DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

19980429 049

USING SIMULATION TO MODEL TIME UTILIZATION OF ARMY RECRUITERS

THESIS

James Douglas Cordeiro, Jr.
First Lieutenant, USAF

Mark Anthony Friend
Second Lieutenant, USAF

AFIT/GOR/ENS/98M-06
AFIT/GOR/ENS/98M-12

WORKING PAPER SERIES

DEPARTMENT OF OPERATIONAL SCIENCES

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

SCHOOL OF ENGINEERING
Wright-Patterson Air Force Base, Ohio

DTIC QUALITY INSPECTED 4

AFIT/GOR/ENS/98M-06
AFIT/GOR/ENS/98M-12

USING SIMULATION TO MODEL TIME UTILIZATION OF ARMY RECRUITERS

THESIS

James Douglas Cordeiro, Jr.
First Lieutenant, USAF

Mark Anthony Friend
Second Lieutenant, USAF

AFIT/GOR/ENS/98M-06
AFIT/GOR/ENS/98M-12

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

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.

AFIT/GOR/ENS/98M-06
AFIT/GOR/ENS/98M-12

Using Simulation to Model Time Utilization of Army Recruiters

THESIS

Presented to the Faculty of the School of Engineering of the Air Force Institute of Technology, Air University, In Partial Fulfillment of the Requirements for the Degree of

Master of Science

James Douglas Cordeiro, Jr., B.A., M.S.
First Lieutenant, USAF

Mark Anthony Friend, B.S.
Second Lieutenant, USAF

Air Force Institute of Technology
Wright-Patterson AFB, Ohio
March 1998

U.S. Army Recruiting Command (USAREC)

Approved for public release; distribution unlimited

AFIT/GOR/ENS/98M-06/98M-12

AFIT/GOR/ENS/98M-06
AFIT/GOR/ENS/98M-12

THESIS APPROVAL

Student: Mark A. Friend, Second Lieutenant, USAF Class: GOR-98M

Student: James Douglas Cordeiro, Jr., First Lieutenant, USAF Class: GOR98M

Title: Using Simulation to Model Time Utilization of Army Recruiters

Defense Date: 6 March 1998

Committee:	Name/Title/Department	Signature
Advisor	J.O Miller, Lt. Colonel, USAF Assistant Professor Department of Operational Sciences	
Advisor	Kenneth Bauer, Ph.D. Professor Department of Operational Sciences	
Reader	Jack Kloeber, LTC, USA Assistant Professor Department of Operational Sciences	

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Acknowledgments

We would like to extend our sincere thanks to our advisors, Dr. Kenneth Bauer, Lt. Col. J.O. Miller, and LTC Jack Kloeber for their assistance and insights, which provided the direction that the study took. They also gave their time to help us prepare for briefings of the Army Recruiting Study given at AFIT and USAREC HQ as well as ironing out the various difficulties we encountered during the course of this research.

Our thanks also go out to the recruiters and station commanders of the Dayton area recruiting stations, which include Huber Heights East and West and Xenia, and also the Middletown, OH, Bethel, OH, Columbus, OH, Marion, OH, and Covington, KY for their patience in filling out the questionnaires and explaining Army Recruiting to us. We are grateful for the support of the Deputy Commandant of the Army Recruiting and Retention School, LTC Floyd K. Maertens, as well as SFC Balser for giving us the opportunity to learn recruiting directly from the source.

Finally, we would like to thank our spouses Lara and Carmen and the "kids", Micah and Mozart, for their understanding and consideration during the long and often grueling process of conducting the study and preparing the document.

Table Of Contents

Acknowledgments ii

Table Of Contents iii

List Of Figures x

List Of Tables xii

List of Symbols xiv

Abstract xviii

Chapter 1. INTRODUCTION 1

1.1 Statement of the Problem 1

1.1.1 Background 1

1.1.2 The Army's Solution 4

1.2 Objective 5

1.3 Approach 5

1.4 Scope 10

Chapter 2. APPROACH TO MODEL CONSTRUCTION 11

2.1 Identify the Problem and Plan the Study 11

2.1.1 Using Recruiter Conversion Data 11

2.1.2 Construction of the Process Description 16

2.2 Observe the System and Define the Model 18

2.2.1 Planning and Lead Generation 18

2.2.2 Prospecting 24

2.2.3 Sales 28

2.2.3.1 The Sales Presentation 30

2.2.3.2 Pre-Testing 31

2.2.3.3 Referrals 32

2.2.3.4 Ancillary Activities 32

2.2.4 Processing 33

2.2.4.1 Normal Processing 34

2.2.4.2 Immediate Processing 39

2.2.5 Waivers 42

2.2.5.1 The Medical Waiver 42

2.2.5.2 The Moral Waiver 46

2.2.6 Delayed Entry Program Sustainment 49

2.2.6.1 Loss Factors 50

2.2.6.2 Length of DEP 52

2.2.6.3 DEP Sustainment Procedures 52

2.2.7 Overall Value of the Study Methods 54

Chapter 3. METHODOLOGY 56

3.1 Model Concepts 60

3.1.1 Analytical Methods 61

3.1.1.1 Random Number Generation 61

3.1.1.2 Input Analysis: The Triangular Distribution 63

3.1.1.3 Input Analysis: The Exponential Distribution 65

3.2 Simulation Tools 66

3.2.1 MODSIM 67

3.2.1.1 MODSIM Program Format 67

3.2.1.2 Simulation Constructs in MODSIM 68

3.2.2 SimProcess 70

Chapter 4. MODEL DESIGN AND CONSTRUCTION 73

4.1 Programming Concepts 73

4.1.1 Model Overview 73

4.1.1.1 Object Orientation 74

4.1.1.2 Ranked Lists and Interrupts 77

4.1.1.3 How Statistics are Collected 79

4.1.1.4 The Top-Level Design of the SimProcess Model 80

4.1.1.5 Interrupts in SimProcess 81

4.1.1.6 Requisition of Resources for Entity Generation 83

4.1.1.7 Model Assumptions: 84

4.2 Implementation of the Recruiting Model 85

4.2.1 Generating Applicants: Prospecting and Walk-Ins 85

4.2.1.1 MODSIM-Specific Issues in Applicant Generation 86

4.2.1.2 Generation of Applicants in SimProcess 89

4.2.2 Selling the Army 90

4.2.2.1 Overview of the Sales Process Implementation 90

4.2.2.2 Applicant Loss and Interrupts 92

4.2.2.3 Modeling Issues 93

4.2.2.4 MODSIM-Specific Issues 93

4.2.3 Processing Applicants 94

4.2.3.1 MODSIM Processing 94

4.2.3.2 The SimProcess Implementation of Processing 97

4.2.4 Sustaining DEP Applicants 103

4.2.4.1 Acceptance of Entities into DEP Sustainment 104

4.2.4.2 DEP Sustainment in MODSIM 104

4.2.4.3 DEP Sustainment in SimProcess 106

4.2.5 Collateral Duties in SimProcess 107

4.2.6 Modeling Issues 108

4.2.6.1 Model Use 108

4.2.6.2 Model Output Interpretation 110

4.2.6.3 Process Features Not Addressed by the Model 111

4.3 Verification and Validation (V & V) 114

4.3.1 Verification 116

4.3.1.1 Writing and Debugging 116

4.3.1.2 Code Verification by Multiple Individuals 116

4.3.1.3 Checking Output 117

4.3.1.4 Tracing 118

4.3.1.5 Animations 118

4.3.1.6 Use of Simulation Packages 119

4.3.2 Validation 120

4.3.2.1 Conversations with System Experts and Observations of the System 121

4.3.2.2 Existing Theory 123

4.3.2.3 Turing Test 123

4.3.2.4 Validation During Simulation Runs 124

Chapter 5. OUTPUT ANALYSIS 126

5.1 Approach to Analyzing Army Recruiting Output 126

5.1.1 The Experimental Design 127

5.2 Results 129

5.2.1 Comparison of the Mean Number of Contracts Produced 129

5.2.2 Regression Analysis 132

5.2.3 Effect Screening 133

5.2.4 Effect of Waivers 136

Chapter 6. CONCLUSION 138

6.1 Summary 138

6.2 Recommendations 141

Appendix A. ANECDOTAL DATA 144

A.1 Data 144

A.1.1 Quality of Life Issues 144

A.1.2 The Job 145

A.1.3 The Recruiter's Views on Society 147

A.1.4 Technology 148

A.2 Questionnaire 149

Appendix B. SIMULATED RECRUITER SCHEDULES 153

Appendix C. MODSIM CODE 158

C.1 Recruiting Station 158

C.1.1 Definition Module 158

C.1.2 Main Module 163

C.1.3 Implementation Module 166

C.2 Ranked List 221

C.2.1 Definition Module 221

C.2.2 Implementation Module 222

C.3 Calendar 223

C.3.1 Definition Module 223

C.3.2 Implementation Module 224

C.4 Statistics 228

C.4.1 Definition Module 228

C.4.2 Implementation Module 230

C.5 Input 233

C.5.1 Definition Module 233

C.5.2 Implementation Module 234

C.5.3 Main Module 255

Bibliography 256

Vita 258

List Of Figures

Figure 1. Ten-Step Approach to a Simulation Study (Law & Kelton) 9

Figure 2. The Recruiting Process 15

Figure 3. Planning 23

Figure 4. Prospecting Process Flow 25

Figure 5. Sales Process Flow 29

Figure 6. Normal Processing Flow 38

Figure 7. Immediate Processing Flow 41

Figure 8. Medical Waiver Process Flow 45

Figure 9. Moral Waiver Process Flow 48

Figure 10. DEP Process Flow 50

Figure 11. Model Flow Diagram (Part I) 58

Figure 12. Model Flow Diagram (Part II) 59

Figure 13. The Triangular Distribution 65

Figure 14. Exponential Density Function with Lambda = 1 66

Figure 15. Node Types 70

Figure 16. Pseudo-Interrupt 72

Figure 17. Station Model 77

Figure 18. Simprocess Top-Level 81

Figure 19. Applicant Generation 84

Figure 20. SP Prospect Model 90

Figure 21. SP Sales Model 91

Figure 22. SP Processing Model 98

Figure 23. SP Immediate Processing Model 99

Figure 24. SP Normal Processing Model 101

Figure 25. SP Medical Waivers 102

Figure 26. SP Moral Waivers 103

Figure 27. SP DEP Model 106

Figure 28. Collateral Duties 108

Figure 29. Warmup Concept (from SimProcess User's Manual) 109

Figure 30. Steps in Constructing a Valid Model (Sargent) 115

Figure 31. Graphical Presentation of Results with Bias Removed 130

Figure 32. Graphical Presentation of Results with Bias Included 130

Figure 33. Collateral-Prospecting Interaction Effect 134

Figure 34. Significant Effects Using a Quantile Plot 135

Figure 35. Questionnaire (Page 1) 150

Figure 36. Questionnaire (Page 2) 151

Figure 37. Questionnaire (Page 3) 152

List Of Tables

Table 1. Station Conversion Data for one Quarter 12

Table 2. SimProcess Node Types 71

Table 3. V & V Activities (Based on Law & Kelton, Chapter 5) 115

Table 4. Design Points 128

Table 5. Simulation Output (Reps = 30, Time = 2 Years, Bias Removed) 129

Table 6. Parameter Estimates of Model Fit to Unbiased Data 132

Table 7. Comparison of Parameter Estimates of Model With and Without Processing 133

Table 8. Parameter Estimates of Model With Negligible Effects Removed 135

Table 9. Parameter Estimates of Model With Negligible Effects Removed 136

Table 10. Summary of Regression Performed with Significant Effects 136

Table 11. Point 1 153

Table 12. Point 2 154

Table 13. Point 3 154

Table 14. Point 4 155

Table 15. Point 5 155

Table 16. Point 6 156

Table 17. Point 7 156

Table 18. Point 8 157

Table 19. Point 9 157

List of Symbols

Acronyms and Abbreviations

Abbreviation	Definition
AFIT	Air Force Institute of Technology
ACF	Army College Fund
AD	Active Duty
ADSW	Active Duty for Special Work
AFQT	Armed Forces Qualifying Test
AMEDD	Army Medical Department
ARADS	Army Recruiting and Accession Data System
ASVAB	Armed Services Vocation Aptitude Battery
BT	Basic Training
CAST	Computerized Adaptive Screening Test
C/F	Carried Forward
CIHS	Currently In High School
CLL	Consolidated Lead List
CLT	Company Leadership Team
COI	Center(s) of Influence
DBM	Dominant Buying Motive
DEP	Delayed Entry Program
DOB	Date of Birth
DOD	Department of Defense
DOS	Days of Service

DTP	Delayed Training Program
EST	Enlisted Screening Test
FEBA	Facts, Evidence, Benefits, and Agreement
FLOOR	MEPS Processing
FY	Fiscal Year
FYTD	Fiscal Year to Date
GC	Guidance Counselor
GSA	Grad Senior, I-IIIA
HQ USAREC	Headquarters, United States Army Recruiting Command
HRAP	Hometown Recruiter Assistance Program
HS	High School
HSSR	High School Senior
IST	Individual Sustainment Training
JADOR	Joint Advertising Directors of Recruiting
JOIN	Joint Optical Information Network
LEADS	Lead Evaluation and Distribution System
LRL	Lead Refinement List
MEPS	Military Entrance Processing Station
MET	Mental Evaluation Testing or Mobile Examining Team
MOS	Military Occupational Specialty
NCO	Noncommissioned Officer
OCS	Officer Candidate School
PDR	Prospect Data Record (USAREC FM 200-C)
PR	Performance Review

PS	Prior Service
PSAT	Preliminary Scholastic Aptitude Test
PT	Physical Training
RA	Regular Army
Rctg Bde	Recruiting Brigade
Rctg Bn	Recruiting Battalion
Rctg Co	Recruiting Company
ROTC	Reserve Officer Training Corps
RPI Maintenance	Ensuring availability of Army brochures and pamphlets
RS	Recruiting Station
RSB	United States Army Recruiting Support Battalion
RSM	Recruit Ship Month
SAT	Scholastic Aptitude Test
SPMS	Station Production Management System
SY	School Year
TAIR	Total Army Involvement in Recruiting
TDA	Tables of Distribution and Allowances
TPU	Troop Program Unit
TSC	Test Score Category
USAR	United States Army Reserve
USAREC	United states Army Recruiting Command
VACPOT	Vacancy Potential Transcripts
VIP	Very Important Person
WOFT	Warrant Officer Flight Training

YTD	Year to Date

Abstract

It is well-known fact Army recruiters work very long hours in a demanding environment. In many cases, recruiting stations are geographically isolated from military bases, with recruiters often tolerating a high cost of living, crime, and other such adverse conditions that characterize the communities they work in. The job itself demands self-starting, motivated individuals with a wide range of skills, from street-savvy to salesmanship, in order to succeed. A number of factors in recent years have made military recruiting more difficult, which include scandals involving highly-placed soldiers and changes in attitudes towards military service among eligible men and women. A recent mission increase has exacerbated this problem even further for the many recruiters who must shoulder this burden. Unlike previous studies which have concentrated on the effects of advertisements and other determinants of enlistments in the Army, this study instead focuses on the individual recruiters themselves, with the ultimate purpose of defining the relationship between the various recruiter tasks and the end product - qualified Army recruits.

The key step towards the accomplishment of this goal was the determination of which factors influence recruiter effectiveness. In the course of developing a model and subsequent computer simulation of the recruiting process, a thorough process flow description of the major recruiter tasks was generated. Task completion times were estimated on the basis of empirical studies of actual recruiting stations in anticipation of their use as model input parameters. All of this information was then incorporated into working Simprocess and ModSim computer simulations of a single recruiting station with an arbitrary number of recruiters. Finally, sensitivity analysis of recruiter output with respect to various input parameters was accomplished using the techniques of simulation output analysis.

Chapter 1 - Introduction

1.1 Statement of the Problem

The Army has been an all volunteer force since 1 July 1973 when, due to controversy surrounding the Vietnam War, the draft was finally discontinued[1]. In FY 1997, for only the second time since its conversion to an all volunteer force, the Army came very close to not meeting its annual recruiting goal of 89,700 new recruits. This situation reflects a recent trend in which fewer and fewer young people are choosing an Army career, electing instead to compete for seemingly higher-salaried positions in the civilian community. Several factors play into the difficulties plaguing Army recruiting. One is the markedly strong national economy that characterizes the latter part of the decade of the 1990s. Another is an image problem resulting from both the loss of the warrior culture [21] and recent scandals involving sexual harrassment of female recruits and personnel. The purpose of this study is not to suggest a strategic solution at the leadership levels of the Army, but rather to study the effects of changes at the recruiting station level. The men and women at the recruiting front, so to speak, ultimately determine the performance of Army Recruiting as a whole. For this reason, the recruiter must be considered a critical element of any effort to counteract the negative impact of changes in societal trends.

1.1.1 Background

In addition to the factors stated above, the rather large increase of 26,700 in the recruiting quota from 1995 to 1997 has severely strained the Army's efforts to meet its mission goals[9] . The current methods and capabilities of the U.S. Army Recruiting Command (USAREC), while

[1] According to [26] , the last draftee entered basic training in June 1973.

sufficient during previous recruiting years, have proved unsatisfactory in light of this development. For one, USAREC has not substantially increased its personnel at the station level, and as a result, existing recruiters have to cope with increasing mission demands. Even if Army leadership authorize personnel increases to deal with the higher quotas, one cannot completely discount the possibility of an even heftier increase in succeeding years. Thus, as has been the theme in the civilian workplace during the past decade, the Army must consider changes that will allow it to "do more with less". This, in turn, implies that greater efficiencies in time management must be introduced at the station level.

Perhaps the important factor affecting recruiting station output is the individual recruiter. Surveys that we have taken from ten different stations seem to resonate with the common theme of long hours with little recognition, low pay, and a great pressure to produce. There are various options to consider in trying to improve the plight of the recruiters, particularly in the area of process reengineering. Some recruiters have suggested a team approach in which recruiters would accomplish tasks in support of station, rather than individual missions. This viewpoint is corroborated by the research of Benjamin J. Roberts, who suggested that "... performance is enhanced when there is greater congruence between the methods of production and the way individuals work together"[22, 216] . Great strides can also be made in the way individuals organize their own time, hence the significance of learning about the sensitivity of recruiter output to how daily tasks are organized.

Army recruiters and training personnel have also become the focus of unwanted attention recently due to allegations of sexual impropriety in the recruiting stations. News reports frequently feature stories of drill sergeants abusing female recruits, and an Army survey revealed that 4 percent of all female soldiers had been a victim of an actual or attempted rape or sexual assault within the previous 12 months. In many cases, the persons responsible were drill instructors[31] [27] . The problem with this form of intense media coverage is the inevitable public relations nightmare, par-

ticularly where eligible young men and women are concerned. Several recruiters we interviewed had mentioned that the topic had been brought up by applicants they had interviewed, thus indicating that people are very aware of this problem.

In addition to the damaging reports of endemic rape and sexual abuse , the Army is experiencing problems projecting an image attractive to young men. Ironically, in its desire to promote gender equality, the Army has lost the "macho appeal" that has traditionally attracted those youths in the highly image-conscious 18 to 25 year-old range. The rationale is that the Army's aggressive recruitment of women to perform the combat tasks originally reserved for men has, in a sense, jeapordized the Army's ability to project itself as being a tightly-knit fraternal organization[21] . The substantial drop in the percentage of Hispanic recruits, which is a benchmark indicator of recruiting success, can be attributed, in part, to this effect. One service that has largely managed to avoid this form of stereotyping has been the Marines. Its success is evinced in the substantial disparity between the Army's 6% composition of Hispanics and the Marine's 14% composition[21] .

While the current difficulties in recruiting are largely a result of the factors discussed previously, there is also a close correlation between enlistment rates and the national economy. The Army can easily meet its recruiting quota when the economy is doing poorly and the unemployment rate is high [5] . However, during periods of economic strength and a correspondingly low unemployment rate, young people have in general, more career options available to them. This makes the Army a tough sell to youngsters unaccustomed to the discipline and level of physical activity required in a military setting. Department of Defense surveys indicate that only 12 percent of the eligible 16-21 year old males were interested in joining the Army, as opposed to the 17 percent figure in 1989[21] .

1.1.2 The Army's Solution

The Army has taken steps to meet its recruiting mission in these tough recruiting years. It has decreased its goal of recruiting non-GED high-school graduates to just 90 percent, which is the minimum allowed by the Department of Defense. The proportion of category 4 applicants (the lowest scoring group on the ASVAB) accepted into the Army has risen from 2 percent to 4 percent in 1997. This decrease in quality did not, however, come without a price. Although the above measures have helped in the short term, comparison studies between recruit quality and incidences of discipline problems reveal a direct correlation, which in turn affects the overall morale and performance of the fighting forces[26] . Some examples of the more notable changes the Army has implemented are the increase of maximum individual enlistment bonuses from $8,000 to $12,000, the augmentation of the recruiting corps by 325 individuals, and a greater investment in recruiting advertising[9] . Despite the promise these measures hold for increasing the overall recruitment figures, they may still not be sufficient to enable the Army to reach its future recruiting goals.

The combined effect of increases in the Army's recruiting goal and a relatively poor recruiting environment necessitates a more proactive attitude in regards to defining the problems and arriving at effective solutions. USAREC has adopted the current trend of basing management policies on the concept of Total Quality Management and has devised plans to implement a wide-reaching reform program called Recruiting 2000. A specific examples of practical changes made in this program is the abolition of the Joint Optical Information Network (JOIN) in favor of an interactive CD-ROM based presentation system using the JRISS laptop computers. The JOIN laserdiscs contained a multitude of 3-5 minute depictions of Army occupational specialties (MOS), special options, and benefits available to new recruits. One of the problems that have plagued JOIN (in addition to the unwieldy bulk of the equipment) was that the potential recruit needed to be present at the recruiting station in order to use the system. The current laptop-based CD-ROM library allows the recruiter to

present the information to a potential recruit anywhere, even in the recruit's own home. Standardized electronic forms, test registering, and e-mail are also being implemented nationwide in recruiting stations in the hope that recruiters can spend more time with the human side of recruiting, and thus increase their production.

1.2 Objective

The objective of this project is to construct and analyze the output of a computer simulation model based on a detailed workflow analysis of the Army recruiting process. This project will also address the relationship between time spent on recruiting tasks and the end output, qualified Army recruits. It is possible that through observation and experimentation, improvements to recruiter time utilization can be discovered and implemented.

1.3 Approach

The authors utilized the following ten-step approach to a simulation study suggested by Law and Kelton[10] :

1. *Identify the problem and plan the study (Chapter 1, Section 2.1):* The first step in any study or project is the statement of the objective. The objective should address why the study is being undertaken as well as what the study hopes to accomplish. Specific issues to be addressed should also be included. Once an objective statement has been formulated, a plan to meet the objectives should be devised. The plan should incorporate as much detail as possible to include (1) the number of people needed to complete the study, (2) the major and minor issues affecting model development, and (3) the criteria for measuring the effectiveness of the solution reached. Additionally, a time line should be generated with the estimated completion date for each stage in the simulation study.

2. *Observe the system and define the model (Section 2.2):* The system to be simulated - which is in this case the recruiting process - must be observed, and all resulting data should be collected. Based on these observations, specific processes to be modeled can be identified and the necessary probability distributions identified. When developing the model, care should be taken to avoid excess detail. The model should only contain enough detail on the system in question to meet the study's objectives. Additional detail in the model adds unnecessary complexity to its structure, thus increasing the model run time and making verification and validation more difficult.

3. *Ensure the model is valid (Chapter 2):* This addresses the need to model the system accurately. It is in the interest of the analyst to involve both process experts and decision makers in all aspects of the development of the model. The model itself should be reviewed at each major stage by someone with expertise in the process being modeled in order to ensure that the simulation reasonably corresponds to the real-world system. The input of the decision maker(s) is especially important since they have at least a say, if not the authority, to accept or reject the model. Verification and validation alone do not necessarily guarantee acceptance of the model.

4. *Construct a computer program based on the model (Chapter 4):* The next step is to implement the abstract model as a computer simulation. Simulation programming languages reduce the amount of coding needed by incorporating many of the standard mechanisms of a simulation into the design of the language. The analyst can avoid the necessity of coding a variety of functions from scratch, thus speeding the verification process. Nevertheless, simulation languages do possess disadvantages, of which the most troublesome is perhaps the need to overcome steep learning curves, particularly with brand-new platforms. Another potential downside is the loss of the flexibility that general-purpose programming languages

provide to the patient modeler. An analyst must weigh the relative advantages of both types of platforms and choose one that will best suit the operating constraints of the modeling project[2]. Once the program is completed, the analyst must verify that it corresponds satisfactorily to the previously validated model. This often involves a thorough re-checking of the program logic to confirm that model processes are neither omitted nor misrepresented. As a rule, the larger and more complex a program is, the harder it is to ascertain whether or not it is working correctly. Often, a considerable amount of debugging is necessary to ensure the proper functioning of a computer simulation.

5. *Get sample data through pilot runs (Chapter 5):* Pilot runs are performed to help validate the completed program. They can be used to test the sensitivity of the simulation output to variations in the parametric input to the program.

6. *Check if the newly verified model is valid (Section 4.3):* Pilot runs should be conducted on the input variables to test the simulation's results with respect to variations in the input variables. If the output obtained via test runs does not reasonably correspond to that of the real system (given the same parameters), there is a distinct possibility that faulty logic or other errors may yet be present within the code. Care should be taken to ensure that both the extremes as well as more normally seen input patterns are tested. If the real system and the simulation do correspond, then the model can be thought as being "valid".

7. *Design experiments to meet objectives (Chapter 5):* In many cases, the large number of different combinations of input variables makes an exhaustive approach impractical. An experimental design approach, on the other hand, permits the analyst to circumvent the need to perform hundreds of runs while sacrificing little, if any, statistical accuracy. In many instances,

[2] We have chosen MODSIM and SimProcess (trademarks of CACI, Incorporated) for the purposes of building the Army Recruiting simulation model.

output from these initial runs is used to generate further program runs. When designing experiments, decisions must be made concerning the length of the warm-up period, the initial conditions of the input variables, the number of production runs, and the duration length of each run.

8. *Make production runs of the experimental design (Chapter 5):* Simulation runs of the experiments designed in the previous step are performed to output the desired measures, which will subsequently be analyzed for trends with respect to changes in model parameters.

9. *Analyze the output data generated (Chapter 5):* The mean, variance, standard deviation, as well as the parameters for the distributions of the measures of effectiveness for the simulation can be computed from the output generated from the production runs. The most effective system based on the various input parameters can then be determined through the use of statistical methods such as Response Surface Methodology (RSM).

10. *Document and present the simulation results (Chapter 4):* It is crucial that the modeler document all assumptions made during model and subsequent simulation development. Knowing the simplifying assumptions will help the decision maker make informed decisions based on the simulation output. The results of the output analysis performed in the previous step should be presented to the decision maker in as clear and concise a manner as possible. Acceptance of these results is a function of the believability of the simulation as well as the effectiveness of the presentation.

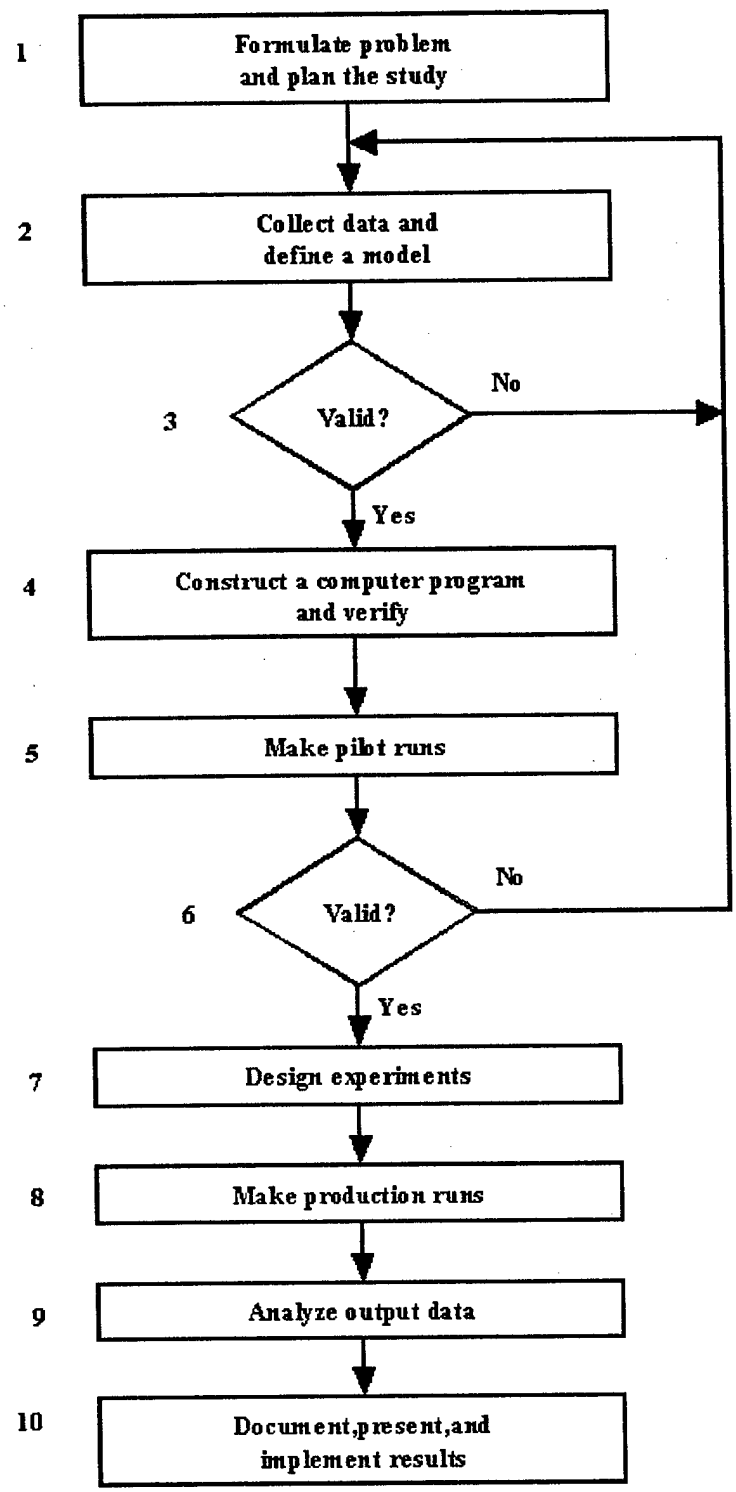

Figure 1. Ten-Step Approach to a Simulation Study (Law & Kelton)

1.4 Scope

This study will only focus upon the analysis of a single recruiting station with one station commander and one or more recruiters, not with groups of recruiting stations. The study will be concerned with the number and type of recruits produced by recruiting stations based on priority. It will also examine the duration of tasks and the effectiveness of the recruiters.

Chapter 2 - Approach to Model Construction

2.1 Identify the Problem and Plan the Study

It is a well-known fact that Army recruiters work very long hours in a demanding work environment. To make matters worse, the recent mission increase, together with certain sociological-political factors, is hampering the Army in its efforts to meet its mission quota for the number of new recruits. The purpose of this study is to ascertain whether simulation can help pinpoint tasks or processes that critically impact recruiter effectiveness. This information will allow the Army to assist its recruiters in achieving their full potential, thereby optimizing the effectiveness of USAREC as a whole. The study will focus on the relationship between time spent on recruiting tasks and the desired output, namely qualified Army recruits. By determining the factors that induce the largest changes in recruiter output, we can proceed to suggest appropriate changes to recruiter time utilization.

As an aid to gaining a thorough understanding of the recruiting process, a detailed process flow diagram was generated based on the study of Army regulations and visits to a local recruiting station. Five visits were made to the station to conduct the on-site observations, during which recruiters were observed conducting their daily activities and their insights and opinions garnered. All of this information was then consolidated and used to construct a detailed process flow diagram. Validation of the process flow was accomplished both by Army recruiters at the same local station and an instructor at the Army's Recruiting and Retention School (Ft. Jackson, SC). Their suggestions for change were subsequently incorporated into the existing process flow diagrams.

2.1.1 Using Recruiter Conversion Data

The level of detail contained within this process flow indeed turned out to be inappropriate for wholesale inclusion into the actual simulation model. The process flow did, however, greatly

expedite the construction a higher level process flow model [30, 22] which did form a useful framework for the computer model. The process flow for the model was loosely based on the Mission Box Plan (MBP)[3] that recruiting station commanders complete each Recruit Ship Month (RSM). Station commanders use the MBP in order to help plan station activities in support of the mission box requirements (or goal) for the RSM. The MAP form itself contains the past 12 months of historical production data, which is calculated from the data obtained from the Conversion Data Chart (USAREC Fm 635-B), a sample portion of which is shown in Table 1. The Conversion Data Chart breaks down recruiter activities into 6 major blocks: number of appointments made, number of appointments conducted, number tested, number that passed the test, number processed through MEPS (Floor), and number contracted. These activities are further broken down by the number of individuals belonging to the following categories: high school graduates (Grad), high school seniors (Sr), category "A" graduates (GA), and category "A" seniors (SA). A number of simple calculations are performed to determine the exact number in each category per contract, first for each recruiter, and henceforth for the entire station. These statistics provide a numerical estimation of a recruiter's (station's) effectiveness as well as an indicator of exactly what a recruiter (station) must accomplish in order to meet the current month's mission requirements. The conversion data is incorporated into the MBP, which is then validated at the Company level and stored in the station's files for as long as 12 months.

	Appt Made		**Appt Cond**		**Test**		**Test Passed**		**Floor**		**Contract**	
RSM & YR	Grad	Sr	Grad	Sr	Grad	Sr	GA	SA	GA	SA	GA	SA
Jan-97	28	9	15	12	3	2	1	1	1	1	0	1
Feb-97	30	16	20	8	4	2	3	2	4	2	4	1
Mar-97	27	15	20	5	5	3	3	1	3	2	2	1
QTR CONV	14.2	13.3	9.2	8.3	2.0	2.0	1.2	1.3	1.3	1.7	1.0	1.0

Table 1. Station Conversion Data for one Quarter

[3]Formerly referred to as the Mission Accomplishment Plan, or MAP.

The number of appointments made is a measure of effectiveness directly associated with the primary recruiting task of prospecting. One of the most challenging duties required of an Army recruiter is the need to "work the field", or go into the community to seek military-eligible youngsters. Various methods are employed to facilitate these efforts, such as maintaining a presence in area high schools and casing shopping malls, video arcades, and other places where groups of teenagers congregate. As can be expected, the Army is most interested in recruiting those applicants who are high school graduates that have scored 50 or above on the AFQT (Grad Alphas). Also considered desirable are high school seniors who have performed well on the exam, although the lengthy wait for their graduation imposes the risk of change of situation that may result in the loss of an applicant. Others, namely those who have scored poorly or who have merely obtained a high school equivalency (GED applicants), are obviously not as sought-after, but are nevertheless allowed to join in limited numbers - particularly if an insufficient number of individuals from the previous groups have been recruited. The most significant drawback of recruiting from this category of applicants, as a recruiter from the Middletown, OH station had pointed out, is that they are not counted in a recruiter's conversion data. This inevitably results in a downward bias in an individual recruiter's output, which may in turn entirely overstate the amount of work required per contract.

On the other hand, the number of appointments conducted is a measure of how prolific a recruiter is at the next primary task: sales. It is during this stage of the process that the recruiter employs his or her skills at persuasiveness to entice an applicant to, at the very least, consent to AFQT testing - if not a lifetime career in the armed forces. It is important to mention that most recruiters lose the largest proportion of their applicants after the sales interview, and so the conversion figures remain relatively high at this stage. The reason for this high loss rate at this stage is because pre-processing screening occurs during the sales interview; certain stations are reputed to disqualify greater than ninety percent of their applicants for demographic reasons. These figures are again

discriminated into the categories of senior and graduate in order to keep track of how successful recruiters have been at high school recruitment.

The next four groups - number tested, number that passed, number processed through MEPS, and the number of contracts - constitute the bulk of the processing task. Once the AFQT/ASVAB test is administered, the applicants are then further categorized (in terms of mental proficiency) into senior/graduate levels A, B, C, and D based on their test scores. For example, graduate and senior alphas must pass the examination with at least a score of 50 while Betas must achieve a score of at least 35. The number of A and B applicants are then duly recorded in the appropriate block in the MAP. The next block in the conversion data chart is what is termed the "floor". Applicants have completed this step when they have undergone a physical examination at MEPS, passed, and selected a job. Once these and all previous requirements have been satisfied and the oath of enlistment taken, the applicant is deemed to have been contracted.

The conversion data is subsequently tracked over the current recruiting year and is used to predict the required level of effort at each stage to produce one contract. For example, historical data may indicate that recruiter A needs to make 14 appointments in order to generate 1 contract whereas recruiter B only needs 8 appointments. The conversion data also allows the station commander to ascertain each recruiter's strengths and weaknesses. Armed with this information, a station commander can provide assistance or motivation when necessary or perhaps even plan a strategic matching of recruiters to tasks for which they have displayed a particular proficiency. For our purposes, the conversion data chart served a dual purpose. The first was to pinpoint crucial areas in the recruiting process in which applicants were lost, which played a crucial role in determining the structure of the high-level process flow. The other purpose was to facilitate the verification of the field data that we collected from station recruiters. Any claims as to the times required to complete recruiting tasks can be transformed into expected output, which can then be compared to the actual conversion data.

In this fashion, we could reduce the risk of using grossly incorrect parameters for time distributions in the actual simulation model. For instance, automatically calculated fields in the electronic questionnaire that we distributed to recruiters displayed the expected number of contracts, which could then be cross-checked with the conversion chart data.

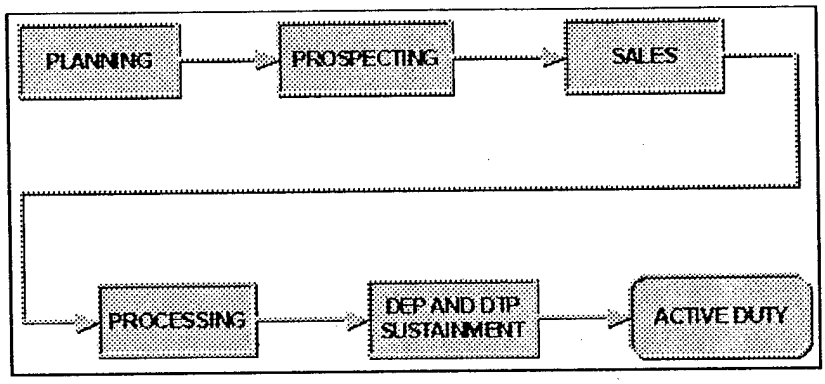

Figure 2. The Recruiting Process

What, then, is the signicance of conversion data to recruiter time utilization in general? Each recruiter has different abilities with regards to the five critical recruiting tasks described in Army regulations (See Figure 2). For instance, some recruiters may be very skilled at selling recruits but poor at making appointments. Others may be great at making appointments but lack the ability to keep the applicant motivated through the sometimes harrowing bureaucratic activities that precede basic training. In other words, different recruiters - perhaps even within the same station - will lose varying numbers of applicants at any given stage of the recruiting process. Knowing the percentage of applicants lost at these chokepoints will allow the end user of the model to explore different scenarios and thus determine which processes have the highest impact on recruiter effectiveness. Suppose that an analyst looking at conversion data observed that the recruiter in question was losing applicants during the processing phase. This would indicate that the recruiter has essentially wasted all of the time invested on the applicant up to that point. Due to the time-intensive nature of the

activities that lead up to processing, losing an applicant at this stage in the recruitment process has a significant effect on the overall number of applicants brought into the Army by that recruiter.

2.1.2 Construction of the Process Description

The high level process focuses on five major concerns. The first is that of the accuracy of the process flow representation. Due to the completely interdependent nature of the various stages in the recruitment process, each must be modeled as closely to the real-world process as possible. The need for such exactness rests in the fact that moderate levels of inaccuracy (in the simulation) tend to become amplified from stage to stage due to the dependence of each latter stage on the output of the former. The second item of interest is the priority a recruiter assigns to each essential task. It is inherently obvious that it is necessary to prospect in order to bring qualified applicants into the Army. However, because a recruiter may be taking care of applicants at different stages, time spent on other activities takes corresponding amounts of resources away from prospecting. For example, processing a new applicant requires a significant investment of the recruiter's time. On the other hand, neglecting Delayed Entry Program (DEP) sustainment wholly in favor of processing unduly increases the risk of a loss. The key to success is clearly a matter of time management - that is, of balancing priorities in such a way as to optimize the number of contracts. In order to meet the study's objectives, the model must allow the end user to explore various recruiting scenarios based on different task priorities. We expect several measures, such as the length of the workday and the number of contracts, to be sensitive to the assignment of priorities.

The third item of interest is how the passage of time should be modeled. Wait and service time distributions, together with the assignment of priorities, determine the duration and precedence of recruiter activities during the course of a simulation run. These activities, in turn, determine the number of applicants a recruiter can contract during any given RSM. To see how this can occur,

consider the medical waiver process. Any applicant who requires a medical waiver may wait months before a decision is rendered. Even if the applicant receives the waiver and enlists, he or she will not be counted for the RSM in which the applicant first spoke to the recruiter. In some cases, such a delay could possibly determine whether or not the recruiter meets mission requirements for a particular month.

The fourth item of interest is the applicant loss percentage at various places within the recruiting process. These probabilities will clearly have a direct effect on the number of contracts produced by a recruiter, and thus on the veracity of the simulation model itself. The reasons for applicant loss vary depending upon the applicant in question, but they can nevertheless be classified into two groups. The first group consists of individuals who have decided against enlisting in the Army because of such factor as a recruiter's poor salesmanship or lack of attention to DEP sustainment. Individuals from the other group have been disqualified for medical or moral reasons, irrespective of their intentions. These losses may occur throughout the recruitment process, all the way up to the moment when the applicant leaves for basic training. The more apt a recruiter is to prevent, or at least control these losses, the more efficient the recruiter will be at time utilization. Therefore, knowledge of when and to what extent losses occur allows investigation into improving the process. From the perspective of the analyst, these loss probabilities are a unique reflection of the talents of the recruiter at different tasks, and are hence invaluable in constructing a valid model.

The last item of interest concerns the amount of time a recruiter spends with the applicant during each task in the process. As noted earlier, recruiters are characterized by their strengths at different tasks - a fact that also manifests itself in varying service time distributions for each recruiter. The recruiter is ultimately constrained by finite time resources, both within the working day and the month in which mission standards must be met. Therefore, too much time spent at one task leaves little for another, making the job of recruiting a juggling act of extreme delicacy. The knowledge

of service times (in addition to the aforementioned loss probabilities) is essential in completing the picture of a recruiter for modeling purposes.

In summary, we require the following parameters:

1. The number of applicants lost at each stage in the recruitment process.
2. The time it takes a recruiter to generate an appointment through telephone prospecting.
3. The time it takes a recruiter to generate an appointment through face to face prospecting.
4. How often walk-in applicants arrive.
5. Percentage of walk-in applicants who fail the pre-sales interview.
6. The average number of applicants who need waivers.
7. The time needed to go through the waiver processes.
8. The percentage of applicants who are immediately processed.
9. The percentage of applicants who are normally processed.
10. The percentage of DEP losses at each successive month for a particular recruiter.
11. The time between DEP meetings and their duration.

Notice that the last two items are correlated, since DEP loss percentages increase with the number of missed meetings.

2.2 Observe the System and Define the Model

2.2.1 Planning and Lead Generation

Planning and Lead Generation is the first of the five critical recruiting tasks, and it is here that the influence of a station commander is most pronounced. Proper planning is the foundation

for all other recruiting activities. Every recruiter is required, by regulation, to maintain a quarterly, weekly, and daily planner. Because time management is so crucial to the functioning of the station, every hour of every working day is plotted in exacting detail in the recruiter's daily planner. It is the station commander's (SC) duty to ensure that a proper amount of time is spent in the critical lead generation and prospecting tasks.

Recruiting station commanders keep track of their recruiters through daily performance reviews (DPR). During these meetings, the commander, together with the recruiter in question, reviews the daily planning guide, Prospect Data Record files, Lead Refinement List, DEP (delayed entry program) sustainment logs, processing list, applicants projected to take the AFQT, applicants scheduled for processing, and school folders[30] . This level of thoroughness enables the commander to pinpoint potential problem areas. From this information, he or she may then formulate a plan of action to mitigate or even prevent applicant losses. Aside from the obvious purpose of providing oversight to the station commander, these sessions also provide the recruiter with an opportunity to ask for the commander's advice and assistance. In summary, DPRs allow the station commander to (1) track events of importance and (2) provide feedback to individual recruiters that will enable them to improve their recruiting skills. How each of the DPR areas actually affect recruiter, and hence station performance, is given below.

One especially crucial item of business is the review of the DEP tracking log. If a DEP applicant decides against the Army due to improper DEP sustainment, then the considerable amount of resources used to recruit the applicant will be entirely wasted. The consequences of losing an applicant from the DEP maintenance pool are particularly severe, as the recruiter will have to make up for the loss of the applicant. This may entail repeating the entire process, perhaps many times, in order to recoup the loss. A detailed analysis of DEP maintenance will be provided later in the chapter.

A Prospect Data Record (PDR) (USAREC Fm 200-C) essentially chronicles every step an applicant takes through the recruiting process. It contains information about the applicant's health, law violations, as well as all personal information known about the applicant and the applicant's family. A PDR also documents by what means the applicant was prospected, as well as every meeting that took place with the applicant. In short, the PDR is a wealth of historical information that may have dozens of potential uses, which include the ability to discern patterns that may suggest how to avoid losses or repeat successes.

The Lead Refinement List (LRL), together with the PDR, daily planner and processing list, give comprehensive feedback to recruiters on how closely they follow their daily planner and also whether or not they are maintaining records properly. This method of overview is called closing the loop, an example of which can be given as follows. Suppose that a station commander discovers from perusing a recruiter's PDR that the source of a particular lead is from a school LRL. The station commander can next look at the processing checklist to ascertain whether or not the processing list information, which documents the applicant's progress through the recruiting process, agrees with the PDR information. The station commander can then verify where the recruiter received the lead by checking the LRL. The LRL documents the time, date and result of contacts made with people on the LRL. The commander can finally compare the information from the LRL with the recruiter's daily planner to ensure that the recruiter obtained the appointment at a scheduled time for prospecting.

The SC review of the daily planning guide perhaps imposes the greatest direct influence upon the activities of individual recruiters. The previous day's plan is reviewed to see if the recruiter in question actually followed the plan. If not, then the station commander annotates the plan with a valid reason for the deviation. When reviewing the recruiter's schedule for the following day, the SC makes sure that the recruiter has set enough time for prospecting and that the plan contains no

tactical errors. If the commander disagrees with the recruiter's projected schedule, he then makes the appropriate revisions. The SC will also discuss with the recruiter any applicants that are scheduled for processing due to the complexity of processing logistics. In summary, the station commander is kept aware of all recruiter plans and activities through the daily performance review. The impact of SCs is also partly due to their role in the training of the recruiters. The ability of a station commander to lead his or her troops is reflected, to some extent, in the amount and quality of the training provided to them. If a station commander maintains an effective training plan and ensures that the recruiters make maximum use of their available time, the recruiting station will have a greater chance at success.

Nevertheless, the goal of a SC during the planning stage should not be to add, delete, or modify daily tasks, but rather to utilize his or her experience to improve their execution. Regardless of the plan, recruiter activities in general will remain the same except for possibly their order and duration. The same five critical tasks are mandated by USAREC regulations regardless of personal priorities or leadership styles, and USAREC enforces their performance through a personal accounting down to the lowest levels. The daily meetings do, however, provide a way in which the station commander can exercise his or her leadership prerogative and actually make a tangible difference in the way the station operates.

The final task classified by regulations as a planning activity is Lead Generation[30] , which is considered to be any method through which a recruiter obtains potential applicants' names, addresses and/or telephone numbers (See Figure 3). There are endless ways in which a recruiter can generate leads, but among the most conventional methods are generating high school lists, ASVAB testing rosters, and utilizing Centers of Influence (COIs). For the sake of simplicity, and because we are not interested in discriminating between the various forms of lead generation, we have chosen in

the process model to aggregate the different tasks into a "Lead Generation" category[4]. Moreover, the close relationship to the critical task of prospecting has prompted the inclusion of lead generation under prospecting in the detailed process flow. Consequently, this activity will be discussed in further detail in the next section.

[4]A follow-on study could easily embellish the model with these details, as we shall discuss later in Chapter 4.

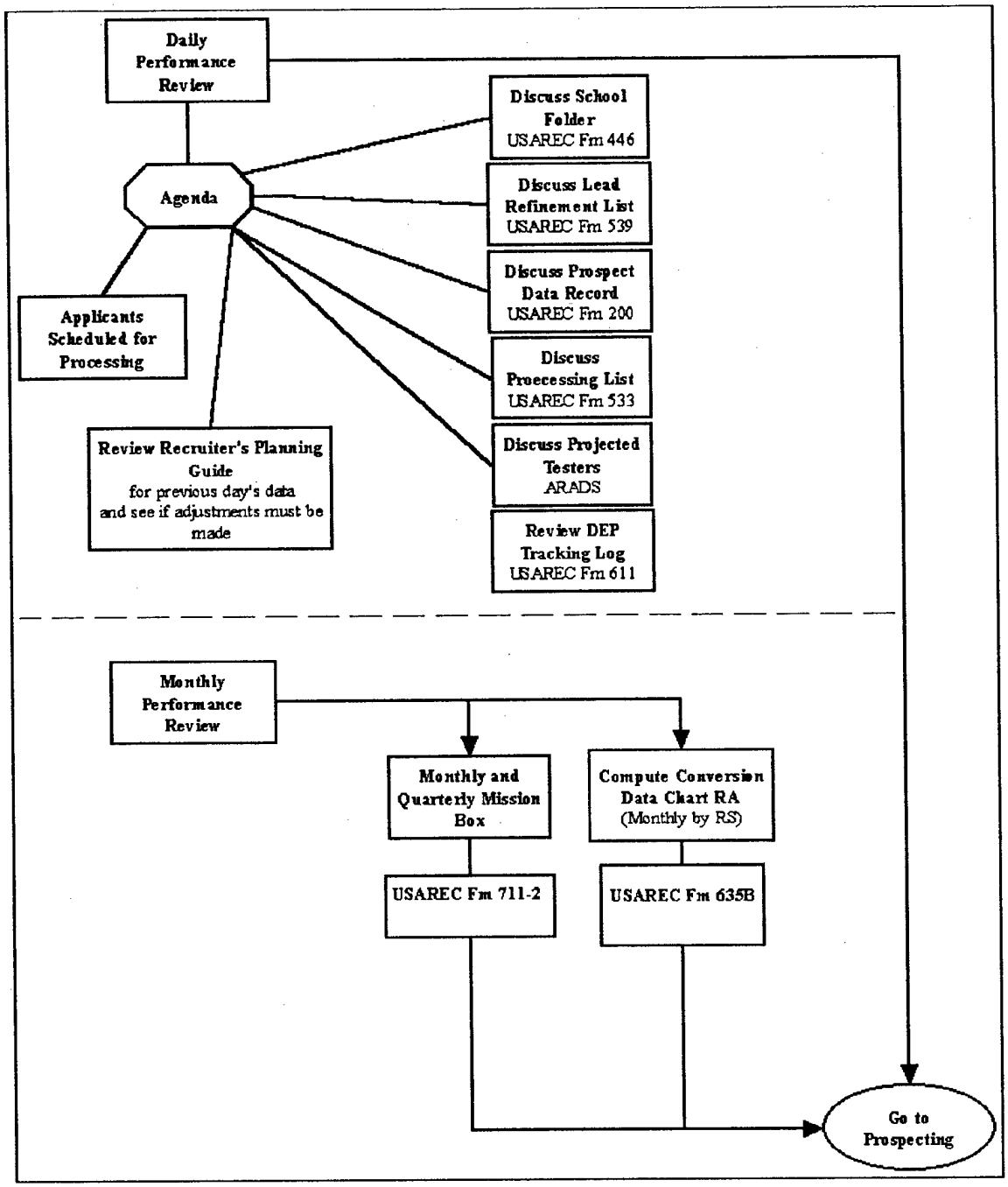

Figure 3. Planning

2.2.2 Prospecting

The prospecting process is the means by which recruiters obtain the applicant input that drives all other recruiting activities (See Figure 4). Without an effective prospecting policy, recruiting stations would not be able to contact, much less enlist, enough people to meet their mission requirements. The recruiter must first, however, lay the initial groundwork for fruitful prospecting by generating enough leads from a variety of sources. Schools, shopping centers, referrals, and even influential members of the community are sources that are constantly relied upon. After finding promising leads, recruiters may pursue them in one of two ways, either through face-to-face contact or telephonic conversations. Recruiters generally rely mostly upon telephonic prospecting because one can reach a relatively large number of people in a much shorter time than it takes to actually drive to, say, a shopping mall or area high school. On the other hand, aggressive face-to-face prospecting combined with excellent salesmanship skills can reap potentially greater rewards overall. Since each method is limited in effect by the abilities of a recruiter, one must find a suitable combination of both to maximize their success in this phase of the recruiting process.

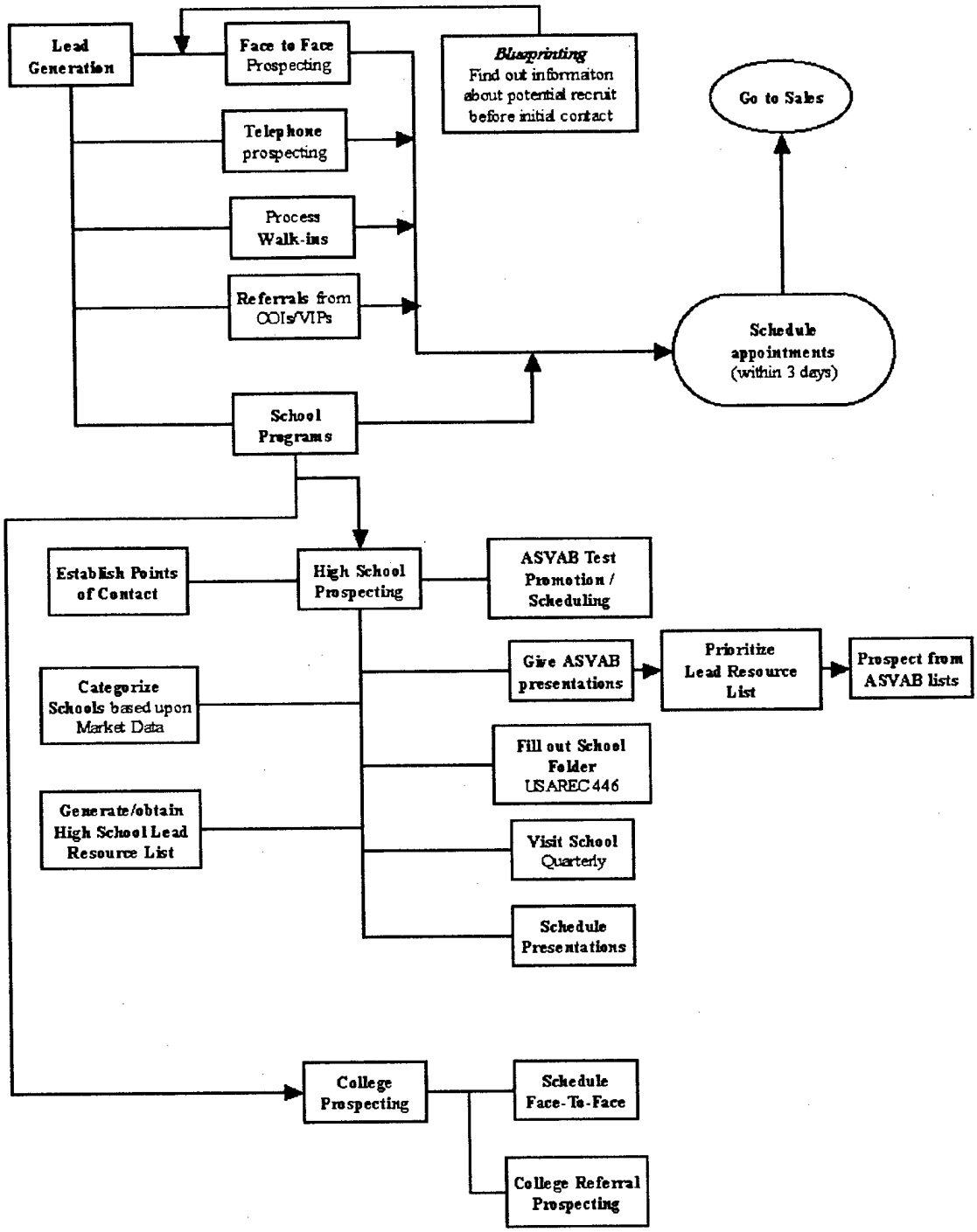

Figure 4. Prospecting Process Flow

High school recruiting is by far the major source of Category A recruits. Applicants in this mental category, namely those high school graduates who score 50 percent or greater on the ASVAB, are obviously the most sought-after by the Army. The Army thus directs correspondingly greater efforts towards lead generation in the high schools than from other sources. In order to assist a station in its prioritization efforts, high schools are rated by historical data on their productivity and are then assigned to the appropriate recruiters. For each high school, recruiters generate what is referred to as a LRL, which contains the names, addresses, and phone numbers of seniors and juniors. Recruiters must also cultivate points of contact in the school in order to establish rapport and thus ensure the continued cooperation of school officials in support of recruiting efforts. One of the most influential of these contacts in terms of their potential affect on student career decisions is the school's career counselor. School counselors can refer qualified and interested students to the Army, provided that they are not prejudiced in some way against a particular recruiter or even military service in general. Recruiters also need to ensure that the school provides for or mandates ASVAB testing since qualified prospects can be found from test result lists. The key to success in accomplishing these goals is simply to establish and maintain a network in the schools, which entails visiting the schools they are responsible for frequently. Regulations mandate at least quarterly visits, but a good recruiter will conduct school visits as often as it is feasible to do so.

On the administrative side, the individual recruiter is also tasked with keeping the LRL and the school folder, USAREC Fm 446, up-to-date. The school folder documents all recruiting efforts and resulting production statistics associated with one particular school. To be more specific, the folder includes information about the number of applicants from the school who have enlisted, ASVAB testing dates, and centers of influence (COIs) whom the recruiter can rely upon to suggest leads. The folder is, in fact, treated so emphatically by those involved in the recruiting business because of the invaluable assistance it provides recruiters in organizing their prospecting efforts at the school.

Without the folder, the business of school prospecting would most likely degenerate into a side activity conducted at the whim of the recruiter.

The recruiter should, at this stage, possess the two most important products of school recruiting efforts: the LRL and the list of students who have passed the ASVAB test. The ASVAB test-result list provides a subset of eligible students, which in turn can be cross-referenced against the LRL to obtain phone numbers. In addition to high school class lists, these items constitute a crucial part of the recruiter's effort to recruit Category A students. Another place one might expect to find "A" candidates is at the local community or four-year college, though these students may already have set career plans. Prospecting from this source is usually done in a face to face manner since student list disclosure does not extend past the secondary schools. These factors combine to make recruiting in colleges a less-fruitful endeavor, and is hence not as highly stressed.

Up to this point, we have only discussed how recruiters initially contact applicants, but not what happens during the contact and afterwards. The primary goal of the recruiter at this stage is to convince eligible applicants into coming to the station for a sales interview. The key word here is "eligible" as the recruiter should not waste any more time than what is necessary in speaking to applicants who clearly do not qualify for Army service. Therefore, screening is another important component of the prospecting task. We make the disclaimer here that the model does not address all of the aspects of telephone or face to face prospecting, but rather assumes that the correct steps between key milestones are followed. Only the essence of the task is captured in the simulation design, namely through utilizing the loss probabilities and service times involved. To reduce the number of applicants not showing up for appointments, all sales interviews are scheduled within 72 hours of prospecting to reduce the number of applicants not showing up for appointments.

For modeling purposes, there are four primary applicant sources: telephone prospecting, face to face prospecting, walk-in applicants, and referrals. Productivity from each type of input varies

depending on the region demographics, station commander effect, recruiter ability, and a number of other factors. Irrespective of relative productivity, however, a recruiter should ideally coax each source to its maximum level of output in order to achieve the greatest amount of efficiency possible.

The detailed process flow shows the four main categories of recruits proceeding to the sales process. However, for the more high-level model, referrals are grouped together with face to face and telephone prospecting. Even if a referral is given to a recruiter, the recruiter must nevertheless speak with the applicant and henceforth schedule the applicant for a sales interview.

2.2.3 Sales

When prospecting, the goal of the recruiter is to identify those individuals who may have even a passing interest in a military career, or would at least be willing to learn more about the Army. Therefore, once a recruiter has persuaded someone to "stop by and have a chat" about the Army, he or she should be striving to draw a firmer commitment from this individual, perhaps by piquing his or her interest or even "inspiring" the person to achieve his or her potential by joining. No single technique works for every individual, and therefore the recruiter must correctly distinguish the effective from the ineffective on an individual basis. An entire theory of recruiting salesmanship has been formulated by USAREC[29] for policy and training purposes. The techniques rely upon basic psychology, time management and organization, and how to tailor the sales pitch according to market conditions and applicant profiles. For a detailed look at the sales process, refer to Figure 5.

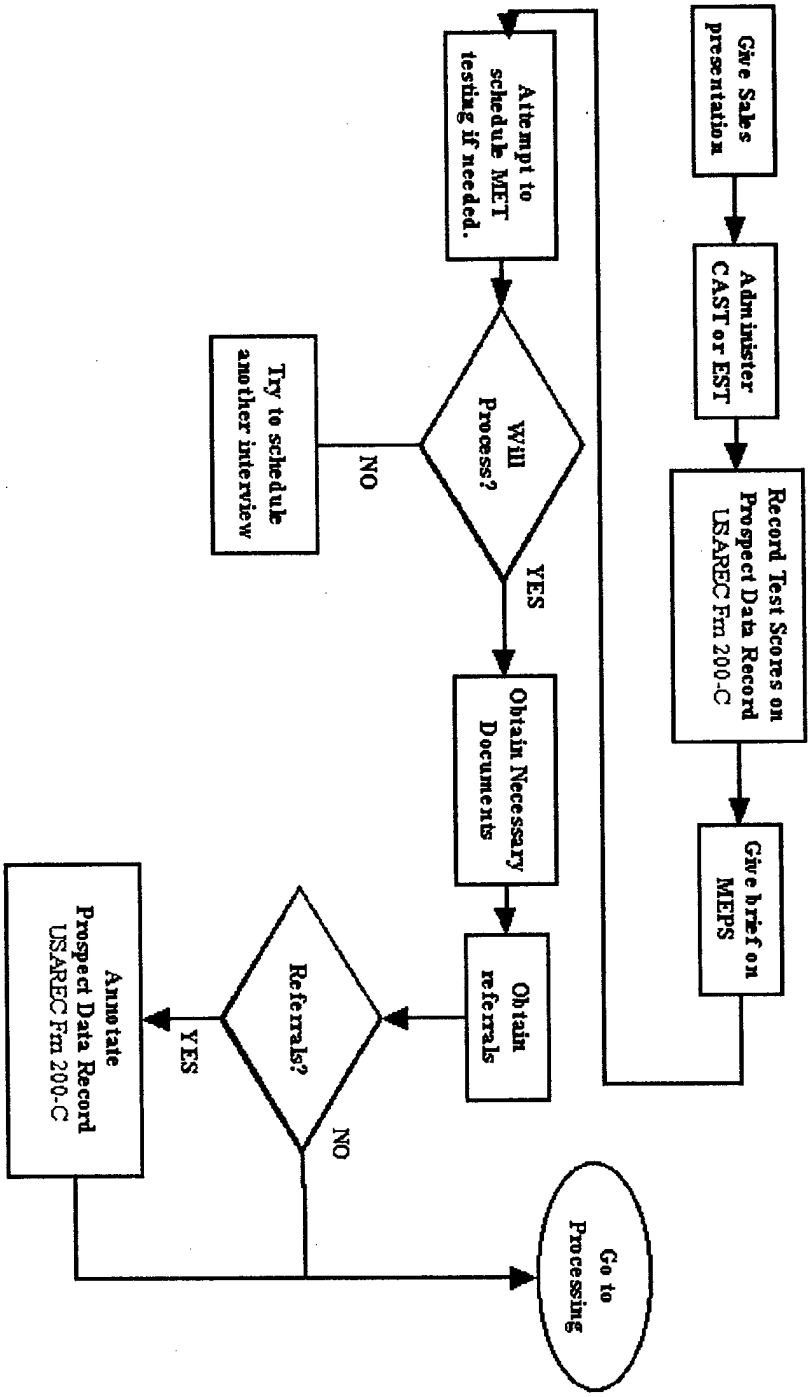

Figure 5. Sales Process Flow

2.2.3.1 The Sales Presentation

The presentation itself is the crux of the entire sales process because it is here that the applicant is either enticed into or discouraged from continuing the process. As a result, every detail of recruiter behavior during the sales pitch must be calculated and controlled according to fundamental psychological principles and sound organizational skills. The sales appointment may have originated as a direct result of a recruiter's prospecting activities, in which case the applicant should already have been screened for obvious disqualification factors. Or, the applicant may have simply walked into the recruiting station and asked to speak to a recruiter for informational purposes. This applicant must then be prescreened at the station before the sales interview is conducted. At the Huber Heights recruiting station, recruiters have estimated that roughly *sixty percent* of walk-in applicants have been eliminated through prescreening due to various disqualifying factors.

In the case of a walk-in applicant, the recruiter may assume (with good reason) that he or she already possesses at least some degree of interest in or at least curiousity about the Army. Hence, the interview may not require as much effort as that for a prospected individual who, after all, is present at the interview due to the initiative of the recruiter. Regardless of the type of applicant, the recruiter must present the Army as possessing whatever a particular individual is looking for, whether it be a specific occupation, benefits, education, or even simple comraderie. The recruiter must then fence objections from the individual. These may manifest themselves verbally or non-verbally, and thus the recruiter must be alert to tell-tale indicators of how effective the sales interview has been. A good closing routine or statement - one that leaves a lasting (positive) impression on the applicant - will alleviate remaining doubts and perhaps even inspire the individual to meet the challenges of Army enlistment. The key to a successful sales pitch lies in understanding what motivates people, and then tailoring one's presentation to answer these motivations.

The length of the sales presentation varies greatly, depending on the indecisiveness of the applicant and the skill of the recruiter. Some applicants may have already made up their mind upon walking through the door, and so could quite possibly state their intention to join within a few minutes. Another may not be fully decided, but is yet not adamant about refusing to join; this applicant may take hours or even require successive appointments (for "re-selling"). The initial appointment(s) requires the administration of a pre-examination which lengthens the appointment, but does not necessarily require the recruiter to actively devote time to the process. However, as we shall discuss next, the pre-test is useful as a screening tool which reduces the risk of devoting too many resources to an applicant who may not be qualified.

2.2.3.2 Pre-Testing

Convincing an applicant to join is one matter, but gauging the qualifications of the individual is yet another, even more crucial responsibility of the recruiter. If the applicant does not have the ability, inclination, or background to pass the enlistment exam (ASVAB and other skills tests), then all of the efforts the recruiter has made up to this point to sell the Army to the applicant will have been for naught. One way to roughly predict whether or not an applicant will pass the ASVAB is by administering the Computerized Adaptive Screening Test (CAST) or the Enlistment Screening Test (EST)[30] . If an applicant passes this examination and professes a willingness to continue with the enlistment process, then a recruiter may attempt to schedule an ASVAB test date and MEPS physical, while simultaneously completing the enlistment package (see 1). Otherwise, a more tortuous route must be taken. For instance, an applicant who fails the pre-test must be provided guidance on how to study for the ASVAB. Subsequent failures are followed by mandatory waiting periods to reaccomplish the examination, which in turn drastically slows down the process. Reluctant applicants, on the other hand, must be scheduled for further sales appointments whose success can, at

best, only be hoped for. All of these remedial activities may occupy substantial blocks of a recruiter's already busy schedule, and must hence be anticipated when setting mission goals or planning a daily schedule. The pre-screening tests allow a recruiter to pinpoint which individuals may require the extra effort. He or she can then determine if it is within the best interests of the Army to pursue the recruitment of these individuals any further.

2.2.3.3 Referrals

A probable source of further leads can be the applicant him or herself during a sales interview. Even should the applicant decide against joining, the recruiter may still have obtained at least one more prospect from the time expended on this particular interview. Better yet, if an applicant provides names of persons who may be interested in the Army, and one or more of these referrals decides to join, then the applicant may become eligible to enlist as an E-2 or even E-3; some even obtain decorations in addition to the bonus enlistments. This fact in and of itself may serve as a good bargaining point for the recruiter during the sales interview. Although the referral program has been fraught with alleged abuses (such as the attribution of non-existent referrals to certain enlistees), referrals result in a mutually beneficial result to both the Army and the applicant, and should thus be pursued by the recruiter whenever possible.

2.2.3.4 Ancillary Activities

The recruiter should communicate to the applicant the need to obtain various forms required for processing. These forms include the birth certificate, court documents, marriage certificates, and other identifying documents that may be required to complete the various forms in the enlistment packet (see 1). Since gathering these forms may in itself occupy much time, the recruiter must insist on their prompt delivery (within reason). The results of the CAST or EST should be recorded on

the PDR (USAREC Form 200-C), as well as any referrals obtained from the applicant during the sales interview(s).

2.2.4 Processing

After the sales interview, the applicant will have embarked on one of three courses of action: (1) decide against the Army, (2) take the ASVAB prior to making a firm commitment, or (3) enlist. However, the desire to commit does not preclude the possibility that the applicant may yet be disqualified. There are still the background checks, the ASVAB and other skill examinations, and a physical examination. Additional complications may arise if the applicants require a moral or medical waiver, which are certainly not guaranteed and usually require significant amounts of time and effort.

Army regulations describe one approved course of action in regards to the processing activity. However, the study revealed that recruiters actually employ two separate methods, which we shall term "immediate processing" and "normal processing". The methods do not differ insofar as the number of tasks involved; rather, the order and timing in which the steps are completed provide the distinguishing factor. As we have found from our field investigations, the path chosen actually depends upon the experience of the recruiter involved. Inexperienced recruiters most often go "by-the-book", while those with a few more months or years under their belt will most likely improvise in order to maximize the efficiency of the operation. Immediate processing usually comes into play if, after completing a sales interview, an applicant indicates they wish to join the Army. The experienced recruiter will capitalize on this decision by doing everything possible to ensure that the applicant completes processing as quickly as possible. The process officially sanctioned by Army regulations constitutes the "normal processing" method, which we will describe in Section 2.2.4.1.

The Enlistment Packet

- *Enlistment Processing Worksheet* (USAREC Fm 794): Actually a folder containing the documents below. Used as a checklist to ensure completion of the packet.
- *Questionnaire for National Security Positions* (SF 86): Application for a security clearance.
- *Enlistment Eligibility Questionaire* (USAREC Fm 1104): Used to determine the extent, if any, of law violations which may pose a barrier to enlistment.
- *Request for Permissive TDY to participate in Home Recruiters Assistance Program (HRAP)* (DA Fm 4187): Permission for new recruits to travel back to their home of record to assist with recruiting efforts under the HRAP program.
- *Armed Forces Fingerprint Card* (DD Form 2280):
- *Police Record Check* (DD Fm 369)
- *Record of Military Processing - United States Armed Forces* (DD Form 1966): Enlistment papers.
- *Applicant Medical Prescreening* (DD Fm 2246)
- *Probation Officer and/or Court Records Report* (USAREC Fm 1037)
- *RAP Sheet* (DIS 1)
- *ARADS* (USMEPCOM 714A): Used for automated test-scheduling.

2.2.4.1 Normal Processing

The following paragraphs refer to the diagram given in Figure 6. If the applicant did not balk after the sales interview, then he or she has at least agreed to take the ASVAB examination (though more interviews may be needed to convince an unusually indecisive person). This phase of the process is particularly crucial because the outcome of the examination ultimately determines placement as well as entrance criteria. The recruiter will schedule another meeting with the applicant in order to discuss the impact of the test scores upon the applicant's eligibility status. Not achieving the cutoff score of 31 may not necessarily eliminate the applicant's enlistment chances, but will nevertheless severely restrict job opportunities. The station commander may recommend that the applicant retake the test if he or she did not achieve the minimum acceptable score. However, since approving a re-test is entirely the SC's prerogative, the applicant is not guaranteed this opportunity, particularly if the chance of success is small. However, if the SC grants permission, the applicant

must wait at least 30 days until the next attempt. Each subsequent re-test must be followed by a six-month waiting period.

If the applicant passed the examination and has stated a desire to enlist, the recruiter initiates the Enlistment Processing Worksheet (USAREC Fm 794). The USAREC Fm 794 is actually a folder containing all of the items necessary to process the applicant, including testing schedules, security clearance, eligibility determination, and other administrative documents (see the following section for more details about specific contents).

If, during the course of completing the USAREC Fm 794, disqualifying factor(s) are found, then the waiver process is initiated in place of the normal sequence of processing activities. The question now becomes one of the severity of the offenses committed rather than the mere possibility of their having been committed. Experienced recruiters are usually able to discern the waiverable conditions from the unwaiverable, and may therefore be in a position to counsel applicants on the appropriate course of action. Such counseling should be truthful, and should moreover contain the recruiter's assessment of the tractability of the situation. In all actuality, the recruiter should not waste any further efforts on applicants who are clearly ineligible. If the applicant agrees with the recruiter's assessment, then the recruiter annotates the action in USAREC Fm 533 and the Prospect Data Record (PDR). Applicants must, however, be granted waiver consideration if they so request; this situation, of course, compels the recruiter to proceed with the waiver processing paperwork. The outcome of the waiver request is annotated in USAREC Fm 533 and in the applicant's PDR. If the waiver is granted, then the applicant can continue processing. We note here that applicants must be medically qualified before a moral waiver is attempted, since the latter process is often much more involved.

Once an applicant has passed the ASVAB test and the recruiter has completed the Enlistment Processing Worksheet, the entire enlistment package is submitted to the station commander for re-

view. The applicant is then scheduled for a physical examination at the nearest MEPS station. Some recruiting stations are fortunate in that the MEP locations are located conveniently nearby. In the case of Dayton, Ohio, the nearest MEPS is located in Columbus, which is approximately a one hour drive in each direction for a recruiter with van duty. Outlying recruiting stations have an even more difficult time with the distances involved. In light of this inconvenience factor, one can quickly realize the urgent need for the processing paperwork to be in order before the applicant arrives at the MEPS station. Improperly completed enlistment packages often equate to an extended stay at a hotel near the MEPS station until the necessary changes have been accomplished. If the applicant passes the physical examination, they proceed immediately to guidance counselors (at the MEPS location) who assist the applicants in matching their qualifications and desires to the available military occupational specialties (MOS). The final step is the oath, which then initiates the applicant into the Delayed Entry Program.

A host of unanticipated contingencies may arise during the MEPS stay, such as when an applicant is qualified but does not enlist (QNE). Another example would be of an applicant who has tested positive for alcohol or drugs or who has otherwise failed the physical examination. Any of these would pose an almost definite loss to a recruiter who has already invested considerable amounts of time and effort into the recruitment of these individuals. Even more distressing is the fact that many of these situations could have been avoided had the proper screening been performed during the sales or processing stages. Consequently, the recruiter should perform the necessary medical and moral pre-screening actions *before* the applicant sets foot in the MEPS. Such preparation will surely reduce the likelihood of an applicant loss during the MEPS stage of the processing activity.

Recruiters treat medical waivers in the much the same manner as they do moral waivers. If the recruiter feels that the applicant's condition is not waiverable, then he or she will most likely attempt to convince the applicant of the hopelessness of the situation. However, the applicant may

(by regulation) insist upon consideration, just as in case of a moral waiver. It became apparent while observing the process that recruiters put forth different levels of assistance to applicants undergoing the waiver process. Again, the dichotomy of experienced versus inexperienced recruiters dictates the course of action that is eventually pursued. New recruiters, for example, may not have the knowledge to discern between those who have a chance at a favorable decision and those who do not. Although a recruiter has the obligation to assist an applicant who is attempting to obtain a medical waiver, it does not, from a practical standpoint, benefit the recruiter to commit inordinate amounts of effort to unpromising cases. The law of diminishing returns regulates the proportion of effort expended relative to each recruit. Faithful adherance to this principle will almost certainly enable the recruiter to avoid unwelcome surprises late in the recruiting process.

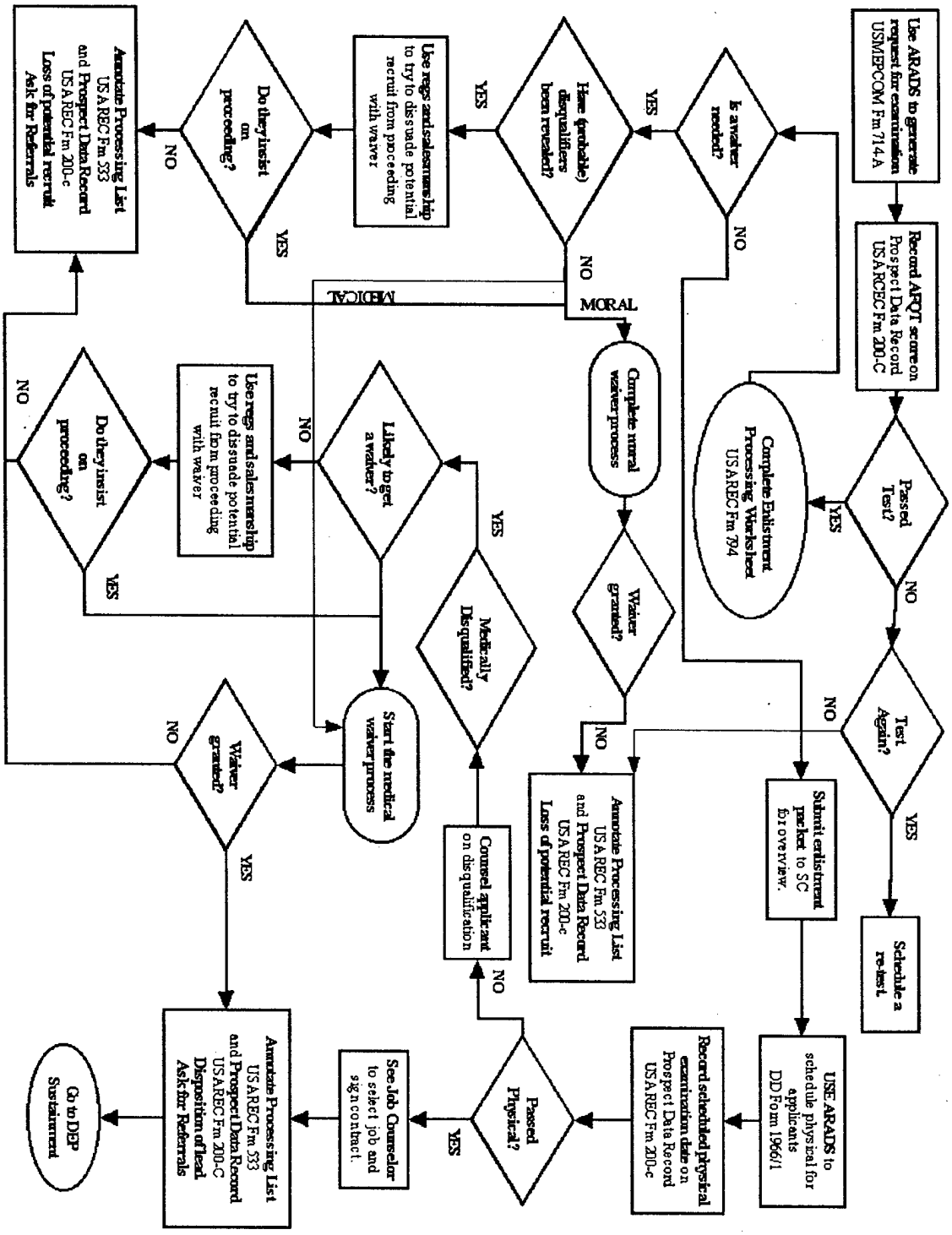

Figure 6. Normal Processing Flow

2.2.4.2 Immediate Processing

Refer to Figure 7 for the following discussion. If an applicant expresses a firm commitment to joining the Army, the experienced recruiter will attempt to enlist the applicant as soon as possible before the applicant changes his or her mind. In what we shall term "immediate processing", the recruiter promptly begins working on the Enlistment Processing Worksheet before scheduling the applicant to take the ASVAB test. Normally, the recruiter will schedule the ASVAB and MEPS physical separately in order to moderate the pace for the sake of more uncertain applicants. This course of action, however, will incur approximately 3 to 7 days of waiting time per appointment. Since time is of the essence, the recruiter will have an applicant test and obtain a physical on the same or consecutive days at the same location (the Columbus MEPS for Ohio recruiting stations), thereby reducing the time needed to complete this sequence of activities. The one catch to a seemingly ideal arrangement is that the trip to the MEPS will be wasted if there is no reasonable assurance that the applicant will pass the ASVAB. For this reason, the SC should approve an immediate processing only if the applicant passes the sales interview pre-test with sufficiently high scores. Should this applicant go on to pass both the actual test and the physical, he or she will stand a fair chance to select an MOS and enlist that very same day. If, on the other hand, the applicant does not pass either the test or the physical, he or she then has the option to apply for a waiver - just as applicants processing normally do.

Of primary concern is the effect of time utilization on the overall output of qualified recruits by a given recruiting station. It is useful for a recruiter in the field or a policy maker to know the specific procedures and regulations that comprise the processing stage. For the purposes of this study, however, we shall emphasize *how much* time is spent rather than how the recruiter spends his or her time. For instance, model parameters will require the average time to complete an enlistment packet rather than the names of specific forms completed. This grouping of processes is referred to

as *aggregation*, and is extremely useful for incorporating effects that, by themselves, are inconsequential, but which together describe a unit whose effects are of interest to the model builders and decision makers. Using this justification, we have aggregated the completion of various forms into a process node we have denoted as "processing paperwork".

For both the immediate processing and the normal processing there are several decision points where applicants either decide against or are else disqualified from joining the Army. For the first contingency, the recruiter's only recourse is to rely on the power of persuasion. USAREC provides a solution to the second problem by leaving open the possibility of obtaining a waiver, provided that the applicant is motivated enough to obtain all of the necessary information required to complete the waiver package. The primary impact of waiver applications on the processing phase is, of course, the added time needed in terms of both labor and other delays. Motivation and time are the key factors in determining applicant loss rates, and thus the waiver process, which does so much to decrease both, marks a major decision point. Nevertheless, disqualification itself does not explain all applicant losses. A certain number of applicants successfully complete the ASVAB are also lost due to any number of factors[5]. Some of these may include the less-than-enjoyable MEPS experience or dissatisfaction with a perceived lack of truthfulness on the recruiter's part. Irrespective of the reason, QNEs are usually considered to be an indicator of poor recruiting practices if they occur with any kind of frequency. Applicants who remain with the Army at this stage are considered to belong to what is called the "Delayed Entry Program", or DEP (refer to later sections of this chapter).

[5]Commonly referred to as "QNE", or "Qualified, but did Not Enlist".

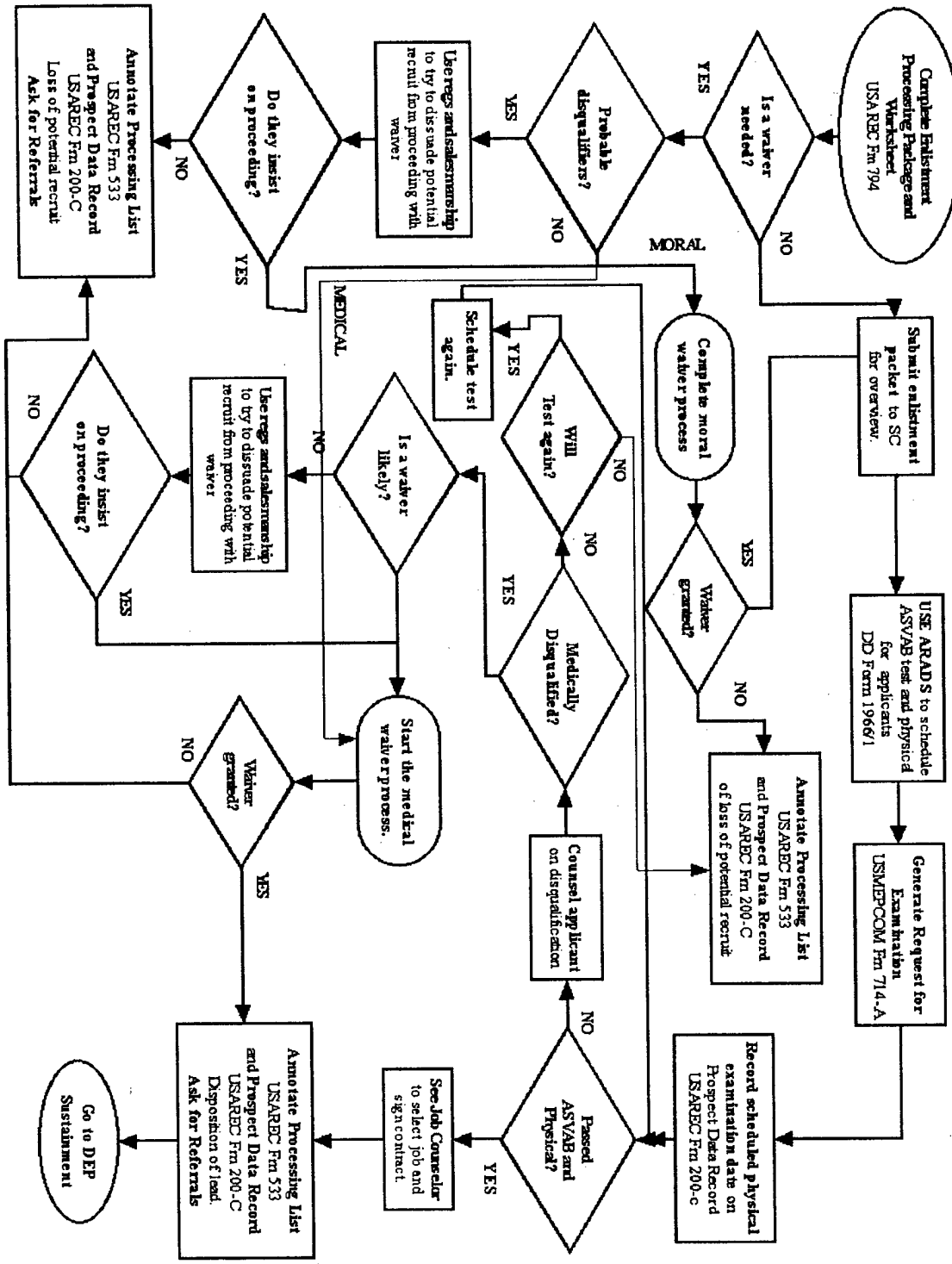

Figure 7. Immediate Processing Flow

2.2.5 Waivers

Often, applicants who are barred from enlisting due to one or more disqualifying physical or moral factors may request special consideration from USAREC. The rationale that forms the basis for the waiver program is that a "by-the-book" review of an applicant's record may prove to be overly narrowsighted. In other words, the ability of an individual to perform his or her duties successfully may not necessarily be compromised by the conditions that originally forced the disqualification. The waiver process itself may take as little time as a few weeks or as much time as several months depending upon the case at hand. This is due to the fact that a large amount of time is spent collecting information for the waiver request, as well as the subsequent time needed for the application to circulate among the approval authorities at the battalion, brigade, or even headquarters level. Therefore, a sizable amount of a recruiter's time may be spent preparing the waiver application, itself a very lengthy compilation of required forms, which in turn often demand obtaining further documents, character references, and the like. Some recruiters have dealt with this issue by placing the bulk of these responsibilities upon the applicant; nevertheless a recruiter is obligated (in the case of a moral waiver) to honor an applicant's request for consideration, and is thus ultimately responsible for the waiver application from its inception to the conclusion. The following paragraphs provide more specific information about the various forms and events which comprise the waiver process.

2.2.5.1 The Medical Waiver

Applicants who fail to qualify for enlistment based on a physical condition may be eligible for consideration to receive a medical waiver. The process may take up to sixty days for non-prior applicants and ninety days for priors. Each waiver request must be supplemented by an abundance of documentary evidence such as medical examination results and medical records if the applicant has ever been a member of a reserve or active duty component. A number of conditions definitely

preclude the granting of a waiver, and indeed usually prevent consideration of the applicant. Some examples are artificial joints and plates, schizophrenia, heart ailments, and pregnancy. An overweight body builder is a common example of a waiverable medical condition which results from the non-applicability of general weight standards to a specific group of people.

Preparing the medical waiver package constitutes the bulk of the recruiter's role in the waiver process. An effective way for a recruiter to manage his or her time is to delegate a portion of this task to the applicant insofar as obtaining information is concerned. A cover letter, of course, must accompany the various forms, the most important of which are described in more detail as follows (see Figure 8).

Medical Waiver Package

- *Report of Medical Examination* (SF-88): Completed by a physician at MEPS. The form itself must contain a recommendation by the examining physician that the applicant be considered for a waiver.

- *Report of Medical History* (SF-93): Also completed by a MEPS physician. Supplement to actual medical records and other supporting documentation.

- *Other Administrative Documents*: Includes Discharge papers (USMEPCOM 714ADP), National Agency Questionnaire (NAQ - DD Form 398-2), Record of Military Processing - United States Armed Forces (DD Form 1966), and Certificate of Release from Active Duty.

Path of Medical Waiver Package

In the case of a non-prior applicant, the recruiter submits the completed package to the Battalion (after possible review by the Station Commander). After waiver experts have evaluated the package for completeness, the package is forwarded to Recruiting Command headquarters (HQ USAREC). The approval authority in this case is the Command Surgeon. For prior applicants, the application must first be routed through the Personnel Command (PERSCOM), which also possesses the approval authorization. The application is then sent to the Command Surgeon at USAREC for a mod-

ification of the medical profile. This added layer of reviewing officials adds an official maximum time of 30 days to the 45 to 60 days for a non-prior waiver package.

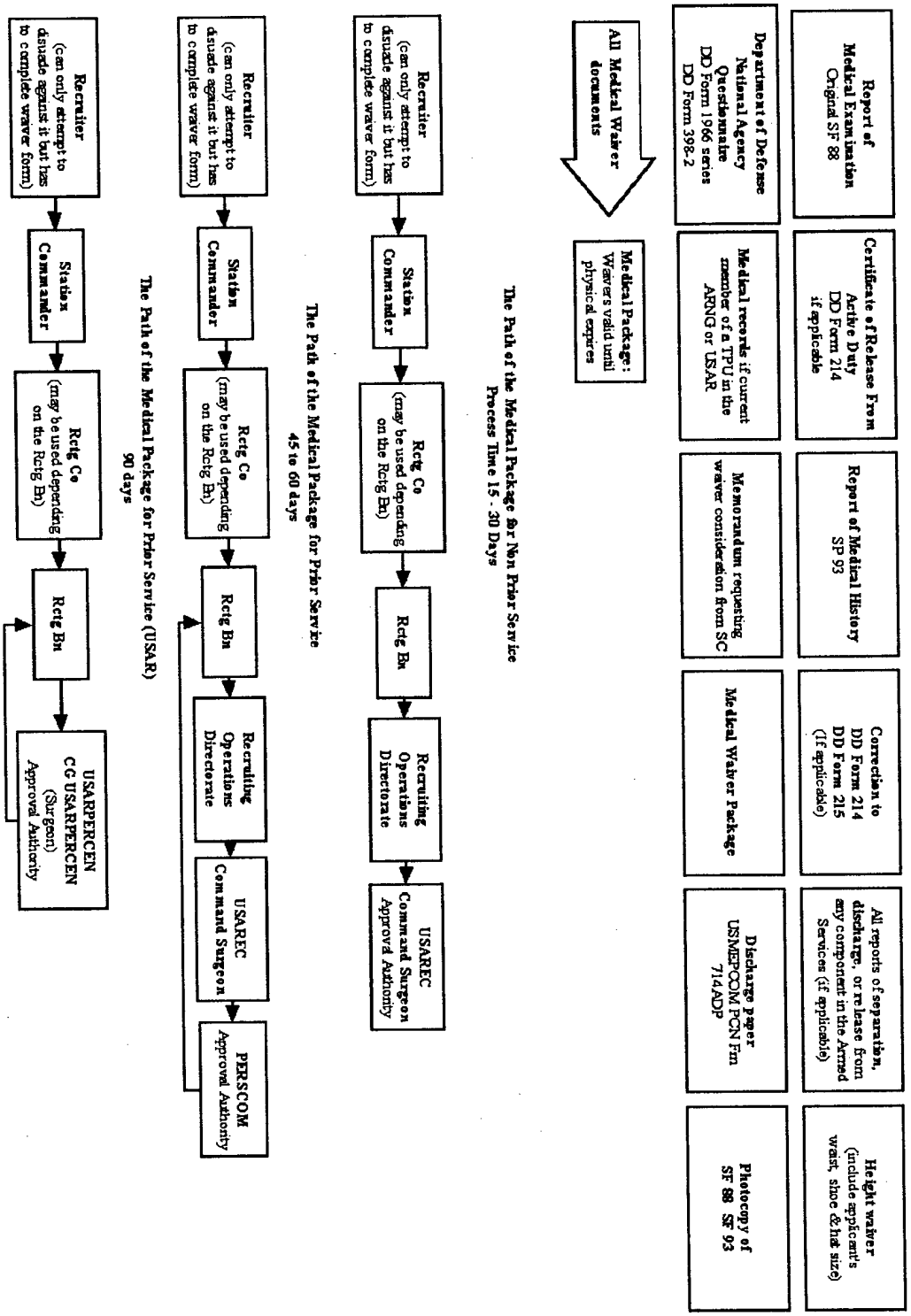

Figure 8. Medical Waiver Process Flow

Preaccession Drug and Alcohol Waivers

Each applicant is tested at MEPS for alcohol and drug use at the initial physical exam and (if placed in the Delayed Entry program) before shipping out to basic training. Applicants who test positive for marijuana (THC) may be retested within six months. If tested positive for other drugs, this waiting period becomes one year. These applicants may be considered for a waiver if the retest results in a negative finding; otherwise, a two-year waiting period is mandated according to the USAREC Regulation 601-56. The approval authority rests with the battalion commander, and waiver packet materials are similar to those required for the medical waiver.

2.2.5.2 The Moral Waiver

The "Whole Person Concept" is intended to screen out those individuals prone to committing repeat and/or serious offenses based upon past conduct. This, in turn, serves the dual purpose of ensuring that as few resources as possible need to be diverted from the military mission in order to discipline individuals and, in addition, to "... assure soldiers and recruits that they will not be thrown into close association with ... chronic offenders or (those) who have committed serious offenses." There are, nevertheless, instances in which applicants have either been rehabilitated or else whose offenses are neither serious nor chronic. The moral waiver process, though useful as a way to project flexibility into the recruiting process, nevertheless presents a considerable challenge to the often time-starved recruiter. Every document relating to the disposition of the applicant's case in either the civil or military courts must be collected. In addition, any supporting documentary evidence - such as character references - must also be included. It is interesting to note that any applicant is entitled to request consideration for a waiver (if so motivated). Though time consuming, completing a waiver packet is preferable to uncovering a disqualification later in the recruiting process after a considerable amount of effort has already been invested. See Figure 9 for more details.

The Moral Waiver Packet

- *Moral Waiver Worksheet and Continuation (* USAREC Form 670/670A): Background sheet containing a list of all offenses and checklist needed to determine eligibility for a waiver. Concludes with a recommendation section that is completed by a waiver expert.

- *Police Record Check (* DD Form 369): Accomplished by the recruiter for every government level (city, county, and state).

- *Records of Civil Proceedings:* Includes Reports From Probation/Parole Officer/Court Records Check (USAREC Form 1037), Copy of Court Document, and Correctional Facility Officer's Report (USAREC FL 41).

- *References* (DD Form 370): Obtained from both school and work. Used to help establish a character profile which is crucial to a well-informed decision about whether or not to grant a waiver.

- *Administrative Forms*: Includes National Agency Questionnaire (DD Form 398-2) for background checks / security clearance, Report of Medical Examination/History (SF 88/93) for substance abuse waivers, Discharge papers (USMEPCOM 714ADP), and Record of Military Processing - United States Armed Forces (DD Form 1966).

Moral Waiver Routing

Waivers for minor offenses such as traffic/non-traffic violations, misdeameanors, and pre-accession drug and alcohol detection can be approved at the battalion level. For felonies, DUIs, and minor offenses judged slightly more serious in point value (reference the point allocation system), the approval authority is located at HQ USAREC, after a preliminary review at the battalion level. Processing time for the first class ranges from 5-10 days whereas the other group of waivers may take up to a month to review. One particular factor that may influence the time to review moral waivers is when a recruiting impropriety (RI) has been alleged. A commissioned officer must then be summoned to perform an interview (ostensibly with the recruiter) to determine if such is the case. The moral waiver application may be delayed indefinitely pending the outcome of the investigation and dispensation of penalties (see USAREC Reglation 601-45).

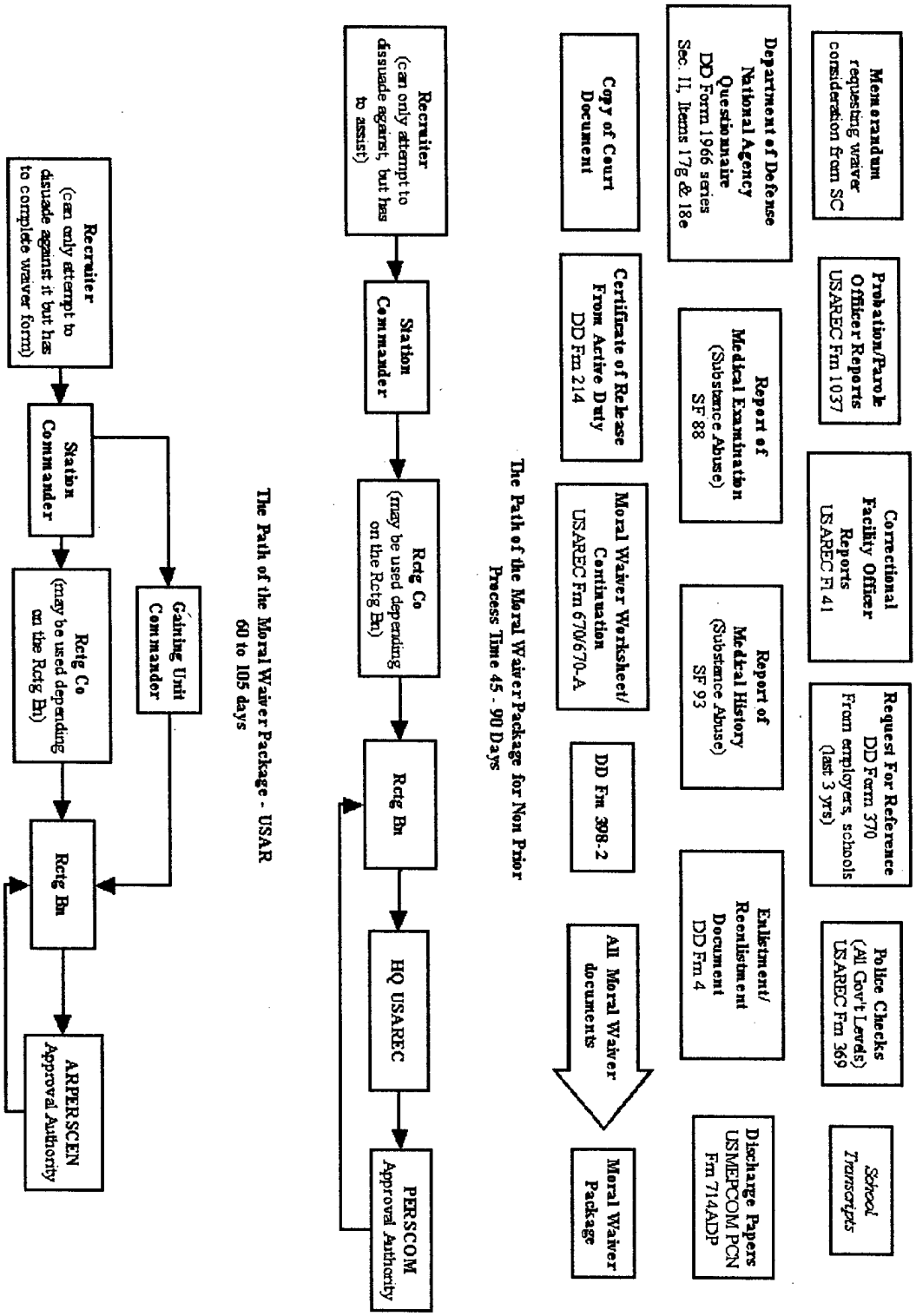

Figure 9. Moral Waiver Process Flow

2.2.6 Delayed Entry Program Sustainment

This final stage of the recruiting process is an extremely crucial one for its purported goal of retaining a hard-won recruit through the weeks and months of the Delayed Entry Program. No matter how diligent or clever a recruiter has been in performing the previous critical tasks, he or she could conceivably undo these efforts by neglecting the necessary follow-through processes embodied in DEP Sustainment. Recruits often face intervals of up to twelve months waiting for either basic training or a specific job (MOS) to become available. Although this practice assists both the Army and the recruit by providing the desired job in such a way that the Army can meet its mission requirements, there is an obvious disadvantage in that the recruit is allowed to terminate the enlistment proceedings at any time (before the final ship date). Also, circumstances such as a change in the eligibility status of a recruit become more likely the longer the delay imposed upon a recruit. According to one study, the risk of a "DEP loss" increases on average by approximately 3.5 percent for each month[8, 261] . Thus the aim of the DEP Sustainment process is to keep an individual motivated to enlist, a task which involves regular and sustained contact between the recruiter and applicant. These meetings may involve anything from a mere status report to involved social functions often occupying the resources of an entire battalion of recruiting stations. One fact, however, is undisupted: an applicant who feels ignored will become a greater DEP loss risk. Hence it is paramount that a recruiter realize this natural relationship and act accordingly to prevent the loss of what could amount to a year's worth of effort. The following task breakdown is mandated by USAREC Regulation 350-6 (see Figure 10).

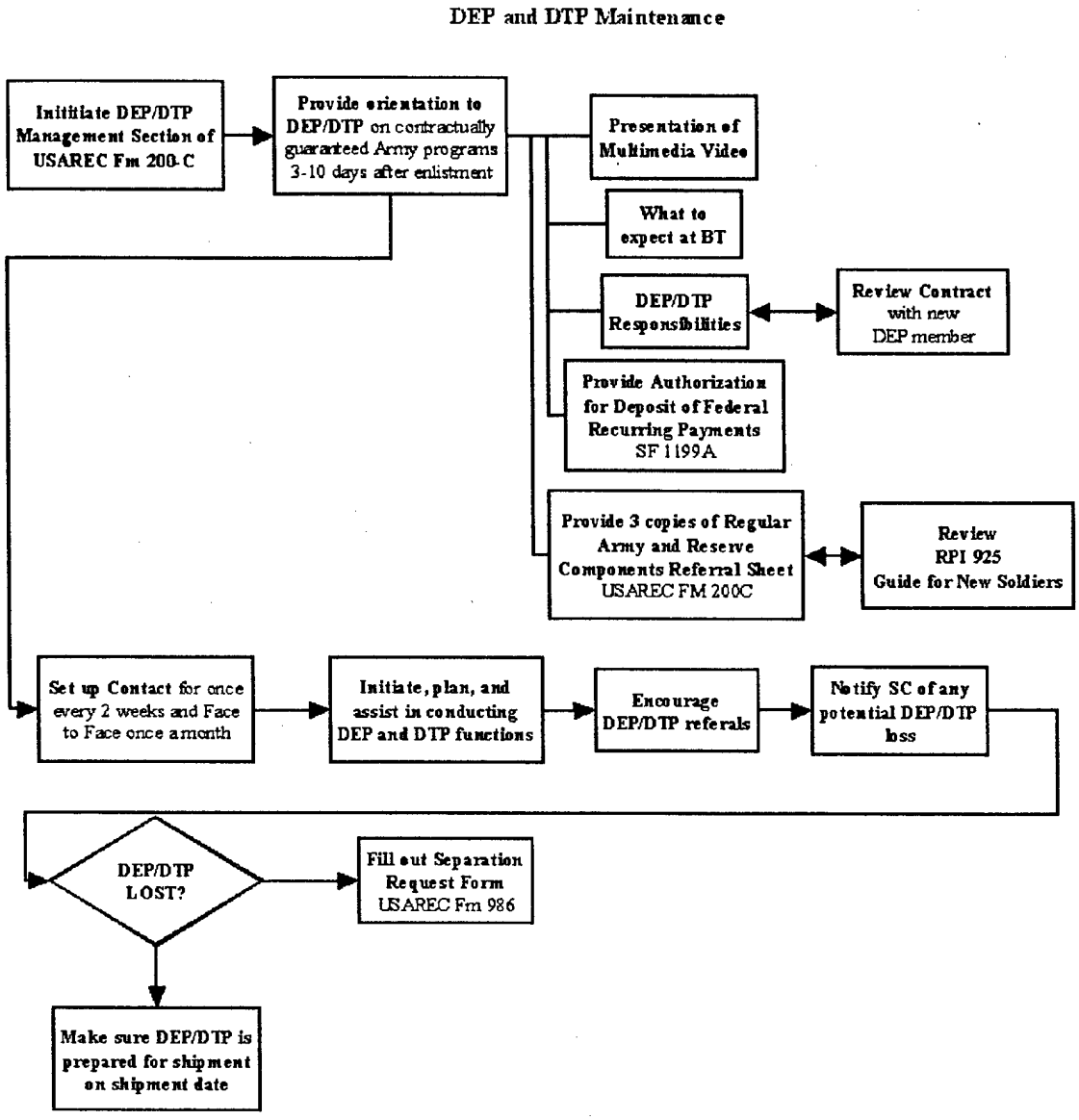

Figure 10. DEP Process Flow

2.2.6.1 Loss Factors

At the DEP stage of the recruiting process, the applicant has completed all or most of the processing and, more importantly, has entered into an enlistment agreement. However, there may yet be a period of a few weeks to several months until the scheduled shipping date. At any time

when the applicant is on delayed entry status, one of two contingency situations may occur. The first danger is that the recruit may change his or her mind and renege on the enlistment contract. What the recruiter must understand is that joining the Army is a major life decision fraught with more uncertanties than what would confront most new civilian employees. The recruiter must thus instill a sense of comraderie in being a member of the U.S. Army. Of course, the best way to do this is to gradually acclimate the applicant to the people and the environment that he or she will deal with once basic training has been completed. Although the recruiter cannot hope to faithfully reproduce this experience in every way, there are still various techniques that one may use in order to maintain the recruit's interest level without an inordinate expense of time and resources. The most important of these is to keep in touch with each DEP recruit. Exceeding (as much as possible) the mandated once-per-two-week telephone contact and once-per-month face contact mandated by the regulations (USAREC Reg 350-6) will reduce the risk of DEP loss greatly. What some recruiters have found helpful are company or battalion-coordinated DEP functions such as picnics in which demonstrations of Army hardware are conducted and combat personnel demonstrate their skills to the recruits.

Another underlying reason for a DEP loss is the change of a recruit's status, medically or otherwise, that would result in a disqualification. Frequent consultation with the recruit would certainly alert a recruiter to potential problems and thus facilitate preventative measures. Prevention, of course, would be the most favored route as the other alternatives would either involve the initiation of a waiver application or, even worse, the loss of a recruit after much time and effort has already been invested. Failing all other efforts to retain the recruit, the recruiter must recognize the warning signs of DEP loss early so that he/she and the station commander can plan a contingency action such as a shift in priorities to other, more promising activities. Certain applicant profiles are associated

to greater DEP loss tendencies (high school seniors, for instance)[8] , and thus offer an excellent predictor as to which applicants one should divert the most resources.

Fear, uncertainty, and disqualification are not the only factors driving DEP loss. Kearl and Nelson have observed that "...time in the DEP permits active search for alternative job opportunities although adding little first-hand information about Army life or training"[8, 255] . This fact highlights the most important duty of the recruiter during DEP sustainment, which is namely the need to both inform and motivate. Sustainment is indeed a continuous sales presentation that builds upon the factors that initially motivated the applicant to commit to the Army. The moral is such that a judicious investment in time to DEP sustainment from the early phase on will obviate the need to beg and plead applicants on the eleventh hour.

2.2.6.2 Length of DEP

The length of an applicant's stay in the DEP varies widely depending upon his or her specific situation. High school juniors who have made a commitment to enlist may be in the DEP for more than a year while some graduates may wait only a few weeks (or even days). The Kearl-Nelson study [8] mentions that, on average, the risk of a DEP loss increases by approximately 3.5% per month. Various recruiters whom we have interviewed during the course of this study have noted that this risk may be alleviated by conscientious DEP sustainment. This consists of the elements noted above, namely frequent meetings and telephonic contacts throughout the course of the applicants hiatus on the DEP.

2.2.6.3 DEP Sustainment Procedures

The first activity of DEP Sustainment is usually the orientation interview, which should occur anywhere from three to ten days after enlistment. Several key activities are performed during this interview. The first is usually a video presentation[6] of various topics relating to Army life and

[6]Formerly presented using the Joint Optical Information Network (JOIN) media, which is a set of LP-sized laserdiscs

occupations. Most importantly, the applicant will be shown what to expect during basic training, including the content and intensity of training activities. The recruiter will then proceed to show video clips relating to the occupations for which the applicant has professed an interest in (if not already accomplished). These activities, though seemingly simple, remove many of the uncertanties that lead to false conclusions and wild imaginings, which in turn increase the risk of DEP or post-enlistment loss. This interview constitutes the first of what should be a series of DEP contacts; subsequent contacts need to be planned and agreed upon with the consent of the applicant, and their frequency should exceed the minimum required by the recruiting regulations (consult the preceeding paragraph). A recruiter should keep in mind, however, that as more applicants survive to the DEP stage, more time is required to perform DEP Sustainment activities. One way to alleviate this problem (as mentioned before) is to plan station or battalion DEP functions regularly or, if this is not possible, to schedule as many recruits at once to attend face-to-face meetings.

Another item of business that needs to be accomplished during the orientation meeting is a discussion of recruit responsibilities while on the DEP. The enlistment contract should be explained and key requirements emphasized in such a way as to establish the recruit's new status as a full-fledged member of the U.S. Army. For instance, the recruiter might phrase the requirement that the applicant appear at the recruiting station at regular intervals as a direct order by his or her commanding officer to report. From this standpoint, the recruit is compelled to view him or herself as a member of a team and to behave accordingly, instead of acting only upon his or her individual desires (which may very well be to not appear until the shipping date). It is important to set this tone at the initial orientation interview because of the important influence of *attitude reinforcement* upon the succeeding term of the DEP. In other words, a lackadaisical attitude on the part of the recruiter will, in turn, translate to a correspondingly fickle attitude on the part of the recruit.

---

containing a variety of video topics. A new video standard is being developed at the time of this printing.

Despite all of the good intentions and self-sacrifice inflicted upon himself / herself by the recruiter, some DEP loss will inevitably occur. The loss itself need not be a total disruption and waste of time if the signs are observed early enough in the process to plan a contingency action. It is important that the recruiter maintain unceasing contact with the station commander on the status of each DEP recruit so that another set of eyes (perhaps more experienced) may pick up these early-warning indicators. The most obvious of these is failure to attend the regularly scheduled DEP meetings or answer the telephone calls of the recruiter. Others include failing grades in school or changes in the physical state of the recruit (weight, injuries, etc.). Regardless of the indicator, it is imperative that the recruiter recognize these problems and either attempt to fix these or, if intractable, plan ways in which to recoup the loss that this recruit poses to the individual and overall mission. As stated before, a certain percentage of losses can be expected, and these may vary depending on the effectiveness of the recruiter. Therefore, the recruiter cannot be consumed with achieving perfection, but rather the reduction of losses. In this way, a proper distribution of time between DEP Sustainment and other activities can be planned so as to prevent an overemphasis on DEP activities.

2.2.7 Overall Value of the Study Methods

We often refer in this chapter to the station visits and regulations referred to in support of the process study. Indeed they have proved themselves to be indispensable in terms of the insight that was gained into the recruiting system. During the initial phase of the Recruiting Study, reading the applicable USAREC regulations were a convenient way in which to quickly attain a basic-level knowledge of mandated recruiting procedures. This allowed the authors to speak to recruiters on a more even footing during subsequent data-gathering trips, when more specific issues such as task time durations needed to be addressed. A comparison could then be made between the USAREC-

documented process and actual practice. More importantly, information from the regulations formed a basis for the subsequent detailed and model process flows.

It is misleading, if not hazardous, to rely solely upon the regulations in order to formulate a description of the recruiting process. Recruiters themselves first learn about recruiting from attending training courses such as those offered at the Recruiting and Retention School at Fort Jackson, SC. The academic instruction was amply supplemented by practical and anecdotal information shared by instructors who are fully versed in the processes and pitfalls of the recruiting job. In other words, the training allowed the authors to bridge the gap between simple textbook learning and a practical insight into the actual system. Attending the classes also helped to legitimize the study by providing the authors with a background similar to that of the average station recruiter. Although one cannot become a recruiter through training alone, one may at least communicate with actual recruiters from the viewpoint of shared experiences and knowledge.

Station visits and interviews fulfilled the need to learn the process from system experts - the recruiters - who inevitably are a wealth of information that cannot be gleaned from other sources. For instance, procedures, task durations, and task priorities vary from station to station, and even from recruiter to recruiter. Such data must therefore be compiled and subsequently analyzed in order to characterize such parameters in general terms. This in turn allows the fulfillment of several critical requirements of the modeling process, which are (1) a thorough study of the system, which includes the development and validation of the detailed process flows and (2) the development and validation of the Recruiting Process Model. In addition, the anecdotal data resulting from many lengthy conversations with recruiters provided background for and insight into the issues that surround the quantitative descriptions. These varied and sundry elements subsequently combine into a cohesive picture of the Recruiting Process from which we could extract a workable computerized model. This next step is the subject of Chapters 3 and 4.

Chapter 3 - Methodology

It is said that every person has unique personality traits and abilities. During our research in the recruiting process it was brought to our attention that this uniqueness has an impact on the duty performance of Army recruiters that far exceeds that of many other occupations. We were faced with the challenge of trying to incorporate this human element into our model. In terms of the average recruiting station, the qualities of the station commander plays a large role in the determination of the overall efficiency of a recruiting station. The station commander effect is felt in every stage in the recruiting process due to his or her involvement in most aspects of station functioning. Another major effect is the local demographics of the area in which the recruiting station was located. Certain areas of the country are more supportive of the idea of military service than others, a fact which enables recruiters located in these regions to be more successful with less effort.

Many leaders in the field of simulation have noted that simulation is suited more for the comparison of alternatives than for providing definitive answers [10] . While one may be inclined to consider a simulation to be a black box that provides a predictable response to any given input, what one must understand is that the output of a stochastic simulation is a collection of random variables. It is also their judgement that, in order to compare alternatives, a simulation does not need to and, in most cases, cannot model reality exactly. A good simulation encompasses only the level of detail necessary to be able to provide answers about the process in question. We generated the simulation of Army recruiting by keeping these well-established guidelines in mind.

It would be possible to spend significant time and resources attempting to quantify the effects that different station commander abilities and demographics have on the efficacy of recruiter efforts. While a study of this relationship would be an interesting and useful tool in understanding the determinants of efficiency of station output, we have based our model solely on recruiter performance

without taking into consideration such interactive effects. The model user must therefore take this into consideration when analyzing the simulation output.

In order to provide a versatile and robust model, the user of the program will be allowed to modify the number of recruiters and various recruiter and station attributes. As a group, these model parameters represent the overall efficiency of recruiters as well as the length of time delays inherent within the recruiting process. Although such factors as station commander influence and area demographics do not appear explicitly within the design of the model, these attributes may implicitly contain such information since they are based on actual data collected from several recruiting stations. Thus, a study of these implicit factors can be accomplished via an experimental design in which the simulation output generated from station data characterized according to, say, demographics are compared.

Figures 11 and 12 show the model-paradigm that we distilled from the detailed process flows that were the product of our research into the recruiting process. Notice that many of the tasks are grouped, or aggregated, together in order to avoid including excessive detail in the design of the working simulation model. The diagram was invaluable to the authors in designing and implementing the Army Recruiting simulation, the details of which we shall forego until Chapter 4, Model Design.

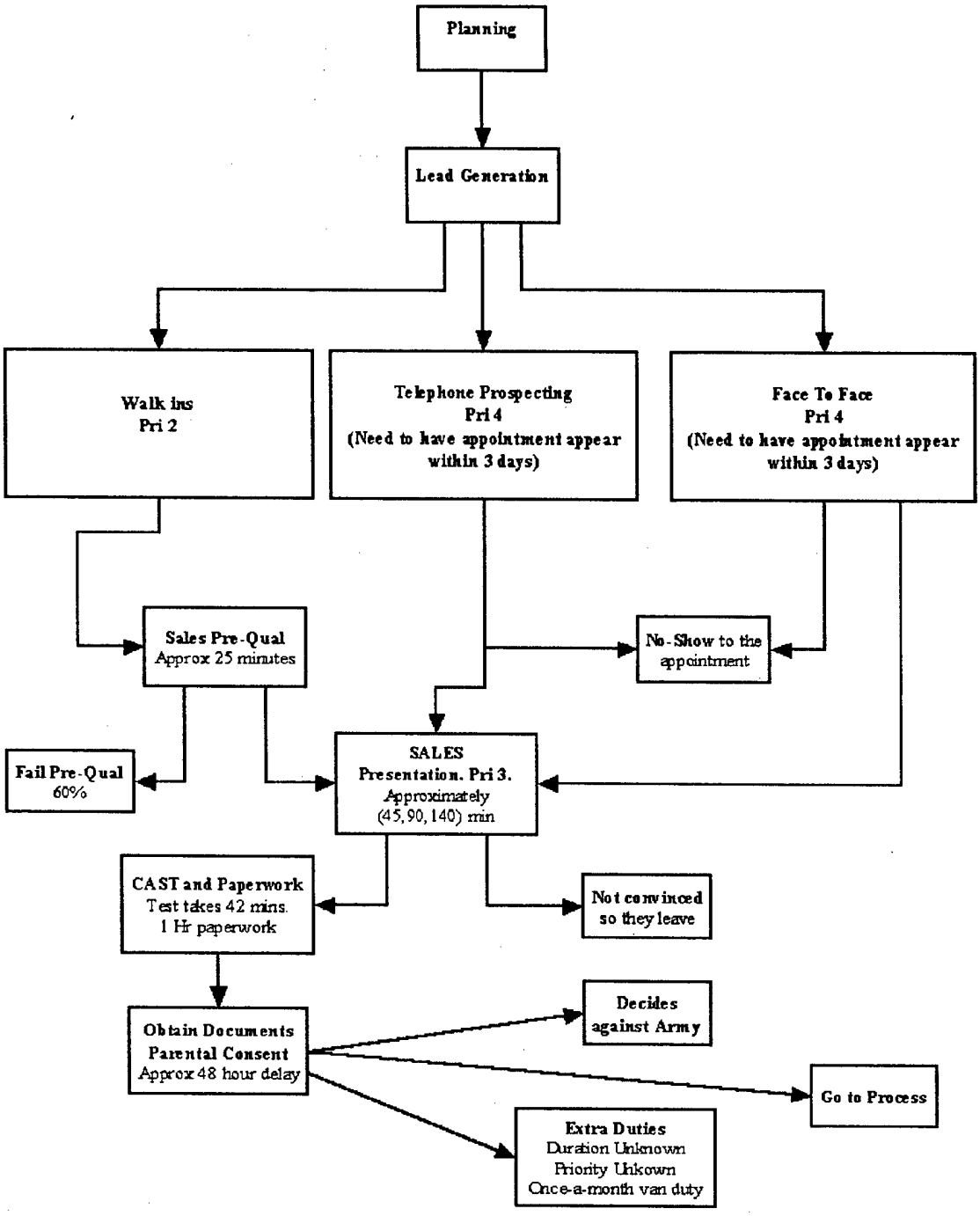

Figure 11. Model Flow Diagram (Part I)

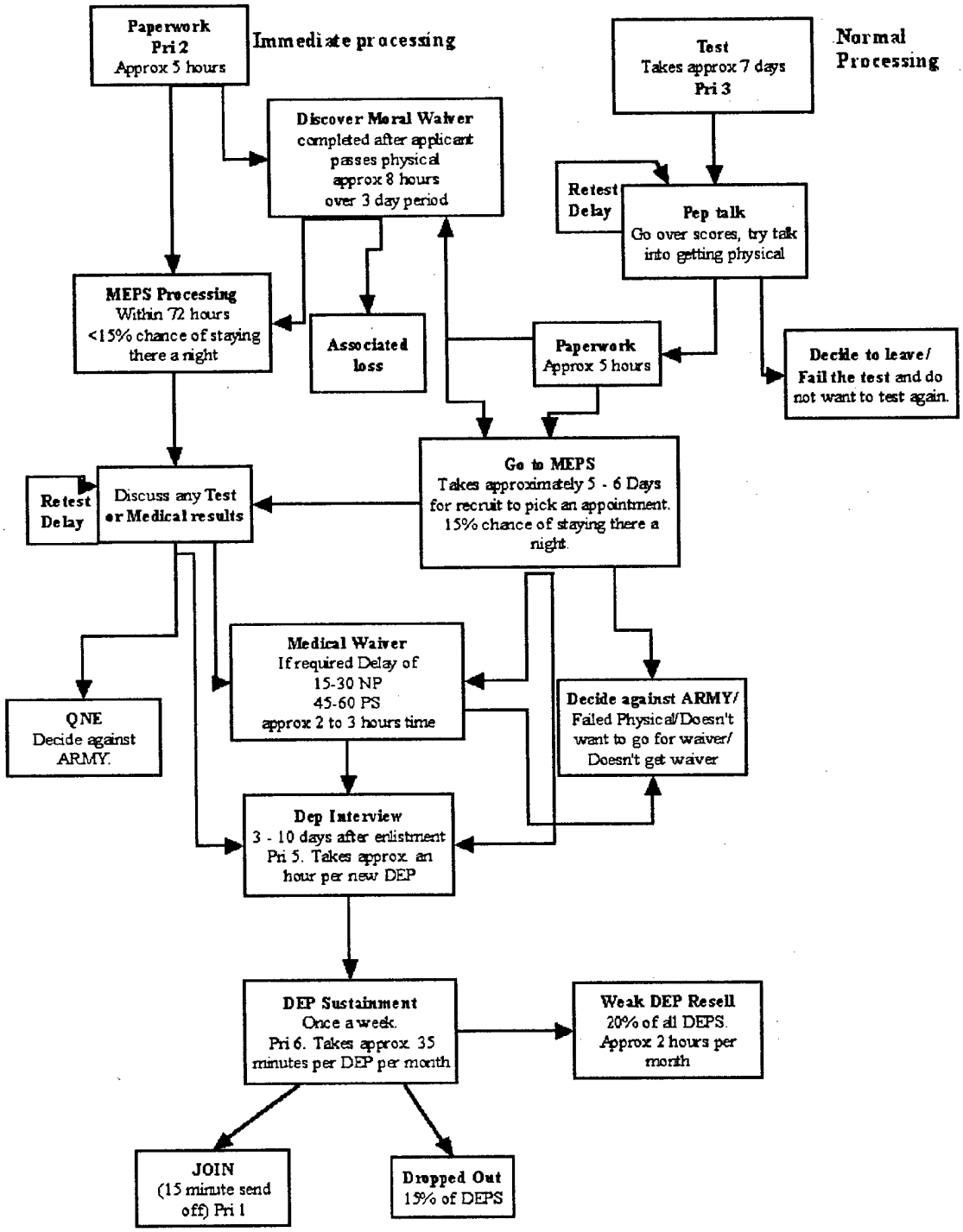

Figure 12. Model Flow Diagram (Part II)

3.1 Model Concepts

Having completed the detailed study of the actual system, it is now possible to design the computer simulation model itself. There are two factors that one must take into consideration during model formulation. The first deals with the model as an abstraction rather than as an exact reflection of reality. The abstractness is a manifestation of the use of mathematical constructs in representing real-world processes. For instance, uncertainty is modeled using probability and statistics, and through the use of algorithms to generate sequences of random numbers. Another consideration is the complexity of the process being modeled. It was due to this complexity that a simulation method was chosen over an analytical one in the study of the recruiting process. It is certainly possible that an overzealous attention to detail can overwhelm the analyst if certain simplifying assumptions to the original process are not made implicit within the design of the model. Without these, the law of diminishing returns dictates an increase in complexity that exceeds any resulting gain in accuracy. Thus, it is important to decide, not only what methods to use in simulating a process, but also to what extent one should remain faithful to the details of the process being modeled.

The use of mathematical/analytical concepts allows one to represent an existing or preconceived system in terms of quantitative relationships. Although purely analytical solutions exist for certain problems (and are preferred if they exist), the extreme complexity of the recruiting process rules out that approach. Nevertheless, the appropriate application of such concepts as queuing systems and input distributions allows the construction of a computer model based on these concepts. Given that the model accurately represents the system in question, various input parameters in the computer simulation can be modified, and the results over the change in parameters can be compared [10, 115] . The mathematical models which form the basis of the computer-driven simulation allows one to predict the behavior of the recruiting process without resorting to the observation of actual recruiting stations.

The use of simplifying assumptions can be justified by the fact that the gain in model fidelity is far outweighed by the resultant growth in complexity when too much detail is added. This complexity results in both the lack of understandability of the model and a corresponding increase in the time required to run the simulation on a computer. According to Law and Kelton, one should include only enough detail to "capture the essence" of a system [10, 107] . Even more crucial is the existence of time and cost constraints which limit the resources available for the simulation study. It is therefore up to the modeler to determine whether the gain in accuracy warrants inclusion of additional detail. To build the actual computer simulation, it was necessary to refine the description of the system presented in Chapter 2 by incorporating process simplifications and analytical concepts to form the basis of a workable model. The results of this "weeding-out" process are given in the following discourse, in which the model process flow is described together with the basic assumptions that form the basis of the flow.

3.1.1 Analytical Methods

Since much of the recruiting process is influenced by random elements (beyond recruiter control), it was necessary to utilize statistical methods in simulating various aspects of the recruiting process. Some examples of random occurrences are: interarrival distribution of walk-in applicants, probability of applicant loss at various stages, probability of passing the AFQT, and frequency of task performance (collateral duties, appointments, etc.). We begin with a discussion of random number generation and then continue with the probability distributions used in modeling these events.

3.1.1.1 Random Number Generation

The generation of random numbers is a fundamental activity of any simulation, as they determine when and how long an event occurs, as well as its outcome. Random numbers may be fundamentally characterized as random variates drawn from the uniform distribution bounded by 0 and 1,

which is denoted by $Unif(0,1)$. There is an equal probability of coming up with any number between these two bounds, hence the name uniform. Uniform random numbers $U_i \sim Unif(0,1)$ can be generated in various ways, the most common being the linear congruential method. We define $U_i = \frac{Z_i}{m}$, where

$$Z_i = (aZ_{i-1} + c) \bmod m, \;\; i = 0, 1, 2, ... \tag{3.1}$$

and a, c, and m are integers. Given an initial seed Z_0, the equation above iteratively produces integers with a period $\leq m$. Full period can be achieved by choosing a, c, and m appropriately (see Law and Kelton[10] for a complete description of these conditions). An abundance of other generators are also in use such as multiplicative LCGs (with the c term omitted) and hybrid methods employing two or more simpler generators. The decision to use one generator over another is usually determined by the ease of implementation of the generator, in addition to the speed of execution and its period (which may need to be very large in order to produce a sufficient number of independent replications of a simulation).

Simulated events may be prompted to occur (or not occur, as the case may be) by drawing $U \sim Unif(0,1)$. If, say, an event can be characterized as occuring $\alpha * (100)$ – percent of the time, we can simulate an occurence of this event if $U < \alpha$. This was used a number of times at decision points in the model to determine, for instance, whether or not an applicant entity would exit the system. Equally useful is the ability to draw random variates according to a certain probability distribution (different from the uniform distribution). There are a number of techniques used to do this, including the Inverse Transform method (which utilizes the inverse cumulative distribution function (CDF)) and the Acceptance-Rejection method of von Neumann. We refer the reader to Law and Kelton [10] for a discussion of these topics. Random variates are used within the model mainly for producing random time intervals, such as interarrival times of applicants to the recruiting

station or meeting durations. As such, the production of random variates is essential to model the time spent in various activities during the course of each simulation run.

This discussion of the basic concepts of random number generation is simply intended as background information since the simulation platforms used, namely MODSIM and Simprocess, include built-in random number generators. Both have the capability of producing random numbers from a wide variety of distributions based on an initial seed. The literature for MODSIM claims a period of 2^{31}, which is around two billion - more than sufficient for, say, a thousand replications of the model, given that a sufficient number of streams[7] are employed. Various random number streams are employed within the model for the following purposes:

1. Interarrival times for walk-in applicants.
2. Sales and DEP meeting durations as well as intervals between DEP meetings.
3. Times between prospected applicants.
4. Decision points that determine whether or not an applicant remains in the system.
5. Decision points that determine whether or not a waiver is granted.
6. Time to complete processing paperwork.

3.1.1.2 Input Analysis: The Triangular Distribution

Most of the random variates produced in the MODSIM and SimProcess models are drawn from the Triangular distribution (see Figure 13), which is denoted by $Triang(a, b, c)$. The parameters correspond to the lower bound, mode, and upper bound respectively, thus forming a type of truncated distribution. Its recommended use is for circumstances where little data is available other than rough estimates of the parameters (keeping in mind that the mode may be different from the mean).

[7]The word "stream" denotes a sequence of random variates produced by a generator using one particular seed value.

Perhaps the best argument for its use may be the ease in which random variates may be generated using the Inverse Transform and Composition methods. As seen in Figure 1, the ease of generation is the result of the distribution function being linear, which in turn means that the CDF is derived from the integration of linear functions. As an example, we first note that, given the triangular distribution corresponding to the random variable $X \sim Triang(0, 1, \frac{(c-a)}{(b-a)})$, we can obtain $X' \sim Triang(a, b, c)$ through the following transformation

$$X' = a + (b - a)X$$

[10, 494] . It is therefore sufficient to consider $X \sim Triang(0, 1, c)$, whose CDF and inverse-CDF are given by

$$F(x) = \begin{cases} \int_0^x g_1(t)dt & \text{if } 0 \leq x \leq 1 \\ \int_0^1 \left[g_1(t)dt + \int_1^x g_2(t)dt\right] & \text{if } 1 \leq x \leq c \end{cases} \tag{3.2}$$

$$F^{-1}(u) = \begin{cases} \sqrt{cu} & \text{if } 0 \leq u \leq c \\ 1 - \sqrt{(1-c)(1-u)} & \text{if } c < u \leq 1 \end{cases} \tag{3.3}$$

where $u \in [0, 1]$. Random variates are therefore easily obtained using streams of random numbers (that is, numbers drawn from the $Unif(0, 1)$ distribution). Recent research has indicated that, in addition to the relative ease of obtaining random variates, the triangular distribution exhibits reasonable accuracy in conforming to unimodal data. For these reasons, and because it is easier for recruiters to characterize task completion times in terms of best, worst, and average, we decided to use the distribution to model times.

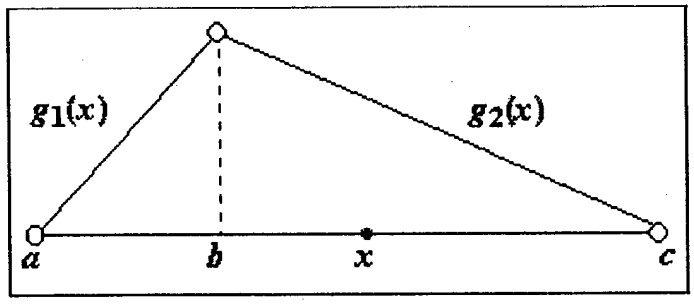

Figure 13. The Triangular Distribution

3.1.1.3 Input Analysis: The Exponential Distribution

The exponential distribution (see Figure 14), denoted by $Exp(\lambda)$, is most commonly used to model interarrival times of customers (or applicants in our case) at the constant rate λ. The density function and CDF are given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geqslant 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geqslant 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

The exponential distribution has a mean of $\frac{1}{\lambda}$ and variance of $(\frac{1}{\lambda})^2$, which implies a larger variance in interarrival times as the mean increases. This is just what we would expect in a real system in which the interarrival times of customers may vary greatly. In certain respects, the recruiting station may be thought of as a $M/G/k$ queuing system with k servers (recruiters) and customers (walk-in applicants) arriving at some rate. Although the service times are scheduled, the walk-in applicants nevertheless enter the system independently of the volition of the recruiters. Prospected applicants, however, arrive at a rate that is dependent upon the skill of the recruiter, and hence may not be modeled in this fashion[8]. Some properties of the distribution impart certain advantages,

[8]These interarrival times are considered to be "applicant generation times" for a particular recruiter and, as such,

chiefly being that interarrival times are allowed to vary widely based on only a single parameter. Another interesting property of the exponential distribution is its memorylessness; that is, $P(X > s + t | X > t) = P(X > s)$ for $X \sim Exp(\lambda)$ and $s, t \geqslant 0$. This implies that the probability of a customer waiting an additional time s after time t units of time have already transpired is the same as it was as when the wait began. These conditions tend to make the random variates mimic the unpredictable behavior of arrivals to a system, which of course makes it the most appropriate distribution to use under these circumstances. The following graph depicts the exponential density function $f(x) = \lambda \exp(-\lambda x)$ for $\lambda = 1$.

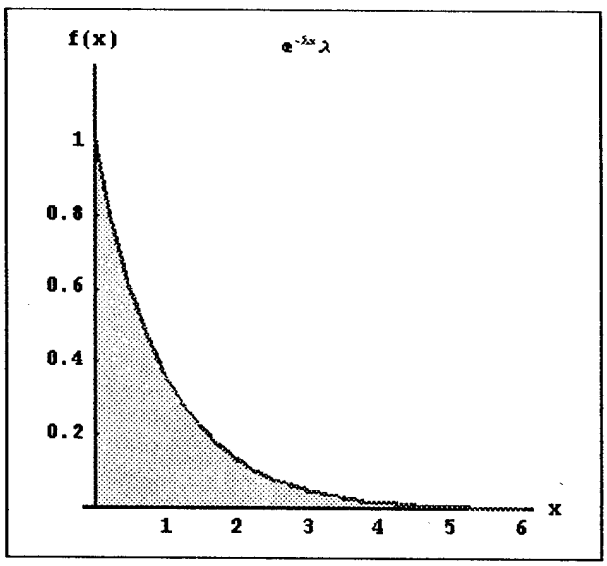

Figure 14. Exponential Density Function with Lambda = 1

3.2 Simulation Tools

may be of shorter duration for more successful recruiters. Consequently, we used the triangular distribution to model this effect.

3.2.1 MODSIM

Most who have used third-generation compiled programming languages would be familiar with the MODSIM environment. The structure of the language resembles that of Pascal, and like its predecessor, MODSIM strives to be as readable as English text. Aside from the usual features sported by general-purpose languages, MODSIM provides powerful simulation-specific routines and mechanisms that free the model-builder from the necessity of building his own library of simulation functions from scratch.

3.2.1.1 MODSIM Program Format

MODSIM is highly modular in the sense that several different blocks of code contained in different physical files must be maintained in order for the program to run. The first block is called the Main Module, which contains the top-level instructions that govern the operation of the simulation, as well as multiple replications of the same. The main module may also call procedures to collect inter-replication statistics. The one limitation imposed on the use of this module is that, although object fields may be referenced from here, no object methods are either declared or defined. In the interests of good top-down design, one should also avoid placing large amounts of code here; rather, related instructions should go into procedures and functions, which are defined in the Implementation Module.

Also mandatory in the MODSIM environment is the Definition Module, which contains function / procedure / method prototypes and state field / variable definitions. This module provides the ideal place to document these various elements because they are not buried within thousands of lines of operational code. Nevertheless, the programmer must take extreme caution to ensure that all declarations here exactly match those in the other modules, or serious run-time anomalies may occur.

The last module, Implementation, contains the overwhelming bulk of the simulation code. For each of the object and procedure definitions, including object methods, the implementation code is given. Thus, the module is really structured in the same way as a C library module and is similarly referenced by calls from other modules, as well as between methods in the same Implementation Module. It may turn out that the module contains such a large amount of code that it would become more feasible to break the module up into several pieces, each containing implementation code for a subset of the defined objects. We found this to be extremely convenient during the building of the model since we felt it adds to ease of understanding of the program flow by providing a logical organization for the implementation code. Thus, the current model contains four different implementation modules: (1) IStatMod, which contains statistics collection procedures, (2) IrstationMod, which contains recruiter and entity object information, (3) ICalendarMod, which contains time-manipulation procedures, and (4) IrankedListMod, which is a short module devoted to the definition of the ranked-list data structure in the Recruiting Model.

3.2.1.2 Simulation Constructs in MODSIM

MODSIM code operates in basically two separate and distinct modes. The first is simulation mode, in which instructions are carried out in artificial "simulation time" - just as if they were events in a normal time continuum. The other mode is simply one in which instructions are carried out independently of time, such as sorting lists or printing statistics to a screen. By writing code that operates on both levels, one can run a simulation while performing various ancillary tasks such as compiling data and initializing input variables. It is this feature of the language that makes it so useful to the model builer. It is certainly possible to build a simulation using a general-purpose language such as C++, and in fact, there are several distinct advantages to doing so. However, in a situation in which time and funds play a major role in defining the study, MODSIM offers the

advantage of possessing the appropriate simulation infrastructure while providing the flexibility and speed of a compiled programming language.

We will only cover those features that are unique to MODSIM to keep the discussion at a manageable length. Of primary interest are the simulation capabilities of the language, the crux of which is time management. As mentioned previously, methods can be invoked in either simulation (in which time passes) or non-simulation mode. Simulation time may elapse only within the context of what are denoted TELL or ASK FOR methods. TELL and ASK FOR methods are groups of instructions that constitute events in the simulation. When a TELL method is invoked, its execution is scheduled on the FEL at a certain point in simulation time. The scheduled time is defaulted to the current time unless otherwise specified by the programmer. ASK FOR methods are similar to TELL methods in every way except that every instruction after the invocation is delayed until after the ASK FOR method has finished executing. In contrast, TELL methods do not delay execution of subsequent instructions, and hence allow for completely asynchronous execution within the simulation.

The usual example of an ASK FOR method is a request for resources, in which the execution of the method stops until a resource is obtained. Thus, this class of method is extremely useful for controlling an execution sequence in simulation time. TELL methods on the other hand are extremely useful for scheduling events asynchronously. For instance, to start a simulation, the programmer can simultaneously set into motion multiple events by invoking TELL methods. The start times may all be instantaneous or else may be set to different future values. Regardless of start times, the execution of one TELL method does not in any way hinder or delay the execution of another.

Another feature that proved to be extremely useful is the ability of one method to interrupt the execution of another. Since the principle of time management is centered around the ability to prioritize and modify schedules on the fly, interrupts are essential elements of a simulation with a

significant human factor component. For the purposes of the Army Recruiting process simulation, whenever an applicant of higher priority demands a recruiter resource, the current task is interrupted and the resource is transferred from the lower-priority applicant. In order to accomplish this, a construct called a ranked list is employed, which we shall discuss in greater detail in the following Methodology section.

3.2.2 SimProcess

SimProcess is a hierarchical event driven simulation language and integrated process tool. It provides the ability to dynamically model a system to show the flow of people, materials, or information as it passes through the system. Models are constructed using processes, activities, entities, resources, connectors, and pads (see Figure 15). Each activity and process in the language is represented graphically allowing the modeler to visually layout the path entities will traverse through the system. This visual approach to modeling significantly reduces the effort required for program verification. The visual approach also allows the modeler to graphically demonstrate program logic without bogging it down in obscure written code. The model created uses the following model constructs:

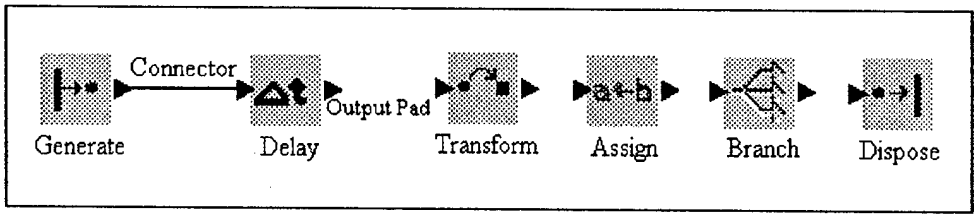

Figure 15. Node Types

Node	Function
Generate	Creates the entities that flow through the system
Connector	Used to connect activities and processes in the model Connectors direct the path of the entities through the system
Delay	Cause time to pass in the system and may require a resource before allowing time to pass
Resource	Information, persons, or materials needed by entities to pass through processes and activities in the system
Pads	Entities flow into and out of the processes and activities through the input pad and output pads
Transform	Changes program entities from one program entity type to another entity type
Assign	Used to assign values to the attributes of entities or to assign a priority to entities
Branch	Sends entities on one of any number of paths based on either a priority, entity type, or entity attribute
Dispose	Disposes the entities which flow into it and frees the memory the entity used

Table 2. SimProcess Node Types

Although the SimProcess simulation package is very powerful, we had to overcome several platform limitations in order to successfully model the activities of a recruiting station. In Chapter 4, we will discuss the work-arounds (reference the loop structure in Figure 16) that we needed to employ in order to incorporate such features as interrupts, which were not yet included in the present version of SimProcess. We shall also discuss the inability of the SimProcess modeler to require a resource to at a generate node. This is readily accomplished in MODSIM, but can only be approximated with SimProcess, although we do not anticipate a great difference between the results of the models due to this inadequacy.

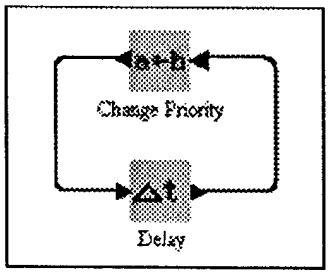

Figure 16. Pseudo-Interrupt

Nevertheless, SimProcess, for all of its design insufficiencies, has proven itself invaluable for short-term model development. In contrast to MODSIM, which required the constant attention of two developers over a period of several months, the SimProcess simulation was built in days (albeit with the experience of building the previous model) by one of the authors. The advanced graphical-user-interface and advanced presentation capabilities also makes it an invaluable tool for presentation to decision-makers who are not well-versed in analysis concepts. In addition, SimProcess includes a number of pre-packaged output and input analysis tools that automate these critical activities. Whereas data from MODSIM must be imported into an Excel spreadsheet and analyzed using author-developed macros, SimProcess results are instantly monitored and output into fairly readable reports that can be exported to presentation software. It is thus an issue of tradeoffs as to whether or not to use either one of these platforms - namely flexibility versus convenience. We will present the results of independent replications of both the SimProcess and MODSIM implementations, thereby creating a basis upon which one can judge their relative merits. This will be discussed later in Chapter 5.

Chapter 4 - Model Design and Construction

4.1 Programming Concepts

The Army Recruiting Model is formulated along the lines of the detailed process flow description given in Chapter 2. However, as mentioned at the beginning of this chapter, many details were not included in the actual computer simulation for the sake of execution speed and code simplification. In addition, since the Recruiting Model was developed using two simulation tools (MODSIM II and Simprocess), different model paradigms had to be developed in order to accommodate the unique features of two types of simulation software. Of the two platforms, MODSIM II is perhaps the most flexible since it allows the programmer to control the utilization of resources and entity throughput more completely than he/she could in Simprocess. However, the striking visual display of Simprocess allows a more intuitive grasp of the simulation by both the analyst and the decision maker alike. What follows is a step-by-step description of the model as it pertains to each of the Five Critical Recruiting Tasks: Planning, Prospecting, Sales, Processing, and DEP Sustainment. We shall also describe in this section (1) the assumptions used in developing the simulation and (2) how the two implementations of the model differ from each other. Model descriptions apply to both the MODSIM and SimProcess simulations unless otherwise indicated. For the interested reader, we have also included code listings (Appendix B) and model diagrams (See Appendix A).

4.1.1 Model Overview

On the most general level, the Army Recruiting Model is an event-driven simulation based on a continuous time-scale. This means that the events, which are appointments or other recruiter tasks, are scheduled on the Future Event List (FEL) at any real-valued time. The simulation itself represents a single recruiting station with a set number of recruiters (a parameter) which are modeled as resources. The applicants are represented as object entities that are instantiated (generated) through

two separate subroutines that model walk-ins to the station and recruiter prospecting. These entities then flow through the system and require recruiter resources to complete their journey through the process. Termination occurs when the FEL is empty (hence the label "event-driven"). We induce termination by artificially halting the flow of entitities (applicants) after a year of simulated time has elapsed. We then collect relevant statistics and run further replications of the simulation as needed. These output statistics will be analyzed using statistical techniques, to include simulation output analysis, multivariate analysis, and response surface methodology in order to determine the sensitivity of model output to various model parameters.

The critical tasks of Prospecting, Sales, Processing, and DEP Sustainment define the structure of the simulation, just as they do for the real system. The absence of Planning from the lineup is the result of a decision to incorporate daily planning meetings as a model parameter. In other words, the number of hours allocated to planning each day is a constant which is input into the model by the user and is then subtracted from the number of hours available to perform the other critical tasks. We did not find any evidence of a large variance in meeting times from our discussions with recruiters, a fact which prompted us to consider modeling these meeting times with random number streams as a non-value-added activity.

4.1.1.1 Object Orientation

No discussion of the model would be complete without mentioning the object-oriented architecture of both MODSIM and Simprocess. A well-known computer scientist, Grady Booch, describes an object as having "... state, behavior, and identity; the structure and behavior of similar objects are defined in their common class"[4, 83] . In less abstract terms, state refers to the values of the fields within the object, behavior to the operations/methods associated to the object, and identity the unique instance of a particular object. An object definition can be thought of in terms

of what is called a *class* representing all similar objects. For instance, one can consider the blanket term "automobile" to be that class of objects defined by certain characteristics such as color, make, and model. The class "complex number" represents all numbers with a real and imaginary part; in a computer programming context, the object is a collection of these real and imaginary fields, together with the methods "Add", "Subtract", and other binary operations. The identity of the object is manifested in its particular location in memory, with its state reflected by the value of its fields.

One can characterize the classes "Station", "Recruiter" and "Applicant" in a MODSIM or Simprocess context by defining them as objects. Consider first the object representation of an applicant entity. In order to construct the object, one must consider the traits that are pertinent to the recruiting process, such as the stage of the process the applicant is involved with, whether the person is a graduate, high school student, or GED equivalent, and what priority the applicant holds for a particular recruiter. All of the methods inside the applicant object simply initialize or alter the state (fields) of the object based upon conditions at a particular point in (simulated) time, such as whether or not the applicant has passed the ASVAB. In an entity-driven simulation such as the SimProcess model, the flow of these applicants through the process determines drives the various processes, as contrasted to its event-driven MODSIM counterpart (which, as you may recall, runs as long as the FEL is not empty). However, applicant entities by themselves do not determine their path through the simulation. Rather, their interaction with recruiters set task duration times and prompt decisions to be made during the course of the simulation run.

Again, a difference in simulation tools dictates the nature of the recruiter representation within the model. In the MODSIM version, the recruiter is implemented as a formal object with state variables and methods. In addition to containing parameters for triangular and exponential distributions for various task durations, it contains the recruiter resource, itself an embedded native MODSIM object. It is this recruiter resource that provides a common link to the Simprocess implementation

and is crucial to modeling recruiter time utilization. The recruiter resource is required in order to perform any of the recruiting tasks relating to a single applicant entity. Any applicant going through the system must wait for this recruiter resource to become available (if it is of lower priority) or else take away the resource from another applicant by means of an interrupt mechanism that we shall discuss in detail in the next section. The methods associated with the recruiter object (in the MODSIM implementation) are among the most important in the model, since they are the critical recruiting tasks of Prospecting, Sales, Processing, and DEP Sustainment. Recalling the definition of the behavior of an object as the nature of its interactions with other objects, these methods do indeed embody the recruiter-applicant relationship within the Army Recruiting Model.

The highest level object is the recruiting station entity (see Figure 17), which contains a selected number of the recruiter objects / resources. While the station is simply implied as the top level of the Simprocess model, it is, in contrast, formally defined in the MODSIM implementation. The state fields are comprised of the actual recruiters, the station schedule and workweek, and assorted statistical counters. It is important to note that walk-in applicants are actually generated by means of a station object method, a fact that actually makes sense in real-world terms. In a real recruiting station, walk-in applicants do not necessarily know beforehand which recruiter will handle their case. So, in a sense, the station itself "creates" these applicants and assigns them to recruiters based on the priority of their current task (if any). The station functions as the linchpin of the entire model wherein all events of the simulation occur. Moreover, it remains instantiated (allocated and accessible to the programmer within computer memory) from replication to replication because it contains the statistical counters for the entire model - as opposed to the more transitory status of recruiter and applicant objects.

It is worth noting here the mechanism by which MODSIM object fields are initialized when these objects are created. MODSIM allows the definition of a special method denoted as ObjInit

which is run automatically each time a particular object instance is instantiated. Therefore, the programmer may insert variable initialization comments, calls to other methods, or any other action that must be performed immediately after object instantiation. This plays a crucial role in setting the state fields and statistical counters for applicant objects (applicantObj), recruiter objects (recruiterObj), and statistics objects (statObj). The ObjInit method could not, however, be utilized for the station object (stationObj) because its fields take on values from other objects that are not instantiated when the station object is first created.

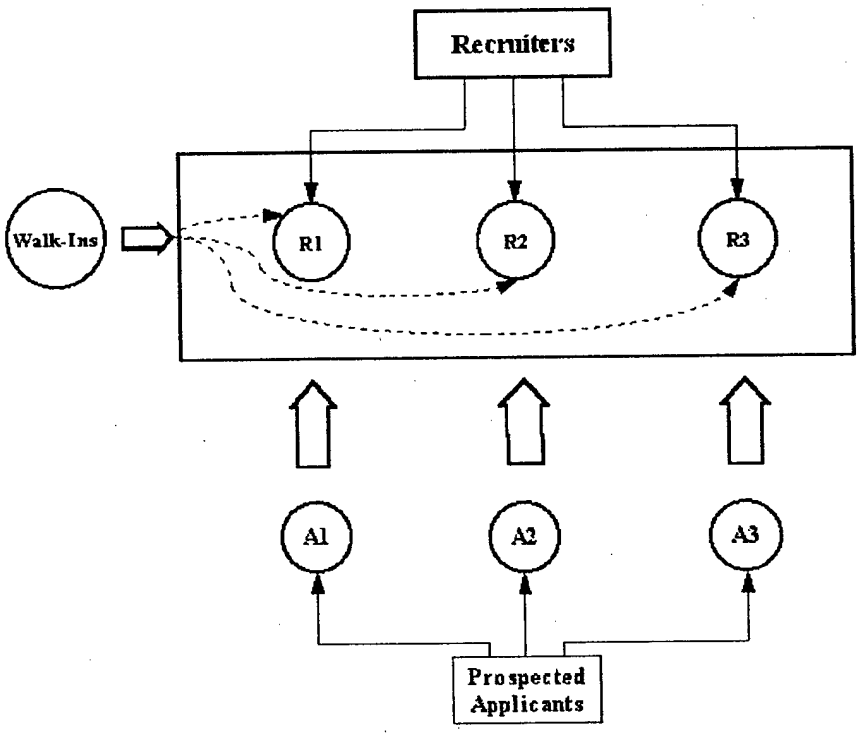

Figure 17. Station Model

4.1.1.2 Ranked Lists and Interrupts

An essential ingredient of the object-oriented MODSIM implementation of the model is the concept of a ranked list. The ranked list is essentially a data structure that contains references to

all entities (in our case, applicants) who are utilizing the services of a resource (recruiter). To each element on the list is associated a priority number, or rank, as the name implies. This definition suggests that a unique ranked list be associated with each recruiter object, a fact which is accomplished through inheritance of the ranked list class by the recruiter class. Since a recruiter cannot perform more than one task at the same time, the ranked list contains only one applicant at a time. The importance of the ranked list relies on the fact that it facilitates interrupts by processes that have higher priority. Any applicant that requires a recruiter resource must first check if the list is empty. If the list indeed contains no other applicant objects, then the requestor may simply grab the resource. If, on the other hand, it contains an element, and that element is of lower priority, then the requestor may go on to interrupt the other applicant and its associated process and put itself on the ranked list. The interrupted applicant must, of course, be re-scheduled for the same process and subsequently await its chance to grab the recruiter resource based on its priority. Interrupting processes is the key to modeling the uncertainty and choices that confront the recruiter struggling to conform to a planned schedule. If no provision were made to include interrupts, processes would run to completion regardless of other events occurring within the simulation. This type of model would exhibit such anomalous behavior as unrealistically long queues, and hence an upward bias towards losses. Interrupts are initiated by higher-priority applicants through means of a call to method Grabrecruiter (See Appendix A).

Interrupts play a key role in the definition of working days in the MODSIM model. In the real system, recruiters will most likely tend to quit whatever task they are working on at the end of the day. Certainly, a recruiter might stay longer if a contingency situation should arise. However, no recruiter can work 24 hours per day, and so the simulation model should contain analogous nightly downtimes. These downtimes are implemented in the procedure "Newday", which interrupts all activities currently in progress (found by accessing the contents of each recruiter's ranked list) after a

delay equal to the number of hours in a work day (which is a model parameter). Applicant generation is then manually restarted at the beginning of the succeeding day, and all other interrupted process are rescheduled and subsequently forced to compete for recruiter resources on a priority basis.

4.1.1.3 How Statistics are Collected

One goal of running a succession of simulation trials is to collect statistics for the purpose of sensitivity analysis. MODSIM provides monitors that automatically collect a variety of statistics over multiple replications of the simulation (IStatObj for integer variables and RStatObj for real variables). One needs to define a counter to be a monitored object, associate it to a monitor variable and then call methods associated to the monitor variable for statistical output. For example, if numTheses is an INTEGER variable for which we wish to collect statistics, we would perform the following steps in the variable declaration section of the relevant method:

```
VAR numTheses      : LMONITORED INTEGER BY IStatObj;
    numThesesStat  : IStatObj;
```

An LMONITORED variable is monitored for changes each time the variable's value is modified while an RMONITORED variable is monitored for changes each time the variable's value is accessed[9]. After the sequence of replications is complete, one can access the stored statistical data by calling the IStatObj methods Mean, Variance, GetHistogram, etc.

Since statistics on production, service times, and the like needed to be collected for each recruiter as well as the entire station, the aforementioned statistical variables needed to be included within the state fields for each recruiter object. Moreover, a very large number of monitored counters needed to be defined. These two situations prompted the construction of a statistics object that

[9]The L and R prefixes refer to the fact that a variable whose value is being changed occurs on the left, while a variable whose value is being accessed occurs on the right of an assignment statement.

contains the monitored counters, a method that initializes the counters, and another that increments them once for each replication (since we are collecting statistics over replications, not the number of recruiters or customers). Not including the statistical counters within the state fields of recruiter objects provides the crucial advantage of being able to dispose of these objects between replications for purposes of memory management. In a more esthetic sense, a separate statistical object is more in keeping with the object-oriented design philosophy of the simulation while also reducing the amount of clutter within the code.

4.1.1.4 The Top-Level Design of the SimProcess Model

A recruiter's daily work is a complex process involving a myriad of demands for his or her time. It would be next to impossible to attempt to explicitly model every possible recruiter activity during the year. Not only would it be difficult to list them, but it would also be difficult to collect data on the time requirements for such an overwhelming number of activities. The key to the SimProcess model, just as in the MODSIM model, is the aggregation of lower-level details into larger groups which ultimately form each of the five critical tasks. The top level of the SimProcess model is thus based on a compartmented structure of four sub-processes, which correspond to the critical tasks of prospecting, sales, processing and DEP maintenance (see Figure 18).

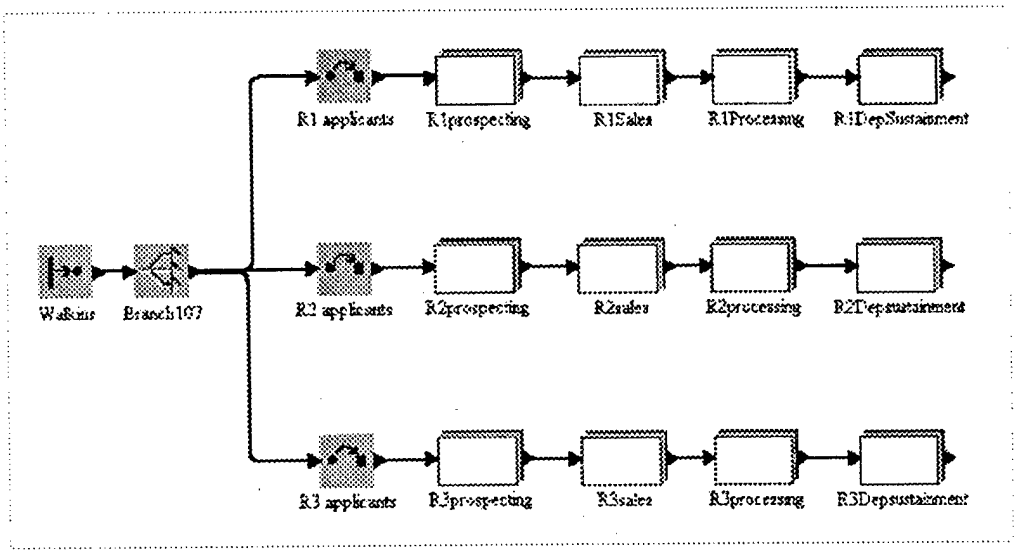

Figure 18. Simprocess Top-Level

The top level of the program shows the three recruiters as three main parallel processes representing three distinct recruiter types (defined by different values of the same model parameters). It is important to note that any number of each recruiter type can be defined at the start of the simulation. Walk-ins are modeled as a separate process that distributes applicant entities to recruiters with equal probability. Once a walk-in has migrated to a particular recruiter along one branch, it is transformed into the same applicant type as that of the entities generated in the prospecting block.

4.1.1.5 Interrupts in SimProcess

One of the limitations of SimProcess is that, although it allows entities in the system to compete for resources, it does not allow activities to be interrupted. This limitation needed to be overcome in order to successfully model the recruiting process with the level of fidelity required. An example of the effect of this limitation can be found in processing, where necessary paperwork could take more than a single working day to complete. If the program were unable to interrupt this activity, nothing else could be accomplished for the entire day. Breaking up a single task into several equal time

portions by means of a loop construct allowed such events as lunch, performing collateral duties, or contingencies, to "interrupt" the task. The reason why such an interruption can occur is that, at the end of each segment, the recruiter resource is released and the applicant entity is scheduled again for the same activity. This arrangement subsequently forces the applicant entity to compete again for the resource. The more time segments are introduced, the better the approximation to a true interrupt becomes.

This implementation led to an unanticipated problem. In SimProcess, entities in a queue are handled by first in first out (FIFO) queuing rules if they have the same priority. This queuing discipline forces applicant entities that have released the recruiter resource after one time segment to wait for every other applicant in line to be served before completing another time segment. This problem, however, is readily solved by increasing the applicant's priority, which in turn allows the applicant to return to the front of the line. This modification works because of the manner in which SimProcess handles events that occur in zero time. When the applicant releases the recruiter resource at the delay node, time ceases to exist for that applicant. The applicant is thus able to pass through the attribute node, receive a new priority, and arrive at the back of the queue formed at the delay node. SimProcess then places the applicant in its prioritized location in the queue before making any further resource requests. Since it possesses the highest priority in the list, the applicant will be placed at the head of the queue at that activity node. If the program does not have any higher priority activities pending, the applicant will again obtain the recruiter resource and complete another time segment.

An issue that we must consequently address is whether or not to allow interruptions of prospecting recruiters. This, in all likelihood, does not occur in an actual recruiting station since the recruiter's daily planner allows for uninterrupted prospecting - unless a walk-in applicant requires

assistance[10]. Planning for such blocks of unbroken activity calls for the ability to schedule daily tasks. As it turns out, boosting priorities to a high enough level will achieve the uninterrupted execution of a task for its allotted time interval. A problem arising from this implementation is that, in the real system, a recruiter will frequently lack the time to simultaneously prospect and complete other, more important activities. In this situation, most recruiters will not schedule any prospecting time in their daily schedule in order to avoid such conflict. Since this decision is made jointly by the recruiter and the station commander, the determination of schedule will vary according to the recruiting station being modeled.

4.1.1.6 Requisition of Resources for Entity Generation

The importance of the preceding method is highlighted when addressing SimProcess's second fundamental limitation. In reality, unless an applicant walks into the recruiting station and asks to join the Army, a recruiter must expend significant time and effort to entice applicants into the station for a sales presentation, a process that we have previously defined as "prospecting". The generation of prospected applicant entities must therefore occur only when a recruiter resource is present. Unfortunately, SimProcess does not provide the option of requiring a resource in a generate node, which in turn forces the placement of a delay node after the generate node, as can be seen in Figure 19[11]. A queue of "pseudo-applicants" waiting to be prospected subsequently forms in front of this delay node during the course of a simulation run. This situation might present serious difficulties if not for the zero-time prioritization of the queue, which prevents bona-fide prospected applicants from competing for resources with yet-to-be prospected pseudo-applicants in the queue.

[10] Actually, we found that even tasks that we had modeled as being non-interruptible were constantly being interrupted. This reflects yet another discrepancy between official policy and reality, as well as between what a recruiter says and eventually does.

[11] Delay nodes can be set to require a user-defined resource in order to elapse a simulated time-interval.

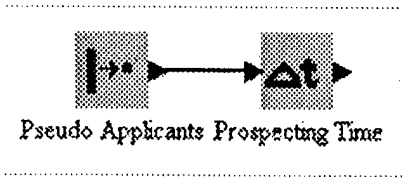

Figure 19. Applicant Generation

The final SimProcess programming limitation is that an activity node cannot request a particular resource based on the type of entity it is serving. This idiosyncracy prompted the creation of several distinct, but identical copies of the recruiting process, or branches, corresponding to each type of recruiter resource. Each branch is an autonomous unit and therefore operates in a parallel structure. The fact that each branch represents a type, and not an individual, means that multiple recruiters of each type may be defined without resorting to the creation of further copies of the process. Despite the added work that this seeming deficiency in SimProcess had caused, the resulting architecture nevertheless did turn out to adequately reflect that of the real system. The independent parallel branches operate in accordance with the USAREC view of the recruiter as a process owner rather than a specialist. In addition, statistics can be more easily collected since individual reports on the different recruiter types are automatically generated. The only drawback is that SimProcess does not generate reports on *individual* recruiters, a fact which compelled us to do our trial runs assigning only one recruiter to each branch.

4.1.1.7 Model Assumptions:

Applicants for each recruiter have the same probabilities for:

- Passing the ASVAB
- Medical waiver approval
- Moral waiver approval
- Base probability for breaking DEP contract

Applicants for each recruiter experience the same delays for:

- Getting an applicant an appointment to test.
- Getting an applicant an appointment for MEPS when immediate processing and normal processing.
- Waiting for medial waiver results
- Waiting for moral waiver results
- Time spent in DEP maintenance

Walk-ins are distributed evenly between all of the station's recruiters.

4.2 Implementation of the Recruiting Model

4.2.1 Generating Applicants: Prospecting and Walk-Ins

In order to provide work for our recruiter resources, we must first generate the applicants. In real-life, an applicant, of course, is "generated" by means of either recruiter prospecting or applicant volition (as in the case of walk-in applicants). These scenarios can be translated into model terms through the following paradigms. The first, which simulates the task of prospecting, involves the requisition of a recruiter resource each time an applicant object is instantiated. The resource is held by the applicant for a certain amount of prospecting time based on a triangular distribution (unique to this particular recruiter). Fields in the applicant object are initialized, after which the applicant is passed to the sales procedure. The other paradigm deals with the simulation of walk-in applicants, which is a relatively simple task. For the duration of the simulation run, applicant objects are instantiated with an interarrival rate governed by an exponential distribution (see Methodology section), and then passed by the station object to the recruiter doing a task with the smallest priority. In the case of a tie, an integer is drawn from $Unif(1, r)$ (r = number of recruiters doing equally low-priority tasks) to determine the lucky recruiter. It is important to note that walk-in applicants do not require a recruiter resource to become instantiated. Nevertheless, after entering the simulation, they are passed to sales in the same manner as prospected applicants.

Note that face-to-face and telephonic prospecting are not physically distinguished in the MOD-SIM model, though they may be characterized by different parameters for task-duration distribu-

tions. It is simply assumed that the recruiter uses a certain amount of lead generation time, after which he brings an applicant into the system. The most serious problem confronting this paradigm is that the model allows interruptions of prospecting, which does not account for the possibility of driving time during which the recruiter cannot be interrupted! Of course, one can prevent interruptions of prospecting by setting the priority high enough - perhaps even higher than every other task.

In both the MODSIM and Simprocess implementations, the number of hours of prospecting is limited on a daily basis in order to more realistically emulate the prospecting schedule of an actual recruiter. This assumption, in fact, can be justified by the fact that a recruiter will rarely spend every free moment prospecting. Moreover, the daily prospecting time can be treated as a model parameter which can be varied to test output sensitivity, thus providing the end-user of the simulation with yet another degree of flexibility. The limitation process also encompasses the termination of prospecting at the end of each day and initialization at the beginning of the next day, as long as a higher priority activity or the prospecting time ceiling does not interfere. The only drawback to this setup is that one cannot schedule prospecting at specific times during the day due to the limitations of MODSIM (this problem is easily overcome with the scheduling functions of Simprocess). This forces the recruiter workday to become front-loaded with prospecting activities, particularly if a recruiter does not have higher priority tasks to deal with.

4.2.1.1 MODSIM-Specific Issues in Applicant Generation

The three subroutines that together form the activity of applicant generation are "Generate", "GenerateWalkins" and "Prospect". The first and last are methods of the recruiter object (RecruiterObj) and so provide input solely to the particular instance of the RecruiterObj type for which the method was originally invoked. For example, when the simulation starts, the Generate method, say, is called by the main program for each recruiter in the station. Suppose the Generate method

corresponding to Recruiter 1 produces an applicant. For the remainder of the time this applicant is in the system, it will continue to use only the recruiter resource associated to Recruiter 1. This independent operation of each recruiter object is, in essence, a manifestation of the parallel design principle. The GenerateWalkins routine, unlike the other two, is a method of the recruiting station object, which thereby enables it to provide input to any of the recruiters.

Method Generate

We shall first discuss the general operation of the Generate method, which works in tandem with the Prospect method to produce input to the system. One fact worth noting is that recruiters are assigned (as a parameter) a maximum number of recruiting hours in a single day. When the ceiling is reached on a particular day, prospecting is interrupted and then restarted on the succeeding day. Thus, applicant generation must also conform to this schedule. The Generate routine produces *pseudo-applicants* [12] on a daily basis (in simulation time) until either the maximum recruiting hours have been expended or the end of the day has occurred (refer to Section 4.1.1.2). The pseudo-applicant will now wait for a recruiter resource based on the prospecting priority by calling the "grabrecruiter" method. Once the resource is obtained, the pseudo-applicant is sent to prospecting. The pseudo-applicant does not yet possess applicant status (its "apstage" field does not contain a value) since it has not held the recruiter resource for the required length of prospecting time.

Method GenerateWalkins

The concept behind this station object method is even more simple than that of Generate. Unlike the behavior of arrivals due to prospecting, walk-in applicant arrivals are not bound by recruiter scheduling constraints; they may, in fact, appear at the station at any time during the working day and week. Hence, the walk-in routine continuously generates pseudo-applicants throughout the day

[12] Those applicant objects who have not officially entered the system.

without a set daily ceiling. The interrarrival rate is modeled as an exponential random variable due to the inherent similarity of this situation with the arrival of customers to a queuing system. Another difference (as mentioned previously) is that the pseudo-applicants produced here may be sent to any of the recruiters. The decision as to which recruiter should handle the walk-in applicant is based on the priority of their current task. The algorithm searches through the priority field for each of the station recruiters (stored in an array) and stores references to the recruiter(s) with the lowest priority. In case of a tie, a $Unif(0, 1)$ random number is drawn so that one of these recruiters will be chosen with equal probability. From here, the pseudo-applicant is tagged as a walk-in applicant and sent directly to Sales (since no prospecting delays are needed).

Method Prospect

Once a pseudo-applicant from Generate has been created and initialized, it is then sent to Prospect in order to incur a prospecting delay. Its interarrival time is a model parameter that corresponds to the distribution of the time it takes a particular recruiter to generate an applicant - in other words, lead generation time. This parameter is modeled as a random variate from a triangular distribution in which the mode reflects the recruiter's most frequently observed time and the other parameters contain the shortest and longest observed times. Once the prospecting duration for the pseudo-applicant has been determined, the procedure makes sure that the length of the day is not exceeded and that enough prospecting time remains to generate the applicant. If so, what is now a full-fledged applicant holds on to the recruiter resource for the required amount of time, after which another random number is drawn to determine if the applicant will even show up at the sales interview. If the applicant object remains in the system (i.e. if the applicant will attend the interview), then it re-competes for the resource in order to proceed to Sales. Failure of either condition will prompt the (pseudo) applicant to relinquish the resource and leave the system.

4.2.1.2 Generation of Applicants in SimProcess

The prospecting process (as noted previously) encompasses lead generation, and hence operates on the implicit assumption that all necessary lead generation work has been accomplished. Quality lead generation efforts are reflected in the number of applicants generated by a recruiter, and thus the ability of a particular simulated recruiter to prospect can be set through modification of the relevant model parameters.

As seen in Figure 20, pseudo-applicants start at the generation node and pass through a branching node that separates the pseudo-applicants into telephone and face to face prospecting applicants. Afterwards, applicants are assigned the appropriate recruiter based on prospecting priority. Once their priority has been assigned, applicants then proceed to a delay node to compete for the recruiter resource and then wait the assigned prospecting duration before proceeding to the sales interview.

In order to make the program match reality as closely as possible, it was necessary to come up with a way to schedule both face-to-face and telephone prospecting. This was accomplished by defining separate recruiter resources for each of telephone and face-to-face prospecting. Because SimProcess allows the scheduling of downtimes for resources, these specialized prospecting resources could be made available for the desired intervals of time during the working day. In addition, the prospecting sequence depicted here requires both the main recruiter resource as well as one of the prospecting resources in order for the delay to function. As a result, prospecting is limited to the hours in which the face-to-face and telephone prospecting resources are available. Additionally, this arrangement prevents the recruiter resource from being simultaneously used for both prospecting and another activity.

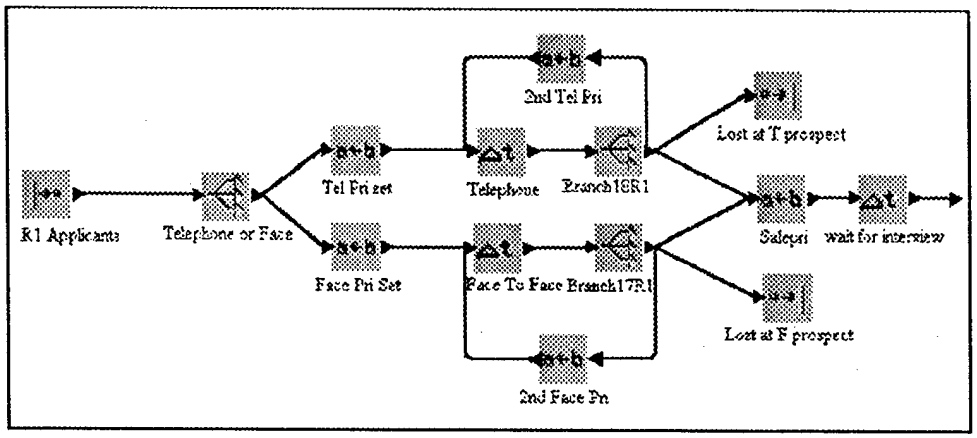

Figure 20. SP Prospect Model

Many of the prospected applicants are lost before they arrive at their sales interview. This factor is incorporated into the model by allowing applicants to be disposed on the basis of a recruiter-specific probability. If applicants are lost, they are then assigned the recruiter-dependent sales priority and experience a 24 - 72 hour delay before arriving at the recruiting station for a sales interview. This delay is taken from the USAREC policy regarding the prescribed interval of time between prospecting and the subsequent sales interview.

4.2.2 Selling the Army

4.2.2.1 Overview of the Sales Process Implementation

In both MODSIM and SimProcess, the simulation model of Sales receives an infusion of applicant entities from two sources, namely the walk-in and prospected portions of applicant generation (see Figure 21 for the SimProcess version of Sales). It was possible to aggregate the two types of sales interviews for the purposes of simplification, but observations of the actual system revealed that the loss percentage differed significantly on the type of applicant. As mentioned before, the Huber Heights station averaged a sixty-percent loss of walk-ins after the pre-qualification sales interview, which exceeded the rate for other applicants. The difference can be traced to the fact that

walk-ins have not been screened over the telephone, and are thus unknown quantities at the time they enter the station. We therefore felt that it would be highly beneficial, and not too difficult, to implement the sales interview as two sub-processes. Otherwise, loss statistics would contain a distinct downward bias, and the salesmanship of the recruiter would be confounded with the loss rate due to screening.

Once an applicant entity arrives at sales, the simulation determines whether the applicant was prospected or else simply walked in. Prospected applicants essentially undergo a pre-qualification interview which is incorporated into the first few telephonic contacts. These questions often reveal factors that may preclude the applicant from ever entering the Army, thus making it unprofitable for the recruiter to deal with the applicant any further. Walk-in applicants, however, have not yet spoken to a recruiter, and thus require an on-site prescreening. Not only does this unexpected task add more time to the sales interview, and it is also characterized by an exorbitant failure rate. The program addresses these differences by adding a further delay to the sales interview duration of walk-in applicants and then disposing a percentage of them. The percentage of walk-in applicants that fail the pre-qualification interview is station-dependent, and is therefore incorporated as a model parameter.

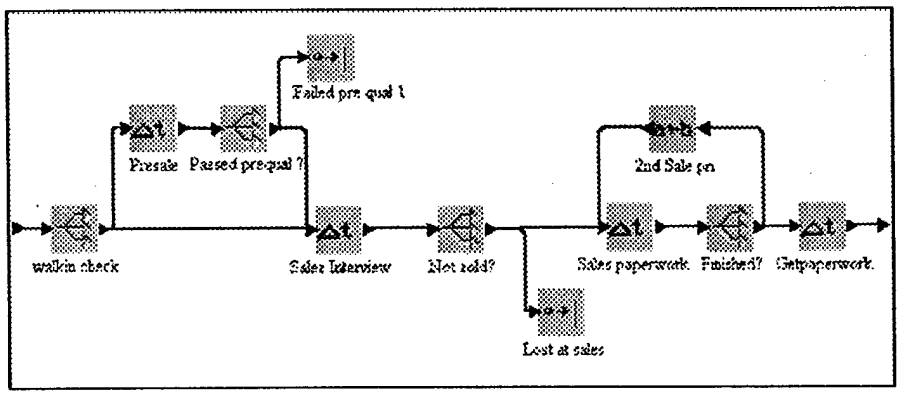

Figure 21. SP Sales Model

4.2.2.2 *Applicant Loss and Interrupts*

As with prospecting, a certain percentage of loss is also associated with sales and, moreover, is highly recruiter-dependent. Hence, each recruiter (recruiter type in the SimProcess implementation) is associated with its own unique loss percentage that is, in itself, a model parameter. Recruiters at the Huber Heights station have noted that the greatest percentage of applicant loss occurs after the sales interview. Therefore, setting the loss parameter correctly plays a large role in determining the accuracy of the model. Once an applicant entity manages to clear this portion of the simulation, it will proceed to the Processing phase[13]. Others will be disposed of permanently from the system.

The most significant difference between the MODSIM and SimProcess implementations can be found in the role of the sales process during interrupts. MODSIM treats sales in the same way that it treats any other priority-assigned activity, and thus allows it to both interrupt and be interrupted. On the other hand, the SimProcess version of the sales process does not allow interruptions via the pseudo-interrupt loop. This means that the interview is allowed to run its course once the applicant arrives at the sales interview delay node. By USAREC regulations, the sales interview must be scheduled within 72 hours of the prospecting time and are normally not cancelled (except in case of a dire emergency).

Recruiters will usually attempt to start work on the enlistment package during the sales interview. This provides the recruiter with the opportunity to quickly determine the supporting documents that might be required of the applicant, in addition to assisting the recruiter in his efforts to finish the processing phase as quickly as possible. As a consequence, one could consider this initial adminstrative task to be part of the sales process. This philosophy was integrated into the SimProcess model by including a sales paperwork time delay. However, unlike the rest of the sales interview, the recruiter does not need the applicant's presence in order to complete the paperwork,

[13]In the MODSIM implementation, applicant entities will wait for a recruiter resource before proceeding to Processing.

and so we have made this an interruptible task in the SimProcess version. MODSIM likewise allows interruption of the admininstrative portion, though the task is included within the Processing rather than the Sales procedure.

4.2.2.3 Modeling Issues

As implied in the Methodology chapter, triangular distributions are utilized for all activity delays in Sales, including the sales / pre-sales interview and sales paperwork delays. Decision points are represented by drawing random numbers according to probabilities that are incorporated as model parameters (e.g. applicant loss after the sales interview). The statistics collected at this stage include mostly counts of entity losses as well as averages for each of the delays. Since we are concerned with system output sensitivity rather than output from individual processes, knowledge of these statistics are important for their use in verifying that the correct parameters were entered into the model. Of special interest is the loss after the sales delay, as this imposes the greatest cost to the system in terms of entity attrition.

4.2.2.4 MODSIM-Specific Issues

The method in which all activities of the Sales process are modeled in the MODSIM model is "Sell" (See Appendix C. Model Code). With the exception of the preparation of the enlistment package in sales (in the SimProcess simulation) and the matter of interrupts, both sales models generally coincide. A few minor differences yet remain to be discussed. In the MODSIM model, the Sales method draws a random number that determines whether the applicant will go to Normal or Immediate processing shortly after the applicant object is first instantiated in the applicantObj method setFields; in other words, this characteristic is contained in the state fields of each applicant. This methodology is in keeping with the fact that going to either Immediate of Normal processing is a major descriptor of an applicant. For instance, in the actual system, the applicant could be suf-

ficiently motivated and qualified, in which case he or she would undergo the Immediate processing step. All others would be sent through normal processing (see Section 1). One last task remains before Sales terminates, namely that the applicant entity must wait again for the recruiter resource before continuing to the Processing stage.

4.2.3 Processing Applicants

Processing is possibly the most involved method because of the variety of activities that it encompasses. To be more specific, it is comprised of the two major subprocesses of Immediate and Normal Processing, each of which includes administrative work, testing, waivers, and the MEPS physical. The design of the Processing model is the result of an extensive series of conversations with system experts (recruiters and USAREC officials) and references to several USAREC regulations [30] [28] . We recall from the discussion in Chapter 2 that recruiters indicated the existence of the Immediate and Normal Processing procedures, the choice of which is contingent upon applicant intentions as well as eligibility for military service. Both the SimProcess and MODSIM models feature this dichotomous arrangement because of the anticipated sensitivity of the system to the difference in throughput times and loss probabilities between these procedures. We will discuss both the similarities and differences between the MODSIM and SimProcess simulations, after a description of certain implementation considerations.

4.2.3.1 MODSIM Processing

The methods included under the blanket description of "Processing" in the MODSIM model are "Testing", "Moralwaiver", "Medicalwaiver", "Normalprocess", and "Immediateprocess". Though their functions are self-explanatory, there are a number of issues concerning the path of applicants through these methods as well as some fairly subtle assumptions. All time delays and probabilities are modeled as before (triangular and uniform variates, respectively) and incorporate data collected

from observations of various recruiting stations. We begin with a discussion of the two upper-level processing methods, Normalprocess and Immediateprocess.

Normal Processing

If applicant entities are not disposed of after the initial sales interview, they proceed to either Normal or Immediate Processing according to the value of their field parameter (applicant.processcat) that was initialized when the applicant was first instantiated. Moreover, the assumption is made that these applicants have brough the necessary documents with them (e.g. birth certificate, Social Security card, etc.). Those who arrive at Normal Processing correspond to those applicants in the actual system who are not as yet firmly committed to the idea of an Army career or have not displayed an aptitude to pass the AFQT through failing the pre-test. To make matters more difficult, several additional barriers still stand in the way of enlistment, namely the possibility of (1) failing the AFQT, (2) requiring a moral waiver, or (3) requiring a medical waiver as a result of a bad MEPS physical examination. Because of the aforementioned conditions that preclude their participation in Immediate processing, Normal processing applicants take the AFQT examination before accomplishing anything else. The applicant first releases the recruiter resource (which it obtained in sales) and then goes to the Test routine.

Unlike most of the other methods, Test does not require a recruiter resource (based on the fact that the AFQT is usually proctored by MEPS personnel in the actual system), which is the reason for its realease by the applicant in question. The method itself draws a random number based on a test-pass probability (a model parameter) and flags the applicant entity according to whether or not it passed the examination. All applicants - regardless of test status - are sent back to their respective processing method, with the difference being that only those who have passed acquire a recruiter resource. For these applicants who have passed the examination, there is an additional delay of

exactly five days in simulation time before returning to Processing, which reflects the wait for a MEPS appointment. The applicants who failed are provided the opportunity to retest after a three-day delay, but successive failures result in a permanent removal from the system. The option is left open in the code for further delays, but discussions with recruiters seemed to indicate that failure on the second try often slashed the success probability on future tries down to almost nil.

As mentioned in the previous section, the MODSIM model diverges from its SimProcess counterpart in a major way by requiring the sales paperwork / enlistment package delay in Processing rather than in sales. Applicant entities who have passed the AFQT examination obtain the recruiter resource for the random period of time needed to complete the paperwork, after which a decision is made concerning whether or not to continue. Every task associated with completing the package, including police checks and driving time, is included within this delay. If an interruption occurs, the method will keep track of the number of hours the applicant has been processed out of the original delay, after which the applicant will be rescheduled[14]. This cycle will continue until the applicant has completely expended its processing time.

The end of administrative tasks marks the beginning of the waiver process. An applicant's waiver status has already been determined in the applicantObj method setFields, and so this portion of the code simply routes the entity to the appropriate waiver method(s) (medical or moral) depending on the value of the waiver flags within the applicant's state fields. For instance, if the needMoral flag has a value of 1, then the applicant needs moral waiver consideration. The assumptions of this stage are that the applicant's police and court record is known through recruiter background checks and, in addition, the applicant has been to the MEPS and received a medical evaluation[15]. These activities provide the real-life foundation for determining whether or not a waiver is needed. The

[14]This is done by means of method Grabrecruiter (see Model Code, Appendix A and Section 4.1).

[15]Recall from the two previous paragraphs that a MEPS delay was scheduled in the method Test.

Moralwaiver and Medicalwaiver methods then proceed to elapse the time needed to accomplish related paperwork and then dispose of the applicant if the waiver is not granted.

Immediate Processing

Applicants who arrive here, unlike their Normal Processing counterparts, are assumed to have passed the pre-ASVAB CAST or EST screening test and have also made a firm commitment to enlist. Therefore, the recruiter will make every effort to hurry the process, which involves a slight rearrangement of the tasks involved in Normal Processing. Instead of initally scheduling the applicant to take the AFQT, the recruiter completes the enlistment packet and then schedules the test and MEPS physical on the same day. Finally, waiver applications are accomplished, together with the associated waiting times for decisions. Of course, the applicant entity is disposed if the waiver(s) is not granted or a decision is arbitrarily made to not continue. In addition to the difference in structure to the Normal method, loss probabilities are set to be lower while test success probabilities are higher for the Immediate applicant entities due to the assumptions that govern their presence in this branch of Processing.

4.2.3.2 The SimProcess Implementation of Processing

Many of the steps outlined above are followed in the same manner in the SimProcess model of Processing. However, the SimProcess has benefited from the authors' experience in coding the MODSIM model, coupled with further field investigations and consultation with system experts. In the following discourse, we shall attempt to cover the various ways (often subtle) in which the two implementations differ from each other. These differences are usually the result of an attempt to more realistically reflect the decisions that actual recruiters will make in the course of an applicant's tenure in the recruiting station.

The model includes an initial delay that models the time needed to obtain documents, unlike MODSIM (see the first paragraph of Normal Processing). The Processing step assumes that all applicants processed at a specific station require the same amount of time to gather these documents and return to the recruiting station. The parameters used to determine this waiting time can be modified before starting the simulation.

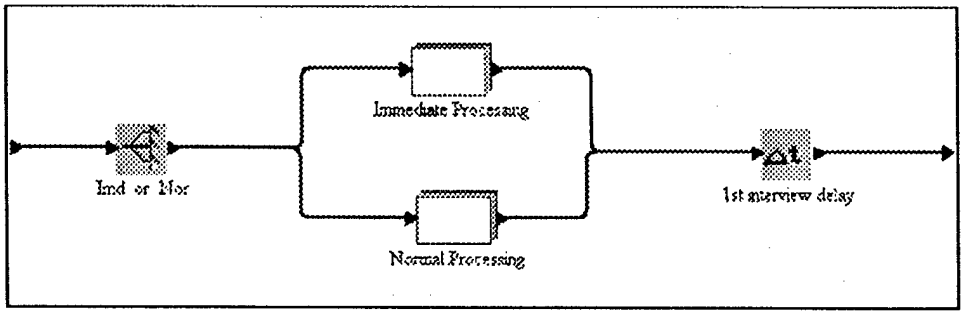

Figure 22. SP Processing Model

As can be seen from Figure 22, applicants branch by the type of processing they are proceeding to when they arrive at the processing activity, rather than when they are instantiated (i.e. created at the generate node for the recruiter in question). We will first discuss the SimProcess implementation of Immediate Processing as compared to that of MODSIM.

Immediate Processing (SimProcess Version)

As soon as an applicant arrives at the immediate processing block (see Figure 23), it is assigned (through a transformation node) the priority that the recruiter resource contains as a local attribute[16]. The next step is to complete the enlistment package, which is modeled as a pseudo-interrupt loop (see Section 4.1). As in the MODSIM program, the enlistment package delay subsumes all associated tasks.

[16]The SimProcess version of state field.

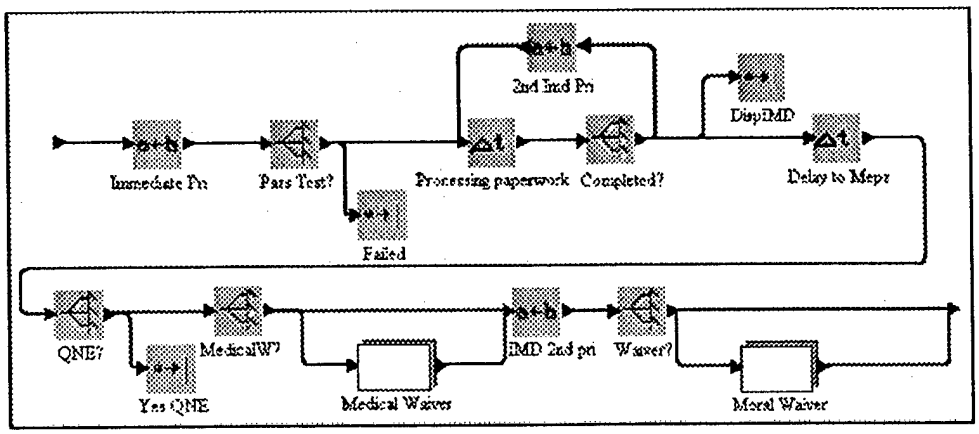

Figure 23. SP Immediate Processing Model

If an applicant does not decide against the Army when the paperwork is completed, he / she is then scheduled for both the MEPS physical evaluation as well as the ASVAB test. Each applicant at the station going through Immediate Processing experiences time delays sampled from the same distribution before proceeding on to MEPS. There are several ways in which a recruiter can lose an applicant while the applicant is at MEPS. For instance, if an applicant successfully passes the ASVAB and the physical, he or she consults a job counselor to select a specialty. Applicants who know neither which jobs they are qualified for nor which are available are at special risk for QNE ("qualified but did not enlist") if they do not receive the offer they had anticipated. Losses resulting from testing are rarely seen among applicants in Immediate Processing due to the screening effect of the CAST / EST pre-examinations (see Section 4.2.3.1). Consequently, the model assumes that, if the applicant has passed either the CAST or the EST, he or she will also have passed the ASVAB.

The processing phase now branches into two further sub-processes: medical and moral waivers. We have observed that the frequency of moral and medical waiver applications vary substantially according to demographics. With this reality in mind, the probabilities of entering either of these branches are included as model parameters. The order of the waiver sub-processes likewise reflect

a real-world consideration. For example, a recruiter will often encounter a situation in which the applicant may require moral waiver consideration. However, in light of the considerable amount of time and energy needed to process the moral waiver cases, the recruiter will (or should) send the applicant to MEPS first in order to ensure at least a medical qualification. Once all necessary waivers have been obtained and the applicant has not been lost at any decision points, the model assumes that the applicant has enlisted. He or she is now officially a member of the Delayed Entry Program (DEP).

Normal Processing (SimProcess Version)

Refer to Figure 24. As in the MODSIM model, both the Immediate and Normal Processing branches are very similar. A few differences do, however, bear mentioning. Applicant entities in Normal Processing experience two additional delays, the first dealing with the the AFQT and the other being the time spent in a post-exam meeting with the recruiter after the test. Recall that the primary assumption about applicants in this phase of processing is that they have as yet to make a firm commitment to enlist. Therefore, we follow the same sequence as in the MODSIM Normal Processing model, namely that the recruiter (1) schedules the applicant to take the test, (2) attempts to extract a commitment, and, if this is successful, (3) works on the enlistment package and schedules a MEPS appointment. The delay to MEPS in normal processing is usually slightly longer than that experienced by applicants going through immediate processing. The steps in both the Immediate and Normal branches coincide from this point forward.

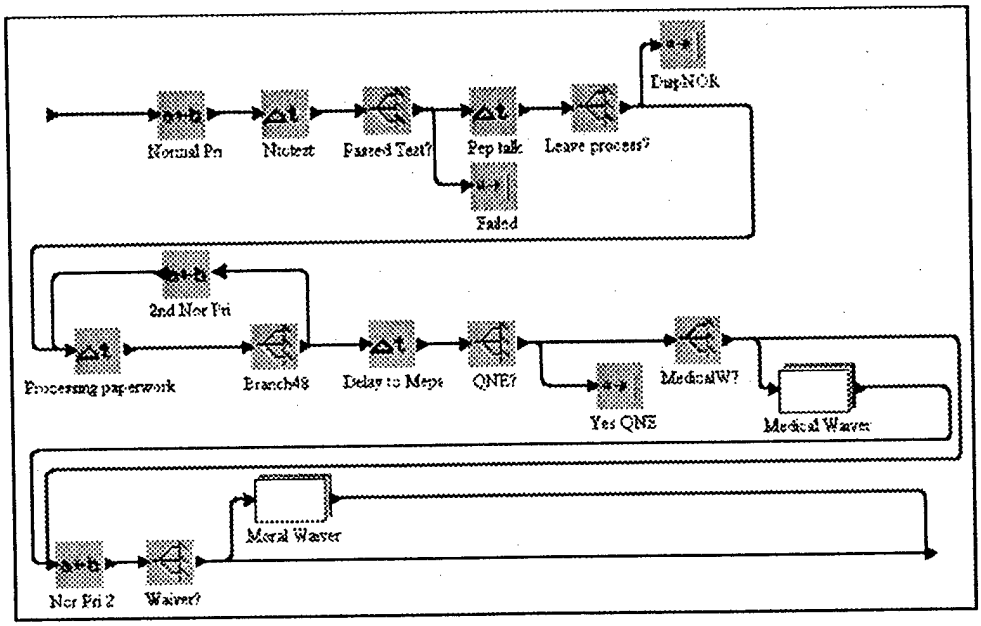

Figure 24. SP Normal Processing Model

MedicalWaiver(SimProcess)

The medical waiver process (reference Figure 25) is initiated after the applicant is disqualified by means of a physical examination at MEPS. The applicant may wish to apply for a waiver of the standards. Needless to say, the process requires a significant investment of both the recruiter's and the applicant's time (and energy). An experienced recruiter can often discern the waiverable from the non-waiverable cases, and advise the applicant accordingly. Nevertheless, the applicant has the right to request waiver consideration, a decision which likewise obligates the recruiter to help. An actual recruiter may vary the amount of involvement in the recruiter's waiver application depending on his or her opinion about the chances for an approval. However, the model assumes the recruiter is actively supporting the applicant in the waiver process.

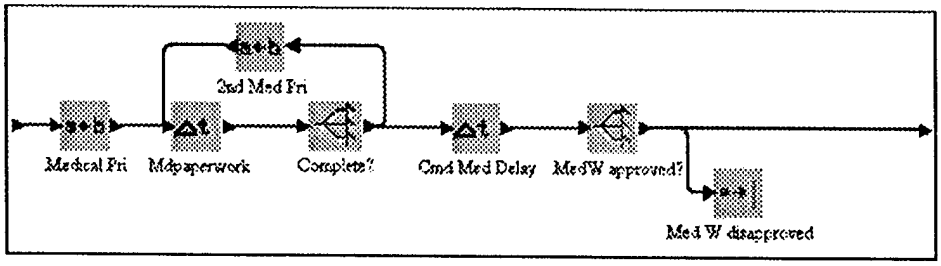

Figure 25. SP Medical Waivers

When applicant entities arrive at the medical waiver process, they are again assigned the medical waiver priority associated with their respective recruiters. The now-familiar pseudo-interrupt looping structure characterizes the administrative portion of the process, which is followed by a delay corresponding to the time needed for the USAREC Command Surgeon to review the package and make an approval decision. If the Command Surgeon concurs, then the applicant can pick an MOS, take the oath of enlistment, and join the DEP. The probability that the waiver will be approved is not a model parameter because the approval authority is the same for all stations. The value is, nevertheless, accessible through a direct modification of the SimProcess code.

Moral Waivers (SimProcess)

As noted previously, moral waivers require a more substantial amount of effort than do medical waivers. Despite this seemingly demoralizing factor, the model assumes that the recruiter fully supports the applicant's waiver request. We have learned from past conversations with recruiters that administrative tasks (i.e. completing the waiver package) are usually completed over a period of three days. This process is captured in the model through a division of the time needed to complete a moral waiver into three separate delay nodes, with each representing a 24-hour period of time, as shown in Figure 26. Although moral waivers are not customarily a high-priority task, the model ensures that at least one time segment is not interrupted once the work has begun. Once the waiver

package has been submitted, the applicant must wait for PERSCOM to review the package and make a determination. If the waiver is approved, the applicant then joins the DEP. As with medical waivers, any changes to the probability of waiver approval must be accomplished through direct modification of the SimProcess code.

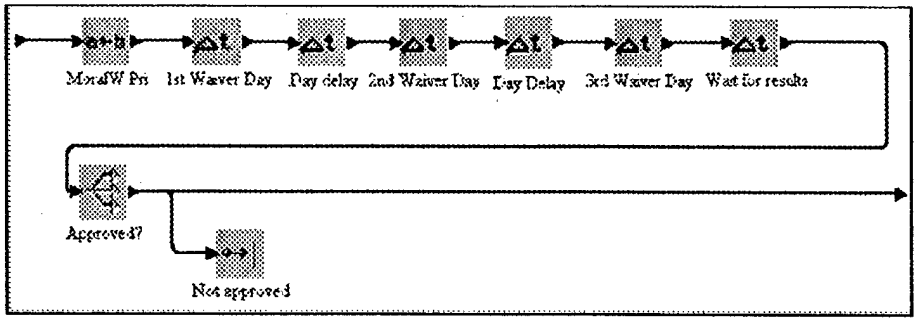

Figure 26. SP Moral Waivers

4.2.4 Sustaining DEP Applicants

The Delayed Entry Program (DEP) sustainment (or maintenance) is an area that is frequently underestimated by recruiters for its affect on applicant retainment. Common wisdom seems to imply that more time spent on sustainment activities equates to correspondingly lighter DEP losses. Indeed, the experiences of the past 24 years of DEP show that recruiters who keep their applicants involved in the recruiting process all the way to the shipping date will be more successful in retaining their applicants. However, some recruiters have mentioned the existence of an adverse effect, namely that the DEP is often so costly in terms of time management that it tends to plunder resources from other areas of the recruiting process. It would be interesting to see whether or not this adage actually holds true with respect to the model output.

4.2.4.1 Acceptance of Entities into DEP Sustainment

DEP Sustainment plays a unique role in the simulation model in that applicants continually cycle through the method for a relatively long period of time. Each cycle itself represents one DEP sustainment contact, which USAREC mandates at least once every two weeks. The applicants continue to cycle through until their alotted time in DEP has expired.After all processing activities have been completed, applicants become official enlistees awaiting either a basic training class date or the availability of advanced training slots for their intended profession. Within 10 days of the end of processing, an orientation DEP briefing describing the responsibilities of being enrolled in the DEP is scheduled for the benefit of the applicant in question. In both the SimProcess and MODSIM models, this initial DEP meeting has a higher priority than that of subsequent meetings. The reason for this assignment of priority is that the recruiter wants to establish a significant enough rapport with the applicant so as to prevent the so-called "buyer's remorse" from creating a loss. This higher priority for the first interview is applied to applicants upon their arrival to DEP sustainment / maintenance. When an applicant in the model has completed the initial DEP briefing it is assigned a priority relative to the importance the recruiter places on DEP maintenance and waits a period of time (one week in SimProcess and a time duration of depIntArv in MODSIM) for its first regular DEP sustainment contact.

4.2.4.2 DEP Sustainment in MODSIM

The methods pertinent to DEP Sustainment are denoted "Sustainment" and "depMeet", with the first handling the control sequence and the other elapsing the amount of time transpired during each sustainment meeting. When the applicant entity first enters Sustainment, DEP duration, applicant priority in relation to DEP, and other necessary variables are initialized. As noted previously, the first meeting is given the highest priority (recruiter's DEP priority + 10) with subsequent

meetings equal to the recruiter's priority alone. The first meeting is scheduled for a period of time distributed exponentially with a mean of four days, and the applicant is sent to depMeet to obtain the recruiter resource for a triangular duration with parameters input by the user.

Once in depMeet, the recruiter resource is acquired using grabDepRecruiter, and the model waits for the generated meeting duration. This method is an ASK FOR rather than a TELL method because we want Sustain to wait for the completion of depMeet before proceeding (see Section 3.1.1). The method checks if less than two weeks have passed since the last meeting, and if so, it decreases the loss probability as a bonus. The depMeet method then returns control to Sustain.

The method depMeet keeps track of the amount of time each applicant has been in DEP and assesses a monthly DEP loss increment onto the applicant's current loss probability. The monthly increment was set to 3.5%, a figure which was based on the Kearl-Nelson study[8, 261] . As usual, interruptions remain a possibility, and correspond in the actual system to missed or neglected meetings. Therefore, a penalty is assessed if an interruption occurs, and is calculated as (Monthly DEP Loss Probability) * (Time Since Last Meeting / 30 Days). This amount is added onto the current DEP loss probability for the next cycle through the method. However, the effect of increased loss probability is offset by a corresponding boost in priority, which is given to the applicant to ensure that the next meeting will not be missed. Thus, this increment is set so that the next meeting is of greatest priority for the recruiter.

Finally, a random draw to see whether or not the applicant survives DEP to the next cycle is accomplished (note that this is done *before* the applicant is assessed the monthly DEP loss probability increment). The appropriate statistics are collected, and the applicant is scheduled for the next meeting. Any applicants who survive until their allotted time in DEP has expired are counted as a contract for the recruiter in question. If simulation time expires before this can happen, the applicant is simply removed from the system and is not counted.

4.2.4.3 DEP Sustainment in SimProcess

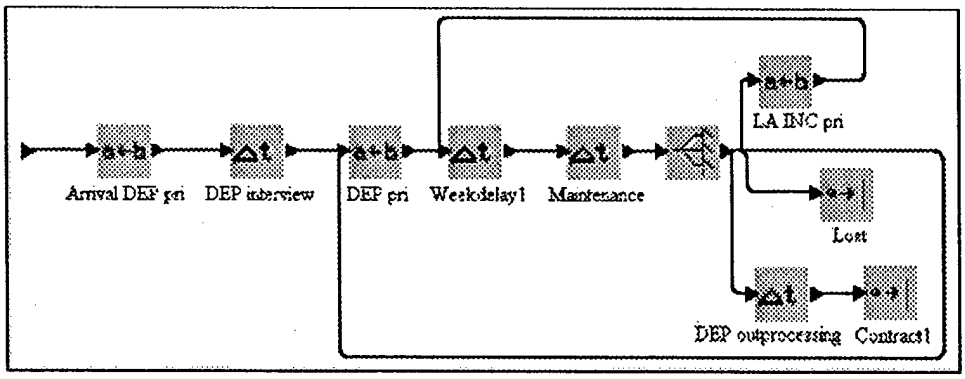

Figure 27. SP DEP Model

We have discovered that, in practice, members of DEP are contacted on a weekly basis. Two of these are carried out by phone and the other two are conducted face-to-face. These repeated contacts are implemented in the model through a simple loop structure, as can be seen from Figure 27. Applicant entities carry an attribute with them that stores the type of contact that was conducted on the previous pass through the loop. The applicant then experiences a delay which is a dependent upon the recruiter-specific meeting-duration probability distribution as well as the type of contact made, which is stored in a state field of the applicant.

The case of a missed meeting is handled in a similar way to that of the MODSIM implementation. If an applicant had not been seen for at least the nine previous days, the applicant will receive priority boost to reduce the chance of a repeated missed contact. Simultaneously, the probability that the applicant will become a DEP loss is increased by a user-defined percentage. The DEP applicant maintains the increase in priority for one cycle before resuming contacts at the usual priority given to all DEP applicants for this particular recruiter.

DEP sustainment is often regarded as a low priority task by station recruiters - in the model as well as in real life - and hence sustainment activities carried out during the simulation must be

carefully scrutinized. One of the undesirable scenarios that bears watching is that members of DEP might not be seen often enough or even not at all if a simulated recruiter is inadvertently assigned (unrealistically) high prospecting or sales priority. Another remote possibility is that an applicant may be seen only once in several months, and yet still make it to the shipping date. This may occur as a result of an applicant obtaining the recruiter resource and conducting an appointment before an increase in DEP loss probability for missed appointments can be assessed[17]. As a precautionary measure, the simulation will notify the user if a DEP applicant is contracting and has completed less than half of his or her scheduled DEP meetings.

The final step in the sustainment process is a DEP out-processing briefing. This briefing lasts approximately an hour, during which time the applicant is given instructions concerning what to bring and how to behave upon arrival at basic training. When the meeting concludes, the applicant leaves the system and the number of contracts for the recruiter is incremented.

4.2.5 Collateral Duties in SimProcess

As noted previously, it would very difficult for any model to include all of the activities a recruiter performs during the year in minute detail. It is, however, essential that all of the time expended by a recruiter is accounted for by some means within the model. Therefore, any time that cannot be attributed to any recruiting task discussed previously in this chapter (except for Planning) is aggregated into what we will term as *collateral duties* (see Figure 28). This generic activity is currently given the highest priority in the model and is scheduled for each recruiter on a daily basis. The expenditure of time on collateral duties is accomplished within the model by means of a collateral entity, which obtains the recruiter resource and waits a user-defined/default delay. After this delay has transpired, the collateral entity requests the resource again (after a few hours

[17] An applicant must obtain the recruiter resource before either assessing the penalty or increasing the sustainment priority when a meeting is missed.

have passed) in order to simulate a lunch-break. Time spent performing collateral duties is likewise included in the recruiter utilization time.

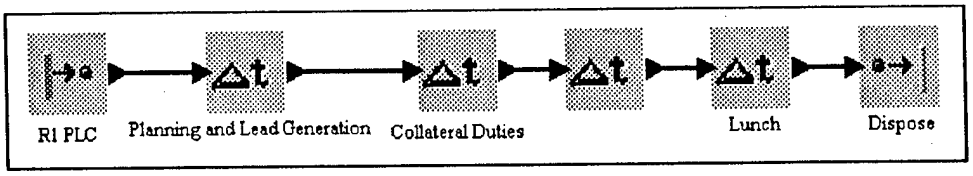

Figure 28. Collateral Duties

4.2.6 Modeling Issues

It is a tenet of simulation analysis that one must be aware of the need to use a model only for the purposes for which it was intended. In this section, we will discuss several issues that a user must be aware of with regard to the Army Recruiting Model. One of these issues deals with the actual operation of the model, insofar as the logistics of use are concerned. Another important area which may hold many pitfalls for a user unfamiliar with the system is output interpretation. Although the authors strived to make the simulation as much of a user-friendly system as possible, users still need to be aware of certain subtleties in the model construction to fully understand the output that they obtain. This awareness must include a recognition of the fact that the model cannot incorporate all levels of detail of the actual recruiting. In other words, a certain degree of knowledge of the particular strengths and limitations of the model will enable a user of the system to apply the model results properly to the fulfillment of his or her particular goal.

4.2.6.1 Model Use

The first issue we will discuss is that of the proper usage of the simulation. One should pay heed to the following recommendations while preparing the SimProcess simulation for trial runs.

1. *Modifying the collateral duty time.* This may also require the modification of the delay before

lunchtime. For example, if the collateral entities are created at 08:00 and a collateral time parameter is set to 4 hours per day, the user must make sure that the delay before lunch is 0.0 in order to allow lunch to take place at 12:00. If this is not done, the user could inadvertently schedule lunch to take place during prospecting time. This can occur because, by default, collateral duties are given the highest priority, and therefore lunch would keep the recruiter from performing his/her regularly scheduled prospecting.

2. *Simulation run length.* In SimProcess, the simulation run length includes the warm up time. Warm up periods, by virtue of their statistics not being included in the overall trial statistics, are the means by which initial bias conditions are eliminated. However, one must be aware of the fact that, in the Simulation Run Settings dialog box, a one-year warm-up preceding a one-year trial should be entered as a two-year simulation run-length. In other words, the *actual* simulation run length should be increased by the same amount as the warm up length in order to ensure that statistics are collected for the amount of time that the user intended. Figure 29 below visually illustrates how the developers of SimProcess implemented the warm-up period.

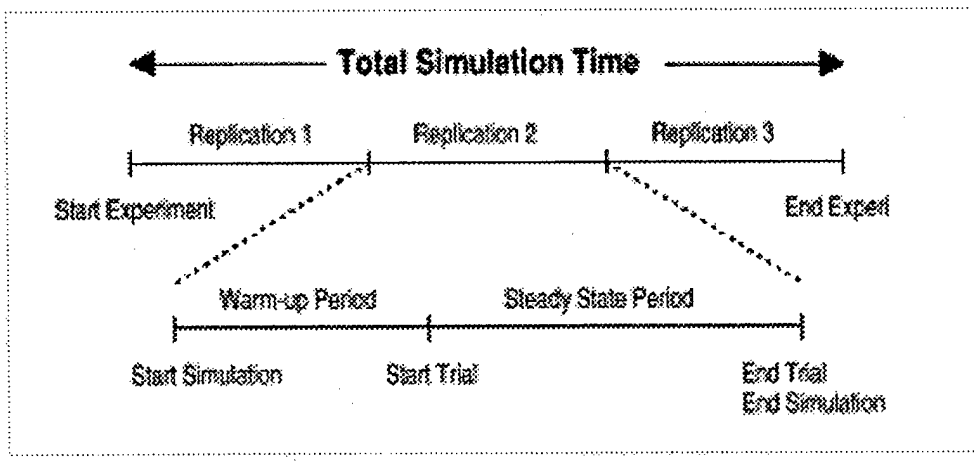

Figure 29. Warmup Concept (from SimProcess User's Manual)

4.2.6.2 Model Output Interpretation

The following descriptions should make program output easier to interpret.

1. *Recruiter utilization rate does not ever reach 100%.* The resource utilization rate in SimProcess is the ratio between the amount of time the resource is occupied and the total time the simulation is running, just as one might expect. However, the catch is that recruiter resources cannot be utilized during scheduled downtimes (non-working hours). This constrains the maximum possible utilization that can be attained to be the percentage of time a recruiter resource is available during a year versus the total number of hours in a year. Consequently, maximum utilization varies depending on the availability of recruiter resources. Certain idiosyncracies inherent in SimProcess's resource allocation procedures tend to make the matter worse. For instance, SimProcess does not allow for the release of resources by entities upon the start of a downtime period. This prevents a recruiter resource from entering into its scheduled downtime period until the applicant entity releases it (i.e. the task is complete). This can quite possibly result in a utilization rate that is inaccurately biased upwards. Take, for instance, the following hypothetical situation confronting a user of the simulation. As previously noted, the simulation recruiter resources are available from 08:00 to 20:00 Monday through Friday (default "working hours"). The maximum utilization rate for a recruiter resource that was never idle - based on this default working day - is 29.76%, which is determined by dividing the 50 hours of working time by 168 total hours in a week. The simulation may, nevertheless, output a figure that is greater than 30% because of the fact that applicants retain resources past regularly scheduled downtime periods.

2. *Inordinately long queues.* When observing wait durations for sales interviews, one may come across times that range anywhere from 12 to 36 hours. This can be an occasion for some

surprise due to the fact that sales interviews are of very high priority, and should therefore take a matter of minutes or hours rather than days. However, upon further reflection, the reason for the excessive durations became fairly clear. The extra time is the result of queues forming over weekends, which must occur since the resource is not available during these downtimes. More specifically, if an applicant arrives at the sales process and finds that it cannot obtain that resource by the downtime, the applicant may spend the weekend in queue waiting for the sales interview. This means that the entire time is included within the report for time-in-queue, which in turn biases the average wait for the sales interview upwards. However, the fact that this anomaly is consisten across recruiter-parallel processes implies that one does not lose the ability to form comparisons between different alternatives.

4.2.6.3 Process Features Not Addressed by the Model

As mentioned previously, it is usually infeasible to model every aspect of the system in question, especially a system as complex as that of the Army recruiting process. In this section, we document those aspects of the recruiting process that we chose not to model for various reasons, most prominent of which is the phenomenom of diminishing returns for the amount of time invested. Further efforts into improvements to the model may address all or most of these issues.

1. *Face-to-Face Prospecting Times:* The difference between face-to-face prospecting times for high school students and graduates (as implied by the data in the questionnaires) was not captured. One important question should be answered before including this into the model. Is there a substantial variance in the quality of face-to-face prospects at high schools as compared to graduates? How is this difference reflected in the probabilities of the applicants appearing at a sales interview? Recruiters are unanimous in their opinion that applicants can be discriminated by their graduate status, but are not so unanimous as to the extent in which this can be done.

2. *Sales Interview No-Shows:* Based conversations with both recruiters and station commanders, it appears that there is a difference in quality between face-to-face and telephone-prospected applicants. Recruiters have indicated that appointments generated through face-to-face prospecting are less likely to become "no shows". The reason for this tendency is that a recruiter can visually screen applicants before approaching them, thus eliminating those who clearly do not meet Army entrance standards. The model does include as model parameters the probabilities associated with both forms of prospecting. However, the lack of data constrained us to assume equal loss probabilities when running the models and performing output analysis.

3. *Heavy Recruiting Station Turnover:* The model user is forced to choose one of two options. The first is to not include a warm up period for the simulation, which of course implies starting the model under empty and idle conditions. The other option is to include the warm up period, which consequently models a situation in which all recruiters have the same experience. In an actual recruiting station, however, there will almost certainly be a mix of recruiters with varying experience levels and different times-on-station.

4. *Changing Effectiveness of Recruiters*: As a recruiter gains more experience, he or she will most likely gain a corresponding degree of effectiveness. The simulation can only assume a constant degree of effectiveness in each task.

5. *Seasonality of Recruiting*: The ability of recruiters to generate appointments and put people into the Army is highly dependent upon the time of the year. Their major market, high school students and recent graduates, are more or less receptive to military recruiting efforts during different times of the year / school calendar. Certain events, such as Christmas break or graduation (when most students have solidified their post-high school plans), may restrict the availability of students to military recruiters. The model does vary the number of applicant

entities based upon anticipated seasonal effects.

6. *Station Commander Effect:* As mentioned in the introduction to Chapter 3, the model does not contain an explicit parameter that encapsulates station commander (SC) effect. However, it is possible to incorporate this factor by modifying task durations and loss probabilities across the board for each recruiter in the simulation. If the user should decide to incorporate SC effects, then he or she should develop a consistent set of criteria by which specific parameter changes can be related back to specific SC types that the user desires to model. An obvious avenue for further research would be develop an SC effect multiplier that could automatically change the values of multiple parameters. Such convenience may turn out to be quite costly, as it would require an involved statistical analysis of large amounts of historical data relating station performance to the perceived effectiveness of the station commander.

7. *Annual Leave and Training*: Every recruiter, as a member of the Armed Services, is allowed 30 days paid vacation during the year - as well as time off for national holidays. Moreover, it is likely that the recruiter will have to attend training at some point during the year. One of the assumptions the model makes is that the recruiter is available and working 52 weeks per year. Although the simulation does not explicitly include holidays as model parameters, the user may reduce the run-length to, say, eleven months in order to account for these additional downtimes.

8. *Other Duties:* In addition to the tasks associated with the Five Critical Recruiting Processes, there are a myriad of other events - both incidental and planned - that occur on a daily basis in a recruiting station. These events exist in such numbers as to preclude their explicit inclusion into the model. Some of the tasks not included in the model definition are: closing the loop, school visits, filling out school folders, daily performance review, cultivating centers of influence, van duty, community events, physical training, DEP functions, and training. They are, however,

implicitly accounted for by means of aggregation into categories which we denoted as Planning and Additional Duties. The user may vary the amount of time a recruiter will spend doing these activities on a daily basis.

4.3 Verification and Validation (V & V)

While designing and building a simulation, the model builder should constantly take steps to ensure that (1) the code works as anticipated and (2) the model output itself reasonably conforms to the output of the system being modeled. Verification and Validation (V&V), or as Osman Balci terms it, Verification, Validation, and Testing (VV&T), constitutes the means to these ends. The formal definition of these concepts in Balci's article on the subject[3, 1] :

Validation:	"... substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the study objectives."
Verification:	"... substantiating that the model is transformed from one form into another, as intended, with sufficient accuracy."
Testing:	"... demonstrating that inaccuracies exist or revealing the existence of errors in the model."

For our purposes, we shall include our discussions of Testing in each of Validation and Verification. This is due to the fact that the Testing step is not meant to be accomplished with a mutually exclusive purpose in mind, but rather as a means to accomplish V&V as effectively and efficiently as possible. V&V may each be accomplished in a variety of ways that vary according to their level of rigor. Table 3 gives a breakdown of the V&V tasks accomplished during the building of the Army Recruiting Model[10, 302] .

Tasks	
Verfication	**Validation**
Writing and Debugging	Conversations With Experts
Code Verification by Multiple Individuals	Observations of the System
Checking Output	Existing Theory
Tracing	Experience and Intuition
Animation	Turing Test
Use of Simulation Packages	Model Calibration

Table 3. V & V Activities (Based on Law & Kelton, Chapter 5)

We shall henceforth use this outline to structure the discussion of the V&V tasks that we performed in support of our model-building efforts. Sargent, in his article on Verification and Validation [23, 57] , provides an diagram (see Figure 30) that clearly summarizes the preferred model-building process.

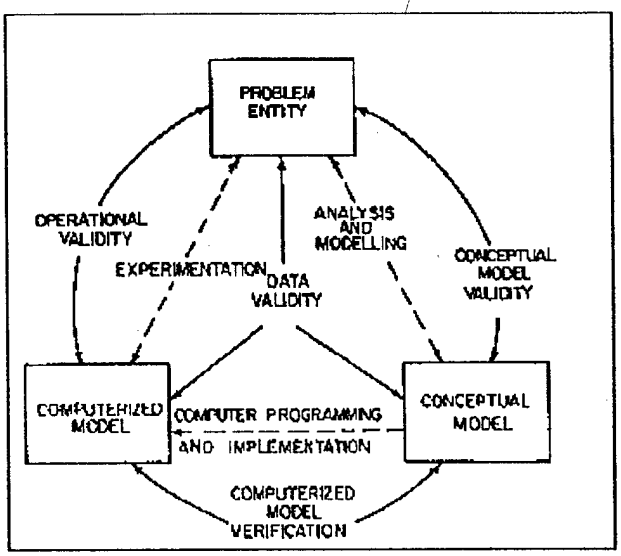

Figure 30. Steps in Constructing a Valid Model (Sargent)

One can never emphasize enough the necessity of accomplishing V&V tasks throughout the life cycle of the model, from the very first studies of the system to actual distribution of the model to the end-user. Indeed, it is too difficult, and often hopeless, to initiate V&V during the course of code production or (worse yet) after a simulation prototype has been developed. Therefore, every

effort has been made to continuously apply the tasks listed in the table during each major step of the Army Recruiting simulation study.

4.3.1 Verification

The verification step is characterized by its goal of producing a model that *functions* correctly. Many of the problems occur during the actual writing of code, and hence a good proportion of the activities tend to focus on the application of accepted programming techniques. This section gives a description of the verification efforts accomplished in support of the study.

4.3.1.1 Writing and Debugging

The modularity of the Army Recruiting process model lends itself well to programming in a modular fashion. For instance, there was no great leap of logic from the abstract concepts of recruiting steps such as "Prospecting" or "DEP Sustainment" to the writing of the corresponding object methods. Therefore, the simulation model could be developed gradually by writing modules one at a time, debugging, and then proceeding to write and incorporate the next module. Since the MODSIM implementation, in its latest version, is comprised of over 10,000 lines of code, this method of coding drastically reduces the complexity of the debugging task. Since the SimProcess model was designed using the same modular structure, coding and debugging were accomplished in exactly the same, gradual fashion which characterized the MODSIM model-building process. The byword "code sparingly and compile frequently" holds special significance for those who have ever been involved in the development of large simulation models.

4.3.1.2 Code Verification by Multiple Individuals

One of the most significant barriers to verification stems from the myopic viewpoint of the simulation developer. Very close involvement with the simulation tends to restrict impartiality, thereby blinding the developer to serious errors in the structure of the simulation. It is advisable to con-

duct a structured walk through the program with other members of the project. A similar structured walk through was conducted on the simulation with two experts trained in the use of SimProcess and MODSIM in order to ensure that such misconceptions did not creep into the design. Since the two authors built the intial MODSIM model together, they were able to constantly cross-verify each other's code, which grew out of the necessity of incorporating modules written by one or the other person. Each developer needed to keep abreast of every modification accomplished by the other so as to make the modules fit as seamlessly as possible into the simulation.

4.3.1.3 Checking Output

Examination of the program output can indicate the existence of program "bugs" if they are present. Since the operation of the simulation is so dependent on the generation of random variates, making sure that the output streams correspond to the desired distributions is a crucial verification task. A good predictor of the reliability of the simulation is the examination its response to the inclusion of parameters that represent extreme cases in the actual system. Analysis of the output will often reveal if extreme cases are not dealt with correctly. For example, if the number of recruits exceeds the bounds of believability in response to, say, the largest amount of prospecting time that was observed in a recruiting station, then the model has not passed the test for robustness.

Time stamping in SimProcess gives the user the ability to observe the passage of simulation time between any two points in the program. When an entity enters activity an activity designated as the starting point of interest, it is assigned a time stamp. Upon entering the designated exit activity, the time between the points is then determined. The standard output report provided with the SimProcess package gives the average simulation time observed for these entities between the two points of interest, which proved extremely useful in validating output relating to the time spent

accomplishing major recruiting tasks. It also served well as a tool for verifying that the random variates used for time delays conformed to the appropriate probability distributions.

4.3.1.4 Tracing

The output alone will usually not be sufficient grounds to affirm or deny that a model is working correctly. In order to be reasonably sure that the simulation operates as it should, it may become necessary to know the values of the various state variables, including the contents of the FEL and other queues [10, 303] . This will allow the modeler insight as to how the various elements of the program (i.e. algorithms) actually function. Tracing can be done as a response to errors in the program that appear at runtime or otherwise evince themselves as anomalies in the output. However, since many errors do not reveal themselves in obvious ways, one must also use tracing to verify the correctness of each part of the program, regardless of whether or not an error was perceived.

Extensive program tracing was done during the course of the development of both the Sim-Process and MODSIM models to convince the authors that everthing was working as it should. Output statements were sprinkled liberally in the code, with the result that, at runtime, one could actually follow in great detail the path of an applicant through each method. It is also possible to have SimProcess output values to the screen, which was extremely useful in situations where the program flow could not be discerned from watching the animation.

4.3.1.5 Animations

The visual nature of SimProcess lends itself readily to the animation technique of verification. Once animation is enabled, one can view entities progressing from the generation node, through the various program constructs, and on to disposal. In addition, entity counts are available at each node when the animation mode is activated. Therefore, when using program animation in conjunction with the tracing methods discussed above, we were able to ascertain whether or not entities were

proceeding along the correct paths through the simulation. Entity counts that indicated the presence of excessively large queues were an indication of delay-time distributions being set incorrectly. Simply stated, animation lends itself readily to the visual method of verification which can, in most cases, be more compelling than simply analyzing output.

Animation proved useful in confirming a malfunction of the SimProcess software. We had become suspicious of the pre-packaged ability to run warm-up periods for single replications due to the excessive output being generated by the simulation. Upon observing the animation and entity counts at each of the Contract nodes, we did discover that, indeed, entities from the warm-up period were being included in the final report. Recall that the stated purpose of this warm-up is to remove the effect of the initial transient bias from the simulation in order to ascertain the output at steady-state. Because the extra entities from the warm-up period were being included, we developed a work-around that disposed of these entities manually. This resulted in only the trial-run entities being included in the output summary, which is, of course, the desired effect.

4.3.1.6 Use of Simulation Packages

As mentioned in Section 3.2, simulation packages such as MODSIM and SimProcess greatly reduce the effort needed to construct a simulation. All such packages include pre-fabricated methods that control functions necessary to the operation of a simulation program - such as the FEL, for instance. Contrast this to the situation of a model builder using a general-purpose language such as C++, who must develop all such simulation mechanisms from scratch. This provision of ready-made overhead results in a smaller volume of code, which in turn reduces the risk of error that seems to increase exponentially with the lines of code. On the other hand, the user assumes the risk that the simulation package itself contains bugs that were incorporated by *its* designers. Although we found numerous examples of this latter case, we could readily compensate by developing relatively

simple work-arounds. Our experience thus dictates that, at least for this model, the benefits - namely a quick development time and ease of verification - vastly outweighed the risks associated with the use of the software.

Over the course of study, we used three different versions of the SimProcess software. The quality of the simulation package improved with each release, as evidenced by the fact that several errors we observed in previous releases were eradicated. As an example, one of these earlier versions of SimProcess demonstrated the unusual trait of allowing the user to define delays using undeclared variables, an unheard of characteristic in a such a highly type-sensitive language. It was thus necessary to make sure that delay times assumed correct values by using such trace methods as viewing output statements and using time-stamps to record the amount of simulation time that elapsed during the execution of certain processes[18]. Many more errors are sure to be present in this and future versions, and so we must accept the possibility of model output inaccuracy as a calculated risk.

4.3.2 Validation

While a simulation may seem to function correctly if simply taken at face value, one may still question its ability to reproduce results with reasonable accuracy or, more importantly, to predict future behavior. The goal of the Validation step is to make sure that the model reflects the real-world accurately enough so that effective decisions can be made on the basis of model output [10, 301] . However, it is not feasible to assume that every aspect of the actual system can be incorporated into the model, or to expect a one-to-one correspondence between model and system output. Recall that the goal of a simulation study is to produce a model that captures the essence of the system, rather than "copying" the system. It must be with this goal in mind that we approach validation, since we must otherwise face the hopeless task of accounting for every discrepancy that could possibly

[18]For yet another example of a SimProcess flaw, see the previous section on animations.

occur. The following subsection covers some of the basic Validation techniques outlined in Chapter 5 of Law and Kelton.

A word must be said about model calibration, which we listed as a Validation activity. Since a goal of the validation step is to try to make the simulation behave as closely as possible to the system being modeled, one must consistently modify - or calibrate - the simulation in order to improve the output. This step was, for the authors, an immediate and natural reaction to any valid suggestions for improvement that we received from any source, including recruiters, instructors, USAREC experts, as well as actual system data. Due to the all-encompassing nature of this task, one may correctly assume that this task follows implicitly from any of the tasks we will discuss below.

4.3.2.1 Conversations with System Experts and Observations of the System

A fundamental tenet of validation is embodied in the following statement: "When developing a simulation, an analyst should not work in isolation. Instead, throughout the simulation effort, the analyst should work with system experts intimately familiar with the system to be simulated" [10, 308] . This is what Sargent [23, 57] refers to as *face validity*. In the case of the Army Recruiting Model, however, Army regulations, rather than human experts, provided the catalyst for the construction of the process flow diagrams. These diagrams initially reflected the official version of the recruiting process, but were later modified (after conversations with various recruiters) so as to make them more indicative of how recruiters actually practiced their trade. The recruiter-instructors at the U.S Army Recruiter Retention School at Fort Jackson, S.C. also provided valuable suggestions for improvements. Needless to say, the information that we gleaned from recruiting classes themselves substantially influenced the composition of the process flow diagrams and the resulting simulation models.

The detailed process flow was in turn used in conjunction with descriptive data collected from recruiters (over five successive visits to the Huber Heights RS) to form an aggregated pre-simulation model[19]. The aggregated model was then examined by both recruiters and recruiting personnel, critiqued, and then molded into its final version. It was this version that provided the structure of the MODSIM model, which took approximately two months to construct. The experience gained from the design and construction of this model proved to be extremely helpful in the subsequent development of the SimProcess model, which incorporates most of the assumptions of its predecessor, albeit with many improvements to model fidelity. The SimProcess version underwent the same scrutiny by recruiters from various recruiting stations in the Third Brigade. Their suggestions were likewise included in further revisions of the physical simulation model that we eventually provided to USAREC.

Descriptive data collected from the recruiters was not gathered in a haphazard fashion, but rather in the form of a questionnaire (See Appendix A). This data provided a database from which to generate input for the simulation based on responses to questions about average times and frequencies with which recruiting tasks were performed. Many recruiters expressed that they had difficulty in asking questions about task completion times because of their high variability. Task priority was another problem because, in practice, recruiters changed priorities based on the context of the situation in which the task is prioritized. Some of the factors that play a role in influencing priority assignment are differences in schedule, urgency of one particular case over another, or station commander leadership style. These difficulties led us to consider the questionnaires to be a guide, rather than a rigorous collection of solid data. Nevertheless, recruiter responses greatly assisted us in validating model by providing us with a range of correct input - and output - data.

[19] "Aggregated" refers to the act of using assumptions to represent groups of activities, thus simplifying the original process flow descriptions.

4.3.2.2 Existing Theory

Processes in a simulation that correspond to certain theoretical systems should be modeled in terms of these systems. Arrival times for a queueing system, for instance, should most likely be implemented using random variates generated from an exponential distribution. As discussed in Section 3.1, input distributions were based on how certain elements of the recruiting process correspond to these theoretical constructs. The distribution for walk-in interarrival time was modeled as an exponential random variable because of the obvious similarity to the arrival of customers to a bank queue, say. The decision to use the triangular distribution for a host of other input distributions was made, on the other hand, to avoid the lengthy process of evaluating literally hundreds of sets of input data to determine the correct distribution. Moreover, the recruiting process does not, by its very nature, produce the necessary data to make reasonable assessments as to the nature of these input distributions.

4.3.2.3 Turing Test

The model builder may accomplish what is known as a *Turing Test* once output data is available from the simulation in question. The technique simply boils down as a test to see whether or not the analyst, using the simulation output data, can "fool" a system expert into believing that the data was collected from the actual system. The simulation output data should first be put into the same format as data collected from the system. If the expert is able to differentiate between the two sets of output data, then the analyst should ask the expert's advice concerning how to improve the simulation. "Fooling" the expert is, of course, no guarantee that one's model is valid, since all cases may not have been accounted for. The test is, nevertheless, a good way for the model builder to discover inconsistencies within the simulation output that may be more apparent to a system expert [10, 312].

During one of the data collection visits, an assistant station commander was provided with output from one replication of the model. The run length used for this output was a one year period starting with empty and idle conditions. The recruiter stated that the output appeared realistic based on the number of appointments generated by the simulated recruiter. The number of appointments, however, was less than what the assistant SC would normally expect of a recruiter. The cause of this problem was traced to an unrealistically short period of simulation time allotted to the recruiter for the purposes of prospecting. Once the correction was made, the output closely conformed to the recruiter's stated expectations. Subsequent validation efforts concentrated on the comparison of the simulation model output and historical data.

4.3.2.4 Validation During Simulation Runs

At times, validation activities are prompted by sources other than expert opinion. For example, our investigations into using the Welch approach to eliminate initial bias conditions inadvertantly revealed data that appeared to be inconsistent with that of an actual recruiter. As we discussed in the section on Animations (4.3.1.5), the warm-up period in SimProcess is intended to provide the user with a means to eliminate transient bias. In search of a truncation point to remove initial bias from the system, output warm up lengths from 5 to 12 months were investigated. The results from these trials showed a surprisingly linear relationship between warm-up lengths and the number of contracts output from the system, an observation which immediately led us to question the results. Regardless the fact that we later found the SimProcess warm-up capability to be flawed, the output analysis might have indicated a serious design flaw that hitherto went unnoticed. We again refer the reader to Section 4.3.1.5 for a discussion of how we finally managed to resolve this problem.

Using the latest revision of the SimProcess model with a one-year warm up period and one-year simulation run time, thirty replications of output were generated. A comparison between this

data and historical data gathered from visits to recruiting stations did not reveal any significant differences.

Chapter 5 - Output Analysis

5.1 Approach to Analyzing Army Recruiting Output

We now present an experimental design and analysis of the output of the Army Recruiting Model for a particular test case. As stated in the introduction to this study, we are not interested in the precise numbers of recruits produced by the model, but rather how *sensitive* the recruiting process is to the modification of various parameters. Such knowledge will assist a policy-maker in making decisions based upon the predicted behavior of the system when a change or combination of changes is imposed upon it. Hence, the end-user should not be solely focused on ways to increase the output of the model. This near-sighted approach does not take into account the random behavior which the real system exhibits, and the model itself simulates. Instead, one should observe the sensitivity of the output with respect to various parameter settings, as well as how certain factors interact to produce the observed model behavior.

In this chapter, we will present the results of a sensitivity study involving three model parameters to demonstrate the possibilities for future investigations of this nature. The representative model parameters are the amount of collateral time imposed on the recruiter, the amount of time allotted to prospecting, and the amount of processing time allowed during the work-week. An analyst using these criteria might, for instance, want to see the effect on model output of decreasing collateral time and prospecting time simultaneously. One can similarly investigate all other combinations by utilizing a factorial design, and then determining how well one can predict model behavior based on these design points. Such analysis will not always provide hard answers, but should nevertheless offer valuable insight into how the recruiting process actually works.

5.1.1 The Experimental Design

We eventually settled on a 2^3 full-factorial, single center-point design based on the factors of prospecting time, collateral time and processing time (as mentioned above). The purpose of the center point is to test for tendencies towards curvature in the data. The measurement of effectiveness, of course, is the number of contracts each recruiter - and hence the entire recruiting station - produced. There was, indeed, a rationale behind the using the three factors that we did, as we shall now attempt to explain. Prospecting time was chosen because we were interested in testing the hypothesis that it wields a major amount of influence on the output of enlistees. Collateral time is notable for the fact that it encompasses time spent on all activities that do not appear in the model, to include special duties, breaks, van duty, and the like. What alerted us most to this particular factor was that USAREC officials repeatedly expressed an interest in understanding how these non-process-related tasks affect recruiter efficiency. Finally, the amount of available processing time distinguished itself because it reflects differences among stations with regard to how they structure their work-week. As an example, many of the recruiters we interviewed work on completing enlistment packages and other administrative duties for an additional four hours on Saturday so as to free time during the week for more "important" tasks. For the remaining model parameters (held constant over the nine design points), we utilized actual task-completion data provided by USAREC, in addition to data collected on visits to recruiting stations.

Although the levels of the individual factors found in the experimental design are representative ranges that exist in the actual station data, the combination of levels used in the sensitivity study may not necessarily exist in practice. We were alerted to this possibility by the experience that we had running the simulation corresponding to design point 7. As Table 4 shows, the point represents high collateral and prospecting, but low processing, which tells us that these recruiters will generate a larger number of recruits, but nevertheless will have a smaller proportion of their daily schedule

available for DEP sustainment. As a result, the program issued a warning message stating that applicants passing through DEP Sustainment did not attend at least half of their scheduled meetings. This state of affairs is yet another indication of the model's real utility, which is not in charting specific courses of action. The extremity of the situations represented by the design points serve rather to point out to which combinations of factor the system will exhibit a greater or lesser amount of response.

Design Point	Pattern	Prospecting Time	Collateral Time	Processing Time
1	− − −	−1	−1	−1
2	− − +	−1	−1	+1
3	− + −	−1	+1	−1
4	− + +	−1	+1	+1
5	+ − −	+1	−1	−1
6	+ − +	+1	−1	+1
7	+ + −	+1	+1	−1
8	+ + +	+1	+1	+1
9	0 0 0	0	0	0
	Low Level	**15**	**10**	**0**
	High Level	**25**	25	**4**

+ indicates the factor in that position in the matrix is at its high level

- indicates the factor in that position in the matrix is at it's low level

Table 4. Design Points

We conducted two experiments using this design, each of which consisted of 30 replications of runs lasting 2 years. The two experiments are distinguished by the fact that the first takes into account all output while the second excludes transient bias from each replication. Since we eventually decided to truncate the data after one year of simulated time, the first set of trials covers actual runs of two years. The latter, on the other hand, consists of runs lasting only one year. All of our measurements are based on the magnitude of the lowest-common-denominator, namely the set of 30 one-year trials, which entailed averaging the results from the set of two-year trials.

5.2 Results

The nine runs corresoponding to each design point was accomplished on nine Pentium machines running SimProcess simultaneously. In actuality, many more runs were needed due to the skewed results produced by the SimProcess warm-up bug (c.f. preceding section) and locked-up machines. The length of each run varied, of course, with the speed of the processor; for instance, a 233 MHz machine completed a run - that occupied a 133 MHz machine for over four hours - in just under two hours. The cumulative time spent on processing time was therefore well in excess of eighty hours. We will start the discussion with a brief look at a comparison between the output of both experiments through the use of descriptive statistical methods.

5.2.1 Comparison of the Mean Number of Contracts Produced

When comparing the two systems, we were mostly concerned with the magnitude of difference between the number of contracts produced on both the station and recruiter levels. The following charts and accompanying graphs summarize the output of the two thirty-replication experiments.

Without Bias					With Bias
Design	Recruiter			Recruiting	Recruiting
Point	1	2	3	Station	Station
1	6.133	13.333	23.767	43.233	36.117
2	6.533	12.667	25.533	44.733	36.467
3	5.933	12.567	24.167	42.667	34.383
4	6.8	12.5	23.3	42.6	34.483
5	9.2	19.867	36.6	65.667	53.317
6	9.933	20.4	38.333	68.667	55.267
7	8.933	17.367	31.4	57.7	46.167
8	9.367	17.433	31.433	58.233	47.183
9	8.867	14.967	29.267	53.1	43.483

Table 5. Simulation Output (Reps = 30, Time = 2 Years, Bias Removed)

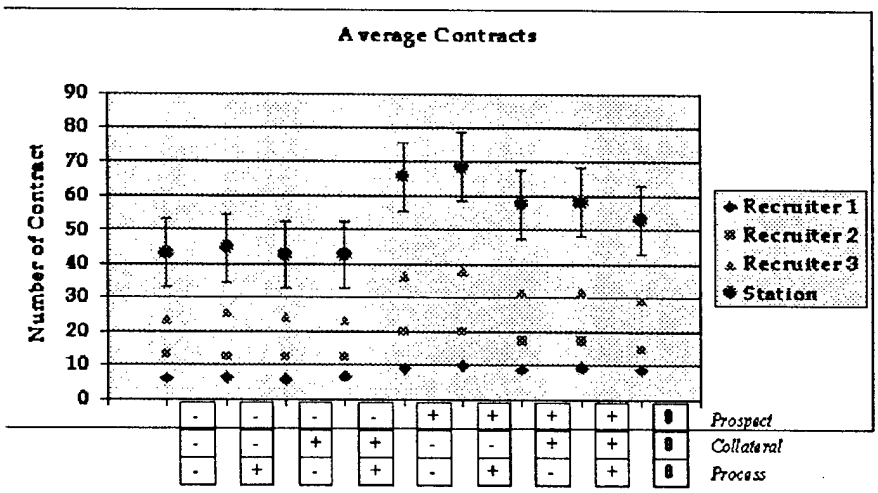

Figure 31. Graphical Presentation of Results with Bias Removed

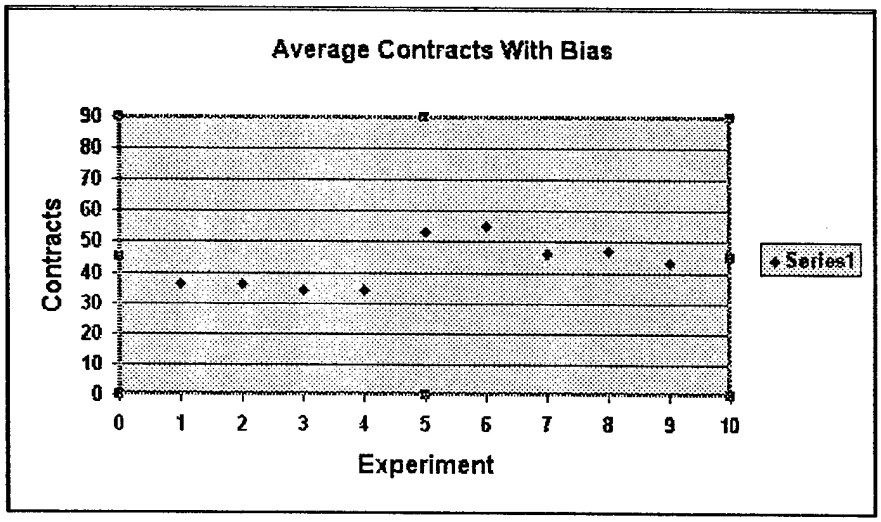

Figure 32. Graphical Presentation of Results with Bias Included

What becomes immediately apparent from the data is the fact that the averages are noticeably larger in the system with bias removed. This effect can be attributed to the fact that, at the start of the simulation run, there may already be a number of applicants going through DEP Sustainment, whereas none exist here for the experiments run without the warm-up period of one year. This

situation is more in line with what we would expect in a recruiting station with a moderate amount of recruiter turnover. In other words, starting from empty and idle conditions may or may not be appropriate based on the TOS (Time on Station) for recruiters in a particular recruiting station. The higher the turnover, the more closely we would expect the data with bias to match reality.

In theory, the use of the method of common random numbers should produce a relatively high amount of correlation between the two systems. We mention this because, in fact, the same random number streams were used from design point to design point and also between experiments. The data corroborated this expectation in that the correlation between the two systems turned out to be around 0.9985. This high degree of correlation provides an adequate basis for constructing a 95% confidence interval using a paired t-test. The two-sided confidence interval itself for the estimation of the average difference between the number of contracts in System 1 (Bias) and System 2 (No Bias) $\bar{C}$ over R differences is given by

$$\bar{C} \pm t(0.05, R-1) * (\frac{S_C}{\sqrt{R}}) \tag{5.1}$$

where $\bar{C} = \frac{1}{R}\sum_{r=1}^{R} C_r$, C_r is the difference between the corresponding number of contracts in the two systems, and $S_C^2 = \frac{1}{R-1}\sum_{r=1}^{R}(C_r - \bar{C})^2$ (the sample variation of the differences C_r). For an explanation of the Paired t-Test, we refer the reader to Banks, Carson, and Nelson[7, 482] . To perform the test, output averages for each of the nine design points were aggregated by experiment, thus forming in two columns of $R = 9$ observations. The resulting confidence interval around the mean difference of the two systems turned out to be 9.97037 ± 1.685976 at a $\alpha = 0.05$ significance level. Since the interval does not include zero, there is evidence for a statistically significant difference between the biased and unbiased systems. Since station output ranges from 34 to 69 - not large in comparison to the average difference of 10 - a difference of 8 or 9 in either direction amounts to a practical difference as well.

5.2.2 Regression Analysis

The next step involves fitting a model to the data for System 2 and seeing how well one can predict output based on the three design parameters. After performing the analysis using the statistical software JMP, we obtain the following parameter estimates given in Table 6. The final column in the table is the significance calculated from the observed t-ratios, which gives the probability that an observation corresponding to greater t-value would occur, given that the null hypothesis is true. Since

$$H_0: \text{ Regression Parameter} = 0 \qquad H_a: \text{ Regression Parameter } \neq 0 \tag{5.2}$$

the low probability suggests that the parameter in question is not equal to zero for any reasonably small significance level. The parameters significant enough to be included in the model are therefore those that do not contain Processing, namely Intercept, Prospect, Collateral, and Collateral*Prospect. The other parameters were removed from consideration, and another standard least squares regression was performed. Table 7 gives a summary of fit for the metamodels based, respectively, on the lack or presence of Processing (for the purposes of comparison):

Parameter	Estimate	Std Error	t*	Prob $> \lvert t^* \rvert$
Intercept	52.95556	0.397147	133.34	< 0.0001
Prospect	9.629167	0.421238	22.86	< 0.0001
Collateral	-2.6375	0.421238	-6.26	< 0.0001
Processing	0.620833	0.421238	1.47	0.1417
Collateral*Prospect	-1.9625	0.421238	-4.66	< 0.0001
Processing*Prospect	0.2625	0.421238	0.62	0.5337
Processing*Collateral	-0.50417	0.421238	-1.2	0.2324
Prospect*Collateral*Processing	-0.1125	0.421238	-0.27	0.7896

Table 6. Parameter Estimates of Model Fit to Unbiased Data

Summary of Fit		
	With Processing	Without Processing
R^2	0.69159	0.686805
R^2 Adj	0.68335	0.683273
Root MSE	6.525787	6.526576
Mean of Response	52.95556	52.95556
Observations	270	270

Table 7. Comparison of Parameter Estimates of Model With and Without Processing

The R^2 value, or the percentage of variance explained by the model, is virtually the same in both cases, which proves that including Processing in the model is probably not worth the effort. We henceforth omit Processing from further consideration in our regression analysis.

5.2.3 Effect Screening

Given the significance of the Collateral-Prospect parameter in the suggested model, it would seem worthwhile to investigate the nature of this interaction further. Interaction plots do indeed reveal a noticeable effect on the output with regard to the high and low values of both factors, as seen in Figure 33. The interpretation of the graph is fairly straightforward in that, the greater the slope of the line, the more interaction exists between the two factors in question. This realization makes apparent that the number of hours spent performing collateral duties has a greater effect on the number of contracts produced when prospecting is at the high level. In terms of an actual recruiting station, this means that collateral duties have a greater impact on those recruiters who devote the most time to prospecting, which, as we have mentioned previously, seems to be the driving force behind the model output.

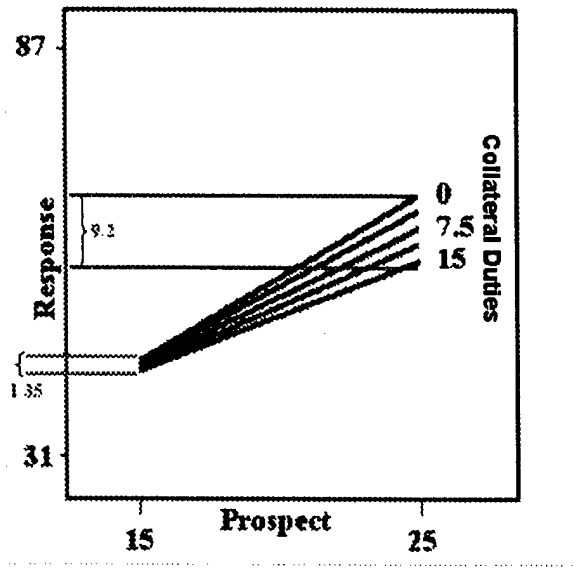

Figure 33. Collateral-Prospecting Interaction Effect

Further screening for significant effects was performed using a normal quantile plot, which is constructed by plotting the normalized parameter estimates against the normal quantile score. The method[20, 104] is useful because of its effective visual representation of effects that are either negligible or significant. Those normalized parameters β_i that are not significant to the regression and with $E(\beta_i) = 0$ tend to fall along a straight line on the plot. Significant effects will have non-zero means and are non-collinear. Figure 34 below thus indicates that the parameters corresponding to Prospect, Collateral, and Prospect * Collateral are significant.

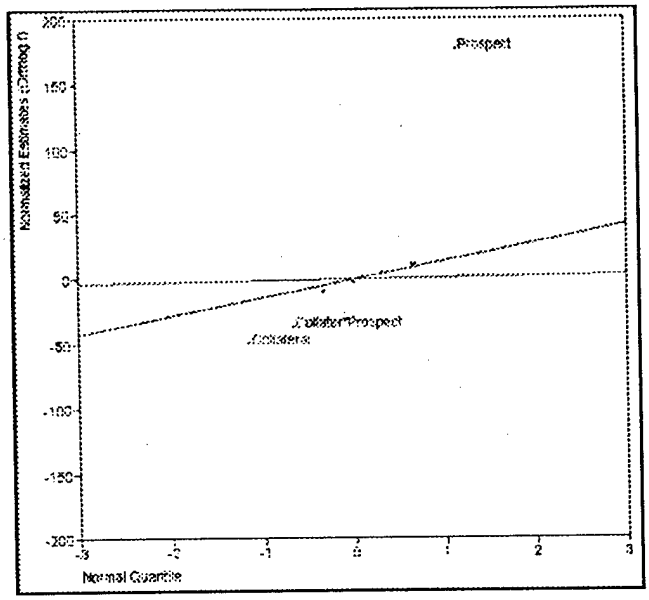

Figure 34. Significant Effects Using a Quantile Plot

Using only the significant factors ascertained from the previous analysis, a least squares linear regression was performed with the parameter estimates given in Table 8.

Parameter	Estimate	Std Error	t*	Prob $> \lvert t^* \rvert$
Intercept	52.95556	0.358792	147.59	< 0.0001
Prospect	9.629167	0.380556	25.30	< 0.0001
Collateral	-2.6375	0.380556	-6.93	0.0010
Collateral*Prospect	-1.9625	0.380556	-5.16	0.0036

Table 8. Parameter Estimates of Model With Negligible Effects Removed

The existence of data points corresponding to multiple replications of the model provided the means by which we could test the lack of fit in the model. From this analysis, we conclude that there is no evidence of curvature over the design region. Table 9 below indicates the basis for this conclusion; notice that we would reject a lack of fit hypothesis under the assumption of a $(1 - 0.9046)$ level of significance.

Source	d.o.f.	Sum of Squares	MSE	F*	Prob>F
Lack of Fit	1	0.0234722	0.02347	0.0163	0.9046
Pure Error	4	5.7694444	1.44236	–	**MaxRSq**
Total Error	5	5.7929167	–	–	0.9931

Table 9. Parameter Estimates of Model With Negligible Effects Removed

As one can see from the Table 10 below, the least squares regression performed on the mean response at each design point explains almost all of the variance found in the response variable.

Summary of Fit	
Response	Y
RSquare	0.993054
RSquare Adj	0.988887
Root Mean Square Error	1.076375
Mean of Response	52.95556
Observations (or Sum Wgts)	9

Table 10. Summary of Regression Performed with Significant Effects

In summary, the metamodel with the parameters given above seems to be adequate to predict average responses when provided with prospecting and collateral times. Processing, on the other hand, does not seem to play a major role in determining the output of contracts. It should nevertheless not be discounted entirely, since it may, in fact, interact significantly with respect to other output criteria. We also will mention that the residuals do appear to conform to a normal distribution according to the Shapiro-Wilk test for normality. This provides further evidence for the appropriateness of constructing a metamodel on the basis of the experimental design we have used in this chapter.

5.2.4 Effect of Waivers

Recruiters often complain about the inordinate amount of time needed to complete waiver applications, which of course impacts the time available to do other tasks. Although several stations that we visited indicated that they processed waiver packages very infrequently, at least one did

mention that the number of waiver applications was significant enough to negatively affect their production. This observation provided the motivation necessary to perform a simple experiment concerning the effect of waivers on recruiter production. The experiment itself consisted of running the model used for design point one for thirty replications (with a warm-up period of one year) under both original conditions and conditions in which there were a heightened percentage of applicants requiring waivers. The first category of applicants requiring waivers, the moral waiver candidates, were increased from 7% to 15% of the applicant pool while those requiring medical waivers were increased from 5% to 15%. The output from both sets of runs were then compared in order to ascertain whether or not there existed a significant difference in the number of contracts produced.

For the measure of correlation between the two sets of runs, we obtained an exceptionally low value of -0.08845, which ruled out a paired t-test approach. The resulting confidence interval at $\alpha = 0.05$ was 2.8333 ± 2.92, which includes zero, and therefore indicates a failure to reject the null hypothesis. Hence there is no evidence for a statistically significant difference between station outputs when the number of waivers is increased. This seems to corroborate the assertion that prospecting alone seems to bear the most significant amount of influence upon the number of contracts. However, as Figure 33 seems to indicate, a greater response to waivers may be found when the recruiter is prospecting at the highest level (25 hours per week). This would suggest further experimentation with this and other parameters at varying levels of prospecting in order to obtain a more complete picture of the effects of these secondary factors.

Chapter 6 - Conclusion

We conclude the simulation study of the Army Recruiting Process with a few insights about what we learned and also what an analyst using the model could learn in future investigations. The knowledge that we obtained during the course of the simulation study is not just limited to the technical processes of model development and output analysis, but also encompasses the entire scope of Army recruiting, to include the human aspect of the job. In these last few sections, we will summarize the results of the study and propose ways in which this knowledge can be used to assist planners in optimizing the performance of recruiters, and hence recruiting stations. We feel that any improvements to output must be accomplished with a corresponding change in the way recruiters organize their time. This can only benefit the recruiters themselves, since it is they who must bear the daily burden of accomplishing the tasks set for them by the powers-that-be.

6.1 Summary

The first few months of the simulation study was focused mainly on the gathering of data and relevant information about the recruiting process. While we could not realistically hope to personally experience the day-to-day routine of the recruiter, we did set a goal to understand the nature of the critical tasks, their true implications for the process itself, and how one might go about modeling these processes. A fairly wide assortment of detailed process flows resulted from the investigation of the recruiting stations, as well as much interesting and useful anecdotal data. The anecdotal data gave us insight into how recruiters used their experiences and common sense to help them "make the mission", so to speak. Although the information could be useful to planners in and of itself, such anecdotal data actually assisted in designing the model, since the regulations leave the determination of daily schedules to the station commanders and recruiters.

The next step was to actually design and build the computer simulation model. As mentioned in Chapter 3, the proper approach to designing a simulation is to include only that level of detail sufficient to provide accurate output. Of course, the determination of what is "sufficient" is what we call the art of simulation. If the model fulfills the stated goal of successfully assisting decision makers in setting policy, then we have met the requirements of the simulation study.

There are a large number of analytic and programming tools that one can use to build a decent computerized model, such as the techniques of input analysis and simulation development packages such as SimProcess and MODSIM. The process of verification enabled us to ensure that the code worked properly once written. Even more importantly, the model design itself was kept accurate in the face of changes to the process[20] by means of continual validation of both the model design flow and the simulation output.

As Law and Kelton and others have stressed, validation should encompass every aspect of the modeling process, from the study of the system through the entire lifespan of the simulation model[10]. For the Army Recruiting Model, every attempt was made to incorporate this philosophy into the simulation study. Recruiters and USAREC experts were consulted during the formulation of the detailed process flows, which contain information about the task composition of each of the five critical recruiting processes. Both historical data provided by USAREC *and* field data collected from station recruiters was used to generate the input files for the simulation, as well as to validate its output. A similar validation process was applied to the model design flow which the authors subsequently used to build the computerized simulation.

An even more rigorous application of both verification and validation was accomplished both during and after the completion of the computerized model. Cross-checking of the code by the

[20]The changes referred to here are either the result of actual policy shifts or the obtainance of hitherto unknown information.

authors and others involved in the project served to reduce the number of implementation errors (which may, in turn, have lead to validation problems). The resulting output of pilot runs of the verified model was then compared to historical data when the same input parameters were used. Constant calibration of the model eventually narrowed the gap between the simulation and the actual system, thus providing the degree of confidence needed to perform trial experiments.

With a sufficiently validated model in hand, we proceeded with investigation of model output to see if we could draw some useful conclusions about the sensitivity of the recruiting process to changes in certain parameters. It was mentioned previously that the purpose of this investigation was to explore the uses of the simulation rather than formulate any absolute truths on the basis of the model output. To this end, we chose three parameters that USAREC analysts stated were crucial measures of recruiter effectiveness: the time available for prospecting, the time available for processing, and the time devoted to additional (collateral) duties. The simulation was run for each point of a 2^3 experimental design (plus center point) and a metamodel was obtained from the output, which was the number of contracts written for the station. The large proportion of variance explained by the model suggested that curvature did not play a significant role in the prediction of average contracts. Moreover, we found that the output was sensitive to the factors of prospecting and collateral duties, as well as to the interaction between them, but not to processing. The real driving force behind the process, however, was the amount of time devoted to prospecting. Collateral tasks have the effect of interfering with the performance of duties related to the Five Critical Processes, and so may have a significant influence on the number of contracts that a recruiter can produce during a period of time. The final factor, processing, can only be accomplished once a recruiter has obtained appointments, and so must rely on the inputs provided by prospecting and walk-ins!

6.2 Recommendations

The intended purpose of the Army Recruiting Model is to provide decision-makers with the means to assess the effect of station-level issues at more strategic levels, and vice-versa. The rationale underlying this objective is that the ability of USAREC to meet its goals is based upon the performance of individual recruiters in the many recruiting stations around the country. It is hoped that the quality of life as well as the production of recruiters might be improved by model information gleaned from either our results or that obtained from future uses of the simulation. In this way, the efforts of the development team, the many recruiters who participated in the study, and USAREC itself will be justified.

The possibilities for the future use of the model are quite large due to the degrees of freedom permitted the user in the number of parameters that can be set. Although we investigated the effect of three factors, many more factors as well as their combinations can be used as the criteria for subsequent analysis. For instance, one may vary the percentage of waivers accrued in a particular station or see what the effects of a high DEP loss might be on the system. An analyst may also choose alternate experimental designs that may reveal different aspects of the model output data. For instance, 3^k experiments such as those that involve the Box-Behnken design described by Myers and Montgomery [20, 318] may be used to address the possible existence of second or higher-order terms.

Another important issue that should eventually be explored is the search for a way to eliminate initialization bias, if this is at all possible to achieve. We chose a one-year truncation point in our simple experiment because of the significant statistical difference between the biased and unbiased output. However, we could not conclusively show that the average output thus achieved was a reasonably accurate estimator of the true mean. Given sufficient time to run the model for long periods of time for a sufficiently large number of replications, a future researcher may apply smoothing

techniques or truncation heuristics in order to investigate the effects of various truncation points upon the measure of effectiveness in question.

When the simulation study was first conceived during the summer of 1997, the authors planned to perform an analytical study to determine how to incorporate the effects of station commander, area demographics, and station performance into the Recruiting Process model. We were not able to perform such a study due to the large-scale data collection effort required for what we envisioned as a multivariate analysis problem. Given sufficient data points, one could determine the dimensionality of the data set and if required, perform a factor analysis to ascertain the true factors that motivate trends within the data. The resultant effects may then be added to the model as parameters that influence other model parameters through direct means, such as multiplication, or by some other transformation defined within the simulation. Although these factors are already implicit through the setting of parameters within the existing model, allowing the user to define the overall characteristics of a station by modifying values for these three characteristics will greatly convenience the user of the model. In addition, the added insight that this study would impart more than justifies the level of effort needed to collect the data and perform the analysis.

Although the model was initially commissioned to address 3rd Brigade issues, the homogeneity of recruiting policies across the command may well support the notion of its applicability to the problems of other brigades as well. We also mention the fact that multiple versions of the model running simultaneously may be used to simulate output at company, battalion, and even brigade levels. In effect, this application transforms the single-station tactical model into a multiprocessing strategic one. On a less ambitious scale, the output from single stations may be used to generate input for another higher-level manpower model used to plan force requirements. This particular arrangement resembles that of the Air Force combat models Brawler and Thunder, which operate

in concert to simulate both the tactical and strategic aspects of theater conflict. The possibilities are limited only by resources and imagination due to the great flexibility imparted by the use of the single-PC platform upon which the Recruiting Process simulation model operates.

While we have stressed the problems confronting military recruitment thus far, we cannot ignore the enormous success of the Army - and the entire Armed Forces in general - in building the exceptionally dedicated and competent corps of professionals who comprise the all-volunteer forces that exist today. The performance of troops during Desert Storm and in Bosnia attests to this fact, as they do even during the present period of personnel drawdowns and large cutbacks to military funding. The station recruiters are inherently important, and even crucial, because of the implication that their performance has on maintaining a strong and capable military force. As such, the leadership must find any way within their means to improve the lot of the military recruiter.

APPENDIX A - Anecdotal Data

A.1 Data

What follows is an informal listing of things said by recruiters during the course of the authors' visits to recruiting stations around the Ohio area. Many recruiters were eager to share this information with us once we informed them that we would relay their suggestions to USAREC. Although these observations were obtained from a relatively small subset of stations within USAREC, they are familiar from our investigations into current literature on the subject of Army recruiting. We hope these suggestions and observations will, at the very least, foster a greater awareness among the readers of this document as to the plight of the Army Recruiter. For further reading, we suggest an article written by Benjamin J. Roberts, "Redesigning Military Recruitment for the Future", which contains great insight about the problems that confront recruiters[22] .

A.1.1 Quality of Life Issues

Recruiters often find that the job places a great amount of stress on their home life, with sometimes catastrophic consequences. According to the station commander at the Bethel, OH station, recruiters must balance the demands of the job (i.e. the station commander) with the demands of the family for his or her attention. The 9 to 18 hour work days do not leave much time for the family interaction that one must have to maintain a reasonably placid home life. As a consequence, recruiters often suffer family break-ups in the form of divorces or problems with errant children. Two of the recruiters that Lt. Friend spoke to, for instance, were going through messy divorces related to the spouse's inability to cope with the recruiter's time-consuming occupation. Of course, these issues affect society in general to a great degree, but many recruiters feel that the stresses of the recruiting job jeapordizes family relationships. Isolation from bases, namely housing, pay, and PX/Commissary services, puts stress on the recruiters and their families. One recruiter stated that

he had trouble correcting pay problems, and succeeded only after months of calls and intercessions by his station commander. Some areas, such as Covington, KY, are relatively expensive because of their proximity to large cities, and thus lower-ranking SGT recruiters suffer for it. In addition, some locations are rife with crime and urban decay. Recruiters and their families are thus placed into less-than-ideal, sometimes even dangerous areas in which raising families is a difficult proposition.

Self-image becomes an important issue for certain recruiters, particularly those who are having difficulty coping with their job. Detailed recruiters, in particular, are often pulled from duty stations that they may have enjoyed or duties that they excelled in, and then land in places they do not like to do a job that they are not particularly good at. As one recruiter put it, these recruiters often are not accustomed to failure, the realization of which recurs each month when they do not meet the mission requirements. These issues are at the root of the Army's difficulty in persuading detailed recruiters to remain in USAREC as members of the professional recruiting corps.

A.1.2 The Job

There are several tasks that, by regulation(s), the recruiter must accomplish. In one particular station, recruiters are required to telephone prospect four hours per day for five days a week. They also must prospect grads during the hours of 9 to 10 (which is not really enforced) and high schools from 3 to 5. Telephone-prospecting is accomplished for a period of four hours on Mondays and Tuesdays (for most, but not all, of the station recruiters). Other days are spent trying more to catch up on paperwork, face-to-face prospecting, and other assorted chores. Face-to-face prospecting is usually done twice a week for about two hours. Recruiters are frustrated because if they prospect and enlist anyone from any group other than the "A" category, they do not receive any credit for the contract.One recruiter felt that, since the goal was to obtain greater numbers of "A" contracts, they should consequently spend less time closing the loop and more time working on tasks directly

related to obtaining recruits. In summary, recruiters should be given some type of credit for enlisting anyone - regardless of category. The same amount of work, after all, is required for these applicants.

An opinion held by most recruiters we spoke to was that a survey cannot encompass the extreme variability of day to day activities. Time spent on recruiting tasks fluctuates widely depending on time of year, the day of the week, or even the time of day. One month, you may write 2 to 3 contracts, but during another month, you might not get any applicants to enlist.

A number of prolems associated with data collection was common to all stations visited by the authors. One of the sections of the questionnaire used for data collection (see Appendix A.2) required recruiters to provide the times necessary to generate appointments by means of both telephone and face-to-face prospecting. Later in the form, the recruiters were asked for the amount of time that they spent on certain recruiting tasks (on a daily basis). Although the instructions asked for daily average times, respondents would almost always provide the entire amount of time needed to complete a task, which often resulted in absurd workdays of greater than 24 hours! This led us to suspect that recruiters were severely overestimating the number of contracts they write during the course of the year. In response to this problem, we incorporated a "reality check" into the form that automatically calculated the number of appointments that would be generated in a year based on the reported data. This resultant value was compared to the documented number of appointments. As expected, we found that recruiters reported an average number of contracts that was consistently 200% to 700% greater than the actual appointment counts.

When questioned about the aforementioned discrepancy on the data sheets, recruiters would often insist that they actually spend the documented amount of time working on obtaining contracts. However, the "reality check" on the form gave us sufficient visual proof of our assertion that they provided faulty data, upon which the recruiter usually obliged us by providing the correct data. This brought us to the realization that the fact that the questionnaire gave the respondents such freedom to

provide the faulty data set provided grounds to doubt the validity of the data that we collected. One recruiter provided further insight into this problem (after being assured that no names were going to be used). He stated that, even though you may have prospecting scheduled on your daily planner, and are honestly attempting to prospect at least 1/3 of the scheduled time, frequent distractions and interruptions make faithful adherance to the calendar an absurd expectation. Therefore, though you may ask recruiters how much they prospect during the course of a week, they will always give you the official time, and omit the time wasted on incidental tasks.

Based on this conversation - as well as observations of the system - it appears that the recruiters may not have been exaggerating their recruiting abilities as much as we originally thought, but rather simply did not account for the amount of time that they lost due to interruptions. The recruiter's method of adjusting how long they require to generate an appointment essentially has the effect of capturing these interrupts, and has been used with some success to obtain acceptable results from the model. The model does account for time spent on miscellaneous activities; for a more detailed explanation, see Section 4.2.5.

A.1.3 The Recruiter's Views on Society

Recruiters often view society negatively, based on the attitudes of the applicants that they must deal with on a daily basis. They see young people behaving in ways they would never have dreamed of when they were youths, and thus proceed to attribute these malfeasances to the corrupting influences of their environment. Drug use (which comprises most of the waiver requests in certain areas of southern Ohio), theft, assault, and other crimes disqualify many of the applicants. Less seriously, but not less of a barrier to recruiting efforts, is the seeming indifference of youth to the ideals of the past, namely of service to country, personal honor, and other values that may have motivated the recruiter to pursue a military career. According to one recruiter from Middletown, OH, most of the

young men aspired to no more than "... owning their own drive-through beer joint". In several areas we visited, there is also an undercurrent of anti-government feeling which transfers over to the recruiters. Some people are still under the impression that the draft is still in effect, and that the recruiters are trying to "kill" their children by sending them to foreign wars. In some places, recruiters are targeted by thugs and others who dislike authority because of their resemblance to policemen in government vehicles. Such incidents, which often occur with noticeable regularity, tend to arouse cynicism amongst some recruiters, which in turn has a correspondingly negative effect upon their morale.

A.1.4 Technology

Recruiters would like access to the World-Wide-Web (WWW). Because of the lack of such capability at work, they must do any sort of job-related browsing at home (if at all). They say that it would enable them to conduct searches, gather information on applicants, and talk to other recruiters and stations. There is an AOL chat room dedicated to recruiters for the purpose of sharing information. Presentation information on the laptops uses people dressed from the 80s and (believe it or not!) applicants pick up on it. Applicants cannot identify with the people in the presentation. Recruiters did mention, however, that a power point presentation of the sales book would be helpful. Another useful innovation using the new laptops would be to electronically generate many of the forms used by recruiters and associate them to an electronic 200-Card, particularly if they are based on spreadsheet (Excel) technology.

The fact that the recruiters have computers and are able to do a sales presentation really captures the applicant's attention. It seems to reinforce the Army's image as being on the cutting edge of technology, as well as representing the Army's promise to provide certain applicants with useful technical training. Some negative impressions were stated by recruiters, mainly by those who had

difficulty adjusting to the technology. However, most of those we questioned were overwhelmingly positive about the provision of laptops.

A.2 Questionnaire

The following pages contain the questionnaire used to gather task-durations from recruiting stations during field investigations. Questions in the first section relate to average total task times while those towards the end relate to average *daily* task completion times. A sequence of computer-generated fields at the end calculate the number of contracts the recruiter should produce in a year - as well as the average workday in hours - based on the recruiter's responses. The form itself was implemented as a Microsoft Excel spreadsheet.

Recruiter Number

In the interest of being able to model a recruiting station correctly it is necessary to obtain a accurately reflects what is actually happening in the field. Answers to this survey will not b shape or form other than to help USAREC HQ get a clear understanding of the times involv recruiting tasks. Records will not be maintained on the recruiter that provided the informati All information will be kept anonymous. You do not have to put your name upon this form. consequence free forum to provide real information, please honestly fill out this form to the ability.

	Shortest (average)	Normal (average)
Time needed telephone prospecting to make one appointment.		
Time needed using face to face prospecting (including driving) to make one appointment when prospecting graduates. Ex 1 hour driving 2 hours face to face prospecting. 3 appointments = 1 hr per appointment		
Time needed using face to face prospecting (including driving) to make one appointment at high schools		
Time needed to conduct a pre-sales presentation (for walk ins).		
Time needed to conduct a sales presentation.		
Time needed to fill out paperwork associated with a sales presentation.		
Time spent waiting for an applicant to bring back paperwork for processing.		

For the following 3 questions include the time needed to

make an appointment and a time the applicant will agree to. This means the number of hours/days from when a person agrees to take a test or go to MEPS until the time the applicant arrives to test or at MEPS

Get an applicant to area for testing.		
Get an applicant to MEPs for testing and a physical.		
Get an applicant to MEPs for a physical only.		
Time needed to fill out paperwork for an enlistment package.		
Time needed to fill out paperwork for a medical waiver.		
Time needed to fill out paperwork for a moral waiver.		
Time needed to conduct an initial DEP interview		
How long does it take to resell members of DEP who are getting cold feet?		
How long does it take to make a telephone DEP contact?		
How long does it take to conduct a face to face DEP contact?		

	Good Month	Average Month
Percentage of DEPs which must be "resold" on the ARMY.		
How long do you spend driving in DEP sustainment efforts?		
Percentage of DEPs which refuse to join or otherwise lost		

Figure 35. Questionnaire (Page 1)

Average number of walk - ins you interview

How many potential applicants do you have to turn away due to criminal records, out of 10 ? Ex. 4 out of 10.

Telephone	Face to Face

On average how often do you contact your DEPS per category.

Telephone	Face to Face

What are some of the other ways you spend time with your DEPS?I.E. Do physical training, pizza parties, etc

Dep functions, House calls Visits Physical training.

When thinking of prioritizing tasks use following mindset. You are filling out your daily plan have applicants at every stage of processing. How do you plan your day? If you do not ha to complete all of your tasks in one day, which items do you leave in your daily plan? In eff tasks left in the daily plan are given a higher priority than the ones scheduled for another day

Task	Rank priority between 1 to 10 Higher number means higher
Helping a walk-in	
Telephone prospecting	
Face to face prospecting	
Conducting a sales interview (walk in)	
Conducting a sales interview	
Filling out applicants paperwork (processing)	
Filling out waivers	
"Reselling" DEP's	
Conducting DEP maintenance	
Non recruiting related duties (van duty etc)	

Task	Time spent on tasks in an a Shortest (average)	Normal (average)
Daily Performance Review		
Telephone prospecting including time to fill out paperwork		
Face to face prospecting including time to fill out paperwork		
Conducting sales interviews (walk-in)		
Conducting sales interviews		
Filling out paperwork associated with sales interviews.		
Filling out applicants paperwork (processing)		
i.e. Enlistment packages etc.		
Filling out waivers		
"Reselling" DEP's		
Conducting DEP maintenance		
Seeing a recruit off to BT		
Additional recruiting duties.		

Figure 36. Questionnaire (Page 2)

Non recruiting related duties (van duty etc)		

How long have you been recruiting?	
Are you a detailed recruiter or a 79R?	
How long is your workday?	
How many days a week do you work on average?	
How long are you given for lunch?	
How many breaks do you take/are given in a day?	
How long are the breaks?	

Based on your reported data the following expected values have been generated you may find them surprising.

	Best	Average
Number of appointments a year expected to be made allowing 30 days leave and 30 days miscellaneous activities.		
Appointments from telephone prospecting.	#VALUE!	#VALUE!
Appointments from face to face prospecting.	#VALUE!	#VALUE!

	Shortest Possible	Average Day length
Day length based on time spent on recruiting tasks.	0	0

Figure 37. Questionnaire (Page 3)

APPENDIX B - Simulated Recruiter Schedules

The following tables represent recruiter schedules in sequential order for each experimental design point presented in Chapter 5. Note that the schedules are ultimately determined by the simulation run, and many time blocks are therefore not given specific labels. The order of the effects in the headers of the tables below are Prospect, Collateral, and Process; i.e. + – + refers to the experiment in which prospecting level is high, the collateral level is low, and the processing level is high.

Key	Meaning
C	Collateral Duties
L	Lunch
T	Telephone Prospecting
F	Face-to-Face Prospecting

Design – – –						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	
0900-1000	C	C	C	C	C	
1000-1100						
1100-1200						
1200-1300	L	L	L	L	L	
1300-1400						
1400-1500			F			
1500-1600			F			
1600-1700	T	T	F	F		
1700-1800	T	T	F	F		
1800-1900	T	T	F	F		
1900-2000						

Table 11. Point 1

Design – – +						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	P
0900-1000	C	C	C	C	C	P
1000-1100						P
1100-1200						P
1200-1300	L	L	L	L	L	
1300-1400						
1400-1500			F			
1500-1600			F			
1600-1700	T	T	F	F		
1700-1800	T	T	F	F		
1800-1900	T	T	F	F		
1900-2000						

Table 12. Point 2

Design – + –						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	
0900-1000	C	C	C	C	C	
1000-1100	C	C	C	C	C	
1100-1200	C	C	C	C	C	
1200-1300	C	C	C	C	C	
1300-1400	L	L	L	L	L	
1400-1500			F			
1500-1600			F			
1600-1700	T	T	F	F		
1700-1800	T	T	F	F		
1800-1900	T	T	F	F		
1900-2000						

Table 13. Point 3

Design – + +						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	P
0900-1000	C	C	C	C	C	P
1000-1100	C	C	C	C	C	P
1100-1200	C	C	C	C	C	P
1200-1300	C	C	C	C	C	
1300-1400	L	L	L	L	L	
1400-1500			F			
1500-1600			F			
1600-1700	T	T	F	F		
1700-1800	T	T	F	F		
1800-1900	T	T	F	F		
1900-2000						

Table 14. Point 4

Design + – –						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	
0900-1000	C	C	C	C	C	
1000-1100						
1100-1200						
1200-1300	L	L	L	L	L	
1300-1400						
1400-1500			F	F	F	
1500-1600	T	T	F	F	F	
1600-1700	T	T	F	F	F	
1700-1800	T	T	F	F	F	
1800-1900	T	T	F	F	F	
1900-2000	T	T				

Table 15. Point 5

Design + – +						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	P
0900-1000	C	C	C	C	C	P
1000-1100						P
1100-1200						P
1200-1300	L	L	L	L	L	
1300-1400						
1400-1500			F	F	F	
1500-1600	T	T	F	F	F	
1600-1700	T	T	F	F	F	
1700-1800	T	T	F	F	F	
1800-1900	T	T	F	F	F	
1900-2000	T	T				

Table 16. Point 6

Design + + –						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	
0900-1000	C	C	C	C	C	
1000-1100	C	C	C	C	C	
1100-1200	C	C	C	C	C	
1200-1300	C	C	C	C	C	
1300-1400	L	L	L	L	L	
1400-1500			F	F	F	
1500-1600	T	T	F	F	F	
1600-1700	T	T	F	F	F	
1700-1800	T	T	F	F	F	
1800-1900	T	T	F	F	F	
1900-2000	T	T				

Table 17. Point 7

Design + + +						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	P
0900-1000	C	C	C	C	C	P
1000-1100	C	C	C	C	C	P
1100-1200	C	C	C	C	C	P
1200-1300	C	C	C	C	C	
1300-1400	L	L	L	L	L	
1400-1500			F	F	F	
1500-1600	T	T	F	F	F	
1600-1700	T	T	F	F	F	
1700-1800	T	T	F	F	F	
1800-1900	T	T	F	F	F	
1900-2000	T	T				

Table 18. Point 8

Design 0 0 0						
Time	Mon	Tues	Wed	Thurs	Fri	Sat
0800-0900	C	C	C	C	C	P
0900-1000	C	C	C	C	C	P
1000-1100	C	C	C	C	C	
1100-1200	0.5C	0.5C	0.5C	0.5C	0.5C	
1200-1300						
1300-1400	L	L	L	L	L	
1400-1500			F	F	F	
1500-1600	T	T	F	F	F	
1600-1700	T	T	F	F	F	
1700-1800	T	T	F	F	F	
1800-1900	T	T				
1900-2000						

Table 19. Point 9

APPENDIX C - MODSIM Code

C.1 Recruiting Station

C.1.1 Definition Module

The following code contains the declarations of all methods and variables pertaining to the Recruiting Station module:

```
DEFINITION MODULE rstationMod;


          FROM RandMod       IMPORT RandomObj;
          FROM rankedListMod IMPORT rankedListObj;
          FROM StatMod       IMPORT IStatObj, RStatObj;
          FROM CalendarMod IMPORT dateObj, ALL daytype,
                                  ALL monthtype;
          FROM IOMod IMPORT StreamObj, ALL FileUseType;
          FROM ResMod        IMPORT ResourceObj;
          FROM rstatMod      IMPORT StatRec, statsObj;

          TYPE
{=============================================================}
  prospectcattype = (walkin, telephone, face);

{    Def: These are the categories that applicants are given. }
{         They are maintained because different applicants    }
{         have different service times as well as for         }
{         statistical purposes.                               }


  approcesstype = (immediate, normal);

{    Def :  These categories differentiate the type of        }
{           processing an applicant will experience.  If they }
{           know they want to join they will be processed     }
{           immediately otherwise they go through the normal  }
{           processing.                                       }

  apstagetype =(prospectstage,sellstage, waiverstage,
                Iprocessstage, Nprocessstage,depstage);

{    Def :  These are all of the stages an applicant might    }
{           go through in the process of becoming a member of }
```

```
{           the Army.  The stages are used to assign the      }

{           applicant to the correct process after it gets the}
{           recruiter resource.                                }

  testtype = (asvab, moral, medical, process,sales);

{    Def:  These categories will be passed into method test }
{          passed to let it know which test the applicant   }
{          passed/received waiver                           }

  applicanttype = (alpha, prior, other);

{    Def:  What category of applicant.  Each applicant has a }
{          category with category Alpha being the most       }
{          desirable.                                        }

  traitArrayType = FIXED ARRAY[1..3] OF REAL;

{    Def:  Array that we use to keep track of various        }
{          distributions such as                             }
{          triangular distribution has a worst average and   }
{          best numbers which need to be kept track of.      }

   stationArrayType = ARRAY INTEGER OF RecruiterObj;

{    Def:  A recruiting station consists of a group or       }
{          recruiters each of which are modeled a recruiter  }
{          object with their own independent methods.        }
{          This allows all of the recruiters to work in      }
{          parallel.  The stationArrayType is just a way to  }
{          keep track of all of the recruiter objects.       }

   statsArrayType = ARRAY INTEGER OF statsObj;

{==============================================================}
{                        OBJECT DEFINITIONS                    }
{==============================================================}

  applicantObj = OBJECT;

          appProspCat : prospectcattype;
               appDesignation : applicanttype;
               processCat : approcesstype;
               apstage : apstagetype;
               arrivalTime,
               {time pseudo applicant becomes an applicant}
```

```
        timeInQueue,      {not used at this time}
           priority,timeInSystem,moralTime,medicalTime,
           sellTime,immediateprob,
           processtime, moralstart,
           medicalstart, normalprob,
           {*}lossProb : REAL;
transID : ACTID;
           status,appID,Testpass, Moralpass,
           Medicalpass, numtestattempts,
           nummoralinterrupts, nummedicalinterrupts,
           needMoral, needMedical, processed : INTEGER;

           ASK METHOD setFields(IN sellT,sellpT,processT,
                      medicalT, moralT : traitArrayType;
                      IN stream1, stream2 : RandomObj; IN
                      prospectpriority : REAL;IN
                      walkinflag : INTEGER);
           ASK METHOD setTransmitID(IN TransID : ACTID);
           ASK METHOD setstage(IN stage : apstagetype);
           ASK METHOD changestatus;
           ASK METHOD setLossProb(IN lprob : REAL);
           ASK METHOD changeprocessed;
           ASK METHOD passedtest(IN status : INTEGER; IN
                                 test : testtype);
           ASK METHOD updateprocesTime(IN timepassed :
                      REAL; IN test : testtype);
           ASK METHOD addwaiverinterrupt(IN test :
                      testtype);
           ASK METHOD startwaivertime(IN test : testtype);
           ASK METHOD changepriority(IN pri : REAL);
           ASK METHOD setid(IN ID : INTEGER);
           ASK METHOD setarrivalTime(IN arvtime : REAL);

 END OBJECT; {applicantObj}


 RecruiterObj = OBJECT (rankedListObj);

            stream1              : RandomObj;
            stream2              : RandomObj;
            stream3              : RandomObj;
          {*}sustainStream       : RandomObj;
            recruiter            : ResourceObj;

    tprospectTime, fprospectTime,
              saleTime,salepaperTime,
              presaleTime,processTime,
```

```
                medicalTime,moralTime: traitArrayType;
            lossProbs : FIXED ARRAY [1..10] OF REAL;
                tosalesdelay, toprocessdelay,tomoraldelay,
                tomedicaldelay: traitArrayType;
            runLength,activitypri,probtestpass,
                timespent : REAL;
                maxPtime, dayPtime, currentPtime,
                workweek : REAL;
                numRecruiters,recID,appnum,walkCount,
                daydone,days: INTEGER;
                prospectpri,sellpri,waiverpri,waiverHighpri,
                walkinpri,Iprocesspri, Nprocesspri,
                deppri : REAL;
                calendar : dateObj;
    DepLossProb, DepLength1, DepLength2,
                DepLength3  : REAL;
    DepIntArv, DepMtgTime1, DepMtgTime2,
                DepMtgTime3 : REAL;
    RstatsRec : StatRec;

  ASK  METHOD ObjInit;
ASK METHOD getrecruiterinfo(IN rsinput : StreamObj;
                      IN number,numdays : INTEGER; IN rlength,
                      maxpospecttime, lworkweek : REAL);
          ASK METHOD sendtoprocess(IN oldapplicant :
                                    applicantObj);
          TELL METHOD grabrecruiter(IN applicant :
                                   applicantObj);
          WAITFOR METHOD grabDEPRec (IN applicant :
                                       applicantObj);
          TELL METHOD newday;
          ASK METHOD incNumApplicants;
          ASK METHOD incrementappnum;
          ASK METHOD resetPtimes;
          ASK METHOD incrementInterruptedtasks;
          TELL METHOD generateApplicants;
          TELL METHOD prospect(IN applicant : applicantObj);
          TELL METHOD sell(IN applicant : applicantObj);
        TELL METHOD moralwaiver(IN applicant :
                                 applicantObj);
          TELL METHOD medicalwaiver(IN applicant :
                      applicantObj);
          TELL METHOD testing(IN applicant : applicantObj);
        TELL METHOD immediateprocess(IN applicant :
                                       applicantObj);
          TELL METHOD normalprocess(IN applicant :
                                    applicantObj);
```

```
      TELL METHOD sustain(IN applicant : applicantObj);
        WAITFOR METHOD depMeet(INOUT applicant :
                        applicantObj; IN mtgWks : REAL);
        ASK METHOD MapplicantTime;
        ASK METHOD getutilization;
      END OBJECT; {RecruiterObj}

      stationObj = OBJECT;
        station                          : stationArrayType;
        recStats                           : statsArrayType;
        numRecruiters, numdays, numReps          : INTEGER;
        runlength, maxprospectTime, lworkweek      : REAL;
        calendar                                 : dateObj;
        numWalkins      : LMONITORED INTEGER BY IStatObj;
        walkStats                              : IStatObj;

        ASK METHOD StationInit;
        ASK METHOD SetNumReps(IN reps : INTEGER);
        ASK METHOD printStats;
    ASK METHOD RecAction(IN action : STRING);
        TELL METHOD generateWalkins;

      END OBJECT {stationObj};

      PROCEDURE SampleStdDev(IN stdDev : REAL;
                        IN n : INTEGER) : REAL;

END {DEFINITION} MODULE {RecruitstationMod}.
```

C.1.2 Main Module

The following code contains the main program of the Recruiting Station module:

```
MAIN MODULE rstation;


        FROM rstationMod IMPORT RecruiterObj;
        FROM rstationMod IMPORT stationObj;
        FROM rstationMod IMPORT applicantObj;
        FROM SimMod      IMPORT StartSimulation,
                                StopSimulation, ResetSimTime;
       FROM IOMod       IMPORT ReadKey,StreamObj, ALL FileUseType;
        FROM SimMod      IMPORT SimTime, InterruptMethod;


       CONST
          REPLICATIONS = 2;

       VAR
           recStation : stationObj;
           ch  : CHAR;
           i, numrecruiters, repCtr   : INTEGER;
           daylength : REAL;
           recsinput : StreamObj;

       BEGIN

         NEW(recStation);
         OUTPUT("STARTING WALK-INS");
         ASK recStation TO StationInit;
         ASK recStation TO SetNumReps(REPLICATIONS);
         numrecruiters:=ASK recStation numRecruiters;

         FOR repCtr := 1 TO REPLICATIONS
           NEW(recsinput);
           ASK recsinput TO Open("recfile",Input);
           ASK recStation TO RecAction("Init");
           TELL recStation TO generateWalkins;

           FOR i:=1 TO numrecruiters
             ASK  recStation.station[i] TO
                 getrecruiterinfo(recsinput,i,recStation.numdays,
                 recStation.runlength,recStation.maxprospectTime,
               recStation.lworkweek); {get info on how good a recruiter it is}
             TELL recStation.station[i] TO newday;
```

```
    {have each recruiter start of prospecting}
  END FOR;
  ASK recsinput TO Close;

  StartSimulation;

  OUTPUT;

  FOR i:=1 TO numrecruiters
    OUTPUT("Recruiter ",i," produced ",
     recStation.station[i].rstatsRec.numberOfapplicants);
    OUTPUT("Total Created ",
          recStation.station[i].rstatsRec.totalcreated);
   { OUTPUT("Lost at prospecting ",
        recStation.station[i].rstatsRec.lostatprospect);
    OUTPUT("Lost at Pre-sell ",
          recStation.station[i].rstatsRec.lostatPsales);
    OUTPUT("Lost at sales ",
           recStation.station[i].rstatsRec.lostatsales);
    OUTPUT("Lost at test ",
            recStation.station[i].rstatsRec.lostattest);
    OUTPUT("Lost at immediate processing ",
            recStation.station[i].rstatsRec.lostatIMD);
    OUTPUT("Lost at normal processing ",
            recStation.station[i].rstatsRec.lostatNOR);
    OUTPUT("Lost at medical waiver ",
            recStation.station[i].rstatsRec.lostatMED);
    OUTPUT("Lost at moral waiver ",
            recStation.station[i].rstatsRec.lostatMOR);}
    OUTPUT("Lost at DEP ",
            recStation.station[i].rstatsRec.lostatDEP);
    OUTPUT;
    ASK recStation.station[i] TO MapplicantTime;
    ASK recStation.station[i] TO getutilization;
    OUTPUT("Mean time applicants spend in processing: ",
            recStation.station[i].rstatsRec.appTime);
    OUTPUT("Mean recruiter utilization: ",
            recStation.station[i].rstatsRec.recutil);
    OUTPUT("Recruiter utilization standard dev: ",
            recStation.station[i].rstatsRec.recdev);
    OUTPUT;

    {************ Increment Statistics for this Replica-
tion *************}
      ASK recStation.recStats[i] TO IncStats(recStation.station[i].rstatsRec
    END FOR;
```

```
      OUTPUT ("Press any key to end program.");
      ch := ReadKey;
      ResetSimTime(0.0);
      ASK recStation TO RecAction("Terminate");
   END FOR;

   ASK recStation TO printStats;
END {MAIN} MODULE {rstation}.
```

C.1.3 Implementation Module

The following code contains the implementations of all methods pertaining to the Recruiting Station module:

```
IMPLEMENTATION MODULE rstationMod;


          FROM RandMod IMPORT FetchSeed,RandomObj;
          FROM SimMod  IMPORT SimTime, InterruptMethod;
          FROM StatMod IMPORT IStatObj, RStatObj;
          FROM IOMod   IMPORT StreamObj, ALL FileUseType;
          FROM CalendarMod IMPORT dateObj, ALL daytype,
                                  ALL monthtype;


   {*}CONST
         DEMOG = -1.0;
         STCOM = 0.0;
         DEPLOSS = 0.035;  {percent Per Month - i.e. 2
                            months = 7 percent}
         DEPMTG = 2.0;
         {Avg interarrival time in weeks for DEP
          appointments}


OBJECT applicantObj;

ASK METHOD setFields(IN sellT,sellpT,processT, medicalT,
                     moralT : traitArrayType; IN stream1,
                     stream2 : RandomObj; IN
                     prospectpriority : REAL;IN
                     walkinflag : INTEGER);

{Method Description: This method has as input two random
streams used to generate uniform random number streams
to aid in setting an applicants  fields. When an applicant
is generated, the fields in the applicant object are set
by this method.  The category of applicant will be set by
distributions revealed in our data collection.  Of special
importance is what priority the applicant has. The applicants
priority will allow it to interrupt lesser priority activities
a recruiter may be engaged in. Walk-ins are a special case
for this method because some of a walk-in's fields are set by
the special method that generates them}
```

```
VAR
tempsptime : REAL;

BEGIN

{Setting by what method the applicant it is based on the
 data}

 IF (walkinflag = 0)

  CASE stream2.UniformInt (1, 100)
   WHEN 1..60:     {  60percent }
    appProspCat := telephone;
   WHEN 60..100:    { 30percent }
    appProspCat := face;
  END CASE;

 END IF;

 {Setting what type of applicant it is percentages based
  on data collected}

 CASE stream1.UniformInt(1, 100)
  WHEN 1..60:     {  60percent }
   appDesignation := alpha;
  WHEN 61..90:    { 30percent }
   appDesignation := prior;
  WHEN 91..100:   { 10percent }
   appDesignation := other;
 END CASE;

 CASE stream1.UniformInt(1, 100)
  WHEN 1..5:     {  5percent }
   needMoral := 1;
  WHEN 6..10:    { 5percent }
   needMedical := 1;
  WHEN 11..13:   { 3percent }
   needMoral := 1;
   needMedical:=1;
  WHEN 14..100:   { 87percent }
   needMoral := 0;
   needMedical := 0;
 END CASE;


 immediateprob:=0.1;
```

```
{prob a person will want to join immediately}
normalprob :=9.0;
{prob applicant goes through the normal processing}

IF (walkinflag = 0)

  IF (stream1.UniformReal(0.0,1.0) <= immediateprob)
   processCat:=immediate;
  ELSE {sets applicants status: immediate or normal}
   processCat:=normal;
  END IF;

ELSE
  processCat:=immediate; {all walkins are immediate}

END IF;

priority:= prospectpriority;
{applicants not walkins need to be prospected}

IF (walkinflag = 0)
 apstage := prospectstage;
ELSIF(walkinflag = 1)
 apstage := sellstage;
 {walkins go directly to sales}
END IF;

status:=0; {applicant is a pseudoapplicant}
processed := 0;
{applicant hasn't been processed}
processtime:=stream1.Triangular(processT[1],processT[2],
                                processT[3]);
Testpass :=0;
{applicant hasn't passed ASVAB/AFQT already}

Moralpass:=2;
{applicant hasn't passed or failed}

Medicalpass:=2;
moralTime:  =stream1.Triangular
                    (moralT[1],moralT[2],moralT[3]);
{Time to fill out moral waiver paperwork}

medicalTime:=stream1.Triangular
             (medicalT[1],medicalT[2],medicalT[3]);
{Time needed to fill out medical waiver paperwork}
```

```
  tempsptime:=stream1.Triangular
                     (sellpT[1],sellpT[2],sellpT[3]);
  {sales paperwork time}

  sellTime := (stream1.Triangular
              (sellT[1],sellT[2],sellT[3]) +tempsptime);
  {Time is takes to complete a sales interview}

  numtestattempts := 0;
  {applicant hasn't tested yet}

  nummoralinterrupts :=0; {default it has not been
                           interrupted}
  nummedicalinterrupts := 0;
  {default it has not been interupted}

END METHOD; {setFields}

ASK METHOD setid(IN ID : INTEGER);

 {Method Description:  This method sets the applicants
  unique applicant ID. This is a debugging feature allowing
  the user track an applicants progress throughout the
  recruiting process}

 BEGIN
  appID:=ID;
 END METHOD; {setid}


ASK METHOD setTransmitID (IN TransID : ACTID);

 {Method Description:  This method sets the transaction ID
  in a recruiter.  It allows the recruiter object to
  interrupt a method that is using the recruiter resource
  maintained by the recruiter object.  When an applicant
  object obtains the recruiter resource and proceeds to a
  process the recruiter object saves the transaction
  (transmit) id of the process.}

 BEGIN
  transID := TransID;
 END METHOD; {setTransmitID}


 ASK METHOD setstage(IN stage : apstagetype);
```

```
 {Method Description:  This method sets the stage the
  recruiting process that an applicant is in.  Before
  going to the next stage in the recruiting process
  the applicant changes it's stage.  This is to allow
  the applicant to be placed at the right process after it
  obtains the recruiter resource.}

 BEGIN
  apstage := stage;
 END METHOD; {setstage}


ASK METHOD changestatus;

 {Method Description:  This method sets the status of an
  applicant to 1 when it is an officially prospected
  applicant and no longer a pseudo applicant.  This method
  is only used for debugging purposes.}

 BEGIN
  status:=1;
 END METHOD; {changestatus}


ASK METHOD changeprocessed;

 {Method Description:  Method used by Immediate process
  and Normal process to signify the paperwork needed to by
  done is finished.}

 BEGIN
  processed:=1;
END METHOD; {changeprocessed}


{*}ASK METHOD setLossProb(IN lprob : REAL);
    BEGIN
     IF lprob > 1.0
      lossProb := 1.0;
     ELSIF lprob < 0.0
      lossProb := 0.0;
     ELSE
      lossProb := lprob;
     END IF;
   END METHOD {setLossProb};
```

```
ASK METHOD updateprocesTime(IN timepassed : REAL; IN
          test : testtype);

 {METHOD Description:  Method used by the process
  methods to keep track of how much time was already
  spent on process paperwork before the process was
  interrupted. Process time is reduced by the
  amount of time already processed.}

 BEGIN

  IF (test = medical)
   medicalTime :=(medicalTime - timepassed);
  ELSIF (test = moral)
   moralTime :=(moralTime - timepassed);
  ELSIF (test = sales)
   sellTime := (sellTime - timepassed);
  ELSIF (test = process)
   processtime:= (processtime - timepassed);
  END IF;

END METHOD; {updateprocessTime}


ASK METHOD addwaiverinterrupt(IN test : testtype);

{Method Description:  Method used to keep track of the
 number of times the applicants recruiter was
 interrupted from working in it's waiver.}

 BEGIN
  IF (test = medical)
   INC(nummedicalinterrupts);
  ELSIF (test = moral)
   INC(nummoralinterrupts);
  END IF;
END METHOD; {addwaiverinterrupt}

ASK METHOD startwaivertime(IN test : testtype);

 {Method Description:  Method used to keep track of the
 number of times the applicant's recruiter was
 interrupted from working in it's waiver.}

 BEGIN
  IF (test = medical)
   medicalstart := SimTime;
```

```
  ELSIF (test = moral)
   moralstart := SimTime;
  END IF;
 END METHOD; {addwaiverinterrupt}

ASK METHOD passedtest(IN status : INTEGER; IN test :
                      testtype);

{Method Description:  Method used by the testing method
 to signify that an applicant passed the ASVAB/AFQT.}

 BEGIN

  IF (test = asvab)
   IF (status=1)
    Testpass:=1;
    INC(numtestattempts);
   ELSE
    INC(numtestattempts);
   END IF;

  ELSIF (test = moral)
   IF (status = 1)
    Moralpass:=1;
   ELSE
    Moralpass:=0;
   END IF;

  ELSIF (test = medical)
   IF (status = 1)
    Medicalpass:=1;
   ELSE
    Medicalpass:=0;
   END IF;
  END IF;

END METHOD; {passedtest}

ASK METHOD changepriority(IN pri : REAL);

{Method Description:  This method changes the priority
 of an applicant to reflect what stage in the recruiting
 process the applicant is in.  The applicants priority
 is used to interrupt the recruiter doing less important
 tasks}

 BEGIN
```

```
    priority:=pri;
  END METHOD; {changepriority}

  ASK METHOD setarrivalTime(IN arvtime : REAL);

  {Method Description:  This method stores when the
   applicant arrived into the system}

   BEGIN
    arrivalTime:=arvtime;
  END METHOD; {setarrivalTime}

END OBJECT; {applicantObj}


OBJECT RecruiterObj;

 ASK METHOD ObjInit;

  {Method Description:  This sets up each recruiter object
  at the beginning of the simulation. It creates the
  recruiter resource simulating the recruiter, sets up the
  statistical variables, and generates seeds to be used by
  randomobjs}

  BEGIN

   NEW (stream1);
   NEW (stream2);
   NEW (stream3);
   {*}NEW (sustainStream);
   NEW (rstatsRec);

   ASK stream1 TO SetSeed (FetchSeed (1));
   ASK stream2 TO SetSeed (FetchSeed (2));
   ASK stream3 TO SetSeed (FetchSeed (3));
   {*}ASK sustainStream TO SetSeed (FetchSeed (4));

  END METHOD; {ObjInit}

  ASK METHOD getrecruiterinfo(IN rsinput : StreamObj; IN
            number,numdays : INTEGER; rlength,
            maxprospecttime, lworkweek : REAL);

   {Method Description:  This is essentially an
   initialization program for each recruiter. The
   information needed is read from three files.
```

```
commonfile,recfile, and depfile. The common file contains all
of the information that is common to all of the recruiters.
The recfile contains recruiter specific information. Finally,
the depfile contains information pertaining to distributions
needed in dep maintenance }

VAR
 daylength : REAL;
 getrecinput : StreamObj;
 temps : STRING;

BEGIN
 NEW (recruiter);
 ASK recruiter TO Create (1);
 ASK recruiter TO SetPendStats(TRUE);
 ASK recruiter TO SetAllocationStats (TRUE);

 NEW(calendar);
 {Creates a new calendar object to keeps up with month
 and day}

 ASK calendar TO DateInit(lworkweek,numdays);
 {calendar requires the week length and number of days in
  week}

 daylength := calendar.hoursPerday;
 appnum :=0;
 {Initializes the applicant number to zero}

 NEW(getrecinput);
 ASK getrecinput TO Open("commonfile",Input);
 ASK getrecinput TO ReadReal(lossProbs[6]);
 {probability of not getting a medical waiver}
 ASK getrecinput TO ReadLine(temps);
 ASK getrecinput TO ReadReal(lossProbs[7]);
 {probability of not getting a moral waiver}
 ASK getrecinput TO ReadLine(temps);
 ASK getrecinput TO ReadReal(probtestpass);
 {probability of passing a test}
 ASK getrecinput TO ReadLine(temps);
 ASK getrecinput TO ReadReal(tosalesdelay[1]);
 ASK getrecinput TO ReadLine(temps);
 ASK getrecinput TO ReadReal(tosalesdelay[2]);
 ASK getrecinput TO ReadLine(temps);
 ASK getrecinput TO ReadReal(tosalesdelay[3]);
 ASK getrecinput TO ReadLine(temps);
 ASK getrecinput TO ReadReal(toprocessdelay[1]);
```

```
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(toprocessdelay[2]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(toprocessdelay[3]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(tomedicaldelay[1]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(tomedicaldelay[2]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(tomedicaldelay[3]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(tomoraldelay[1]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(tomoraldelay[2]);
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(tomoraldelay[3]);
ASK getrecinput TO ReadLine(temps);

ASK getrecinput TO Close;

ASK getrecinput TO Open("depfile",Input);

ASK getrecinput TO ReadReal(DepLossProb);
{DEP loss probability                    }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepLength1);
{Length of DEP - Lower Bound             }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepLength2);
{Length of DEP - Average                 }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepLength3);
{Length of DEP - Upper Bound             }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepIntArv);
{Ave time between DEP mtgs               }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepMtgTime1);
{Length of DEP mtgs                      }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepMtgTime2);
{Length of DEP mtgs                      }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO ReadReal(DepMtgTime3);
{Length of DEP mtgs                      }
ASK getrecinput TO ReadLine(temps);
ASK getrecinput TO Close;
```

```
{Print statements are for debugging purposes only}
PRINT(lossProbs[6]) WITH "****.**";
PRINT(lossProbs[7]) WITH "****.**";
PRINT(probtestpass) WITH "****.**";
PRINT(tosalesdelay[1]) WITH "****.**";
PRINT(tosalesdelay[2]) WITH "****.**";
PRINT(tosalesdelay[3]) WITH "****.**";
PRINT(toprocessdelay[1]) WITH "****.**";
PRINT(toprocessdelay[2]) WITH "****.**";
PRINT(toprocessdelay[3]) WITH "****.**";
PRINT(tomedicaldelay[1]) WITH "****.**";
PRINT(tomedicaldelay[2]) WITH "****.**";
PRINT(tomedicaldelay[3]) WITH "****.**";
PRINT(tomoraldelay[1]) WITH "****.**";
PRINT(tomoraldelay[2]) WITH "****.**";
PRINT(tomoraldelay[3]) WITH "****.**";
OUTPUT;
OUTPUT;
{Above is information that is common to all of the
 recruiters}
runLength:= rlength;
{Simulation run length }
numRecruiters := 1;
{each recruiter object has only one recruiter}
recID:=number;
{the recruiter's id number in the station}
maxPtime := maxprospecttime;
{Maximum amount of time a recruiter is allowed to
 prospect}
dayPtime := 0.0;     {These get initialized to zero}
currentPtime:=0.0;
rstatsRec.totalcreated := 0;
rstatsRec.lostatprospect := 0;
rstatsRec.lostatsales :=0;
rstatsRec.lostatIMD :=0;
rstatsRec.lostatNOR :=0;
rstatsRec.lostatMED :=0;
rstatsRec.lostatMOR :=0;
rstatsRec.lostattest := 0;
rstatsRec.lostatPsales := 0;
rstatsRec.appTime := 0.0;

{Below is information unique to each recruiter}

ASK rsinput TO ReadReal(tprospectTime[1]);
ASK rsinput TO ReadLine(temps);
```

```
ASK rsinput TO ReadReal(tprospectTime[2]);
{telephone prospecting distribution}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(tprospectTime[3]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(fprospectTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(fprospectTime[2]);
{face to face distribution}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(fprospectTime[3]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(lossProbs[1]);
{Probability of not showing up for sales interview}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(lossProbs[2]);
{Probability of not passing pre-sales interview for
 walkins}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(lossProbs[3]);
{Probability of not showing up for processing}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(lossProbs[4]);
{Probability of deciding against army in Immediate
 processing}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(lossProbs[5]);
{Probability of deciding against army in Normal
 processing}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(presaleTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(presaleTime[2]);
{reading in pre sales duration dist}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(presaleTime[3]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(saleTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(saleTime[2]);
{reading in sales interview duration dist}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(saleTime[3]);
```

```
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(salepaperTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(salepaperTime[2]);
{reading in sales paperwork duration dist}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(salepaperTime[3]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(processTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(processTime[2]);
{reading in processing time duration dist}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(processTime[3]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(medicalTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(medicalTime[2]);
{reading in medical waiver paperwork time dist}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(medicalTime[3]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(moralTime[1]);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(moralTime[2]);
{reading in moral waiver paperwork time dist}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(moralTime[3]);
ASK rsinput TO ReadLine(temps);
{Reading in the process priorities}
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(prospectpri);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(sellpri);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(Nprocesspri);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(Iprocesspri);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(waiverpri);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(waiverHighpri);
ASK rsinput TO ReadLine(temps);
```

```
ASK rsinput TO ReadReal(deppri);
ASK rsinput TO ReadLine(temps);
ASK rsinput TO ReadReal(walkinpri);
ASK rsinput TO ReadLine(temps);

{Print statements useful for debugging purposes only}
PRINT(tprospectTime[1]) WITH "****.**";
PRINT(tprospectTime[2]) WITH "****.**";
PRINT(tprospectTime[3]) WITH "****.**";
OUTPUT;
PRINT(fprospectTime[1]) WITH "****.**";
PRINT(fprospectTime[2]) WITH "****.**";
PRINT(fprospectTime[3]) WITH "****.**";
OUTPUT;
PRINT(lossProbs[1]) WITH "****.**";
PRINT(lossProbs[2]) WITH "****.**";
PRINT(lossProbs[3]) WITH "****.**";
PRINT(lossProbs[4]) WITH "****.**";
PRINT(lossProbs[5]) WITH "****.**";
OUTPUT;
PRINT(presaleTime[1]) WITH "****.**";
PRINT(presaleTime[2]) WITH "****.**";
PRINT(presaleTime[3]) WITH "****.**";
OUTPUT;
PRINT(saleTime[1]) WITH "****.**";
PRINT(saleTime[2]) WITH "****.**";
PRINT(saleTime[3]) WITH "****.**";
OUTPUT;
PRINT(processTime[1]) WITH "****.**";
PRINT(processTime[2]) WITH "****.**";
PRINT(processTime[3]) WITH "****.**";
OUTPUT;
PRINT(medicalTime[1]) WITH "****.**";
PRINT(medicalTime[2]) WITH "****.**";
PRINT(medicalTime[3]) WITH "****.**";
OUTPUT;
PRINT(moralTime[1]) WITH "****.**";
PRINT(moralTime[2]) WITH "****.**";
PRINT(moralTime[3]) WITH "****.**";
OUTPUT;
PRINT(prospectpri) WITH "****.**";
PRINT(sellpri) WITH "****.**";
PRINT(Nprocesspri) WITH "****.**";
PRINT(Iprocesspri) WITH "****.**";
PRINT(waiverpri) WITH "****.**";
PRINT(waiverHighpri) WITH "****.**";
PRINT(deppri) WITH "****.**";
```

```
  PRINT(walkinpri) WITH "****.**";
  OUTPUT;
  OUTPUT;

END METHOD; {getrecruiterinfo}

TELL METHOD grabrecruiter(IN applicant : applicantObj);

{Method Description:  This method is called by almost      }
{every other method. This is where all of the applicants   }
{compete for the recruiter resource. The applicants arrive}
{here and based on their priority they compete for the     }
{recruiter resource.  If the recruiter resource is being   }
{utilized by an applicant with a lesser priority process,  }
{the recruiter is taken away from it and given to the      }
{higher priority applicant. The applicant is then sent to  }
{find its correct process in sendtoprocess based on which  }
{stage the applicant is at.  The old applicant waits to    }
{obtain the recruiter resource back before going back to   }
{process where it left off.  If an applicant arrives that  }
{has a lesser priority and the recruiter is busy, then the}
{applicant waits for the recruiter resource.  Once the     }
{applicant obtains the recruiter resource it is sent on    }
{it's way.  Because waivers are such a low priority but    }
{still need to be completed, their priority is increased   }
{if after a suitable time period the waiver has not been   }
{completed.                                                }

 VAR
  oldapplicant : applicantObj;
  depFlag      : INTEGER;

 BEGIN
  oldapplicant:= ASK SELF Last();
  depFlag := 0;
  IF (oldapplicant <> NILOBJ)
   IF (oldapplicant.apstage = depstage)
     depFlag := 1;
   END IF;
  END IF;

  IF ((oldapplicant <> NILOBJ)  OR  (depFlag = 1))
   AND (recruiter.Resources = 0)
   { no circuits are available }
   AND (applicant.priority > oldapplicant.priority)
   {newapplicant is of higher priority }
```

```
OUTPUT("applicant ",applicant.appID, " for recruiter
       ",recID," going to interupt oldapplicant
       ",oldapplicant.appID, " method");
InterruptMethod (oldapplicant.transID);
INC (rstatsRec.interruptedTasks);
ASK recruiter TO Transfer (oldapplicant,applicant, 1);
ASK  SELF TO RemoveThis (oldapplicant);
OUTPUT("applicant " ,oldapplicant.appID, " for
       recruiter ",recID, " was interupted at time
       ",SimTime);
ASK SELF TO Add(applicant);
{puts applicant into its list notifies recruiter taken}
ASK SELF TO sendtoprocess(applicant);

IF (oldapplicant.apstage = prospectstage)
 {Then it was prospecting and wasn't yet an applicant}

 {OUTPUT("applicant " ,oldapplicant.appID, " was pseudo
         applicant for recruiter ",recID, " it was
         interupted and disposed");}
 DISPOSE(oldapplicant);

ELSIF (oldapplicant.apstage <> depstage)

 OUTPUT("applicant ",applicant.appID, " is waiting for
 recruiter ",recID);
 WAIT FOR recruiter TO
  PriorityGive(oldapplicant,1,oldapplicant.priority);
  ASK SELF TO Add(oldapplicant);
  {puts applicant into its list notifies
   recruiter taken}
  ASK SELF TO sendtoprocess(oldapplicant);
  {Go to process stage you are at}
 END WAIT;

END IF;

ELSE  { applicant will wait for recruiter resource}

 OUTPUT("applicant ",applicant.appID, " is waiting for
        recruiter ",recID);
 WAIT FOR recruiter TO
 PriorityGive(applicant,1,applicant.priority);
  IF (SimTime > runLength) TERMINATE; END IF;
  ASK SELF TO Add(applicant);
  {puts applicant into its list notifies recruiter
   taken}
```

```
    ASK SELF TO sendtoprocess(applicant);
   END WAIT;

  END IF;

END METHOD; {grabrecruiter}


 WAITFOR METHOD grabDEPRec (IN applicant :
                                  applicantObj);

 VAR
  oldapplicant : applicantObj;

 BEGIN
  oldapplicant:= ASK SELF Last();

  IF (oldapplicant <> NILOBJ)        AND
     (recruiter.Resources = 0)
     {no recruiters are available}
     AND (applicant.priority > oldapplicant.priority)
     {newapplicant is of higher priority }

   OUTPUT("applicant ",applicant.appID, " for recruiter
           ",recID," going to interupt oldapplicant
           ",oldapplicant.appID, " method");
   InterruptMethod (oldapplicant.transID);
   INC (rstatsRec.interruptedTasks);
   ASK recruiter TO Transfer (oldapplicant,applicant, 1);
   ASK  SELF TO RemoveThis (oldapplicant);
   OUTPUT("applicant " ,oldapplicant.appID, " for
           recruiter ",recID, " was interrupted at time
          ",SimTime);
   ASK SELF TO Add(applicant);
   {puts applicant into its list notifies recruiter taken}
   OUTPUT("applicant ",applicant.appID, " is waiting for
           recruiter ",recID);
   IF (oldapplicant.apstage = prospectstage)

    {Then it was prospecting and wasn't yet an applicant}
    {OUTPUT("applicant " ,oldapplicant.appID, " was pseudo
            applicant for recruiter ",recID, " it was
            interupted and disposed");}
    DISPOSE(oldapplicant);

   ELSE
```

```
    OUTPUT("applicant ",applicant.appID, " is waiting for
            recruiter ",recID);
    WAIT FOR recruiter TO
     PriorityGive(oldapplicant,1,oldapplicant.priority);
     ASK SELF TO Add(oldapplicant);
     {puts applicant into its list notifies recruiter
      taken}
     ASK SELF TO sendtoprocess(oldapplicant);
     {Go to process stage you are at}
    END WAIT;

   END IF;

  ELSE  { applicant will wait for recruiter resource}

   OUTPUT("applicant ",applicant.appID, " is waiting for
           recruiter ",recID);
   WAIT FOR recruiter TO
    PriorityGive(applicant,1,applicant.priority);

    ASK SELF TO Add(applicant);
    {puts applicant into its list}

    IF (recruiter.NumberAllocatedTo(applicant) = 0)
     OUTPUT("No resource to DEP in grabDEPRec!!!!");
    END IF;

   END WAIT;

  END IF;

END METHOD; {grabDEPRec}


ASK METHOD sendtoprocess(IN oldapplicant : applicantObj);

{Method Description:  This method sends applicants to
 their next stage in the recruitment process based on
 their priority and current stage in the process.
 Stage and priority assignments are accomplished
 before an applicant is ready to go on to the next stage
 and after they have completed  the wait for the stage
 they are currently in. }

 VAR
  pstage : apstagetype;
  transmitActivityID : ACTID;
```

```
BEGIN
 pstage:=oldapplicant.apstage;

 IF (pstage = prospectstage)
  transmitActivityID:=TELL SELF TO prospect(oldapplicant);
  ASK  oldapplicant TO setTransmitID (transmitActivityID);

 ELSIF (pstage = sellstage)

  {We know that a walkin occurred or someone is shipping out}
  OUTPUT("applicant " ,oldapplicant.appID, " for recruiter
         ",recID, " may have been interupted - being sent to
          sales");
  transmitActivityID:=TELL SELF TO sell(oldapplicant);

  ASK  oldapplicant TO setTransmitID (transmitActivityID);
  {applicant sent back to be sold, time already spent in the}
  {sales interview will be saved and the applicant will only}
  {need to be sold the remainder of time left when he/she   }
  {obtains the recruiter resource again                     }

 ELSIF (pstage = Iprocessstage)

  OUTPUT("applicant " ,oldapplicant.appID, " for recruiter
         ",recID, " may have been interupted  - being sent to
          IMD process");
  transmitActivityID:=TELL SELF TO
  immediateprocess(oldapplicant);

  ASK  oldapplicant TO setTransmitID (transmitActivityID);
  INC(rstatsRec.numProcessed);

 ELSIF (pstage = Nprocessstage)

  OUTPUT("applicant " ,oldapplicant.appID, " for recruiter
         ",recID, " may have been interupted - being sent
          back to NOR process");
  transmitActivityID:=TELL SELF TO
  normalprocess(oldapplicant);
  ASK  oldapplicant TO setTransmitID (transmitActivityID);
  INC(rstatsRec.numProcessed);

 ELSIF (pstage = waiverstage)

  IF ((oldapplicant.needMoral = 1) AND
     (oldapplicant.Moralpass = 2))
```

```
     transmitActivityID:=TELL SELF TO
       moralwaiver(oldapplicant);

    ELSIF ((oldapplicant.needMedical = 1) AND
          (oldapplicant.Medicalpass = 2))
          transmitActivityID:=TELL SELF TO
          medicalwaiver(oldapplicant);
    ELSE

     OUTPUT("PROBLEM WITH WAIVERS");

    END IF;

     ASK  oldapplicant TO setTransmitID (transmitActivityID);

{*}ELSIF (pstage = depstage)
         transmitActivityID:=TELL SELF TO
         sustain(oldapplicant);
         ASK  oldapplicant TO setTransmitID
         (transmitActivityID);

   ELSE  {we have a problem here}

    OUTPUT(" a priority appeared that was unexpected in
             SENDTOPROCESS ");
   END IF;

 END METHOD; {sendtoprocess}

 ASK METHOD incNumApplicants;

  {Method Description:  This method increments the number of}
  {applicants that the recruiter has generated.  We should  }
  {have it so a user can not only see how well the station  }
  {does but how well different priorities can combine with  }
  {different recruiter strengths.                           }

  BEGIN
   INC(rstatsRec.numberOfapplicants);
 END METHOD {incNumApplicants};

 ASK METHOD incrementappnum;

  {Method Description:  This method increments the applicant  }
  {number which is mainly used for debugging purposes.  Since }
  {this number is always incremented and not decremented every}
  {applicant will have a unique number.  It will not be equal }
```

```
 {to the number of successful recruits generated.                    }

 BEGIN
  INC(appnum);
END METHOD; {incrementappnum}



TELL METHOD newday;

 {Method Description:  This method is used to take care of a}
 {problem found in ModSim. If all of the activities are not }
 {busy then the program terminates. New day is always busy. }
 {It waits the calculated day length found in the calendar  }
 {object. Then it gets rid of any pseudo applicants on the  }
 {ranked list, finally it gets rid of the old generate      }
 {applicants and starts a new one.  I.E. a new day thus     }
 {occurs.                                                   }

 VAR

  temp, oldapplicant,toldapplicant : applicantObj;
  {used to look at the applicant in the ranked list}
   daylength : REAL;
  {day length computed by the calendar object}
  i,flag : INTEGER;
  activity : ACTID;
  {holds the most recent activity ID of generate applicants}

 BEGIN

  daylength := calendar.hoursPerday;
  { OUTPUT("This is the daylength ",daylength," for recruiter
           ",recID);}
  activity := TELL SELF TO generateApplicants;
  {This generate only runs at start of program}

  rstatsRec.numWalkins := 0;

  WHILE (SimTime <= runLength)
   {Loop runs as long as the program has been set to run}
   flag:=0;
   OUTPUT("There is a new day at ",SimTime, " for recruiter
          ",recID);

   WAIT DURATION (daylength * 60.0);
    {waits a day before telling the recruiters to start
```

```
   prospecting again}

  OUTPUT("recruiter ",recID," has waited and it is
         ",SimTime);

  {In generate applicants we set a flag if the recruiter }
  {has used up his or her maximum reasonable prospecting }
  {time for a day.  If the flag is set it also means that}
  {the recruiter no longer has a generate applicants     }
  {running and we need to start up a new one for him or  }
  {her.  Since we are only modeling working hours if an  }
  {applicant is being prospected (not like reality) or if}
  {an applicant is being helped in any other way similar }
  {to reality the recruiter continues what he or she is  }
  {doing.  Time spent on breaks, lunch etc. decrease the }
  {work week length.}

  IF (daydone = 1)
   activity:=TELL SELF TO generateApplicants;
   {comment out and see if just updating times works}
   {have each recruiter start of prospecting}
  END IF;

  ASK SELF TO resetPtimes;
   {resets the times prospected that day back to zero}

 END WAIT;
END WHILE;
 oldapplicant := ASK SELF Last();

 IF (oldapplicant <> NILOBJ)
  OUTPUT; OUTPUT("Recruiter ", recID, " list still
                 occupied!:");
  OUTPUT("Applicant ", oldapplicant.appID, " on list is in
            stage: ", oldapplicant.apstage); OUTPUT;
  ASK SELF TO RemoveThis(oldapplicant);
 END IF;

END METHOD; {newday}

TELL METHOD generateApplicants;

 {Method Description:  Method's purpose is to generate      }
 {pseudo applicants.  This method will run as long as the   }
 {recruiter hasn't prospected his or her max amount of      }
 {prospecting time.  What keeps this program from constantly}
 {running is the fact that a generated pseudo applicant must}
```

```
{get a recruiter resource and go to prospecting before     }
{another pseudo applicant is generated.  Extra logic has   }
{been added to take care of having prospected less than the}
{total amount possible but not having a lot of time left to}
{prospect.                                                 }

VAR
 applicant,oldapplicant : applicantObj;
 transmitActivityID : ACTID;

BEGIN
 WHILE ((maxPtime > dayPtime) AND ((maxPtime - dayPtime) >
        tprospectTime[2]))
  daydone:=0;
  {OUTPUT("maxPtime ",maxPtime," dayPtime ",dayPtime, " in
           generate apps for recruiter ",recID, " Time
          ",SimTime);}

  {Simplifying assumption as long as the recruiter has not }
  {yet prospected that day more than the maxPtime and      }
  {still has at least as much time left to prospect as the }
  {average time it takes to generate a telephone prospect  }
  {we will let him prospect.
                            }
  {OUTPUT("Entered generate applicants at time ",SimTime,"
          Recruiter ",recID);}
  NEW (applicant);
  {Create a potential recruit}
  ASK applicant TO

  setFields(saleTime,salepaperTime,processTime,
  medicalTime,moralTime,stream1, stream2, prospectpri,0);

  {*******************************************************}
  {This section of code will be run if there are waiver    }
  {processes or other processes that have a lower          }
  {priority than the prospecting process.                  }
  {*******************************************************}

  oldapplicant:= ASK SELF Last();

   IF (oldapplicant <> NILOBJ)          AND
      (recruiter.Resources = 0)        AND
      {no circuits are available }
      (applicant.priority > oldapplicant.priority)
      {newapplicant is of higher priority }
      OUTPUT("Inside interupt loop in generate apps
```

```
        ",SimTime);
  InterruptMethod (oldapplicant.transID);
  INC (rstatsRec.interruptedTasks);
  ASK recruiter TO Transfer (oldapplicant,applicant,1);
  ASK  SELF TO RemoveThis (oldapplicant);
  ASK applicant TO changepriority(prospectpri);
  ASK applicant TO setstage(prospectstage);
  ASK SELF TO Add(applicant);
   {puts applicant into its list notifies recruiter
    taken}
  transmitActivityID := TELL SELF TO
  prospect(applicant);
  ASK  applicant TO setTransmitID (transmitActivityID);
 {*}IF oldapplicant.apstage <> depstage
     TELL SELF TO grabrecruiter(oldapplicant) IN 0.0;
    END IF;

ELSE  {applicant will wait for recruiter resource}
      {OUTPUT("waiting for recruiter resource in }
      {Generate apps at time ",SimTime);         }

 WAIT FOR recruiter TO
  PriorityGive(applicant,1,applicant.priority);
  ASK applicant TO changepriority(prospectpri);
  ASK applicant TO setstage(prospectstage);
  ASK SELF TO Add(applicant);
  {puts applicant into its list notifies recruiter taken}
  transmitActivityID := TELL SELF TO prospect(applicant)
  IN 0.0;
  ASK  applicant TO setTransmitID (transmitActivityID);

  {The wait above keeps the loop from continuously   }
  {running because it needs the recruiter to be free }
  {before it can send the pseudoapplicant to the     }
  {prospect method. In prospect a delay occurs before}
  {the pseudoapplicant is really considered to be an }
  {applicant.  This is done so we can have all       }
  {interrupts pull the item off of the ranked list   }
  {object and interrupt by referring to the          }
  {transmitActivityID.                               }

  ON INTERRUPT
   ASK recruiter TO Cancel (applicant,1);
  TERMINATE;
 END WAIT;

END IF;
```

```
  END WHILE;
   ASK calendar TO updatetime;
   OUTPUT("Used up my prospecting time for in generate apps
           for recruiter ",recID, " working day
          ",calendar.monthdate, " time ", SimTime);
   daydone:=1;

END METHOD; {generateApplicants}

TELL METHOD prospect (IN applicant : applicantObj);

 {Method Description:  The prospect method is reached from}
 {generate applicants when a pseudo applicant has been    }
 {created and has gotten the recruiter resource.  Here the}
 {pseudo applicant waits until the time corresponding to  }
 {the type of applicant he or she is has elapsed.  After  }
 {the time has elapsed the recruiter resource is released }
 {and the applicant is sent on to a sales appointment,    }
 {after of course an appropriate delay. Some of the added }
 {logic takes care of a recruiter prospecting for a time  }
 {and getting interrupted before the pseudo applicant     }
 {becomes a true applicant.  Credit is given to the       }
 {recruiter for that time and if the recruiter comes back }
 {to prospect again in the same day he has a "credit" and }
 {has to wait that amount of time less before his pseudo  }
 {applicant becomes a new applicant.                      }


 VAR
  waitduration,tempPtime, tempduration,temptime : REAL;
  transmitActivityID : ACTID;

  {Variable description:                                  }

  {tempduration: The time it would normally have taken the  }
  {recruiter to prospect an applicant.                      }

  {waitduration: Temp duration gets reduced by currentPtime,}
  {the amount of time the recruiter prospected earlier in   }
  {the day before being interrupted. If the recruiter wasn't}
  {interrupted then this is the time recruiter will wait.   }

  {tempPtime: The time the recruiter begins recruiting.     }
  {This is used to give the recruiter credit for time       }
  {prospected is he is interupted.  Simplifying  assumption }
  {the recruiter was engaged in the same type of prospecting}
```

```
{if interupted earlier.  We will not keep track if          }
{currentPtime was caused by a face to face prospecting      }
{being interupted or a telephone prospecting getting        }
{interupted.                                                }

BEGIN

 temptime := SimTime;
 OUTPUT("made it prospecting for recruiter ",recID, " at
        time ",SimTime);
 tempPtime:=SimTime;
 {note when the recruiter starts prospecting allows the}
 {walkin procedure to know what the recruiter is doing.}

 activitypri:=prospectpri;

  IF (applicant.appProspCat = telephone)
   tempduration:=stream3.Triangular(tprospectTime[1],
   tprospectTime[2],tprospectTime[3]);
   {Time units will be minutes}
  ELSE
   tempduration:=stream3.Triangular(fprospectTime[1],
   fprospectTime[2],fprospectTime[3]);
  END IF;

  IF (currentPtime = 0.0)
   waitduration:=tempduration;
  ELSE
   IF ((tempduration - currentPtime) <= 0.0)
    waitduration := 0.0;
    tempduration := 0.0;
   ELSE
    waitduration := (tempduration - currentPtime);
    {Still have some time left to prospect before get next
     applicant}
    tempduration := 0.0;
   END IF;
  END IF;

  IF ((waitduration + dayPtime) < maxPtime)
   {Then there was enough time to generate an applicant}

   WAIT DURATION waitduration

    {**} ASK recruiter TO TakeBack(applicant,1);
    {give back the recruiter resource}
    ASK SELF TO RemoveThis(applicant);
```

```
      rstatsRec.prospectDur:= (rstatsRec.prospectDur +
      (SimTime - temptime));
      INC (rstatsRec.numberOfapplicants);
      INC(rstatsRec.totalcreated);
      INC(appnum);
      ASK applicant TO setid(appnum);
      dayPtime:= (waitduration + currentPtime + dayPtime);
      {adds in how much time the recruiter prospected}
      {OUTPUT("This is the dayPtime in prospect ",dayPtime, "
               for recruiter ", recID, " at time ",SimTime);}
      currentPtime :=0.0;
      OUTPUT("applicant ",applicant.appID," created by
              recruiter ",recID," at time ",SimTime);
      {OUTPUT("applicants wait duration ",waitduration, " for
               recruiter ",recID);}
      ASK applicant TO changestatus;
      {Just sets status to 1 to say no longer pseudoapl}
      ASK applicant TO changepriority(sellpri);
      ASK applicant TO setstage(sellstage);

      IF (stream3.UniformReal(0.0,1.0) < lossProbs[1])
      {is going to be a no show}

       OUTPUT("applicant ",applicant.appID," no show for
               recruiter ",recID," at time ",SimTime);
       DISPOSE(applicant);
       DEC(rstatsRec.numberOfapplicants);
       INC(rstatsRec.lostatprospect);

      ELSE

       ASK applicant TO setarrivalTime(SimTime);
       {notes when the applicant was created}
       TELL SELF TO grabrecruiter(applicant) IN
       stream3.Triangular(tosalesdelay[1],tosalesdelay[2],
       tosalesdelay[3]);
       ASK  applicant TO setTransmitID (transmitActivityID);
       {This sets up sales to receive an applicant with the }
       { proper priority and set up the activity ID         }
       {So this particular sales method can be interupted   }

      END IF;

ON INTERRUPT
       currentPtime:=currentPtime + (SimTime - tempPtime);
       {keeps track of how much time a recruiter prospected
        before being interupted}
```

```
      rstatsRec.prospectDur:= (rstatsRec.prospectDur +
      (SimTime - temptime));
     TERMINATE;
    END WAIT;

   ELSE

    {There wasn't enougth time to generate another applicant}
    dayPtime := maxPtime;  {we will therefore make sure the
    recruiter doesn't try again}
    ASK recruiter TO TakeBack(applicant,1);
    {give back the recruiter resource}
    ASK SELF TO RemoveThis(applicant);
    DISPOSE(applicant);

   END IF;

END METHOD; {prospect}



TELL METHOD sell(IN applicant : applicantObj);

 {Method Description:  Due to the high priority of this      }
 {activity it almost always interrupts another method.  Here}
 {the applicants must wait to be sold or not. Walk-ins are a}
 {special category because they did not have to go through  }
 {the pre-sell telephone interview process.  Therefore a    }
{large portion of them, currently modeled at 60percent  fail a   }
 {prequal interview. If they fail the recruiter doesn't      }
 {have to spend his time selling.  If they are               }
 {acceptable a recruiter still has to go through the         }
 {sales interview, however with a lot less pressure.         }

 VAR
  waitduration, presalesduration, apriority,tempTime : REAL;
  oldapplicant, toldapplicant : applicantObj;
  transmitActivityID : ACTID;

 BEGIN

  tempTime:= SimTime;
  {Keep track when an applicant begins sales interview}

  IF (applicant.priority = walkinpri)   {applicant would be
   coming to sales directly from ask method generate walkins}
```

```
 activitypri:=walkinpri;
 INC(rstatsRec.numWalkins);
 OUTPUT("Walkin applicant ",applicant.appID," arrives at
        sales for recruiter ",recID," at time ",SimTime);

ELSE

 activitypri:=sellpri;  {allows walkin procedure to know
 what the recruiter is doing}

END IF;

IF (applicant.priority = walkinpri)
 presalesduration:=stream3.Triangular(presaleTime[1],
 presaleTime[2],presaleTime[3]);

 WAIT DURATION presalesduration
  OUTPUT("WALKIN applicant ",applicant.appID," successfully
         finished PRESALES interview with recruiter"
        ,recID," at time ",SimTime);
 ON INTERRUPT
  rstatsRec.salesDur:= (rstatsRec.salesDur + (SimTime -
  tempTime));
  TERMINATE;

 END WAIT;
 rstatsRec.salesDur:= (rstatsRec.salesDur + (SimTime -
 tempTime));
 tempTime:= SimTime;  {Reset the tempTime to reflect when a
 walkin actually begins sales interview}

 IF ((applicant.priority = walkinpri) AND
  (stream3.UniformReal(0.0,1.0) <= lossProbs[2]))
  {walkin failed the presales qualification}

  OUTPUT("WALKIN applicant ",applicant.appID," FAILED
          PRESALES interview with recruiter ",recID," at
          time ",SimTime);

  ASK recruiter TO TakeBack(applicant,1);
  {give back the recruiter resource}

  ASK SELF TO RemoveThis(applicant);
  {it is no longer an applicant}
  DISPOSE(applicant);
  {get rid of the object to free up memory}
  DEC(rstatsRec.numberOfapplicants);
```

```
  {no longer an applicant so one less}
  INC(rstatsRec.lostatPsales);

  TERMINATE;
  {do not need to go through rest of code}
 END IF;
END IF;

WAIT DURATION applicant.sellTime
  {This will be based on recruiter info later}
  ASK recruiter TO TakeBack(applicant,1);
  {give back the recruiter resource}
  ASK SELF TO RemoveThis(applicant);
  {it has been sold}{**}
  OUTPUT("applicant ",applicant.appID," finished sales
          interview with recruiter ",recID," at time
         ",SimTime);
  rstatsRec.salesDur:= (rstatsRec.salesDur + (SimTime -
  tempTime));

  IF (stream3.UniformReal(0.0,1.0) < lossProbs[3])
  {is going to be a no show for processing}

  OUTPUT("applicant ",applicant.appID," was not sold by
          recruiter ",recID," at time ",SimTime);
  DISPOSE(applicant);
  DEC(rstatsRec.numberOfapplicants);
  INC(rstatsRec.lostatsales)

 ELSE

  IF (applicant.processCat = immediate)

   ASK applicant TO changepriority(Iprocesspri);
   ASK applicant TO setstage(Iprocessstage);
   TELL SELF TO grabrecruiter(applicant) IN
   stream3.Triangular(toprocessdelay[1],toprocessdelay[2],
   toprocessdelay[3]);

  ELSIF (applicant.processCat = normal)

   ASK applicant TO changepriority(Nprocesspri);
   ASK applicant TO setstage(Nprocessstage);
   TELL SELF TO grabrecruiter(applicant) IN
   stream3.Triangular(toprocessdelay[1],toprocessdelay[2],
   toprocessdelay[3]);
```

```
    ELSE

     OUTPUT("THERE IS A MISSING CATEGORY IN SALES");

    END IF;

   END IF; {for the no show}

   ON INTERRUPT
    rstatsRec.salesDur:= (rstatsRec.salesDur + (SimTime -
    tempTime));
    ASK applicant TO updateprocesTime(SimTime -
    tempTime,sales);
    TERMINATE;
  END WAIT;

END METHOD; {sell}

TELL METHOD testing(IN applicant : applicantObj);

 {Method Description:  This method just uses given          }
 {probabilities to see if the applicant would pass the test.}
 {A recruiter resource is not needed for this method because}
 {a recruiter is not needed in real life.                   }

 VAR
  tester : REAL;

 BEGIN

  ASK calendar TO updatetime;
  tester := stream2.UniformReal(0.0,1.0);
  IF ( tester < probtestpass)
   ASK applicant TO passedtest(1,asvab);
   {applicant passed the test}

   IF (applicant.processCat = immediate)
    OUTPUT("applicant ",applicant.appID," in IMD process
            passed test for recruiter ",recID," at time
           ",SimTime);
   ELSE
    OUTPUT("applicant ",applicant.appID," in NOR process
            passed test for recruiter ",recID," at time
           ",SimTime);
   END IF;

  ELSIF (tester >= probtestpass)
```

```
    ASK applicant TO passedtest(0,asvab);
    {applicant failed the test}
    OUTPUT("applicant ",applicant.appID," failed test for
            recruiter ",recID," at time ",SimTime);
  END IF;

  {What follows is a departure from the program norm of}
  {saving activity id so it can be interrupted. This    }
  {is because there is no wait and a recruiter is not   }
  {needed for this. The only reason why it is a method }
  {is to make it easy to have it occur about 7 days     }
  {after an applicant starts processing.                }

  IF (applicant.processCat = immediate)
   TELL SELF TO immediateprocess(applicant);
  ELSIF (applicant.processCat = normal)

   IF (applicant.Testpass = 1)

    TELL SELF TO grabrecruiter(applicant) IN
    (calendar.hoursPerday * 5.0 * 60.0);
    {applicant will need a wait of avg 5 days to go to MEPS}

   ELSE
    TELL SELF TO normalprocess(applicant);
   END IF;

  ELSE

   OUTPUT("Problem in testing applicant doesn't has a process
           category");
  END IF;

END METHOD; {testing}

TELL METHOD moralwaiver(IN applicant : applicantObj);

 {Method Description:  This method assumes that an applicant}
 {is arriving already put upon the list and already got the }
 {recruiter resource.  Since waivers can be interrupted so  }
 {easily by methods of higher priority we keep track of how }
 {long an applicant's waiver has been worked on. When the }
 {time is complete they are done.  Additionally if too much }
 {time has passed their waiver gets set to a high priority  }
 {to allow them to get the recruiter resource easily. }
```

```
VAR
    tempTime : REAL;

 BEGIN

    tempTime := SimTime;
    {keeps track of how long the waiver has been worked on}

    IF (applicant.nummoralinterrupts = 0)
     ASK applicant TO startwaivertime(moral);
    END IF;

    IF ((SimTime - applicant.moralstart) > 15000.0)
     ASK applicant TO changepriority(waiverHighpri);
    END IF;

    WAIT DURATION applicant.moralTime
     ASK recruiter TO TakeBack(applicant,1);
     {give back the recruiter resource}

     ASK SELF TO RemoveThis(applicant);
     {paperwork complete}

     rstatsRec.waiversDur := rstatsRec.waiversDur +
     applicant.moralTime;
     OUTPUT("applicant ",applicant.appID," finished Moral
             Waiver paperwork with recruiter ",recID," at time
            ",SimTime);

     IF (stream2.UniformReal(0.0,1.0) < lossProbs[7])
      ASK applicant TO passedtest(0,moral);
      DISPOSE(applicant);
      DEC(rstatsRec.numberOfapplicants);
      INC(rstatsRec.lostatMOR);
     ELSE
      ASK applicant TO passedtest(1,moral);
      IF ((applicant.needMedical = 1) AND
          (applicant.Medicalpass = 2))
       TELL SELF TO grabrecruiter(applicant) IN 0.0;
      ELSE

       ASK applicant TO changepriority(deppri);
       ASK applicant TO setstage(depstage);
       rstatsRec.appTime:= (rstatsRec.appTime + (SimTime -
       applicant.arrivalTime));
       ASK SELF TO sendtoprocess(applicant);
```

```
    END IF;
   END IF;

   ON INTERRUPT

    rstatsRec.waiversDur := rstatsRec.waiversDur + (SimTime -
    tempTime);
    ASK applicant TO addwaiverinterrupt(moral);
    ASK applicant TO updateprocesTime(SimTime -
    tempTime,moral);
    TERMINATE;

  END WAIT;

END METHOD; {moralwaiver}


TELL METHOD medicalwaiver(IN applicant : applicantObj);

 {Method Description:  This method assumes that an applicant}
 {is arriving already put upon the list and already got the }
 {recruiter resource.  Since waivers can be interupted so   }
 {easily by methods of higher priority we keep track of how }
 {long an applicant's waiver has been worked on so when the }
 {time is complete they are done.  Additionally if too much }
 {time has passed their waiver gets set to a high priority  }
 {to allow them to get the recruiter resource easily.       }

 VAR
  tempTime : REAL;


 BEGIN

  tempTime := SimTime;
  {keeps track of how long the waiver has been worked on}

  IF (applicant.nummedicalinterrupts = 0)
   ASK applicant TO startwaivertime(medical);
  END IF;

  IF ((SimTime - applicant.medicalstart) > 15000.0)
   ASK applicant TO changepriority(waiverHighpri);
  END IF;

  WAIT DURATION applicant.medicalTime
   ASK recruiter TO TakeBack(applicant,1);
```

```
    {give back the recruiter resource}
    ASK SELF TO RemoveThis(applicant);
    {paperwork complete}
    rstatsRec.waiversDur := rstatsRec.waiversDur +
    applicant.medicalTime;

    OUTPUT("applicant ",applicant.appID," finished Medical
            Waiver paperwork with recruiter ",recID," at time
           ",SimTime);

    IF (stream2.UniformReal(0.0,1.0) < lossProbs[6])

     ASK applicant TO passedtest(0,medical);
     DISPOSE(applicant);
     DEC(rstatsRec.numberOfapplicants);
     INC(rstatsRec.lostatMED);

    ELSE

     ASK applicant TO passedtest(1,medical);
     ASK applicant TO changepriority(deppri);
     ASK applicant TO setstage(depstage);
     ASK SELF TO sendtoprocess(applicant);
     rstatsRec.appTime:= (rstatsRec.appTime + (SimTime -
     applicant.arrivalTime));

    END IF;

    ON INTERRUPT

     rstatsRec.waiversDur := rstatsRec.waiversDur + (SimTime
     - tempTime);
     ASK applicant TO addwaiverinterrupt(medical);
     ASK applicant TO updateprocesTime(SimTime -
     tempTime,medical);
     TERMINATE;

   END WAIT;

END METHOD; {medicalwaiver}


TELL METHOD immediateprocess(IN applicant : applicantObj);

  {Method Description:  This process is used with applicants }
  {that are walkins who generally want to join the Army and  }
```

```
{don't need to be sold as much and people who when they     }
{come in for a sales interview and asked "so do you want to}
{join?" say yes.  The recruiter will have the applicant     }
{take a pre-test and if the applicant passes then they will}
{get sent on to MEPS to take the physical and the test at   }
{the same time.  The testing in this method really refers   }
{to the pretest.  However, based on conversations with a    }
{recruiter, Sgt.  Kendrick it is very rare for someone who }
{has passed the pretest to fail the real test.  Thus we are}
{assuming that the percentages of passing the pretest and   }
{the real test are the same and can use the same method in }
{this model.  Applicants arrive with a recruiter resource   }
{and wait the average amount of time it takes for them to   }
{get the paperwork done.  Then they get scheduled to        }
{test/MEPS.  Meps isn't modeled, needing a waiver is.       }

VAR
 apriority, tempTime,thirtydays,threedays : REAL;
 oldapplicant, Toldapplicant : applicantObj;
 disposedflag : INTEGER;
 {used to flag an applicant decided against the Army}
 transmitActivityID : ACTID;

BEGIN
 disposedflag := 0;
 {protects from disposing an applicant with a resource}
 activitypri:=Iprocesspri;
 {allows walkin procedure to know what the recruiter is
  doing}
 ASK applicant TO changepriority(Iprocesspri);
 ASK applicant TO setstage(Iprocessstage);
 thirtydays:=(calendar.hoursPerday * FLOAT(calendar.wdays) *
 4.0 * 60.0);  {4 weeks}
 threedays:= (calendar.hoursPerday * 3.0 * 60.0);  {3 days}
 tempTime := SimTime;

{********}IF (applicant.processed = 0)

    {The applicant hasn't completed processing paperwork IF  }
    {it enters here it hasn't been processed and already has }
    {the recruiter resource and the applicant is on the list.}
    {Otherwise it doesn't have the recruiter resource and the}
    {applicant is not on the list, for now.                  }

    WAIT DURATION applicant.processtime
     ASK recruiter TO TakeBack(applicant,1);
     {give back the recruiter resource}
```

```
      ASK SELF TO RemoveThis(applicant);
      {paperwork complete}
      ASK applicant TO changeprocessed;
      OUTPUT("applicant ",applicant.appID," finished IMDprocess
              paperwork with recruiter ",recID," at time
             ",SimTime);
      rstatsRec.processDur := rstatsRec.processDur +
      applicant.processtime;

      IF (stream2.UniformReal(0.0,1.0) < lossProbs[4])
       {Decides against the Army}

       OUTPUT("applicant ",applicant.appID," left IMDprocessing
               for recruiter ",recID," at time ",SimTime);
       DISPOSE(applicant);
       DEC(rstatsRec.numberOfapplicants);
       INC(rstatsRec.lostatIMD);
       {just keeping track of how many applicants are lost at
        this stage}
       disposedflag := 1;

      END IF; {for the decides against the Army}

      ON INTERRUPT

        rstatsRec.processDur := rstatsRec.processDur + (SimTime
                                - tempTime);
       ASK applicant TO updateprocesTime(SimTime -
       tempTime,process);
       TERMINATE;

     END WAIT;

{****** } END IF;
     {all of the code above gets bypassed by a person who has
      complete the paperwork}

     {=========================================================}
     {Code used for waivers and tests applicants no longer     }
     {have recruiter resource.                                 }
     {=========================================================}

     IF (disposedflag = 0)

      IF (applicant = NILOBJ)
       OUTPUT("in protected region IN IMD PROCESS AND APPLICANT
               NIL OBJECT");
```

```
END IF;

IF ((applicant.Testpass = 0) AND
   (applicant.numtestattempts = 0))

 TELL SELF TO testing(applicant) IN threedays;
 {first attempt occurs in three days}

ELSIF ((applicant.Testpass = 0) AND
      (applicant.numtestattempts = 1))

 DISPOSE(applicant);
 DEC(rstatsRec.numberOfapplicants);
 {not allowing retest in 30 days yet need more info}
 INC(rstatsRec.lostattest);
 {30 day delay has been tested however}
 disposedflag := 1;
 {TELL SELF TO testing(applicant) IN thirtydays; }
 {second attempt occurs in thirty days}

ELSIF ((applicant.Testpass = 0) AND
      (applicant.numtestattempts > 1))

 OUTPUT("applicant ",applicant.appID," did not pass test
         twice for ",recID," at time ",SimTime, " and was
         killed.");
 DISPOSE(applicant);
 DEC(rstatsRec.numberOfapplicants);
 INC(rstatsRec.lostattest);
 disposedflag := 1;

ELSIF (applicant.needMoral = 1) AND
      (applicant.Moralpass = 2)

 ASK applicant TO changepriority(waiverpri);
 ASK applicant TO setstage(waiverstage);
 TELL SELF TO grabrecruiter(applicant) IN
 stream2.Triangular(tomoraldelay[1],tomoraldelay[2],
 tomoraldelay[3]);
 {applicant will arrive to get medical waiver according}
 {to its triangular distribution                        }

ELSIF (applicant.needMedical = 1) AND
      (applicant.Medicalpass = 2)

 ASK applicant TO changepriority(waiverpri);
 ASK applicant TO setstage(waiverstage);
```

```
    TELL SELF TO grabrecruiter(applicant) IN
    stream3.Triangular(tomedicaldelay[1],tomedicaldelay[2],
    tomedicaldelay[3]);
    {applicant will arrive to get moral waiver according to}
    {its triangular distribution                            }

   ELSIF ((applicant.needMoral = 0) AND
         (applicant.needMedical = 0))

    OUTPUT("applicant ",applicant.appID," did not need a
            waiver in IMDprocess for recruiter ",recID," at
            time ",SimTime);
    ASK applicant TO changepriority(deppri);
    ASK applicant TO setstage(depstage);
    rstatsRec.appTime:= (rstatsRec.appTime + (SimTime -
    applicant.arrivalTime));
    ASK SELF TO sendtoprocess(applicant);

   END IF;  {for waivers/tests}
  END IF;   {protection against attempting to delete an}
            {already deleted applicant                  }

END METHOD; {Immediateprocess}

TELL METHOD normalprocess(IN applicant : applicantObj);

 {Method Description:  This is the process that applicants  }
 {go through when they are prospected and have to be lead   }
 {every step of the way to join the Army.  They need to test}
 {first and then they fill out the paperwork and then they  }
 {go to MEPS.  To model this, applicants have to pass the   }
 {test first before any paperwork is done then get paperwork}
 {done and then go to MEPS.  An applicant has to get a      }
 {recruiter resource just for a second to start the         }
 {test.  This is a simplifying assumption to make modeling  }
 {it easier.  The applicant when he or she arrives at the   }
 {process has a recruiter resource.  If they haven't passed }
 {the test they give back the resource and take the test.   }
 {Otherwise they go on to do the paperwork. If this method  }
 {is interrupted while an applicant is processing, the      }
 {applicant gets credit for the amount of processing that   }
 {has already been accomplished. Once the applicant is done }
 {processing, the resource is given back.  If the applicant }
 {needs a waiver, predetermined in set-fields, the applicant}
 {will go on to compete for the resource and go to waivers. }

 VAR
```

```
apriority, tempTime,thirtydays,sevendays : REAL;
oldapplicant : applicantObj;
disposedflag : INTEGER;
{used to flag an applicant decided against the Army}
transmitActivityID : ACTID;


BEGIN

ASK calendar TO updatetime;
disposedflag := 0;
activitypri:=Nprocesspri;
{allows walkin procedure to know what the recruiter is
 doing}
ASK applicant TO changepriority(Nprocesspri);
ASK applicant TO setstage(Nprocessstage);
thirtydays:=(calendar.hoursPerday * FLOAT(calendar.wdays) *
4.0 * 60.0);  {4 weeks}
sevendays:= (calendar.hoursPerday * FLOAT(calendar.wdays) *
60.0);  {1 week}

{*********************************************************}
{This logic deals with testing which doesn't require the  }
{recruiter                                                }
{*********************************************************}

IF ((applicant.Testpass = 0) AND
       (applicant.numtestattempts = 0))

 ASK recruiter TO TakeBack(applicant,1);
 {give back the recruiter resource}
 ASK SELF TO RemoveThis(applicant);
 TELL SELF TO testing(applicant) IN sevendays;
 {first attempt occurs in three days}

ELSIF ((applicant.Testpass = 0) AND
      (applicant.numtestattempts = 1))

 DISPOSE(applicant);
 DEC(rstatsRec.numberOfapplicants);
 INC(rstatsRec.lostattest);
 disposedflag := 1;
 {TELL SELF TO testing(applicant) IN thirtydays;}
 {second attempt occurs in thirty days}

ELSIF ((applicant.Testpass = 0) AND
      (applicant.numtestattempts > 1))
```

```
 OUTPUT("applicant ",applicant.appID," did not pass test
        twice for ",recID," at time ",SimTime, " and was
        killed.");
 DISPOSE(applicant);
 DEC(rstatsRec.numberOfapplicants);
 INC(rstatsRec.lostattest);
 disposedflag:=1;

ELSIF (applicant.Testpass = 1)
 {*******************************************************}
 {End of test logic code underneath here will only be     }
 {visited when the applicant has tested and passed the    }
 {test or failed the test twice, then disposed flag will  }
 {let the remaining code be bypassed.                     }
 {*******************************************************}
 {As soon as it passed the test it grabbed a recruiter and}
 {came here                                               }

 IF ((applicant.processed = 0) AND (disposedflag = 0))
  {The applicant hasn't completed processing paperwork}

  tempTime:= SimTime;

  {This is set up differently because process time will  }
  {have at all times the amount of time needed to finish }
  {processing.                                           }

  OUTPUT("applicant ",applicant.appID, " NOR process
          recruiter ",recID, " processtime
         ",applicant.processtime," time ",SimTime);

  WAIT DURATION applicant.processtime
   ASK recruiter TO TakeBack(applicant,1);
   {give back the recruiter resource}
   ASK SELF TO RemoveThis(applicant);
   {paperwork complete}
   ASK applicant TO changeprocessed;
   OUTPUT("applicant ",applicant.appID," finished
           NORprocess paperwork with recruiter ",recID,"
           at time ",SimTime);
   rstatsRec.processDur := rstatsRec.processDur +
   applicant.processtime;

   IF (stream2.UniformReal(0.0,1.0) < lossProbs[4])
    {Decides against the Army}
```

```
  OUTPUT("applicant ",applicant.appID," DAGA
          NORprocessing for recruiter ",recID," at time
         ",SimTime);
  DISPOSE(applicant);
  DEC(rstatsRec.numberOfapplicants);
  INC(rstatsRec.lostatNOR);
  disposedflag := 1;

 END IF; {for the decides against the Army}

 ON INTERRUPT
  rstatsRec.processDur := rstatsRec.processDur + (SimTime
                          - tempTime);
  ASK applicant TO updateprocesTime(SimTime -
  tempTime,process);
  TERMINATE;
 END WAIT;

END IF;
 {end of if that checks to see if it has already}
 {been processed                                 }

 {+++++++++++++++++++++++++++++++++++++++++++++++++++}
 {Start of the Waiver code                          }
 {+++++++++++++++++++++++++++++++++++++++++++++++++++}

IF ((disposedflag = 0) AND (applicant.processed = 1))

 IF ((applicant.needMoral = 1) AND (applicant.Moralpass =
  2))
  ASK applicant TO changepriority(waiverpri);
  ASK applicant TO setstage(waiverstage);
  TELL SELF TO grabrecruiter(applicant) IN
  stream2.Triangular(tomoraldelay[1],tomoraldelay[2],
  tomoraldelay[3]);
  {applicant will arrive to get medical waiver}
  {according to its triangular distribution   }

 ELSIF ((applicant.needMedical = 1) AND
       (applicant.Medicalpass = 2))

  ASK applicant TO changepriority(waiverpri);
  ASK applicant TO setstage(waiverstage);
  TELL SELF TO grabrecruiter(applicant) IN
  stream3.Triangular(tomedicaldelay[1],tomedicaldelay[2],
  tomedicaldelay[3]);
  {applicant will arrive to get moral waiver according to}
```

```
     {its triangular distribution                                  }

    ELSIF ((applicant.needMoral = 0) AND
          (applicant.needMedical = 0))

     OUTPUT("applicant ",applicant.appID," did not need
             waiver in NORprocess for recruiter ",recID," at
             time ",SimTime);
     ASK applicant TO changepriority(deppri);
     ASK applicant TO setstage(depstage);
     rstatsRec.appTime:= (rstatsRec.appTime + (SimTime -
     applicant.arrivalTime));
     ASK SELF TO sendtoprocess(applicant);

    END IF;  {for waivers/tests}
   END IF;
    {protection against attempting to delete an already }
    {deleted applicant                                  }

END IF;
   {This is the end of the If statement checking it testing}
   {is done                                                 }

END METHOD;{normalprocess}


TELL METHOD sustain(IN applicant : applicantObj);

 CONST
 PRIORITYCHANGE = 2.0;
 FIRSTMTGCHANGE = 10.0;

 VAR
  depLength, currentTime     : REAL;
  mtgMins, mtgWks, tempTime  : REAL;
  lastDEPTime, nextMonth     : REAL;
  mtgsMissed                 : INTEGER;

 BEGIN

  ASK calendar TO updatetime;
  {******* Initialize variables *******}
  activitypri := deppri;
  ASK applicant TO setLossProb(DepLossProb);
  depLength := sustainStream.Triangular(DepLength1,
  DepLength2, DepLength3);   {months}
  depLength := (depLength / 4.0) * calendar.whours * 60.0;
```

```
currentTime := SimTime;
lastDEPTime := currentTime + depLength;

IF (lastDEPTime > runLength)
 lastDEPTime := runLength;
END IF;

nextMonth := currentTime + (4.0 * calendar.whours * 60.0);

{****************** Initial DEP Meeting ****************}
mtgWks := sustainStream.Exponential( 4.0 / FLOAT(calendar.wdays) );
mtgMins := mtgWks * calendar.whours * 60.0; {Convert mtgWks to minutes}
ASK applicant TO changepriority(applicant.priority + FIRSTMTGCHANGE);
 deltaPri := FIRSTMTGCHANGE;
 mtgsMissed := 1;     {Flags the need to decrement priority}

{*********** Loops until DEP time has expired ***********}

WHILE ( ( (currentTime + mtgMins) < lastDEPTime) AND
      (applicant <> NILOBJ) AND
      (currentTime <= runLength) );

 OUTPUT;
 OUTPUT("Applicant ", applicant.appID, " going through DEP
        at ", currentTime);
 OUTPUT("Meeting Weeks: ", mtgWks, "Date: ",
        calendar.month, " ", calendar.monthdate);
 OUTPUT("last time in DEP: ", lastDEPTime);
 OUTPUT("Recruiter: ", recID);
 OUTPUT;

 WAIT DURATION mtgMins
    IF (SimTime > runLength) TERMINATE; END IF;
       tempTime := SimTime;
       WAIT FOR SELF TO depMeet(applicant, mtgWks)
       IF (SimTime > runLength) TERMINATE; END IF;
       IF (mtgsMissed > 0)
          DEC(mtgsMissed);
          ASK applicant TO changepriority(applicant.priority
  - deltaPri);
          deltaPri := PRIORITYCHANGE;
          OUTPUT; OUTPUT("Applicant ", applicant.appID,
                    " PRIORITY BACK TO ", applicant.priority);
          OUTPUT;
       END IF;
    ON INTERRUPT
      INC(mtgsMissed);
```

```
      ASK applicant TO changepriority(applicant.priority + deltaPri);
         OUTPUT;
         OUTPUT("APP # ", applicant.appID,
                " DEP MTGS MISSED :", mtgsMissed);
         OUTPUT("NEW PRIORITY: ", applicant.priority); OUTPUT;
         IF (SimTime > runLength) TERMINATE; END IF;
    {Greater probability of DEP loss for a missed meeting}
  rstatsRec.depDur := rstatsRec.depDur + (SimTime - tempTime);
  ASK applicant TO setLossProb(applicant.lossProb +
                                     DepLossProb * mtgWks / 4.0);
  OUTPUT;
         OUTPUT("Applicant ", applicant.appID,
 " interrupted waiting for DEP mtg at ", SimTime);
         OUTPUT;
         IF (recruiter.NumberAllocatedTo(applicant) = 1)
            ASK recruiter TO TakeBack(applicant,1);
            {give back the recruiter resource}
            ASK SELF TO RemoveThis (applicant);
         END IF;
      END WAIT;

      ON INTERRUPT
  OUTPUT("Applicant ", applicant.appID, " interrupted
               between DEP mtgs at ", SimTime);
    END WAIT;

    ASK calendar TO updatetime;
    currentTime := SimTime;

    IF (sustainStream.UniformReal(0.0, 1.0) <=
        applicant.lossProb)

OUTPUT;
      OUTPUT("Applicant ", applicant.appID, " DEP loss
             at ", currentTime);
      OUTPUT;

IF (recruiter.NumberAllocatedTo(applicant) = 1)
       ASK recruiter TO TakeBack(applicant,1);
       {give back the recruiter resource}
       ASK SELF TO RemoveThis (applicant);
      END IF;
       DISPOSE(applicant);
       INC(rstatsRec.lostatDEP);
       DEC(rstatsRec.numberOfapplicants);
      ELSE
 {********* Increment loss probability by DepLossProb for
```

```
      each month of DEP **********}

 IF (currentTime >= nextMonth)
  nextMonth := nextMonth +
                    (4.0 * calendar.whours * 60.0);
        ASK applicant TO setLossProb(applicant.lossProb +
                                     DepLossProb);
  OUTPUT;
  OUTPUT("Applicant ", applicant.appID, " New month loss
               prob = ", applicant.lossProb, " at ",
               currentTime);
        OUTPUT("OLD LOSS PROB: ", applicant.lossProb -
               DepLossProb
        OUTPUT("Next Month: ", nextMonth);
        OUTPUT;
       END IF;

{Wait an ave. DEPMTG weeks for next DEP meeting}
      {*********************************************}
  mtgWks := sustainStream.Exponential( DepIntArv );
  mtgMins := mtgWks * calendar.whours * 60.0;
        {Convert mtgWks to minutes}
     END IF;
    END WHILE;

    IF (recruiter.NumberAllocatedTo(applicant) = 1)

     ASK recruiter TO TakeBack(applicant,1);
     {give back the recruiter resource}
     ASK SELF TO RemoveThis (applicant);
     OUTPUT("DEALLOCATED RECRUITER BEFORE DUMPING APPLICANT AT
            ", SimTime);

    END IF;

    IF (applicant <> NILOBJ)

     OUTPUT;
     IF (currentTime < runlength)
        OUTPUT("Applicant ", applicant.appID, " finished DEP on
             ", calendar.month, " ", calendar.monthdate);
INC(rstatsRec.numContracts);
     ELSE
      OUTPUT("DEP Applicant ", applicant.appID, " booted before
              EOS ", calendar.month, " ", calendar.monthdate);
     END IF;
```

```
    OUTPUT;
    DISPOSE(applicant);

   END IF;

   TERMINATE;
END METHOD; {sustain}

WAIT FOR METHOD depMeet(INOUT applicant : applicantObj; IN
 mtgWks : REAL);

VAR
 mtgDuration : REAL;

BEGIN

 {*****************************************}
 {Acquire Recruiter for DEP meeting/function }
 {*****************************************}

WAIT FOR SELF TO grabDEPRec(applicant)

 IF (recruiter.NumberAllocatedTo(applicant) = 0)
  OUTPUT;
  OUTPUT("Applicant ", applicant.appID, "No resource to DEP
          even after wait!!!!");
  OUTPUT;
 ELSE
  OUTPUT;
  OUTPUT("Applicant ", applicant.appID, " Resource obtained
          at ", SimTime);
  OUTPUT;
 END IF;

 mtgDuration := sustainStream.Triangular(DepMtgTime1,
                              DepMtgTime2, DepMtgTime3);

 {**************** Go to DEP Meeting *******************}
 WAIT DURATION mtgDuration
  rstatsRec.depDur := rstatsRec.depDur + mtgDuration;

  {****** Bonus for more frequent DEP meetings ******}
  IF (mtgWks < 2.0)
   ASK applicant TO setLossProb(applicant.lossProb -
                                (1.0 / mtgWks) * 0.0005);
   OUTPUT;
   OUTPUT("Applicant ", applicant.appID, " loss prob bonus ",
```

```
          applicant.lossProb);
   OUTPUT;
  END IF;

  IF (recruiter.NumberAllocatedTo(applicant) = 1)
   ASK recruiter TO TakeBack(applicant,1);
   {give back the recruiter resource}
   ASK SELF TO RemoveThis (applicant);
   OUTPUT("Applicant ", applicant.appID, " finished DEP
          interview at ", SimTime);
  ELSE
   OUTPUT;
   OUTPUT("Recruiter not obtained in depMeet! Possible
          interrupt.");
   OUTPUT;
  END IF;

  IF (SimTime > runLength) TERMINATE; END IF;
 END WAIT;

 END WAIT;
END METHOD {depMeet};

ASK METHOD resetPtimes;
 {Method Description:  This method resets the variables  }
 {which keep track of how long a recruiter has prospected}
 {that day and how long the recruiter had prospected     }
 {before being interrupted.                              }

 BEGIN
  dayPtime:=0.0;
  currentPtime:=0.0;
END METHOD; {resetPtimes}

ASK METHOD incrementInterruptedtasks;
 BEGIN
  INC (rstatsRec.interruptedTasks);
END METHOD; {incrementInterruptedtasks}

ASK METHOD MapplicantTime;
 BEGIN
  rstatsRec.appTime:= rstatsRec.appTime /
  (FLOAT(rstatsRec.numberOfapplicants));
END METHOD; {MapplicantTime}

ASK METHOD getutilization;
 {This method keeps track of the recruiters mean utilization
```

```
    and standard deviation}
   BEGIN
    rstatsRec.recutil := ASK recruiter AllocWtdMean;
    rstatsRec.recdev  := ASK recruiter AllocWtdStdDev;
  END METHOD;

END OBJECT; {RecruiterObj}

OBJECT stationObj;

  ASK METHOD StationInit;

   VAR
    i,rlength : INTEGER;
    daylength : REAL;
    stationinput : StreamObj;
    temps : STRING;

   BEGIN
    NEW(stationinput);
    ASK stationinput TO Open("stationfile",Input);

    ASK stationinput TO ReadInt(numRecruiters);
    {number of recruiters at the station   }
    ASK stationinput TO ReadLine(temps);
    ASK stationinput TO ReadInt(rlength);
    {number of days the simulation will run}
    ASK stationinput TO ReadLine(temps);
    ASK stationinput TO ReadReal(lworkweek);
    {length of the workweek in hours       }
    ASK stationinput TO ReadLine(temps);
    ASK stationinput TO ReadInt(numdays);
    {number of days in a workweek          }
    ASK stationinput TO ReadLine(temps);
    ASK stationinput TO ReadReal(maxprospectTime);
    ASK stationinput TO ReadLine(temps);

    ASK stationinput TO Close;

    NEW(calendar);    {This creates a new calendar object which
                       keeps up with month and day}
    ASK calendar TO DateInit(lworkweek,numdays);
    {you have to tell calendar the weeklength and days in week}
    daylength := calendar.hoursPerday;
    runlength := (FLOAT(rlength) * daylength * 60.0);

    {need to add conversion of days to minutes for runlength
```

```
   runlength :=30000.0;}
  NEW (station,1..numRecruiters);
  NEW (recStats, 1..numRecruiters);
  NEW(walkStats);
  ADDMONITOR(numWalkins, walkStats);

  FOR i:=1 TO numRecruiters
   NEW(recStats[i]);
  END FOR;

END METHOD {StationInit};


ASK METHOD SetNumReps(IN reps : INTEGER);
  BEGIN
  numReps := reps;
END METHOD {SetNumReps};

ASK METHOD RecAction(IN action : STRING);

 VAR
  i : INTEGER;

 BEGIN

  FOR i := 1 TO numRecruiters
   IF (action = "Init")
    NEW(station[i]);
   ELSIF (action = "Terminate")
    DISPOSE(station[i]);
   END IF;
  END FOR

END METHOD {RecAction};

ASK METHOD printStats;

 VAR
  i        : INTEGER;
  numApps  : REAL;

 BEGIN
  OUTPUT("Recruiter Statistics for");
  FOR i := 1 TO numRecruiters
   numApps := ASK recStats[i].numberOfapplicantsStats Mean();

   OUTPUT("   Recruiter ", i, ":");
```

```
PRINT(recStats[i].totalcreated) WITH "
      Total Number of Applicants  ****";
PRINT(ASK recStats[i].totalcreatedStats Mean()) WITH "
      Mean:  ****.**";
PRINT(SampleStdDev(ASK recStats[i].totalcreatedStats
      StdDev(), recStats[i].numberOfapplicants))
      WITH "      Standard Deviation:  ****.**";
OUTPUT;
PRINT(recStats[i].numberOfapplicants) WITH "
      Surviving Number of Applicants  ****";
PRINT(numApps) WITH "      Mean:  ****.**";
PRINT(SampleStdDev(ASK recStats[i].numberOfapplicantsStats
      StdDev(), recStats[i].numberOfapplicants))
      WITH "      Standard Deviation:  ****.**";
OUTPUT;
OUTPUT("      Total Walk-ins: ", recStats[i].numWalkins);
OUTPUT("      Mean: ", ASK recStats[i].walkStats Mean());
OUTPUT("      Standard Deviation: ",
SampleStdDev(ASK recStats[i].walkStats StdDev(),
numWalkins));
OUTPUT;
PRINT(ASK recStats[i].lostatprospectStats Mean()) WITH "
      Mean lost at prospecting:          *****.**";
PRINT(ASK recStats[i].lostatPsalesStats Mean())   WITH "
      Mean lost at pre-sales:            *****.**";
PRINT(ASK recStats[i].lostatsalesStats Mean())    WITH "
      Mean lost at sales:                *****.**";
PRINT(ASK recStats[i].lostattestStats Mean())     WITH "
      Mean lost at test:                 *****.**";
PRINT(ASK recStats[i].lostatIMDStats Mean())      WITH "
      Mean lost at immediate processing: *****.**";
PRINT(ASK recStats[i].lostatNORStats Mean())      WITH "
      Mean lost at normal processing:    *****.**";
PRINT(ASK recStats[i].lostatMEDStats Mean())      WITH "
      Mean lost at medical waiver:       *****.**";
PRINT(ASK recStats[i].lostatMORStats Mean())      WITH "
      Mean lost at moral waiver:         *****.**";
PRINT(ASK recStats[i].lostatDEPStats Mean())      WITH "
      Mean lost at sustainment:          *****.**";
OUTPUT;
OUTPUT("      Ave. time spent performing:");
OUTPUT("         Prospecting: ",
       ASK recStats[i].prospectDurStats Mean());
OUTPUT("         Sales:       ",
       ASK recStats[i].salesDurStats Mean());
OUTPUT("         Processing:  ",
       ASK recStats[i].processDurStats Mean());
```

```
    OUTPUT("          Waivers:      ",
           ASK recStats[i].waiversDurStats Mean());
    OUTPUT("          Sustainment: ",
           ASK recStats[i].depDurStats Mean());
    OUTPUT;
    OUTPUT("       TOTAL CONTRACTS: ", recStats[i].numContracts);
   OUTPUT("       Mean: ", ASK recStats[i].numContractsStats Mean());
    OUTPUT("          Standard Deviation: ",
              SampleStdDev(ASK recStats[i].numContractsStats StdDev(),
                  recStats[i].numContracts));
    OUTPUT;
    OUTPUT("       Total Walk-ins: ", recStats[i].numWalkins);
    OUTPUT("          Mean: ", ASK recStats[i].walkStats Mean());
    OUTPUT("          Standard Deviation: ",
                   SampleStdDev(ASK recStats[i].walkStats Std-
Dev(),
                   numWalkins));
    OUTPUT;
    OUTPUT("       Mean time applicant in processing: ",
           ASK recStats[i].appTimeStats Mean() /
           FLOAT(recStats[i].numProcessed) );

    {  OUTPUT("************************************");}
    {PRINT(ASK recStats[i].recruiter AllocMaximum) WITH "
           Maximum allocated recruiters ****";
    OUTPUT;}
    PRINT(ASK recStats[i].recutilStats Mean() /FLOAT(numReps))
          WITH "      Time weighted mean allocated recruiters
          ****.**";
    OUTPUT;
    PRINT(ASK recStats[i].recdevStats Mean() / FLOAT(numReps))
          WITH "      Time weighted standard deviation
          allocated recruiter ****.**";
    OUTPUT;

    OUTPUT("************************************");
    OUTPUT
   END FOR;

   OUTPUT;
   OUTPUT("TOTAL Number of Walk-Ins for the Station: ",
          numWalkins);
   OUTPUT("   Mean: ", ASK walkStats Mean());
   OUTPUT("   St Dev: ", SampleStdDev(ASK walkStats StdDev(),
          numWalkins));
   OUTPUT("************************************");
  END METHOD {printStats};
```

```
TELL METHOD generateWalkins;

 VAR
  applicant,oldapplicant : applicantObj;
  transmitActivityID : ACTID;
  numLowestPri, rindex, selectRand, applicantnumber,
  walkCount : INTEGER;
  lowestPriority : REAL;
  lowPriRcArray : ARRAY INTEGER OF INTEGER;
  waitStream, selectStream,wstream1, wstream2 : RandomObj;

  {numLowestPri = number of recruiters doing equally low-
                  priority tasks}
  {lowestPriority = the number of the lowest priority
                    activity amongst recruiters}
  {lowPriRcArray = indices of recruiters doing equally low-
                   priority tasks}

 BEGIN
  NEW(waitStream);
  ASK waitStream TO SetSeed(FetchSeed(4));
  NEW(selectStream);
  ASK selectStream TO SetSeed(FetchSeed(5));
  NEW(lowPriRcArray, 1..numRecruiters);
  walkCount := 0;

  WHILE (SimTime < runlength)

   NEW(applicant);
   INC(walkCount);
   WAIT DURATION waitStream.Triangular(400.0, 600.0, 800.0)
   END WAIT;
   OUTPUT("New walk-in at time ", SimTime);
   lowestPriority := 20.0;
{*****************************************************}
{Find recruiters doing the lowest priority activities.  }
{Stores their indices in lowPriRcArray                  }

   FOR rindex := 1 TO numRecruiters
    IF (station[rindex].activitypri < lowestPriority)
      numLowestPri := 1;
      lowestPriority := station[rindex].activitypri;
      lowPriRcArray[1] := rindex;
    ELSIF (station[rindex].activitypri = lowestPriority)
      INC(numLowestPri);
      lowPriRcArray[numLowestPri] := rindex;
```

```
      END IF;
     END FOR;

{*****************************************************}
{Select recruiters doing equal priority activities with  }
{equal probability                                       }
{*****************************************************}

     selectRand := selectStream.UniformInt(1, numLowestPri);
     wstream1:=station[lowPriRcArray[selectRand]].stream1;
     wstream2:=station[lowPriRcArray[selectRand]].stream2;
     ASK station[lowPriRcArray[selectRand]] TO incrementappnum;
     applicantnumber:=
     station[lowPriRcArray[selectRand]].appnum;
     ASK applicant TO setid(applicantnumber);
     ASK applicant TO setarrivalTime(SimTime);
     ASK applicant TO
     setFields(station[lowPriRcArray[selectRand]].saleTime,
     station[lowPriRcArray[selectRand]].salepaperTime,
     station[lowPriRcArray[selectRand]].processTimes,
     station[lowPriRcArray[selectRand]].medicalTime,
     station[lowPriRcArray[selectRand]].moralTime,
     wstream1,wstream2,station[lowPriRcArray[selectRand]].
     walkinpri,1);

     ASK station[lowPriRcArray[selectRand]] TO
                incNumApplicants;
     TELL station[lowPriRcArray[selectRand]] TO
                 grabrecruiter(applicant);

   END WHILE;

   {*********  Get Statistics for this Replication  **********}
   numWalkins := numWalkins + walkCount;

  END METHOD {generateWalkins};


END OBJECT {StationObj};


PROCEDURE SampleStdDev(IN stdDev : REAL; IN n : INTEGER) :
                        REAL;
 VAR
  rn : REAL;

 BEGIN
```

```
  rn := FLOAT(n);
  RETURN rn * stdDev / (rn - 1.0);
 END PROCEDURE {SampleStdDev};

END MODULE. {rstationMod}
```

C.2 Ranked List

C.2.1 Definition Module

The following code contains the declarations of all methods and variables pertaining to the Ranked List module:

```
DEFINITION MODULE rankedListMod;

          FROM GrpMod      IMPORT RankedObj;
          {FROM rstationMod IMPORT RecruiterObj;}
          FROM rstationMod IMPORT applicantObj;


          TYPE
           rankedListObj = OBJECT (RankedObj[ANYOBJ:applicantObj]);

                OVERRIDE
                ASK METHOD Rank (IN a, b : applicantObj) : INTEGER;
            END OBJECT {rankedListObj};

       END {DEFINITION} MODULE {rankedListObj}.
```

C.2.2 Implementation Module

The following code contains the implementation of the methods pertaining to the Ranked List module:

```
IMPLEMENTATION MODULE rankedListMod;

  FROM rstationMod IMPORT applicantObj;

          OBJECT rankedListObj;

             ASK METHOD Rank (IN applicantA, applicantB :
                                 applicantObj) : INTEGER;
             BEGIN

               IF applicantA.priority  >  applicantB.priority
                  RETURN -1
                   { insert applicantA before applicantB }
               ELSE
                  RETURN 1
                  { insert applicantA after applicantB  }
               END IF;

            END METHOD {rank};

         END OBJECT {rankedListObj};

      END {IMPLEMENTATION} MODULE. {rankedListMod}
```

C.3 Calendar

C.3.1 Definition Module

The following code contains the declarations of all methods and variables pertaining to the Calendar module:

```
DEFINITION MODULE CalendarMod;

          FROM StatMod        IMPORT IStatObj, RStatObj;

 TYPE

      daytype = (Tuesday,Wednesday,Thursday, Friday, Saturday,Monday);
       monthtype = (January, February, March, April, May,June,July,
              August,September,October, November,December,pDecember);

  dateObj = OBJECT;

        whours,timeinday,hoursPerday : REAL;
        wdays,daydate,monthdate : INTEGER;
        day : daytype;
        month : monthtype;
      ASK METHOD DateInit(IN hoursinweek : REAL; IN daysinweek : INTEGER);
        ASK METHOD updatetime();
         ASK METHOD ElapseTime(IN monthsElapsed : REAL; INOUT new-
Month :
                                monthtype;INOUT newDay : INTEGER);
       END OBJECT; {dateObj}


END {DEFINITION} MODULE.
```

C.3.2 Implementation Module

The following code contains the implementation of all methods pertaining to the Calendar module:

```
IMPLEMENTATION MODULE CalendarMod;


          FROM RandMod     IMPORT RandomObj;
          FROM SimMod      IMPORT SimTime;
          FROM StatMod     IMPORT IStatObj, RStatObj;
OBJECT dateObj;


  {  whours,hour,hoursPerday : REAL;}
  {  wdays,daydate : INTEGER;}
  {  day : daytype;}

 ASK METHOD DateInit(IN hoursinweek : REAL; IN daysinweek : INTEGER);
    BEGIN
      whours:= hoursinweek;
      wdays := daysinweek;
      hoursPerday:=hoursinweek/FLOAT(daysinweek);
    END METHOD; {DateInit}

  ASK METHOD updatetime();
    VAR
     hours : REAL;
     days : INTEGER;
     dayofweek: INTEGER;
    BEGIN
      hours:= SimTime / 60.0; {SimTime is in minutes}
      days := TRUNC(hours/hoursPerday);
      {Gives how many full days have passed}
      daydate:=days + 1;
      timeinday := hours - (FLOAT(days)*hoursPerday);
      {figure out how many hours in days have passed          }
      {subtract from total hours.minutes that have passed and }
      {you have time in the current day.}
      dayofweek := days MOD wdays;
      {all simulations by default start on a monday}
      day:=VAL(daytype,dayofweek);

     IF (wdays = 5)    {This construct is for the 1998       }
                       {Recruiter year and computes recruiter}
                       {month                                }
```

```
CASE daydate
  WHEN 1..20:    {number of working days in  January 20}
    month:=January;
    monthdate:=daydate;
  WHEN 21..40:   { 20 working days in February }
    month := February;
    monthdate:=(daydate - 20)
  WHEN 41..65:   { 25 working days in March }
    month := March;
    monthdate := (daydate - 40);
  WHEN 66..85:   { 20 working days in April }
    month := April;
    monthdate := (daydate - 65);
  WHEN 86..105:   { 20 working days in May }
    month := May;
    monthdate:= (daydate - 85);
  WHEN 106..130:   { 25 working days in June }
    month := June;
    monthdate := (daydate - 105);
  WHEN 131..150:   { 20 working days in July }
    month := July;
    monthdate := (daydate - 130);
  WHEN 151..170:   { 20 working days in August}
    month := August;
    monthdate := (daydate - 150);
  WHEN 171..195:   { 25 working days in September}
    month := September;
    monthdate := (daydate - 170);
  WHEN 196..215:   { 20 working days in October}
    month := October;
    monthdate := (daydate - 195);
  WHEN 216..235:   { 20 working days in November}
    month := November;
    monthdate := (daydate - 215);
  WHEN 236..260:   { 25 working days in December}
    month := December;
    monthdate := (daydate - 235);
  OTHERWISE
    month := pDecember;
  END CASE;
ELSE
  CASE daydate
  WHEN 1..24:    {24 working days in  January}
    month:=January;
    monthdate:=daydate;
  WHEN 25..48:   {24 working days in February }
    month := February;
```

```
        monthdate:=(daydate - 24);
      WHEN 49..78:   { 30 working days in March }
        month := March;
        monthdate:=(daydate - 48);
      WHEN 79..102:   { 24 working days in April }
        month := April;
        monthdate:=(daydate - 78);
      WHEN 103..126:   { 24 working days in May }
        month := May;
        monthdate:=(daydate - 102);
      WHEN 127..156:   { 30 working days in June }
        month := June;
        monthdate:=(daydate - 126);
      WHEN 157..180:   { 24 working days in July }
        month := July;
        monthdate:=(daydate - 156);
      WHEN 181..204:   { 24 working days in August}
        month := August;
        monthdate:=(daydate - 180);
      WHEN 205..234:   { 30 working days in September}
        month := September;
        monthdate:=(daydate - 204);
      WHEN 235..258:   { 24 working days in October}
        month := October;
        monthdate:=(daydate - 234);
      WHEN 259..282:   { 24 working days in November}
        month := November;
        monthdate:=(daydate - 258);
      WHEN 283..312:   { 30 working days in December}
        month := December;
        monthdate:=(daydate - 282);
      OTHERWISE
        month := pDecember;
     END CASE;
   END IF;
END METHOD; {updatetime}


 ASK METHOD ElapseTime(IN monthsElapsed : REAL; INOUT newMonth
                       : monthtype; INOUT newDay : INTEGER);

 {Adds monthsElapsed to current time and returns day & month}
 {Assumes 30 days in month to calculate days for fractional }
 {part of monthsElapsed}

    VAR
       daysThisMonth : INTEGER;
```

```
    BEGIN
       newDay := monthdate + TRUNC(30.0 * (monthsElapsed -
                 FLOAT(TRUNC(monthsElapsed))));

       IF ( ORD(month) MOD 2 <> 0 )
          daysThisMonth := 31;
       ELSIF (month = February)
          daysThisMonth := 28;
       ELSE
          daysThisMonth := 30;
       END IF;

       IF (newDay > daysThisMonth)
          newDay := newDay MOD daysThisMonth;
          monthsElapsed := monthsElapsed + 1.0;
       END IF;

       newMonth := VAL(monthtype, ((ORD(month) +
       TRUNC(monthsElapsed) - 1) MOD 12) + 1);
    END METHOD {ElapseTime};

  END OBJECT; {dateObj}

END MODULE. {rstationMod}
```

C.4 Statistics

C.4.1 Definition Module

The following code contains the declarations of all methods and variables pertaining to the Statistics module:

```
DEFINITION MODULE rstatMod;
{Contains statistics monitor variables for RecruiterObj}

FROM StatMod IMPORT IStatObj, RStatObj;

TYPE

  StatRec = RECORD

     lostatprospect,lostatsales,lostatIMD                    : INTEGER;
    numWalkins,lostatNOR,lostatMED,lostatMOR,lostattest   : INTEGER;
    lostatPsales,lostatDEP,totalcreated,numContracts      : INTEGER;
    numberOfapplicants,numProcessed,interruptedTasks      : INTEGER;
      applicantsWait,appTime,recutil,recdev                  : REAL;
     prospectDur, salesDur, processDur, waiversDur, depDur : REAL;

  END RECORD {StatRec};



  statsObj = OBJECT;

    lostatprospect,lostatsales,lostatIMD,numWalkins   : LMONITORED INTEGER;
    lostatNOR,lostatMED,lostatMOR,lostattest        : LMONITORED INTEGER;
    lostatPsales,lostatDEP,totalcreated,numContracts  : LMONITORED INTEGER;
    numberOfapplicants,numProcessed,interruptedTasks  : LMONITORED INTEGER;
    applicantsWait,appTime,recutil,recdev, depDur     : LMONITORED REAL;
    prospectDur, salesDur, processDur, waiversDur     : LMONITORED REAL;


     walkStats,lostatprospectStats,lostatsalesStats      : IStatObj;
     lostatIMDStats,lostatNORStats,lostatMEDStats        : IStatObj;
     numberOfapplicantsStats,numProcessedStats           : IStatObj;
    lostatDEPStats,totalcreatedStats,numContractsStats : IStatObj;
     lostatMORStats,lostattestStats                      : IStatObj;
     interruptedTasksStats,lostatPsalesStats             : IStatObj;
     applicantsWaitStats,appTimeStats,recutilStats       : RStatObj;
      recdevStats, processDurStats                       : RStatObj;
```

```
    prospectDurStats, salesDurStats                    : RStatObj;
    waiversDurStats, depDurStats                       : RStatObj;

    ASK METHOD ObjInit;
    ASK METHOD IncStats(IN rStats : StatRec);
  END OBJECT {statsObj};

END {DEFINITION} MODULE {rstatMod}.
```

C.4.2 Implementation Module

The following code contains the implementation of all methods pertaining to the Statistics module:

```
IMPLEMENTATION MODULE rstatMod;
{Contains statistics monitor variables for RecruiterObj}

   OBJECT statsObj;

      ASK METHOD ObjInit;
         BEGIN

            NEW (numberOfapplicantsStats);
         ADDMONITOR (numberOfapplicants, numberOfapplicantsStats);

            NEW (interruptedTasksStats);
            ADDMONITOR (interruptedTasks, interruptedTasksStats);

            NEW (applicantsWaitStats);
            ADDMONITOR (applicantsWait, applicantsWaitStats);

            NEW(walkStats);
            ADDMONITOR(numWalkins, walkStats);

            NEW(totalcreatedStats);
            ADDMONITOR(totalcreated, totalcreatedStats);

            NEW(lostatprospectStats);
            ADDMONITOR(lostatprospect,lostatprospectStats);

            NEW(numProcessedStats);
            ADDMONITOR(numProcessed, numProcessedStats);

            NEW(lostatsalesStats);
            ADDMONITOR(lostatsales,lostatsalesStats);

            NEW(lostatIMDStats);
            ADDMONITOR(lostatIMD,lostatIMDStats);

            NEW(lostatNORStats);
            ADDMONITOR(lostatNOR,lostatNORStats);

            NEW(lostatMEDStats);
            ADDMONITOR(lostatMED,lostatMEDStats);
```

```
        NEW(lostatMORStats);
        ADDMONITOR(lostatMOR,lostatMORStats);

        NEW(lostattestStats);
        ADDMONITOR(lostattest,lostattestStats);

        NEW(lostatPsalesStats);
        ADDMONITOR(lostatPsales,lostatPsalesStats);

        NEW(lostatDEPStats);
        ADDMONITOR(lostatDEP,lostatDEPStats);

        NEW(appTimeStats);
        ADDMONITOR(appTime,appTimeStats);

        NEW(recutilStats);
        ADDMONITOR(recutil,recutilStats);

        NEW(recdevStats);
        ADDMONITOR(recdev,recdevStats);

        NEW (prospectDurStats);
        ADDMONITOR (prospectDur, prospectDurStats);

        NEW (salesDurStats);
        ADDMONITOR (salesDur, salesDurStats);

        NEW (processDurStats);
        ADDMONITOR (processDur, processDurStats);

        NEW (waiversDurStats);
        ADDMONITOR (waiversDur, waiversDurStats);

        NEW (depDurStats);
        ADDMONITOR (depDur, depDurStats);

        NEW (numContractsStats);
        ADDMONITOR (numContracts, numContractsStats);

END METHOD {ObjInit};


ASK METHOD IncStats(IN rstats : StatRec);
   BEGIN
      numProcessed := numProcessed + rstats.numProcessed;
    lostatprospect := lostatprospect + rstats.lostatprospect;
      lostatsales := lostatsales + rstats.lostatsales;
```

```
            lostatIMD := lostatIMD + rstats.lostatIMD;
            numWalkins := numWalkins + rstats.numWalkins;
            lostatNOR := lostatNOR + rstats.lostatNOR;
            lostatMED := lostatMED + rstats.lostatMED;
            lostatMOR := lostatMOR + rstats.lostatMOR;
            totalcreated := totalcreated + rstats.totalcreated;
            lostattest := lostattest + rstats.lostattest;
            lostatPsales := lostatPsales + rstats.lostatPsales;
            lostatDEP := lostatDEP + rstats.lostatDEP;
        numberOfapplicants := numberOfapplicants + rstats.numberOfapplicants;
        interruptedTasks := interruptedTasks + rstats.interruptedTasks;
          applicantsWait := applicantsWait + rstats.applicantsWait;
            appTime := appTime + rstats.appTime;
            recutil := recutil + rstats.recutil;
            recdev := recdev + rstats.recdev;
            prospectDur := prospectDur + rstats.prospectDur;
            salesDur := salesDur + rstats.salesDur;
            processDur := processDur + rstats.processDur;
            waiversDur := waiversDur + rstats.waiversDur;
            depDur := depDur + rstats.depDur;
            numContracts := numContracts + rstats.numContracts;
      END METHOD {IncStats};

   END OBJECT {statsObj};

END MODULE {rstatMod}.
```

C.5 Input

C.5.1 Definition Module

The following code contains the declarations of all methods and variables pertaining to the Input module:

```
DEFINITION MODULE rsinfoMod;

TYPE

    traitArrayType = FIXED ARRAY[1..3] OF REAL;
    {    Def:  Array that we use to keep track of various distrib-
utions such }
   {       as the triangular distribution which has a worst, average and }
   {        best numbers which need to be kept track of.              }


 informationObj = OBJECT;
   ASK METHOD grabinfo;
 END OBJECT; {informationObj}

END {DEFINITION} MODULE {rsinfoMod}.
```

C.5.2 Implementation Module

The following code contains the implementation of all methods and variables pertaining to the Input module:

```
IMPLEMENTATION MODULE rsinfoMod;


FROM IOMod IMPORT StreamObj, ALL FileUseType;


VAR
rsinput   : StreamObj;
stationinput : StreamObj;



OBJECT informationObj;

ASK METHOD grabinfo;

VAR
 GoodT, AvgT, PoorT, GoodF,AvgF,PoorF,
 presaleTime,saleTime,processTime,medicalTime,
 moralTime, tosalesdelay, toprocessdelay, tomedicaldelay,
 tomoraldelay,salepaperTime,dpresaleTime,
 dsaleTime,dprocessTime,dmedicalTime,dmoralTime,
 dsalepaperTime : traitArrayType;
 GoodS, AvgS, PoorS, GoodPS,AvgPS,PoorPS,GoodP,AvgP,PoorP,
 GoodIP,AvgIP,PoorIP, GoodNP, AvgNP, PoorNP,
 Medwaiver,Morwaiver,testpass, breaklength,worktime,
 colateraltime, lunchtime,workweek, tempv,
 prospectpri,sellpri,waiverpri,waiverHighpri,
 walkinpri,Iprocesspri, Nprocesspri,deppri,
 maxprospecttime,dprospect,dsell,dwaiver,dwaiverhigh,
 dwalkin,diprocess,dnprocess,ddep : REAL;
 response, recability: CHAR;
 days, numberrecruiters,numberofbreaks, runlength, i : INTEGER;
 temps : STRING;

BEGIN
  {Following are the default values for the information asked}
  {of the program user.  The user can either choose to accept}
  {the default or they have the option to change the         }
  {information.                                              }
```

```
GoodT[1]:= 15.0;    GoodT[2]:=50.0;       GoodT[3]:=70.0;
{telephone prospecting}
AvgT[1]:= 20.0;     AvgT[2]:=60.0;        AvgT[3]:=90.0;
PoorT[1]:= 30.0;    PoorT[2]:=70.0;       PoorT[3]:=100.0;
GoodF[1]:= 15.0;    GoodF[2]:=50.0;       GoodF[3]:=70.0;
{face to face prospecting}
AvgF[1]:= 20.0;     AvgF[2]:=60.0;        AvgF[3]:=90.0;
PoorF[1]:= 30.0;    PoorF[2]:=70.0;       PoorF[3]:=100.0;

GoodS:=0.35;        AvgS:=0.4;          PoorS:=0.45;
{Probability of not showing up for sales interview}
GoodPS:=0.55;       AvgPS:=0.6;         PoorPS:=0.65;
{Probability of not passing pre-sales interview for walkins}
GoodP:=0.25;        AvgP:=0.3;          PoorP:=0.35;
{Probability of not showing up for processing}
GoodIP:=0.03;       AvgIP:=0.05;        PoorIP:=0.07;
{Probability of loss in Immediate processing}
GoodNP:=0.12;       AvgNP:=0.15;        PoorNP:=0.18;
{Probability of loss Normal processing}
OUTPUT;
OUTPUT("You will later be asked to rate the recruiters");
OUTPUT("in the model on recruiting ability");
OUTPUT;
OUTPUT("This information can have significant effects
        on the program output");
OUTPUT;
OUTPUT("By necessity most distributions used are triangular
        distributions");
OUTPUT;
OUTPUT("Do you wish to modify the information used to
        characterize");
OUTPUT("good, average, and poor recruiters? Y or N");
OUTPUT;
INPUT(response);

IF (response = 'y') OR (response = 'Y')
 OUTPUT;
 OUTPUT("Time it takes a recruiter to generate an appointment
         by telephone prospecting");
 OUTPUT;
 PRINT(GoodT[1],GoodT[2],GoodT[3]) WITH "Good ****.** ****.**
       ****.**";
 PRINT(AvgT[1],AvgT[2],AvgT[3])    WITH "Avg  ****.** ****.**
       ****.**";
 PRINT(PoorT[1],PoorT[2],PoorT[3]) WITH "Poor ****.** ****.**
       ****.**";
 OUTPUT;
```

```
OUTPUT("Time it takes a recruiter to generate an appointment
       by face to face prospecting");
OUTPUT;
PRINT(GoodF[1],GoodF[2],GoodF[3]) WITH "Good ****.** ****.**
      ****.**";
PRINT(AvgF[1],AvgF[2],AvgF[3])    WITH "Avg  ****.** ****.**
      ****.**";
PRINT(PoorF[1],PoorF[2],PoorF[3]) WITH "Poor ****.** ****.**
      ****.**";
OUTPUT;
OUTPUT("Do you wish to modify the default Telephone
        prospecting ratings? ENTER Y or N");
INPUT(response);

IF (response = 'Y') OR (response = 'y')

 OUTPUT("Enter the 3 parameters for the Good telephone
        prospecting times separated by spaces. ");
 INPUT(GoodT[1],GoodT[2],GoodT[3]);
 OUTPUT;
 OUTPUT("Enter the 3 parameters for the Average telephone
         prospecting times separated by spaces. ");
 INPUT(AvgT[1],AvgT[2],AvgT[3]);
 OUTPUT;
 OUTPUT("Enter the 3 parameters for the Poor telephone
         prospecting times separated by spaces. ");
 INPUT(PoorT[1],PoorT[2],PoorT[3]);
 OUTPUT;
END IF;

OUTPUT("Do you wish to modify the default face to face
        prospecting ratings? ENTER Y or N");
INPUT(response);
IF (response = 'Y') OR (response = 'y')

 OUTPUT("Enter the 3 parameters for the Good face to face
         prospecting times separated by spaces. ");
 INPUT(GoodF[1],GoodF[2],GoodF[3]);
 OUTPUT;
 OUTPUT("Enter the 3 parameters for the Average face to face
         prospecting times separated by spaces. ");
 INPUT(AvgF[1],AvgF[2],AvgF[3]);
 OUTPUT;
 OUTPUT("Enter the 3 parameters for the Poor face to face
         prospecting times separated by spaces. ");
 INPUT(PoorF[1],PoorF[2],PoorF[3]);
END IF;
```

```
response:='n';

OUTPUT("Default probabilities of an applicant not showing for
        an event or loss:");
OUTPUT;
OUTPUT("No show to sales interview:");
PRINT(GoodS,AvgS,PoorS) WITH "  Good ****.** Avg ****.** Poor
      ****.**";
OUTPUT;
OUTPUT("Probability of not passing Pre-Sales interview:");
PRINT(GoodPS,AvgPS,PoorPS) WITH " Good ****.** Avg ****.**
      Poor ****.**";
OUTPUT;
OUTPUT("No show to processing:");
PRINT(GoodP,AvgP,PoorP) WITH " Good ****.** Avg ****.** Poor
      ****.**";
OUTPUT;
OUTPUT("Probability of loss in immediate processing:");
PRINT(GoodIP,AvgIP,PoorIP) WITH
      " Good ****.** Avg ****.** Poor ****.**";
OUTPUT;
OUTPUT("Probability of loss in normal processing:");
PRINT(GoodNP,AvgNP,PoorNP) WITH
      " Good ****.** Avg ****.** Poor ****.**";
OUTPUT;
OUTPUT("Do you wish to modify the Probabilities?
        ENTER Y or N");
INPUT(response);

IF (response = 'Y') OR (response = 'y')
 OUTPUT("Enter the new probabilities for no show to sales
         interview separated by spaces");
 INPUT(GoodS,AvgS,PoorS);
 OUTPUT;
 OUTPUT("Enter the new probabilities of not passing Pre-Sales
         interview separated by spaces");
 INPUT(GoodPS,AvgPS,PoorPS);
 OUTPUT;
 OUTPUT("Enter the new probabilities for no show to
         processing separated by spaces");
 INPUT(GoodP,AvgP,PoorP);
 OUTPUT;
 OUTPUT("Enter the new probability of loss in immediate
         processing separated by spaces");
 INPUT(GoodIP,AvgIP,PoorIP);
 OUTPUT;
```

```
   OUTPUT("Enter the new probability of loss in normal
           processing separated by spaces");
   INPUT(GoodNP,AvgNP,PoorNP);
   OUTPUT;
  END IF;

 END IF;
{skipped if user does not want to modify good, bad, or average}

 Medwaiver:= 0.7;  {Probability of not getting a waiver}
 Morwaiver:= 0.3;  {Probability of not getting a moral waiver}
 testpass:= 0.8;   {Probability of passing the test}
 prospectpri := 3.0;        {process priorities}
 sellpri := 5.0;
 waiverpri := 1.0;
 walkinpri := 8.0;
 Nprocesspri := 6.0;
 Iprocesspri :=7.0;
 waiverHighpri := 6.5;
 deppri :=2.0;
 numberofbreaks := 4;
 breaklength := 10.0;     {break length in minutes}
 colateraltime := 60.0;
 {time spent on collateral duties per day in minutes}
 lunchtime := 60.0;
 {time spent on lunch daily in minutes}
 worktime := 60.0;

 {work time encompasses the time spent on daily performance   }
 {reviews, school visits, generating school folders, and other}
 {activities not included in the model                        }

 workweek := 60.0;
 {time spent working in hours per week}
 days := 6;                {day in a work week}
 maxprospecttime:=300.0;   {time in minutes}
 OUTPUT;
 OUTPUT("Default waiver and test values");
 OUTPUT;
 PRINT(Medwaiver) WITH "Probability of not getting a medical
                        waiver ****.**";
 PRINT(Morwaiver) WITH "Probability of not getting a moral
                        waiver ****.**";
 PRINT(testpass) WITH "Probability of passing the ASVAB
                       ****.**";
 OUTPUT;
 OUTPUT("Do you wish to modify these values ?  Y or N");
```

```
INPUT(response);

IF (response = 'y') OR (response = 'Y')
  OUTPUT("Input probability of not getting a medical waiver");
  INPUT(Medwaiver);
  OUTPUT("Input probability of not getting a moral waiver ");
  INPUT(Morwaiver);
  OUTPUT("Input probability of passing the ASVAB ");
  INPUT(testpass);
  OUTPUT;
END IF;

{Delays between processes}
  tosalesdelay[1]:= 2400.0;      {40 hours}
  tosalesdelay[2]:= 2880.0;      {48 hours}
  tosalesdelay[3]:= 4320.0;      {72 hours}

  toprocessdelay[1]:= 2400.0;      {40 hours}
  toprocessdelay[2]:= 2880.0;      {48 hours}
  toprocessdelay[3]:= 4320.0;      {72 hours}

  tomedicaldelay[1]:=2400.0;  {150 hours, approx 15 days}
  tomedicaldelay[2]:=2880.0;  {200 hours, approx 20 days}
  tomedicaldelay[3]:=4320.0;  {300 hours, approx 30 days}

  tomoraldelay[1]:=2400.0;
  tomoraldelay[2]:=2880.0;
  tomoraldelay[3]:=4320.0;

  presaleTime[1]:=15.0;
  {Time it takes to go through a pre-sales interview for}
  {walkins}
  presaleTime[2]:=25.0;
  presaleTime[3]:=30.0;

  saleTime[1]:=45.0;
  {Time it takes to go through a sales interview}
  saleTime[2]:=90.0;
  saleTime[3]:=140.0;

  salepaperTime[1]:=40.0;
  {Time it takes to do paperwork associated with sales}
  {interview}
  salepaperTime[2]:=60.0;
  salepaperTime[3]:=80.0;

  processTime[1]:= 250.0;
```

```
 processTime[2]:=300.0;
 processTime[3]:=350.0;

 medicalTime[1]:= 100.0;
 medicalTime[2]:= 125.0;
 medicalTime[3]:= 175.0;

 moralTime[1]:=400.0;
 moralTime[2]:=480.0;
 moralTime[3]:=530.0;

OUTPUT;
OUTPUT("The following delays are the default delays between
        processes in minutes");
OUTPUT;
OUTPUT("Time between making an appointment and the
        appointment in minutes.");
PRINT(tosalesdelay[1],tosalesdelay[2],tosalesdelay[3]) WITH
      " ****.** ****.** ****.**";
OUTPUT;
OUTPUT("Time between sales interview and processing paperwork
        in minutes.");
PRINT(toprocessdelay[1],toprocessdelay[2],toprocessdelay[3])
     WITH " ****.** ****.** ****.**";
OUTPUT;
OUTPUT("Time from submitting medical waiver to receiving a
        response in minutes.");
PRINT(tomedicaldelay[1],tomedicaldelay[2],tomedicaldelay[3])
      WITH " ****.** ****.** ****.**";
OUTPUT;
OUTPUT("Time from submitting moral waiver to receiving a
       response in minutes.");
PRINT(tomoraldelay[1],tomoraldelay[2],tomoraldelay[3]) WITH
      " ****.** ****.** ****.**";
OUTPUT;
OUTPUT("Do you wish to change the default delays between
        processes? Y or N");
INPUT(response);
IF (response = 'y') OR (response = 'Y')
 OUTPUT("Enter the new default delay to sales interview
         separated by spaces");
 INPUT(tosalesdelay[1],tosalesdelay[2],tosalesdelay[3]);
 OUTPUT;
 OUTPUT("Enter the new default delay to process from sales
         separated by spaces");
 INPUT(toprocessdelay[1],toprocessdelay[2],
       toprocessdelay[3]);
```

```
  OUTPUT;
  OUTPUT("Enter the new default time required for medical
          waivers separated by spaces");
  INPUT(tomedicaldelay[1],tomedicaldelay[2],
        tomedicaldelay[3]);
  OUTPUT;
  OUTPUT("Enter the new default time required for moral
          waivers separated by spaces");
  INPUT(tomoraldelay[1],tomoraldelay[2],tomoraldelay[3]);
  OUTPUT;
  OUTPUT;
 END IF;

OUTPUT;
OUTPUT("The following are the default priorities recruiters
        will give to tasks.");
OUTPUT("If you choose to modify them remember the priorities
        must match reality not regulations");
OUTPUT;
OUTPUT("The higher the priority the more important it is");
OUTPUT;
PRINT(prospectpri) WITH "Prospect priority ****.**";
PRINT(sellpri) WITH "Sales priority ****.**";
PRINT(walkinpri) WITH "Walkin priority ****.**";
PRINT(Nprocesspri) WITH "Normal process priority ****.**";
PRINT(Iprocesspri) WITH "Immediate process priority ****.**";
PRINT(waiverpri) WITH "Waiver priority ****.**";
PRINT(waiverHighpri) WITH "Waiver high priority ****.**";
PRINT(deppri) WITH "Dep Maintenance priority ****.**";
OUTPUT;
OUTPUT("Do you wish to change the default priorities?
        Y or N");
INPUT(response);

IF (response = 'y') OR (response = 'Y')

 OUTPUT("Input new prospect priority");
 INPUT(prospectpri);
 OUTPUT("Input new sales priority");
 INPUT(sellpri);
 OUTPUT("Input new walkin priority");
 INPUT(walkinpri);
 OUTPUT("Input new normal process priority");
 INPUT(Nprocesspri);
 OUTPUT("Input new immediate process priority");
 INPUT(Iprocesspri);
 OUTPUT("Input new waiver priority");
```

```
 INPUT(waiverpri);
 OUTPUT("Waiver high priority");
 INPUT(waiverHighpri);
 OUTPUT("Input dep maintenance priority");
 INPUT(deppri);
 OUTPUT;

END IF;

OUTPUT("Default time duration for recruiter activities");
OUTPUT;
PRINT(presaleTime[1],presaleTime[2],presaleTime[3]) WITH
      "Time needed to conduct a pre-sales interview for
      walkins ****.** ****.** ****.**";
PRINT(saleTime[1],saleTime[2],saleTime[3]) WITH
      "Time needed to conduct a sales interview
      ****.** ****.** ****.**";
OUTPUT;
PRINT(salepaperTime[1],salepaperTime[2],salepaperTime[3]) WITH
      "Time needed to fill out paper related to sales
      interview ****.** ****.** ****.**";
OUTPUT;
PRINT(processTime[1],processTime[2],processTime[3]) WITH
      "Time needed to process an applicant's paperwork ****.**
      ****.** ****.**";
OUTPUT;
PRINT(medicalTime[1],medicalTime[2],medicalTime[3]) WITH
      "Time needed to complete medical waiver paperwork
      ****.** ****.** ****.**";
OUTPUT;
PRINT(moralTime[1],moralTime[2],moralTime[3]) WITH
      "Time needed to complete moral waiver paperwork ****.**
      ****.** ****.**";
OUTPUT;
OUTPUT("Do you wish to change the default process durations?
        Y or N");
INPUT(response);

IF (response = 'y') OR (response = 'Y')
 OUTPUT("Enter new default time to conduct pre-sales interview
         separated by spaces");
 INPUT(presaleTime[1],presaleTime[2],presaleTime[3]);
 OUTPUT;
 OUTPUT("Enter new default time to conduct sales interview
         separated by spaces");
 INPUT(saleTime[1],saleTime[2],saleTime[3]);
 OUTPUT;
```

```
 OUTPUT("Enter new default time to fill out sales interview
        paperwork separated by spaces");
 INPUT(salepaperTime[1],salepaperTime[2],salepaperTime[3]);
 OUTPUT;
 OUTPUT("Enter new default time to process applicant paperwork
        separated by spaces");
 INPUT(processTime[1],processTime[2],processTime[3]);
 OUTPUT;
 OUTPUT("Enter new default time to process medical waiver
        paperwork separated by spaces");
 INPUT(medicalTime[1],medicalTime[2],medicalTime[3]);
 OUTPUT;
 OUTPUT("Enter new default time to process moral waiver
        paperwork separated by spaces");
 INPUT(moralTime[1],moralTime[2],moralTime[3]);
 OUTPUT;
 OUTPUT;
END IF;

dpresaleTime[1]:=presaleTime[1];
{Time it takes to go through a pre-sales interview for
 walkins}
dpresaleTime[2]:=presaleTime[2];
dpresaleTime[3]:=presaleTime[3];
dsaleTime[1]:=saleTime[1];
{Time it takes to go through a sales interview}
dsaleTime[2]:=saleTime[2];
dsaleTime[3]:=saleTime[3];
dsalepaperTime[1]:=40.0; {Default Time it takes to do
paperwork associated with sales interview}
dsalepaperTime[2]:=60.0;
dsalepaperTime[3]:=80.0;

dprocessTime[1]:=processTime[1];
dprocessTime[2]:=processTime[2];
dprocessTime[3]:=processTime[3];

dmedicalTime[1]:= medicalTime[1];
dmedicalTime[2]:= medicalTime[2];
dmedicalTime[3]:= medicalTime[3];

dmoralTime[1]:=moralTime[1];
dmoralTime[2]:=moralTime[2];
dmoralTime[3]:=moralTime[3];

OUTPUT;
OUTPUT("The following values are the default values dealing
```

```
        with time for the recruiters");
OUTPUT;
PRINT(workweek) WITH "Length of workweek in hours ****.**";
PRINT(days) WITH "Number of workdays in a week ****";
PRINT(lunchtime) WITH "Time spent at lunch daily in minutes
                      ****.**";
PRINT(colateraltime) WITH "Time spent on collateral duties
                           daily ****.**";
PRINT(worktime) WITH "Average time spent on
                planning,performance review, etc *****.**";
PRINT(breaklength) WITH "Length of breaks ****.**";
PRINT(numberofbreaks) WITH "Number of breaks in a day ****";
PRINT(maxprospecttime) WITH "Maximum time allowable to
                              prospect ****.**";
OUTPUT("Do you wish to modify these default values?
        Y or N ");
INPUT(response);

IF (response = 'y') OR (response = 'Y')

 OUTPUT("Input length of workweek in hours");
 INPUT(workweek);
 OUTPUT("Input number of workdays in a week");
 INPUT(days);
 OUTPUT("Input Time spent at lunch daily in minutes");
 INPUT(lunchtime);
 OUTPUT("Input Time spent on collateral duties daily");
 INPUT(colateraltime);
 OUTPUT("Input Average time spent on planning,performance
         review, etc");
 INPUT(worktime);
 OUTPUT("Input length of breaks");
 INPUT(breaklength);
 OUTPUT("Input Number of breaks in a day ");
 INPUT(numberofbreaks);
 OUTPUT("Input maximum time allowable to prospect");
 INPUT(maxprospecttime);
 OUTPUT;
END IF;

OUTPUT("How many recruiters are working at the station?");
INPUT(numberrecruiters);
OUTPUT;

OUTPUT("How many days do you want the simulation to run?");
INPUT(runlength);
workweek :=  (((workweek * 60.0) - ( FLOAT(days)*lunchtime +
```

```
                FLOAT(numberofbreaks)*breaklength +
                FLOAT(days)*colateraltime + worktime) ) /
                60.0);
OUTPUT("When rating the recruiters enter either G (good) A
       (average) or P (poor)");
NEW(stationinput);
ASK stationinput TO Open("stationfile",Output);
 ASK stationinput TO WriteInt(numberrecruiters,4);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteInt(runlength,4);
 {time will have to be converted to minutes in program}
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(workweek,4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteInt(days,4); {days in the week}
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(maxprospecttime,4,2);
 ASK stationinput TO WriteLn;
ASK stationinput TO Close;

ASK stationinput TO Open("commonfile",Output);
 ASK stationinput TO WriteReal(Medwaiver,4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(Morwaiver,4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(testpass,4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tosalesdelay[1],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tosalesdelay[2],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tosalesdelay[3],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(toprocessdelay[1],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(toprocessdelay[2],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(toprocessdelay[3],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tomedicaldelay[1],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tomedicaldelay[2],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tomedicaldelay[3],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tomoraldelay[1],4,2);
 ASK stationinput TO WriteLn;
```

```
 ASK stationinput TO WriteReal(tomoraldelay[2],4,2);
 ASK stationinput TO WriteLn;
 ASK stationinput TO WriteReal(tomoraldelay[3],4,2);
 ASK stationinput TO WriteLn;
ASK stationinput TO Close;


NEW(rsinput);
ASK rsinput TO Open("recfile",Output);
dprospect := prospectpri;
{process priorities to save the defaults}
dsell := sellpri;
dwaiver := waiverpri;
dwalkin := walkinpri;
dnprocess := Nprocesspri;
diprocess := Iprocesspri;
dwaiverhigh := waiverHighpri;
ddep := deppri;

FOR i:= 1 TO numberrecruiters

 OUTPUT("How do you rate recruiter ",i," ability to make appointments
         using telephone prospecting?");
 INPUT(recability);

 IF (recability = 'G') OR (recability = 'g')
  ASK rsinput TO WriteReal(GoodT[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(GoodT[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(GoodT[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
 ELSIF (recability = 'A') OR (recability = 'a')
  ASK rsinput TO WriteReal(AvgT[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(AvgT[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(AvgT[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
 ELSIF (recability = 'P') OR (recability = 'p')
  ASK rsinput TO WriteReal(PoorT[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(PoorT[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(PoorT[3],4,2);
```

```
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");
END IF;

OUTPUT("How do you rate recruiter ",i," ability to make
        appointments using face to face prospecting?");
INPUT(recability);

IF (recability = 'G') OR (recability = 'g')
 ASK rsinput TO WriteReal(GoodF[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(GoodF[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(GoodF[3],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
ELSIF (recability = 'A') OR (recability = 'a')
 ASK rsinput TO WriteReal(AvgF[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(AvgF[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(AvgF[3],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
ELSIF (recability = 'P') OR (recability = 'p')
 ASK rsinput TO WriteReal(PoorF[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(PoorF[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(PoorF[3],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");

END IF;

OUTPUT("How do you rate recruiter ",i,"
        appointments to arrive to sales?");
INPUT(recability);

IF (recability = 'G') OR (recability = 'g')
 ASK rsinput TO WriteReal(GoodS,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'A') OR (recability = 'a')
```

```
 ASK rsinput TO WriteReal(AvgS,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'P') OR (recability = 'p')
 ASK rsinput TO WriteReal(PoorS,4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");

END IF;

OUTPUT("How do you rate recruiter ",i," ability to get
        recruits to process from sales?");
INPUT(recability);

IF (recability = 'G') OR (recability = 'g')
 ASK rsinput TO WriteReal(GoodP,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'A') OR (recability = 'a')
 ASK rsinput TO WriteReal(AvgP,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'P') OR (recability = 'p')
 ASK rsinput TO WriteReal(PoorP,4,2);
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");

END IF;

OUTPUT("How do you rate recruiter ",i," walkin's ability
       to pass a sales pre qualification?");
INPUT(recability);

IF (recability = 'G') OR (recability = 'g')
 ASK rsinput TO WriteReal(GoodPS,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'A') OR (recability = 'a')
 ASK rsinput TO WriteReal(AvgPS,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'P') OR (recability = 'p')
 ASK rsinput TO WriteReal(PoorPS,4,2);
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");

END IF;
```

```
OUTPUT("How do you rate recruiter ",i," ability to keep
        applicants through immediate processing?");
INPUT(recability);

IF (recability = 'G') OR (recability = 'g')
 ASK rsinput TO WriteReal(GoodIP,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'A') OR (recability = 'a')
 ASK rsinput TO WriteReal(AvgIP,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'P') OR (recability = 'p')
 ASK rsinput TO WriteReal(PoorIP,4,2);
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");

END IF;

OUTPUT("How do you rate recruiter ",i," ability to keep
        applicants through normal processing?");
INPUT(recability);

IF (recability = 'G') OR (recability = 'g')
 ASK rsinput TO WriteReal(GoodNP,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'A') OR (recability = 'a')
 ASK rsinput TO WriteReal(AvgNP,4,2);
 ASK rsinput TO WriteLn;
ELSIF (recability = 'P') OR (recability = 'p')
 ASK rsinput TO WriteReal(PoorNP,4,2);
 ASK rsinput TO WriteLn;
ELSE
 OUTPUT("incorrect response, sorry");

END IF;

ASK rsinput TO WriteLn;

OUTPUT("Do you wish to change recruiter ",i," process
        durations?  Y or N");
INPUT(response);

IF (response = 'y') OR (response = 'Y')
 OUTPUT("Enter time to conduct pre-sales interview
        separated by spaces");
 INPUT(presaleTime[1],presaleTime[2],presaleTime[3]);
 OUTPUT;
```

```
OUTPUT("Enter time to conduct sales interview separated
        by spaces");
INPUT(saleTime[1],saleTime[2],saleTime[3]);
OUTPUT;
OUTPUT("Enter time to conduct fill out sales paperwork
        separated by spaces");
INPUT(salepaperTime[1],salepaperTime[2],salepaperTime[3]);
OUTPUT;
OUTPUT("Enter time to process applicant paperwork
        separated by spaces");
INPUT(processTime[1],processTime[2],processTime[3]);
OUTPUT;
OUTPUT("Enter time to process medical waiver paperwork
        separated by spaces");
INPUT(medicalTime[1],medicalTime[2],medicalTime[3]);
OUTPUT;
OUTPUT("Enter time to process moral waiver paperwork
        separated by spaces");
INPUT(moralTime[1],moralTime[2],moralTime[3]);
OUTPUT;
OUTPUT;
 ASK rsinput TO WriteReal(presaleTime[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(presaleTime[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(presaleTime[3],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(saleTime[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(saleTime[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(saleTime[3],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(salepaperTime[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(salepaperTime[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(salepaperTime[3],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(processTime[1],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(processTime[2],4,2);
 ASK rsinput TO WriteLn;
 ASK rsinput TO WriteReal(processTime[3],4,2);
```

```
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(medicalTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(medicalTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(medicalTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(moralTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(moralTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(moralTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
ELSE
  ASK rsinput TO WriteReal(dpresaleTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dpresaleTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dpresaleTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsaleTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsaleTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsaleTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsalepaperTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsalepaperTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsalepaperTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dprocessTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dprocessTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dprocessTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dmedicalTime[1],4,2);
  ASK rsinput TO WriteLn;
```

```
  ASK rsinput TO WriteReal(dmedicalTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dmedicalTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dmoralTime[1],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dmoralTime[2],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dmoralTime[3],4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
END IF;


OUTPUT("Do you wish to change the default processing
        priorities for recruiter ",i," Y or N ");
INPUT(response);

IF (response = 'y') OR (response = 'Y')
  OUTPUT("What priority does recruiter ",i,"
          give to prospecting? ");
  INPUT(prospectpri);
  OUTPUT;
  OUTPUT("What priority does recruiter ",i,"
          give to sales? ");
  INPUT(sellpri);
  OUTPUT;
  OUTPUT("What priority does recruiter ",i," give to
          normal processing? ");
  INPUT(Nprocesspri);
  OUTPUT;
  OUTPUT("What priority does recruiter ",i," give to
          immediate processing? ");
  INPUT(Iprocesspri);
  OUTPUT;
  OUTPUT("What priority does recruiter ",i," give to
          waiver processing? ");
  INPUT(waiverpri);
  OUTPUT;
  OUTPUT("What priority does recruiter ",i," give to waiver
          processing when it has to get done? ");
  INPUT(waiverpri);
  OUTPUT;
  OUTPUT("What priority does recruiter ",i," give
          to DEP maintenance? ");
  INPUT(deppri);
```

```
  OUTPUT;
  OUTPUT("What priority does recruiter ",i," give to
          assisting walkins? ");
  INPUT(walkinpri);
  OUTPUT;

  ASK rsinput TO WriteReal(prospectpri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(sellpri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(Nprocesspri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(Iprocesspri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(waiverpri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(waiverHighpri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(deppri,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(walkinpri,4,2);
  ASK rsinput TO WriteLn;
ELSE
  ASK rsinput TO WriteReal(dprospect,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dsell,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dnprocess,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(diprocess,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dwaiver,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dwaiverhigh,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(ddep,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteReal(dwalkin,4,2);
  ASK rsinput TO WriteLn;
  ASK rsinput TO WriteLn;
END IF;

END FOR;
ASK rsinput TO Close;

OUTPUT("Now an attempt to read from the file. ");
ASK rsinput TO Open("recfile",Input);
```

```
WHILE NOT (ASK rsinput eof)
 ASK rsinput TO ReadReal(tempv);
 OUTPUT(tempv);
 ASK rsinput TO ReadLine(temps);
END WHILE;
ASK rsinput TO Close;


{last of the code still need more distributions ect}


END METHOD; {grabinfo}

END OBJECT; {informationObj}

END MODULE. {rsinfoMod}
```

C.5.3 Main Module

The following code contains the main program of the Input module:

```
MAIN MODULE rsinfo;

FROM rsinfoMod IMPORT informationObj;
FROM SimMod     IMPORT StartSimulation, StopSimulation, ResetSimTime;


VAR

 eye : informationObj;

BEGIN

 NEW(eye);
 ASK eye TO grabinfo;

END {MAIN} MODULE {rsinfo}.
```

Bibliography

[1] Anonymous. "Despite Cutbacks, U.S. Army Still Looking for Recruits," *Jet*, *85* 18, 49 March, 1994.

[2] Bachman, Jerald G., et al. *The All-Volunteer Force: A Study of Ideology in the Military*. The University of Michigan Press and John Wiley & Sons Canada, Ltd., 1977.

[3] Balci, Osman. "Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation Study." Accepted for publication in Annals of Operations Research, Vol. 53, December 1994, June 1994.

[4] Booch, Grady. *Object Oriented Analysis and Design With Applications* (2nd Edition). The Benjamin/Cummings Publishing Co., Inc., 1994.

[5] Gilroy, Curtis L., et al. "The All-Volunteer Army: Fifteen Years Later," *Armed Forces & Society*, *16* 3, 329–350 1990.

[6] Grissmer, David W. and Sheil Nataraj Kirby. "A Total Force Perspective on Recruiting and Manning in the Years Ahead," *Contributions in Military Studies*, *154* 189–208 1994.

[7] Jerry Banks, John S. Carson, II Barry L. Nelson. *Discrete-Event System Simulation*. Prentice-Hall, Inc., 1996.

[8] Kearl, Cyril E. and Abraham Nelson. "The Army's Delayed Entry Program," *Armed Forces & Society*, *18* 2, 253–268 1992.

[9] Kennedy, Claudia J. "Army Takes Steps to Boost Recruiting," *Army*, *47* 4, 57 April, 1997.

[10] Law, Averill M. and W. David Kelton. *Simulation Modeling & Analysis*. McGraw-Hill, Inc., 1991.

[11] Light, Larry. "Got Tuition Woes? Maybe the Army Can Help," *Business Week*, *31* 71, 80 July, 1990.

[12] Lovell, et al. "The Allocation of Consumer Incentives to Meet Simultaneous Sales Quotas: An Application to U.S. Army Recruiting," *Management Science*, *37* 3, 350 March, 1991.

[13] Mehay, Stephan L. "Determinants of Enlistments in the U.S. Army Reserve," *Armed Forces & Society*, *16* 3, 351–367 1990.

[14] Moniz, Dave. "Recruiters Become Hunters in Search for New Soldiers," *Dayton Daily News*, 12 April, 1997.

[15] Moore, Alan. "Mission: Sell the Army," *Soldiers*, 28–30 1997.

[16] Moskos, Charles C. and Frank R. Wood. *The Military: More Than Just a Job?*. Pergamon-Brassey's International Defense Publishers, Inc., 1988.

[17] Nauroth, Tony. "Soldier Show At Opryland," *Soldiers*, *37* 3, 42–44 November, 1990.

[18] Neblett, Liz. "And They're Off!," *Soldiers*, *37* 3, 41 November, 1990.

[19] Pfluke, Lillian A. "Every Day is a Fight," *Newsweek*, *128* 22, 27 November, 1996.

[20] Raymond H. Myers, Douglas C. Montgomery. *Response Surface Methodology : Process and Product Optimization Using Designed Experiments*. John Wiley & Sons, Inc., 1995.

[21] Ricks, Thomas E. "Army Faces Recruiting Obstacle: A Less-Macho Image," *The Wall Street Journal*, 20 July, 1997.

[22] Roberts, Benjamin J. "Redesigning Military Recruitment for the Future," *Contributions in Military Studies*, *154* 209–225 1994.

[23] Sargent, Robert G. "Verifying and Validating Simulation Models." *Proceedings of the 1996 Winter Simulation Conference*. 55–64. 1996.

[24] Sarkesian, Sam C. "Military Strategies and Force Structure," *Contributions in Military Studies*, *154* 21–37 1994.

[25] Shyles, Leonard and John E. Hocking. "The Army's "Be All You Can Be" Campaign," *Armed Forces & Society*, *16* 3, 369–383 1990.

[26] Simpson, Kenneth W. "Recruiting Quality Soldiers for America's Army," *Army*, *44* 10, 159–163 October, 1994.

[27] Thompson, Mark. "Offensive Maneuvers," *Time*, *149* 18, 40–42 May, 1997.

[28] United States Army Recruiting Command (USAREC). *USAREC Regulation 601-56: Waiver, Delayed Entry Program Separation, and Void Enlistment Processing Procedures"*, January 1992.

[29] United States Army Recruiting Command (USAREC). *USAREC Pamphlet 350-7: Recruiter Salesmanship*, July 1994.

[30] United States Army Recruiting Command (USAREC). *USAREC Regulation 350-6: Recruiter Production Management System*, May 1996.

[31] Vistica, Gregory L. "Ranks," *Newsweek*, *128* 22, 27 November, 1996.

Vita

First Lieutenant James D. Cordeiro, Jr. was born on July 17, 1967 in Sasebo, Japan. He graduated from the Punahou School Academy in Honolulu, Hawaii in 1985 and subsequently attended the University of California at Berkeley in August of the same year. He graduated in 1989 with a Bachelor of Arts in Mathematics and Oriental Languages, and then proceeded to the University of Washington to study Mathematics at the graduate level. He obtained a Master of Science degree in Mathematics in December, 1992.

Lt. Cordeiro taught Computer Science courses at Hawaii Pacific University in 1993, and then received a commission as a Second Lieutenant in the U.S. Air Force in May, 1994 from the Officer Training School in Montgomery, Alabama. He was assigned to the Standard Systems Center in Montgomery as an acquisitions officer specializing in Comm-Computers. While there, he served on database-development teams for the Standard Systems Center's Management Information System (MIS). In August 1996, Lt. Cordeiro entered the School of Engineering, Air Force Institute of Technology, specializing in Operations Research.

Permanent Address: 94-1112 Akeu Place
Waipahu, HI 96797

Second Lieutenant Mark A. Friend, was born June 12, 1969 in San Diego, California. He graduated from John F. Hodge High School in St. James, Missouri in 1987 and enlisted in the U.S. Navy in the same year. While in the Navy, he studied Spanish at the Defense Language Institute at Monterey, California. Lt. Friend enrolled at Texas Christian University after leaving the Navy and graduated magna cum laude with a Bachelor of Science in Computer Science with a minor in Mathematics. Lt. Friend subsequently received a commission as a Second Lieutenant in the U.S. Air

Force and shortly thereafter, entered the School of Engineering, Air Force Institute of Technology to study Operations Research.

Permanent Address: 719 E 18th Streeet
San Angelo, TX 76903

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
March 1998

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Using Simulation to Model Time Utilization of Army Recruiters

5. FUNDING NUMBERS

6. AUTHOR(S)
1LT James D. Cordeiro, 2LT Mark A. Friend

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT/ENS
2950 P Street
Wright - Patterson AFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/98M-06
AFIT/GOR/ENS/98M-12

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Headquarters, U.S. Army Recruiting Command (USAREC)
ATTN: RCPAE-RP
1307 Third Avenue
Fort Knox, KY 40121-2726

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Advisors:
Lt. Col. J.O. Miller, 255-6565 Ext. 4333, Jmiller@afit.af.mil
Dr. Kenneth Bauer, 255-6565 Ext. 4326, Kbauer@afit.af.mil

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
It is a well-known fact that Army recruiters work very long hours in a demanding environment. In many cases, recruiting stations are geographically isolated from military bases, with recruiters often tolerating a high cost of living, crime, and other such adverse conditions that characterize the communities they work in. The job itself demands self-starting, motivated individuals who require a wide range of skills, from street-savvy to salesmanship, in order to succeed. A number of factors in recent years have made military recruiting more difficult, such as scandals involving highly-placed soldiers to changes in attitudes towards military service amongst eligible youngsters. Added to the mix is a recent mission increase which has exacerbated this problem even further. Unlike previous studies which have concentrated on the effects of advertisements and other determinants of enlistments in the Army, this study instead focuses on the individual recruiters themselves, with the ultimate purpose of defining the relationship between the various recruiter tasks and the end product, namely qualified Army recruits.

14. SUBJECT TERMS
Recruiting, Army, Army Recruiting, SimProcess, enlistment, simulation,ModSim,design of experiments,response surface methodology,linear regression

15. NUMBER OF PAGES
279

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

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	**PR** - Project
G - Grant	**TA** - Task
PE - Program Element	**WU** - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
  Leave blank.
- **NASA** - Leave blank.
- **NTIS** -

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self- explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.