AN ABSTRACT OF THE THESIS OF

Ann M. Mechling for the degree of Master of Science in Industrial Engineering presented on September 6, 1988.

Title: A Simulation Based Decision Support System for Resource Planning

Redacted for privacy

Abstract approved: _____
Sabah Randhawa

The problem of resource planning has long been a topic of interest for operations managers in both production and service industries. Inputs such as raw materials, energy, and capital are limited resources; therefore, managers must make informed decisions regarding the use of these inputs. This thesis addresses the area of man-power resource planning by developing a simulation based decision support tool to aid in the evaluation of alternatives during the decision making process.

The problem of resource planning was first brought to my attention through the Motor Vehicle Division (DMV) Systems and Planning Section in Salem, Oregon. They had conducted a study of field offices in the summer of 1987 in a continuing effort to improve customer satisfaction. One area that was identified as in need of further study was the receptionist function.

A simulation program was developed using the SLAM simulation language to model customer activities in DMV field offices. The program has a front end that allows model variables to be input by the user. The user specified variables include customer arrival rates, number of available servers, number of available receptionists and definition of the receptionist function. The program also executes the simulation for the user and includes a back end that displays relevant statistics.

The program was applied to DMV field offices comparing the effects of a receptionist versus another worker serving customers. It was found that at busy times (high customer arrival rates) the use of a receptionist significantly lowers waiting times. Sensitivity analysis was also conducted on three of the more uncertain variables relating to the receptionist function: service time, percent turn away and percent MVR service time reduction. It was found that waiting times and queue lengths were very sensitive to these variables.

A Simulation Based Decision Support System for Resource Planning

by

Ann M. Mechling

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed September 6, 1988

Commencement June 1989

APPROVED:

Redacted for privacy

Professor of Industrial and Manufacturing Engineering in charge of major

Redacted for privacy

Head of department of Industrial and Manufacturing Engineering

Redacted for privacy

Dean of Graduate School

Date thesis is presented September 6, 1988

Typed by Ann M. Mechling for Ann M. Mechling

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Sabah Randhawa. Not only was he of great assistance in making this a successful research project, he also made my education at OSU a more rewarding experience. His encouragement along the way was appreciated.

I would also like to thank Bob Joerger of Systems and Planning of the Motor Vehicles Division for initiating this project and for his assistance in completing it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF TABLES

A SIMULATION BASED DECISION SUPPORT SYSTEM FOR RESOURCE PLANNING

INTRODUCTION

The problem of resource planning has long been a topic of interest for operations managers in both production and service industries. In both types of industry a transformation process takes place. Inputs such as raw materials, energy, and capital are transformed into outputs such as finished goods and/or services. Because these inputs are not unlimited the managers must make informed decisions regarding the use of the inputs and the methods of transforming them into outputs. Resource planning is one area within the transformation process with which operations managers are concerned. (Schroeder, 1982).

Decision making can be defined as "the whole process of arriving at a decision, from initial identification of the problem through generation and evaluation of alternatives and finally to the choice itself." (Schroeder, 1982). This thesis addresses the area of human resource planning by developing a simulation based decision support tool to aid in the evaluation of alternatives during the decision making process.

Simulation was chosen for several reasons. First, it allows the study of an organization without disrupting

operations. This is particularly important in the service industry where much of the business is based on customer satisfaction. Second, it costs considerably less than experimenting with the actual operations. It may be impractical to hire additional workers, lay off workers or to rearrange the facility layout each week. Simulation allows the decision maker to study the effects of a resource plan on an organization quickly and to compare alternatives. It must be understood that "simulation can only be used to predict or describe what will happen under a given set of circumstances, not what should be done relative to specific decision criteria." (Schroeder, 1982)

Because this is a decision support tool it must be available for managers to use not only now but in the future. Development must be geared toward the end user who may know little or nothing about simulation. It should be simple to use and presented in a format understood by managers. Because of the dynamic nature of businesses, constantly changing as new technologies are developed, the tool must also be complex enough to adapt to a variety of alternatives. Consequently, this simplicity must be balanced with the complexity required to be a useful tool.

The problem of resource planning was first brought to my attention through the Motor Vehicle Division (DMV) Systems and Planning Section in Salem, Oregon. They had conducted a study of field offices in the summer of 1987

in a continuing effort to improve customer satisfaction. One area that was identified as in need of further study was the receptionist function. It was suggested that a simulation queueing model of DMV field offices be developed in order to study the effects of resource changes on field office operations.

PROBLEM ANALYSIS

Literature Review

The development of computer simulation programs and software has been a progressive area in computer programming. Although there was a significant amount of literature describing specific applications of simulation, most of it dealt with robotic work cells and flexible manufacturing systems rather than queueing models for man-power resource planning tools. Most of the literature review therefore concentrated on recent developments in making simulation accessible to inexperienced users and facilitating its use. Some of the major trends in computer simulation have been: simulation languages, simulation program generators, and expert systems (Haddock, 1987).

Simulation languages (SLAM, SIMAN, GPSS, etc.) were developed to facilitate model building and are often written in another level language such as FORTRAN. They provide specific functions, time advancement mechanisms and activity scheduling for the simulation process, thereby relieving the programmer of much of the tedious work. Related to these are the specialized simulation languages (GEMS, MAP, etc.) which are more problem specific, and hence faster to learn, but are more limited in the range of systems that can be modeled. Another area of develop-

ment has been in animated simulation (TESS, CINEMA, etc.) which aids the programmer in understanding model behavior. Although the above languages have made simulation easier for the programmer, knowledge of simulation modeling and language syntax is still necessary in order to create the model. This is a decided disadvantage for the inexperienced simulation user. (Haddock, 1987)

A good example of recent developments in simulation languages is The Extended Simulation System (TESS) as described by Standridge (1985). TESS was designed to aid the SLAM model builder by providing "support for model building, execution, result analysis, and result presentation" within a standardized, integrated framework. The capabilities of TESS include:

"(1) graphically building SLAM II networks and schematic models;
(2) forms entry for simulation control information, simulation inputs, and animation specifications;
(3) database management of user-defined data, model inputs and model outputs;
(4) preparation of reports and graphs;
(5) analysis of simulation results;
(6) and the animation of simulation runs."

The main disadvantage of TESS is that the user is still required to have knowledge about how the simulation and the real system behave, and some knowledge about the simulation language.

Medeiros and Sadowski (1983) have developed a modular approach for modeling robotic manufacturing cells based on the Q-GERT simulation language. The modular approach facilitates model building by providing the modules

already encoded and only requiring the programmer to select and link the appropriate ones for representation of the model. These modules are provided by a library of functions. Although limited at the present to robots working with automated devices, the set of library functions can easily be expanded to cover other activities. The initial work of module construction can be done by simulation experts while others with domain knowledge can construct and evaluate the alternatives. This system has the advantage of great flexibility and one which can grow over time to meet new simulation demands. In addition, a more experienced user may write FORTRAN subroutines to be integrated with the simulation program. The output is provided by the normal Q-GERT summary reports.

Like TESS some knowledge of the programming language is still necessary; however, much of the more repetitious work has been eliminated. The modular approach is

> "most useful when many options are under consideration and thus many related models must be constructed. It is also very useful where such a modeling effort may be used over an extended period of time." (Medeiros and Sadowski, 1983)

Sauer and MacNair (1979) have developed RESQ, a research queueing network software. Unlike many other languages RESQ has three levels of user interface: dialogues, dialogue files and subroutine interface. The dialogue mode is intended for beginning users with little knowledge of RESQ syntax. Tutorial assistance is also

available in this mode in which model definition is prompted by the computer. The dialogue file mode is for more experienced users in which model definition is prompted by the user. The user builds a dialogue file of prompts and replies by combining dialogue templates provided by RESQ. The subroutine interface mode is for the most experienced user as detailed knowledge of RESQ syntax is required. This mode is the most flexible and is used for more complicated models when dialogue files fail. One of the most significant contributions of RESQ to simulation is the ability of the program to adapt to users with differing levels of expertise and to grow with the user.

While simulation languages have made modeling easier by providing specialized functions, the model builder must still have a good background in simulation and model behavior. And, for every new language developed the model builder must learn a new set of commands and syntax. Simulation program generators were developed to overcome the language barrier of simulation languages and to further facilitate model building by actually generating the program code. Mathewson (1984) defines a simulation program generator as:

> "an interactive software tool that translates the logic of a model described in a relatively general symbolism into the code of a simulation language and so enables a computer to mimic model behavior."

According to Haddock and Davis (1985),

> "the predominant advantages provided by a simulation generator are the following:

1. Facilitate model building and data input
2. Facilitate experimentation
3. Incorporate model refinements more rapidly
4. Reinforces evaluation of alternative design/control scenarios.

The trade-offs faced by the programmer/developer are obvious. The more detail that is accounted for, the more tedious and time consuming the data collection and input are. The same holds true for the scope of the systems that can be modeled."

Bengu and Haddock (1986) developed a system that combines a simulation generator and an optimization subroutine. This system develops code for the SIMAN simulation language. The simulation generator interactively designs the experiment file for inventory control systems while the model file remains the same. The optimization subroutine executes the experimental and model file simulation and evaluates total revenue. Using a search procedure it then determines new decision variables (optimal reorder point and quantity) "and repeats the process until the stopping criteria are reached."

There are many advantages to the system described by Bengu and Haddock (1986). The need for user knowledge of high level simulation languages is eliminated by the use of a program generator. The analyst is supported in the decision making process by the optimization module, thereby facilitating experimentation and sensitivity analysis of the inventory control system. Because the generator/ optimization subroutines are fully integrated, the user is not required to interface the program modules. There are some limitations to the present system developed

including: the need for a single, quantifiable performance measure (total revenue); and the unchanging model file which limits the type of systems modeled.

Haddock and Davis (1985) have developed a program generator for the simulation of flexible manufacturing systems (FMS). The generator was written in BASIC and produces both the experiment file and the model file for the SIMAN simulation language. The user-friendly, menu driven generator allows the user to create a model, edit the model, run it, get instructions and exit. All data entry (model specification) is through user-oriented screens and are self-explanatory. The data is then stored in a file which may be edited or run at any time. This has the advantage of being able to store many models for later study without having to re-enter the data. The main disadvantage is that the output is provided by the SIMAN summary reports without any support for sensitivity analysis.

Mathewson (1975 and 1984) describes the DRAFT program generator written in FORTRAN. There are three steps in the use of this program generator: describe the system to be modeled, translate the model into a simulation program using DRAFT, and edit the program. Before using DRAFT, a symbolic description of the model must be made. One example is the entity-cycle diagram. The diagram logic is then used to provide input data to the program generator. All user input is monitored by the computer for errors.

From this the program generator translates the diagram logic (computer data input) into a coded file which then produces the simulation program in a language specified by the user. DRAFT will produce simulation programs in SIMON/ FORTRAN, GASP II, SIMSCRIPT II.5, SIMULA and 2900 ACSL. DRAFT was developed to help the user construct a program "into the best approximation of the final form." Therefore, the final program must be edited in order to insert details that are beyond the power of DRAFT.

The main advantages of DRAFT include: the production of error-free code, the ease of switching between simulation languages, and lowering the level of user knowledge of a specific language. The main disadvantage is that the final code must be edited by the user who may be an inexperienced programmer.

Subrahmanian and Cannon (1981) have developed a program generator for discrete event models to explore the feasibility of simulation generators. Their system consists of three parts: model descriptor language, model structure generator, and translator. The descriptor language consists of key words which represent modules and submodules. A model can then be built by using the key words to link the modules. The model structure generator creates a set of files from the model description for input to the translator. The generator also flags errors at this time. The translator then takes the input from the generator and translates the model into a SIMSCRIPT

program. While the current version of the program generator is limited to simple models and a few basic statistical functions, its modular organization facilitates the addition of other functions and modules.

The third major trend in simulation has been in the area of expert systems (ESs). There are many similarities between ESs and simulation. In each case the knowledge representation and inference engine are separate. ES knowledge representation may be in the form of networks, frames or rules, while its inference engine may be forward- or backward-chaining, among others. Simulation knowledge representation includes events, activities and process interaction, while its inference engine is quite often the next-event time-scheduling algorithm. (O'Keefe, 1986). According to O'Keefe,

> "conducting experiments with discrete-simulation models ... the experimenter uses four classes of knowledge:
>
> (1) Knowledge about the domain in which the model is built - for example manufacturing or health care
>
> (2) Knowledge about statistics - how to interpret results, what measurements are appropriate etc.
>
> (3) Knowledge about how the simulation and the real system behave
>
> (4) Knowledge of the language or package used to implement the simulation.
>
> Many inexperienced simulation users will have good domain knowledge, ... what they will lack is knowledge of types (2) through (4)."

The objective of an ES would then be to provide knowledge of types (2) through (4).

O'Keefe (1986) has developed a pilot Intelligent

Front End (IFE) system to provide type (3) knowledge. An IFE is defined by O'Keefe to be

> "an expert system that sits between a package and the user, generates the necessary instructions or code to use the package following a dialogue with the user, and interprets and explains results from the package."

It was written in ES/P Advisor, a commercially available package, which in turn is written in PROLOG. The objective of this system is to construct an advice giving system and knowledge base on simulation behavior. Although the system at this time contains only a small amount of information in the knowledge base, "less than two dozen pieces of conditional text", additional information may be added as it becomes available. Furthermore, "if it could also obtain and analyse the output, the resulting system would be a very sophisticated IFE." A secondary objective of this pilot IFE was to study how much knowledge programmers have about simulation behavior.

Haddock (1987) describes the possible development of an ES based on a user-oriented simulation generator. The program generator is one that he has programmed for the design and control of flexible manufacturing systems (FMS). It was developed using SIMAN and FORTRAN subroutines that interactively build the simulation model and execute it. The ES solely requires type (3) knowledge from the user and builds on the program generator by providing an analysis of the output. Knowledge about statistics, type (2), is provided by the output analysis

subroutines. Knowledge of the simulation language, type (4), is provided by the program generator. The output analysis has several levels of options such that experienced users have the flexibility to perform more complicated analyses while inexperienced users can stay with the basics. The IFE could be written in any of the popular expert system languages, such as PROLOG and LISP. At the current time "human-computer interaction considerations regarding the acquisition of knowledge from the user are ... under study for this particular system."

The one piece of literature found that relates closely with the DMV field office simulation was an article by Rasmussen. Rasmussen (1984) has developed an interactive program to simulate queues using BASIC. The program is limited to single-phase models with one queue and multiple parallel servers. The user may specify the mean arrival rate and distribution, mean service rate and distribution, number of servers, the status of the queue and servers at the start of simulation, and the sample size. Although this program severely limits the range of models that can be simulated, it facilitates the simulation of many alternatives within the model. This approach to modeling is most advantageous when one system, perhaps of different sizes, will be modeled repeatedly. It has the advantage of being user-friendly and flexible in defining often varied parameters without requiring much knowledge from the user on simulation or the computer

language used.

Although Rasmussen's interactive program is not complete enough for the DMV field office simulation, it does lay some important ground work. By using one standard model and allowing the user to vary specific parameters in a user-friendly environment, requiring user knowledge of model building techniques and simulation languages can be eliminated. Because the field offices are fairly standard, a representative model could easily be built without restricting use of the model by the DMV. This idea of varying parameters was also suggested by Subrahmanian and Cannon (1981) as an improvement to their program generator.

Within a system similar to the one described by Rasmussen (1984) some of the advantages of ESs and program generators can be retained without introducing their complexity of design. No knowledge of the programming language would be required from the user since the model would already be built. The input of parameters could be an interactive dialogue between the computer and the user such that even the most inexperienced user could operate the program. Experimentation and sensitivity analysis would be facilitated due to the ease of varying the parameters in a user-friendly atmosphere. Statistical analysis could also be supported easily by designing a standard output subroutine specifically for the model. Consequently, only type (1) knowledge (O'Keefe, 1986),

domain knowledge, would be required of the user.

This system has another advantage as mentioned by Medeiros and Sadowski (1983). The initial work of model construction can be done by simulation experts while others with domain knowledge construct and evaluate the alternatives. Consequently, the end-user would not work with the coded model and the choice of programming language would be solely the choice of the model builder. The advantages of using a language which supports many simulation functions could then be exploited by the model builder in a language most comfortable to them.

Detailed Problem Description

During the summer of 1987 the DMV conducted a study of field offices in a continuing effort to improve customer satisfaction. The objective of the study was to "analyze the use of the receptionist, self-service counter, drop box, and color coding for forms." (Mills, Jabs and Bouten, 1987). One area that was identified as in need of further study was the receptionist function. A customer survey, work measurement and a facility layout study was done to determine customer opinion of the receptionist and the functions of the receptionist. In general, it was found that both customers and DMV personnel have positive reactions to a receptionist; however, some improvements could be made regarding the

office layout and some guidelines developed for the receptionist function. Specific questions arose as to the effect of a receptionist on waiting times and queue length versus another MVR (Motor Vehicle Representative) behind the counter, and the size of office that would most effectively use a receptionist. It was suggested that a simulation queueing model would be able to answer these questions.

Since the summer of 1986, receptionists have been assigned to some of the larger field offices in response to customer waiting times exceeding the goal of a fifteen minute maximum for average waiting times. The purpose of the receptionist was to:

> "1. Greet the customer.
> 2. Check the documents for accuracy and make sure the customer has everything needed for the transaction.
> 3. If not, ask the customer to return with the necessary information.
> 4. Otherwise, give the customer a number and the appropriate forms and ask him or her to be seated until the number is called." (Mills, Jabs and Bouten, 1987)

By contacting the customers before they enter the MVR queue, those who would be unable to complete their transaction could be prevented from entering and "wasting" their time waiting. A secondary benefit of the receptionist would be to prepare the customer for the transaction (i.e. start filling out forms) and thereby shorten the MVR service time.

Since the initial set-up of receptionists in 1986,

the use of the receptionist has varied somewhat from office to office. Some of the receptionists are classified as clerical specialists while others are MVRs. Work schedules vary from part-time to full-time, and rotating work assignments to non-rotating.

The purpose of this project was to develop a simulation queueing model of DMV field offices in order to study the effects of resource changes on field office operation. Specific objectives were to determine: in what size office a receptionist would be most effective versus another MVR behind the counter, the effect of a receptionist classified as an MVR versus a clerk specialist (i.e. what duties they should perform), and whether or not a receptionist is cost effective in terms of shorter waiting lines and/or improved customer service. Measurements of interest include: queue length, waiting time and employee utilization.

Although developed for DMV operations, this model was designed as a generic one in which other queueing situations may be modeled. Any operation in which entities arrive to a work place, may or may not be intercepted by a router, wait in one of several queues to be serviced at one of many stations is a candidate for the model developed. A good example would be modeling the tellers in a bank or airline ticket counters.

A discussion follows describing field office activities and DMV requirements for the simulation program.

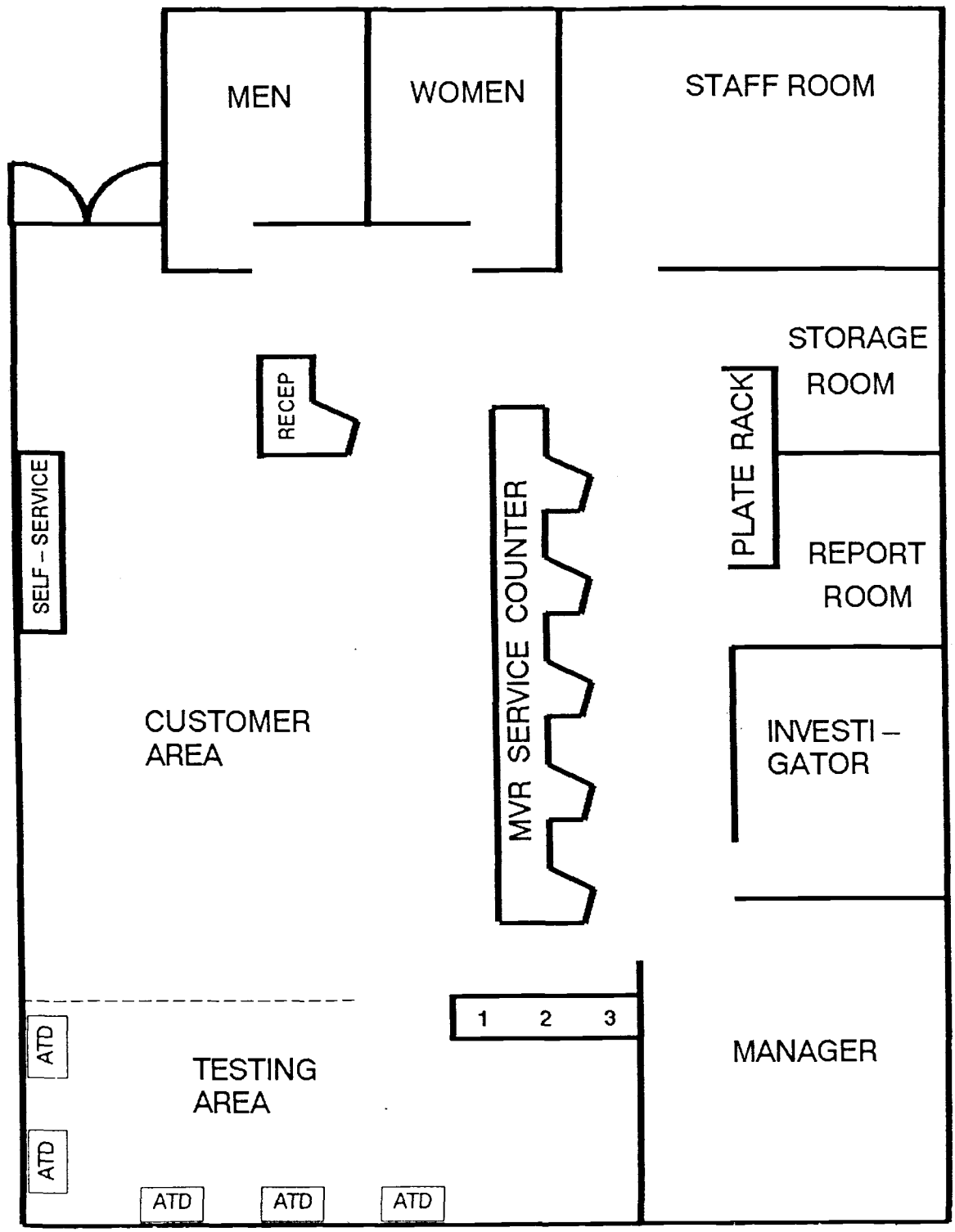
DMV Field Office Activities

In general all DMV field offices are of the same design and conduct the same activities. Figure 1 shows a typical field office layout with or without the optional receptionist.

The self-service counter is for those transactions which the customer may complete without contacting an MVR. A ticket dispenser is situated near the entrance for customers to "take a number" and wait. The main counter is where the MVRs are stationed. Each MVR has a station consisting of a terminal, a cash drawer, a forms file and other items necessary for serving customers. The testing area includes the camera (license photos), ATDs (automated testing devices), ATD monitor and vision test. This area is operated by one or more MVRs depending on the size of the office. The main differences between offices are the volume of transactions, physical size of the office, and the number of workers. In addition, some of the larger offices have a receptionist. The receptionists may be part-time or full-time and, if classified as an MVR, may be rotated with the other MVRs.

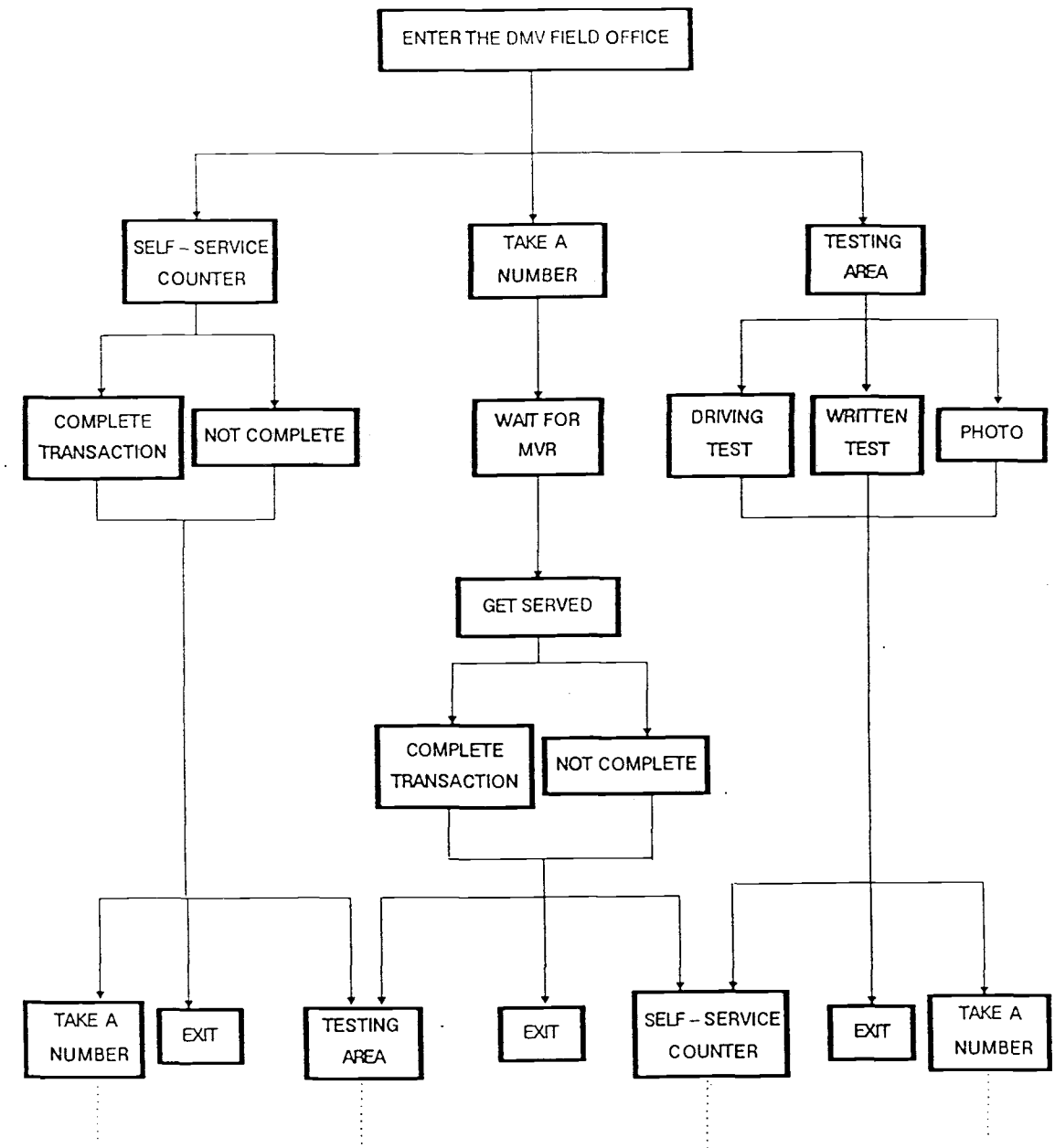
The activities that take place in a field office without a receptionist, in which the DMV customers are involved, are shown in Figure 2. A customer enters a DMV field office and may take one of three actions: go directly to the self-service counter, report to the

Figure 1. Field Office Layout



1 Camera
2 ATD Monitor
3 Vision Tester

Figure 2. Customer Flow Without Receptionist



testing area, or take a number and wait until an MVR calls the number.

The self-service counter is for those transactions not requiring contact with an MVR. They contain pamphlets and manuals for customers to pick up, and the forms needed for the self-service transactions. Some customers will attempt the self-service counter, find that they cannot complete their transaction, and will take a number to wait for an MVR. Another consideration is that not all customers that should use the self-service counter will use it. Some will go directly to the MVR instead. Those leaving the self-service counter (after completing the transaction or not) may have other business in the testing area or with the MVRs, or they may depart the field office.

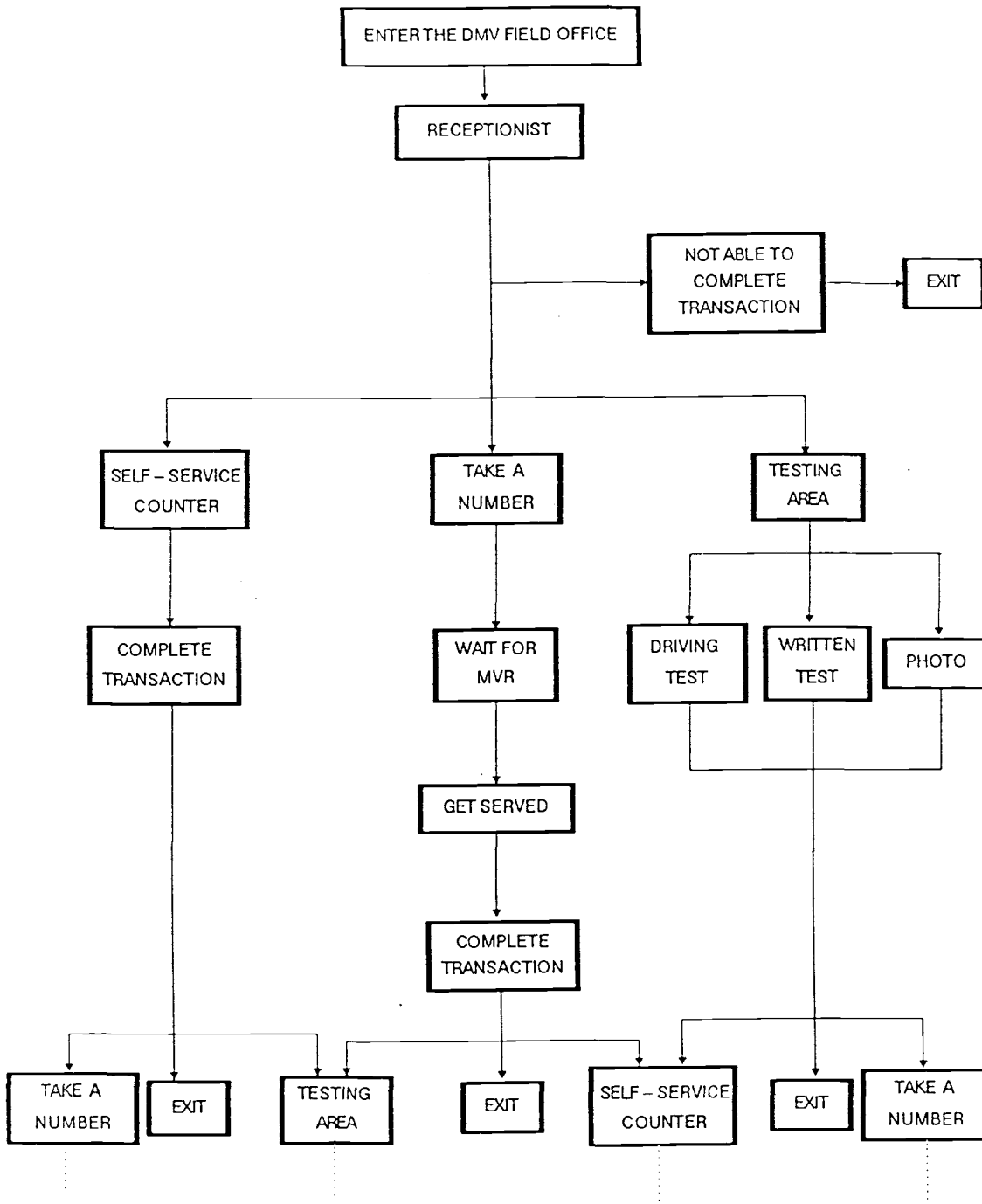
The testing area is operated by one or more MVRs depending on office size. Customers approaching this area come for one of three reasons: a driving test, a written test or a license photo. Usually driving tests are scheduled by appointment in which case the customer checks in. When the examiner (an MVR) is ready they can begin the driving test. After completing the driving test the customer is photographed, if they have passed, the transaction is completed and they leave the testing area. The written tests are conducted on the ATDs after seeing an MVR first. If one is not free at that time the customer will wait for an available ATD to begin testing. After completing the written test the customer is

photographed, if they have passed, the transaction is completed and they leave the testing area. Those customers leaving the testing area may have other business with an MVR or at the self-service counter, or they may depart the field office.

The majority of transactions take place at the MVR counter. A variable number of MVRs may be working at any one time depending on office size and time of day. All MVRs at the counter serve customers in the order they arrive. As customers arrive they take a number from the ticket dispenser and wait in the customer waiting area to be called. When the MVR calls the customer's number they proceed to the counter to begin their transaction. Not all transactions can be completed due to customer lack of preparation or the DMV unable to accommodate that type of transaction. When the customer is finished at the counter they may have other business at the testing area or at the self-service counter, or they may depart the field office.

For those offices with a receptionist the process described above changes slightly as can be seen in Figure 3. As a customer enters the office they are immediately contacted by the receptionist. Although the function of the receptionist varies somewhat from office to office, the initial DMV guidelines described above were used to define the receptionist function. In the case of a customer needing the self-service counter or the testing area they would be directed to the appropriate place

Figure 3. Customer Flow With Receptionist



without a number. The use of a receptionist would eliminate most of the customers who would be unable to complete their transaction.

Program Design Requirements

In addition to the field office activities described above, there were some additional program design requirements. First, the DMV wanted the program to be flexible enough so that they would be able to use it for future analysis. Second, because the Systems and Planning Section of the DMV would be using the program in the future they would like it to be user-friendly toward inexperienced simulation users.

Program flexibility refers to the range of office types that could be modeled. Since all offices are generally of the same design and operations, the program has to allow for varying the office size, volume of business and a limited number of optional activities (for example, having a receptionist or not). The variables included are: customer arrival rate, number of driving tests scheduled, number of MVRs, number of receptionists, type of work the receptionist will do, and how much the receptionist will affect customer preparation for DMV transactions. By allowing for these variations all current and a variety of future DMV field offices could be modeled.

Being user-friendly means that the program must be able to accurately simulate the model and present the results, with little effort on the part of the user. Some sort of user interface is required to allow for modeling the variations with a minimum of input. The less input there is, the fewer the chances for mistakes. The output must also be presented in an understandable format, containing only necessary information.

Based on the field office description and program requirements described above, a simulation program was developed.

METHODOLOGY

Approach to Modeling

Simulation and SLAM

Computer simulation is defined by Pritsker and Pegden (1979) as "a mathematical-logical representation of a system which can be exercised in an experimental fashion on a digital computer." The purpose of using simulation is to predict behavior of a system

> "without building them, if they are only proposed systems; without disturbing them, if they are operating systems that are costly or unsafe to experiment with; without destroying them, if the object of an experiment is to determine their limits of stress." (Pritsker and Pegden, 1979)

Each of these is a valid reason for using simulation to study the DMV field offices. In some offices the receptionist function does not exist and guidelines can be developed for employing receptionists, before implementation. It would also be impractical to analyze the effectiveness of receptionists by moving them around and changing duties daily. Not only would this disrupt operations but it would also be counterproductive to improving customer service. Furthermore, testing the limits of the system may be impossible, not to mention impractical as queues could theoretically grow to infinity. Computer simulation of DMV field offices is a low cost, non-

destructive way to analyze the receptionist function.

The program for simulating the DMV field offices was written using SLAM II (Simulation Language for Alternative Modeling), PC version. SLAM II is an advanced FORTRAN based language that allows discrete simulation models to be built using event orientation (called discrete event programming) and/or process orientation (using network symbols). Continuous simulation models can also be built using continuous event programming. By the judicious combination of the above approaches nearly any system can be modeled, and the advantages of each exploited. For this project a combination of networks and discrete event subroutines were used.

Discrete event programming in SLAM is a very flexible modeling approach in which the model builder defines the events and the changes to a system when an event occurs. These are encoded as FORTRAN subroutines. The SLAM executive controls the event calendar by advancing time and calling the appropriate FORTRAN subroutines and SLAM functions. An example of discrete event programming for people entering a queue follows:

```
      IF (IDLE.EQ.'YES') GO TO 10
      CALL FILEM(1,ATRIB)
      RETURN
10    IDLE='NO'
      CALL SCHDL(2,EXPON(3.5),ATRIB)
      RETURN
        |
        |
      IF (NNQ(1).GT.0) GO TO 20
      IDLE='YES'
      RETURN
```

```
20    CALL RMOVE(1,1,ATRIB)
      CALL SCHDL(2,EXPON(3.5),ATRIB)
      RETURN
```

When a person arrives to the queue a check is made to see if the teller is idle. If the teller is idle then set the teller to busy and schedule an end-of-service event at an exponentially distributed time (mean 3.5 minutes) from the current time; otherwise, file the person into queue one. When the end-of-service event is called a check is made to see if the number of people in queue one is zero. If no one is in the queue set the teller to idle; otherwise, remove the next person from queue one and schedule another end-of-service event.

While this may be a cumbersome way of modeling a queue there are advantages in flexibility. For example, there may be other conditions to be satisfied before setting the teller busy; a person other than the next in line may be selected for removal from the queue; or there may be certain times when a person will not enter the queue when the teller is busy. Within discrete event programming these conditions may be easily specified.

Another advantage to discrete event programming is that any commands used in regular FORTRAN programming are available. Data may be stored in external files, the files manipulated, user interfaces designed, and customized output reports designed. These functions are not directly available in network programming.

Network programming, while not as flexible as

discrete event, is generally easier to learn and more concise. The model is encoded using symbols called nodes and branches. The nodes represent such concepts as servers, queues and decision points. The branches represent the actions taken from the nodes such as service activities and decisions. A model of a system is built by linking nodes together with branches. The same system modeled above as FORTRAN subroutines would look like this in network programming:

```
QUEUE(1);
ACT(1)/1,EXPON(3.5);
```

The QUEUE node models the queue in which a person will stay until a teller is free. The ACT (activity) branch models the utilization of the teller and the duration of the service activity. When the time for service is over the teller is automatically freed and the queue is checked for other people waiting.

While this is obviously shorter than discrete event programming, it lacks flexibility. Although the queue may be ordered as first-in-first-out, last-in-first-out, high value first or low value first, other specifications are not available. Once an entity enters the queue their priority in relation to others in the queue cannot be changed (for example, being moved to the head of the queue in response to a changed due date). In addition, once in the queue they can not leave before being serviced.

Before describing the simulation program in detail, a

brief description of necessary hardware will be given.

SLAM II (PC version) can be used on any IBM PC or compatible microcomputer. It requires 320K of RAM, one double sided/double density disk drive and MS DOS 2.0 or later version. The following additional hardware will improve the performance of SLAM II. The 8087 Math Co-processor will increase execution speed three to five times and is automatically utilized by SLAM. A hard disk will allow larger models to be run and will enhance compiling and model translation. (Lilegdon and O'Reilly, 1986)

Simulation Model

The DMV simulation program was conceptualized based on the need to accurately simulate a DMV field office and to conduct sensitivity analysis. The obstacle faced was that the program would be run by users not familiar with simulation in general, nor with SLAM specifically. Thus, it was necessary to develop an accurate model of DMV field offices that could be modified easily by the user for conducting sensitivity analysis.

From the study of DMV field offices and discussions with office managers it became clear that DMV field offices were of approximately the same layout and that they conduct roughly the same transactions. The differentiating factors were the size of the office and how busy

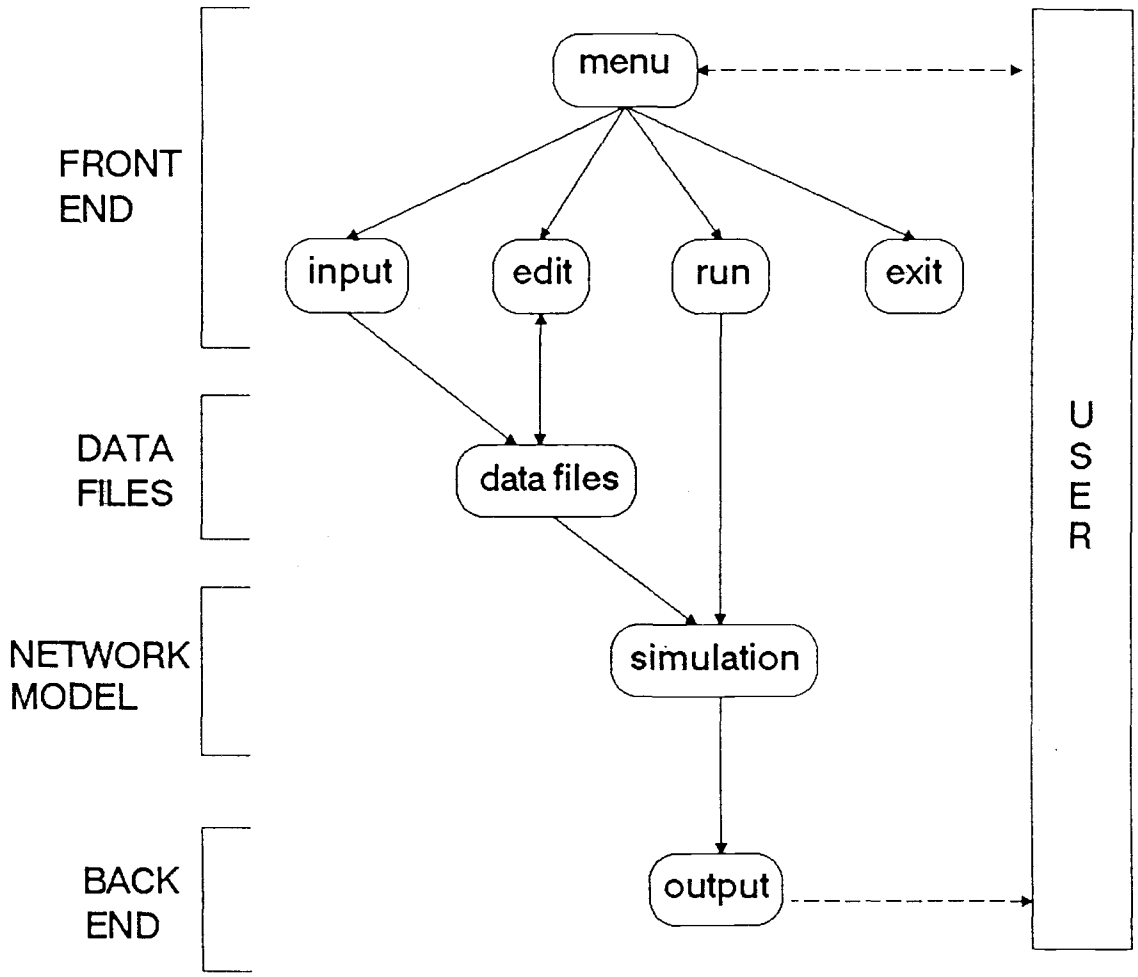
it was. This made it possible for the development of a "skeletal" model for representing the basic office, integrated with a data file for setting specific parameters. Network programming was used to build the "skeletal" model. Discrete event programming and FORTRAN subroutines were used to define, manipulate and integrate the data files, and to design a customized output report.

By developing the program this way the user would not have to interact with the model, thereby eliminating the need for user knowledge of SLAM. Defining the data file could be accomplished through FORTRAN subroutines, thereby eliminating the need for user knowledge of data file structure. Thus the "skeletal" model would remain the same and the user could define the data file in a user-friendly atmosphere. Based on these concepts the DMV simulation program has been divided into four areas: the network/ "skeletal" model, the data files, the user-friendly front end, and the back end for presenting simulation output. A diagram of the program environment is shown in Figure 4.

The front end is the user interface and master control of the program. The advantage of being able to integrate FORTRAN subroutines with the network has allowed for the creation of a menu to control data file manipulation and simulation execution. The only knowledge required of the user is knowledge of the DMV operating environment. No knowledge of SLAM or FORTRAN is

necessary.

Figure 4. Program Environment



The data file, which provides the necessary parameters to the network model, is manipulated through the front end. This allows the "skeletal" network model to be filled in by the user through variables rather than directly editing the program.

The "skeletal" network model controls the actual

simulation of customers, servers and activities within the DMV field office. The simulation of activities was accomplished using the network approach in SLAM. The model specifics are portrayed as variables whose values are supplied by the data files. The advantages of the network method, namely conciseness and ease of development, were able to be utilized.

Finally, the back end controls the output of the simulation. This also takes advantage of FORTRAN subroutines to organize otherwise complex statistical output into an understandable format. Another advantage of writing a separate back end was to eliminate some additional user input and commands normally required by SLAM.

Front End

The front end of the program was designed to give the user control over the execution of the program. A menu has been developed to control the four main activities that could take place in the programming environment: inputting a new data file, editing an existing data file, running the simulation of a specified model, and exiting the program. In turn these menu options call subroutines for performing specified functions. The front end is called from the network through an event node (EVENT 1), before any simulation has actually taken place. Since the front end is the first event to occur upon execution of

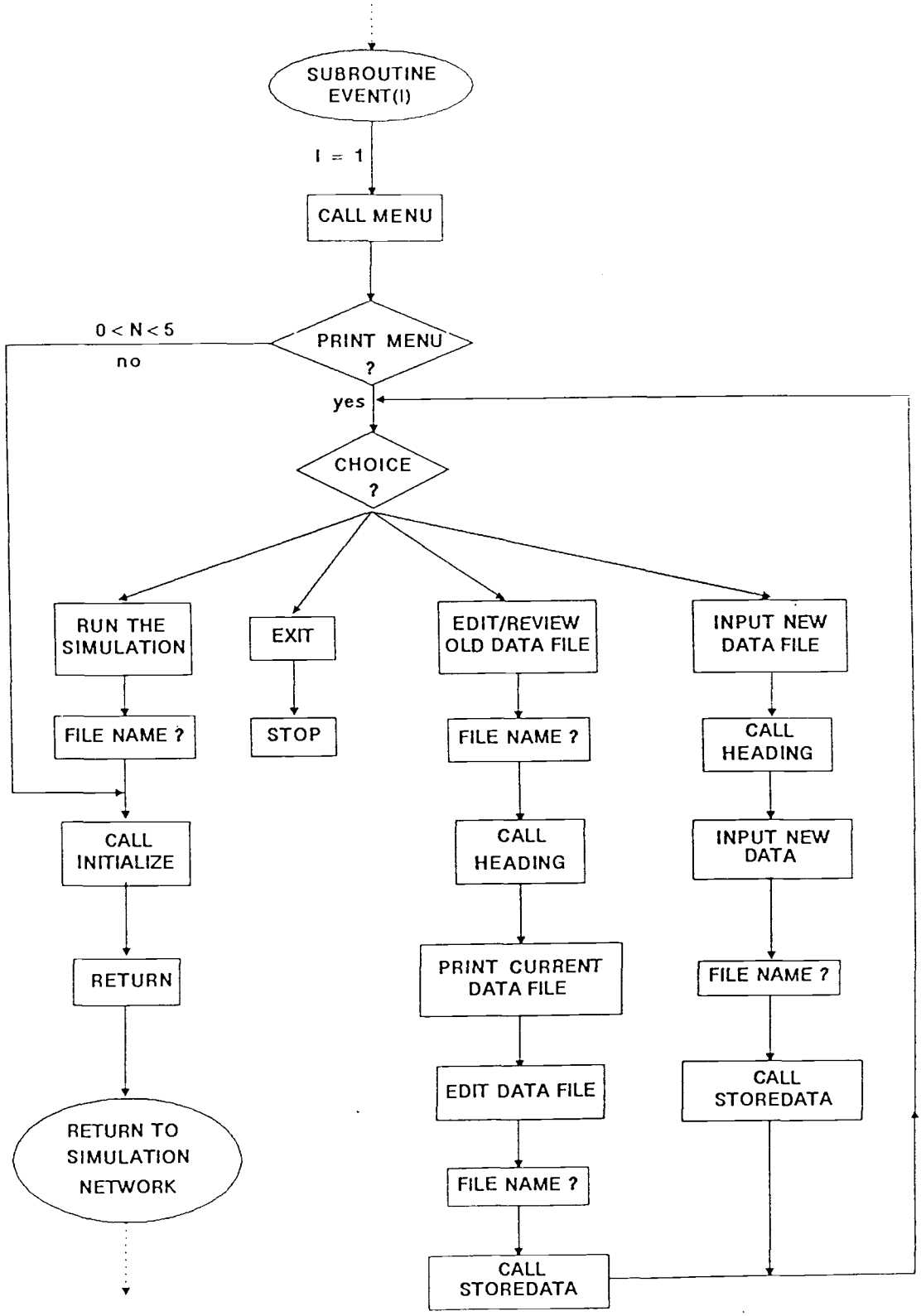
the program, the menu is in control of the program until a return to the network is called. The front end also controls the number of simulation runs made before the menu appears again. More than one run is made so that the variability inherent in stochastic processes is reduced. A flow chart of the front end is shown in Figure 5.

The first step taken in the MENU subroutine is to determine whether the menu should be printed or not. The menu will be printed if the program has just been started or if the required number of runs have been completed with the results displayed. Otherwise, the program is in the middle of simulation and the menu is not printed. The variable N counts the number of simulation runs completed.

If the program is in the middle of simulating the required number of runs (N is greater than zero and less than required number) then printing the menu is skipped. The program goes directly into the RUN option to the step for calling the INITIALIZE subroutine. This resets some of the variables, a return is made to the network and the next simulation run begins.

If none or all runs have been completed then the menu is printed and one of the options may be chosen. After the activities for that option have been completed the program displays the menu again. If an invalid choice was entered, the menu will also reappear. The only way to return to the network and run the simulation is to select the RUN option. The only way to leave the program, short

Figure 5. Front End



of turning the computer off, is to select the EXIT option.

The first option in the menu, INPUT a new data file, will allow the user to enter data from the keyboard in response to prompts from the screen. The data entered into the file will be used as input parameters to the "skeletal" model variables in the RUN option. Instructions are first printed on the screen and then the DMV STAFFING AND ARRIVALS table heading is printed by a call to the HEADING subroutine. This table is to be filled in by the user and will contain model information for each hour of the day, up to eleven hours. Figure 6 shows an empty table.

Figure 6. DMV Staffing and Arrivals Table

DMV STAFFING AND ARRIVALS

HOUR	MVRS AT THE COUNTER	RECEP-TIONIST	CUSTOMER ARRIVAL RATE	# DRIVING TESTS SCHEDULED
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

When this table is completed the user is asked to enter data on the receptionist function, if any receptionists have been scheduled. This includes:

service time, fraction of the customers contacted, whether the receptionist can act as an MVR when idle, and percent MVR service time reduction after contact. Otherwise, default values are assigned to those variables based on current DMV operations. The user is then asked to enter the name of the file for storing the data and the STOREDATA subroutine is called. The program then redisplays the menu.

The second option in the menu is to EDIT/REVIEW an old data file. The user is asked to enter the name of an existing file or to hit ENTER for the default file. The contents of the file are read, and the DMV STAFFING AND ARRIVALS table heading and the table contents are displayed on the screen. The user may then edit the data file a row at a time, with the revised table displayed between each row change. When no more changes in the table are needed, the receptionist function data must then be re-entered if any receptionists have been scheduled. The user is then asked to enter the name of the file for storing data. This may be the same name as before (replace the old file) or a new name may be specified (the old file remains and a new file is created). Again, the STOREDATA subroutine is called and the program returns to the menu.

The third option in the menu is to RUN the simulation. Up until this point the menu has been in control of the program. This option will return control

to the network in order to begin simulation of a DMV field office. Again, the user is asked to enter the name of an existing data file or to hit ENTER for the default file. The user is then asked to verify the data file name before commencing simulation since it takes several minutes for the five simulation runs to finish. The data file values are read directly into the system variables used by the network. The INITIALIZE subroutine is called and the program returns control to the network where simulation begins. Five simulation runs are automatically made by the program before the results or the menu are displayed again.

The fourth option in the menu is to EXIT the program by stopping the program execution.

Data Files

Within the network there are many variables whose values must be provided; therefore, together with the "skeletal" network model developed, a data file is required to run the simulation. This data file contains the information about customer arrival rates, MVR availability and the receptionist function. Many data files may exist, each one referred to by its name (for example, MEDFORD.DAT). The name may have up to seven alphanumeric characters and generally has the DAT extension to denote its data file status. These are

unformatted, sequential access files used by the FORTRAN subroutines. Table 1 shows the information contained in a data file, in the order in which it is stored, and its corresponding system variable (XX) used by the network.

The data files are utilized in three places in the front end. First, they are created through the INPUT subroutine and stored under a given name. Second, they may be edited through the EDIT subroutine and saved under the same name (replace the old file) or saved under a new name (retain the old file and create a new one). Third, a data file is required in the RUN subroutine to assign values to the XX variables before control returns to the network.

Table 1. Data File Contents

VARIABLE	VALUE
XX(50)	change in # MVRs hour 1 (alter value)
XX(51)	change in # receptionists hour 1 (alter value)
XX(52)	regular customer time between arrivals hour 1
XX(53)	drive line customer time between arrivals hour 1
:	:
:	:
:	:
XX(90)	change in # MVRs hour 11 (alter values)
XX(91)	change in # receptionists hour 11 (alter value)
XX(92)	regular customer time between arrivals hour 11
XX(93)	drive line customer time between arrivals hour 11
XX(94)	fraction customers met by receptionist
XX(1)	Y if the receptionist does MVR work when idle N if the receptionist does not
XX(95)	receptionist service time
XX(96)	MVR service time coefficient

Network Model

The "skeletal" model represents the events in which DMV customers take part but does not specify any parameters about the size of the office (for example, number of MVRs and customer arrival rate) or the receptionist function. This information is provided from a data file described in the Data Files and Front End sections.

The events in which customers participate at the DMV may be put into four categories: customer arrival and routing, contact with a receptionist, contact with an MVR and contact with a self-service counter. These categories are four of the five main parts of the network model. The fifth part integrates the FORTRAN subroutines, data file and parameter changes with the network.

The resource and gate concepts in SLAM were used in the network model. These are presented in the table below.

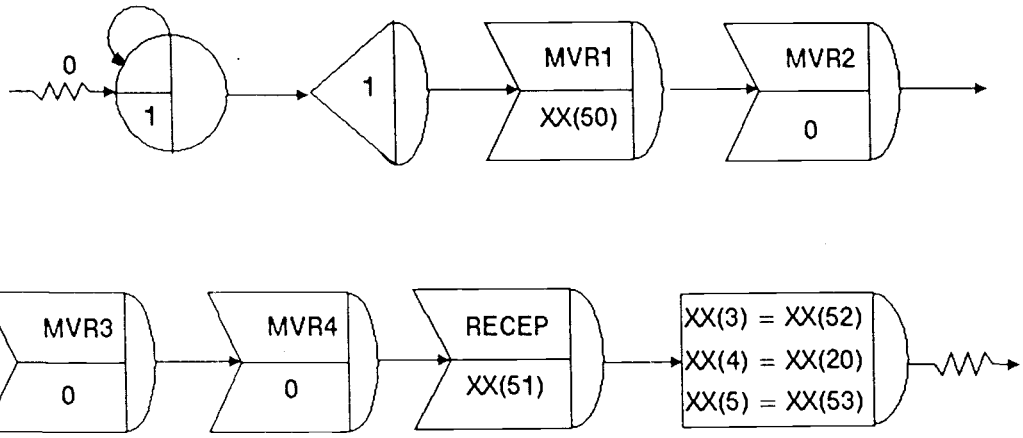
Table 2. Resources and Gates

	DESCRIPTION	FILE #
GATE		
1	physical queue for MVR type 1	1
2	physical queue for MVR type 2	2
3	physical queue for MVR type 3	3
4	physical queue for MVR type 4	4
5	physical queue for receptionist	5
RESOURCE		
1	MVR type 1	11
2	MVR type 2	12
3	MVR type 3	13
4	MVR type 4	14
5	receptionist	15

There are two subnetworks that integrate the user-written FORTRAN subroutines to the network and specify variable values necessary for simulation. The first subnetwork calls the front end menu and initializes the parameters, before the simulation of the DMV field office begins. The second subnetwork changes the parameters on an hourly basis during simulation and collects statistics.

The SLAM network diagram is shown in Figure 7 for the first subnetwork. The first event that takes place in this subnetwork is the creation of a dummy entity that calls EVENT 1. When EVENT 1 is called, the user-written FORTRAN subroutine for the menu is executed and program control is transferred to the FORTRAN subroutines. Control is returned to the network only after the RUN option is chosen in the menu and the XX variables (20-96) have been set. When control returns to the network the next event to occur is the altering of the MVR resources and the receptionist resource up to the desired levels for the first hour of simulation. The initial arrival rates for regular and drive line customers are assigned to XX(3) and XX(5), respectively, and the number of receptionists in hour one is assigned to XX(4). The dummy entity is then terminated and the simulation of a DMV field office may begin.

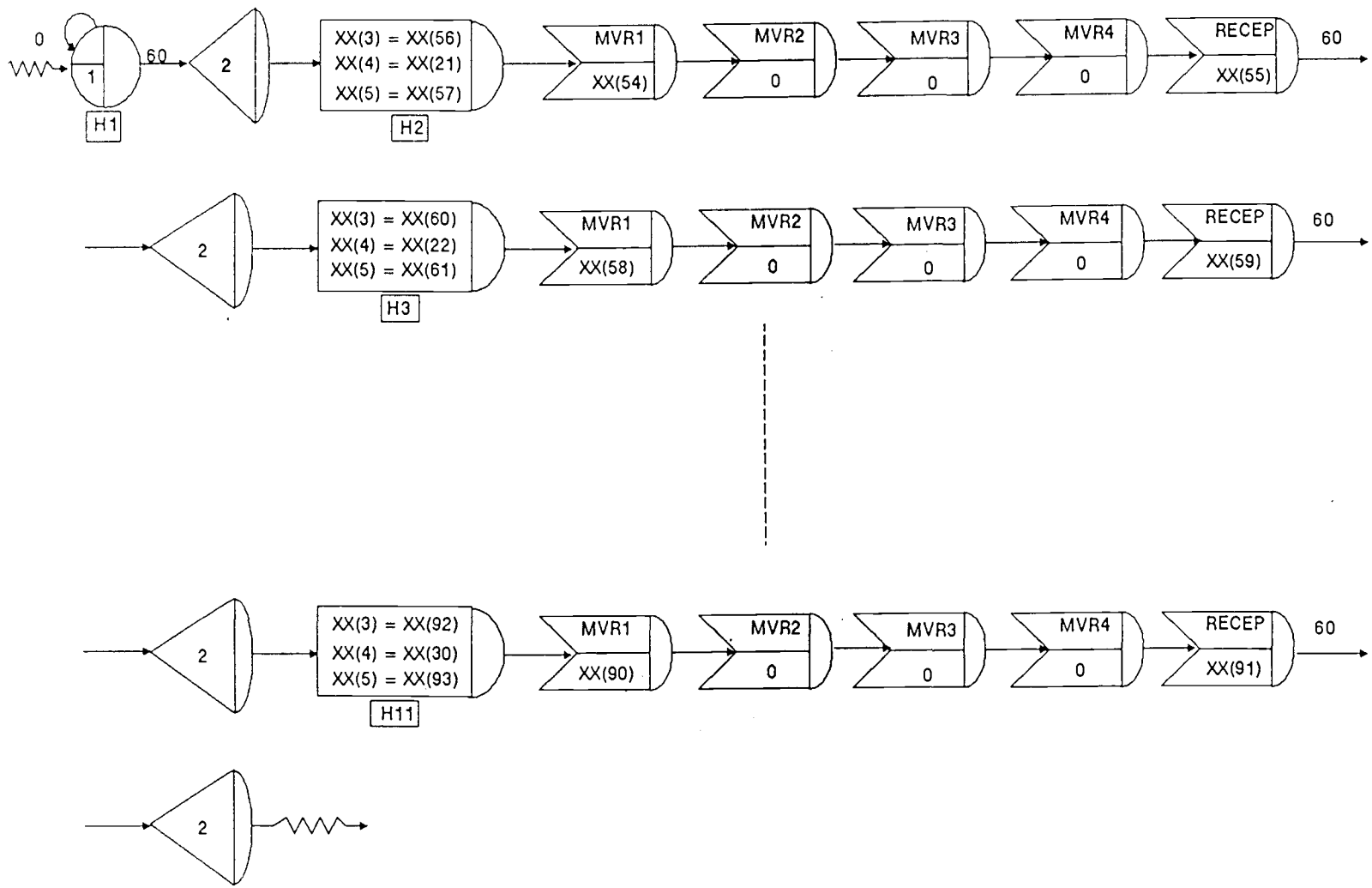
Figure 7. Initializing Subnetwork



The second subnetwork is shown in Figure 8. It models the changes in resources and arrival rates throughout a normal working day. By changing the values of model variables hourly, provided through data file parameters, the variability inherent in arrival rates and resource availability may be modeled. Hourly statistics are also collected by this subnetwork.

Again, a dummy entity is created and then time is delayed sixty minutes. The next event to occur is EVENT 2, the back end, which calls the subroutine for collecting the hourly statistics. The resources are then altered and arrival rates changed for the second hour of simulation. Time is delayed another sixty minutes and the pattern repeats until eleven hours has passed and the simulation run is done. The dummy entity is then terminated.

Figure 8. Updating Subnetwork



The customer arrival and routing submodel models the arrival of customers to a DMV field office and their initial path taken. The SLAM network diagram is shown in Figure 9.

The first event that takes place is the creation of drive line customers, those taking driving tests. Since the DMV schedules these as appointments, a known number of this type of customer will arrive per hour and will be considered separate from the "regular" customers. In addition, they are assigned a queue priority of one denoting going to the head of the queue (high value first) for the first available MVR. The time-between-creations is represented as a constant whose value (user input provided from the data file) will be changed each hour. As they are created, these customers have values assigned to their attributes according to the type of transaction, service rates, queue priority, etc. They then enter the system.

Regular customers, those not taking driving tests, are also created. As their arrivals are not by appointment, the arrival rate is represented as a variable rate whose mean time-between-creations (user input provided from the data file) will also be changed hourly. Their queue priority is set to zero and the type of transaction they will require is decided through predetermined probabilities. They are routed to the appropriate attribute assignments for service type,

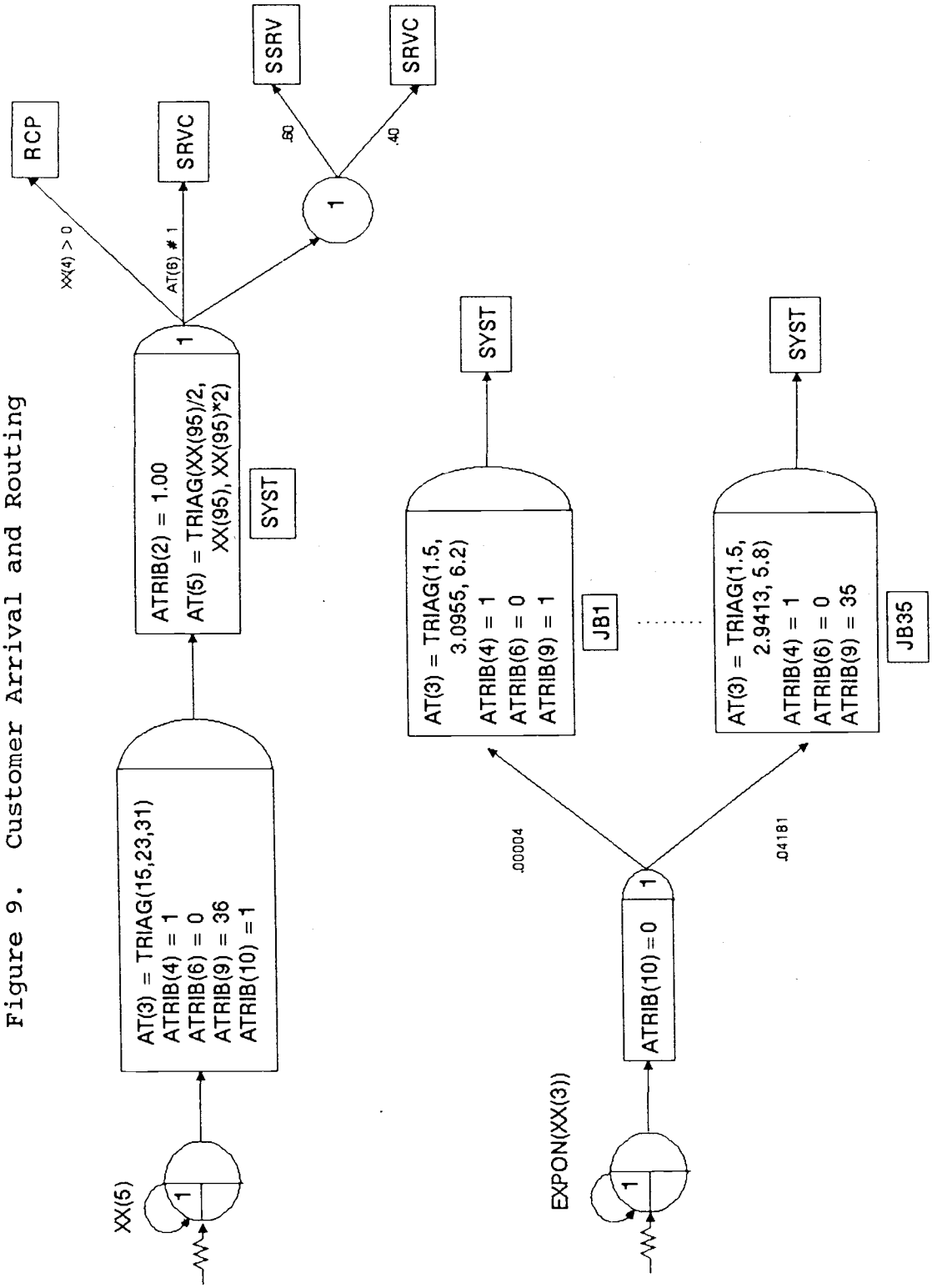


Figure 9. Customer Arrival and Routing

service time, etc., and then enter the system.

After customers have been created and attribute assignments made the customers are routed in the system. If there is a receptionist (XX(4)>0) on duty they will head toward the receptionist function. If no receptionist is on duty and self-service is not possible (AT(6)#1) for their transaction they go directly to MVR service. Otherwise, a percent of those able to use the self-service counter will use it (sixty percent) and the rest will go to MVR service.

The receptionist function models the receptionist station and the events that take place there. The SLAM network diagram is shown in Figure 10.

Within the receptionist function the customers are routed again. The receptionist is only able to contact a percent of the customers and this may vary from office to office. This percentage is specified by the user through the data file (XX(94)) and will be between zero and one hundred percent. Those not contacted will go to either self-service or MVR service as appropriate.

The customers that will be contacted by the receptionist wait in the receptionist queue until the gate (RCPQ) is opened. When opened, all of the customers in the queue are released and will take one of three paths. If the receptionist is scheduled to go off duty (XX(4)=0), the remaining customers in the queue are sent to MVR service. If the receptionist becomes free, they leave the

Figure 10. Receptionist Function

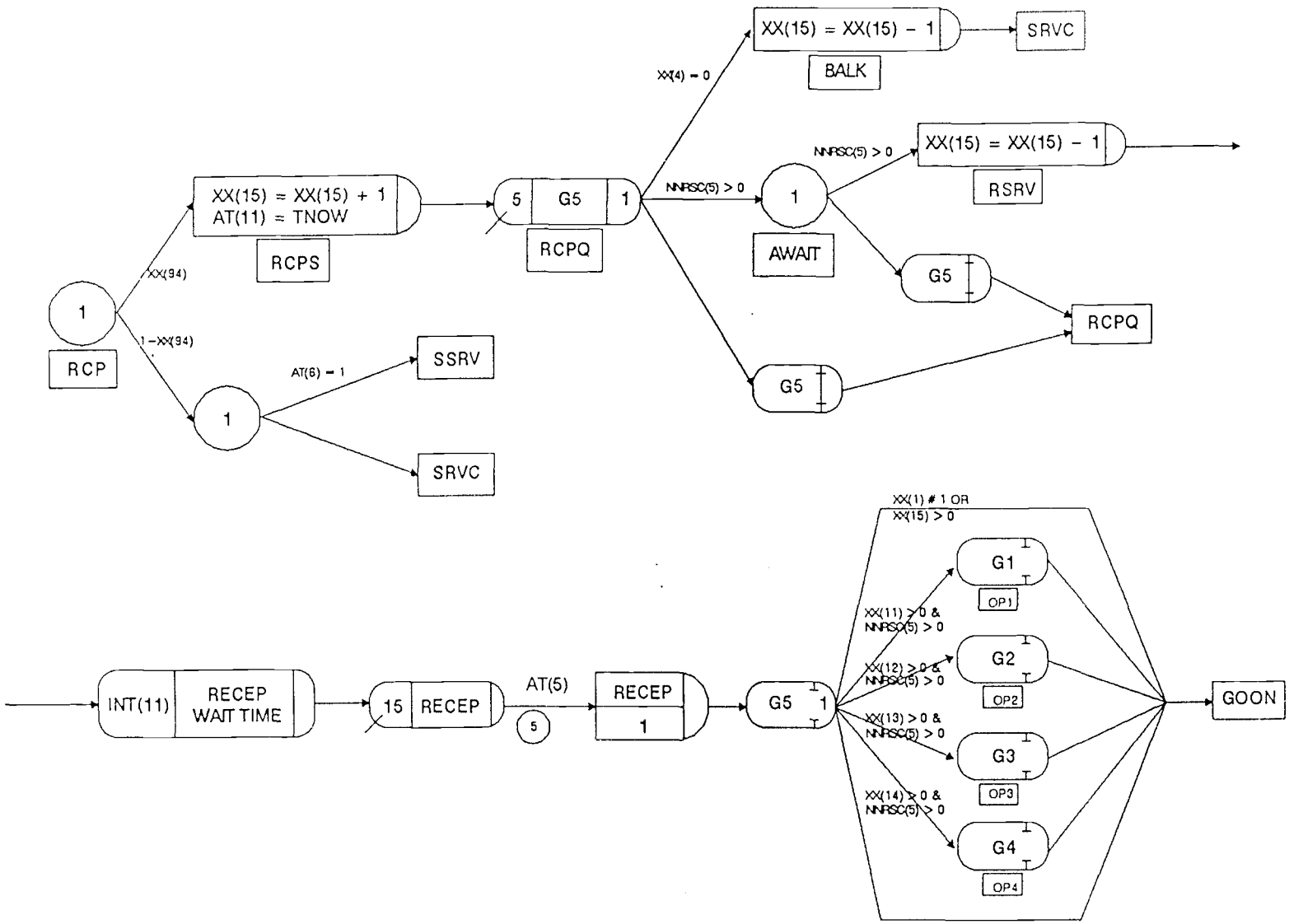
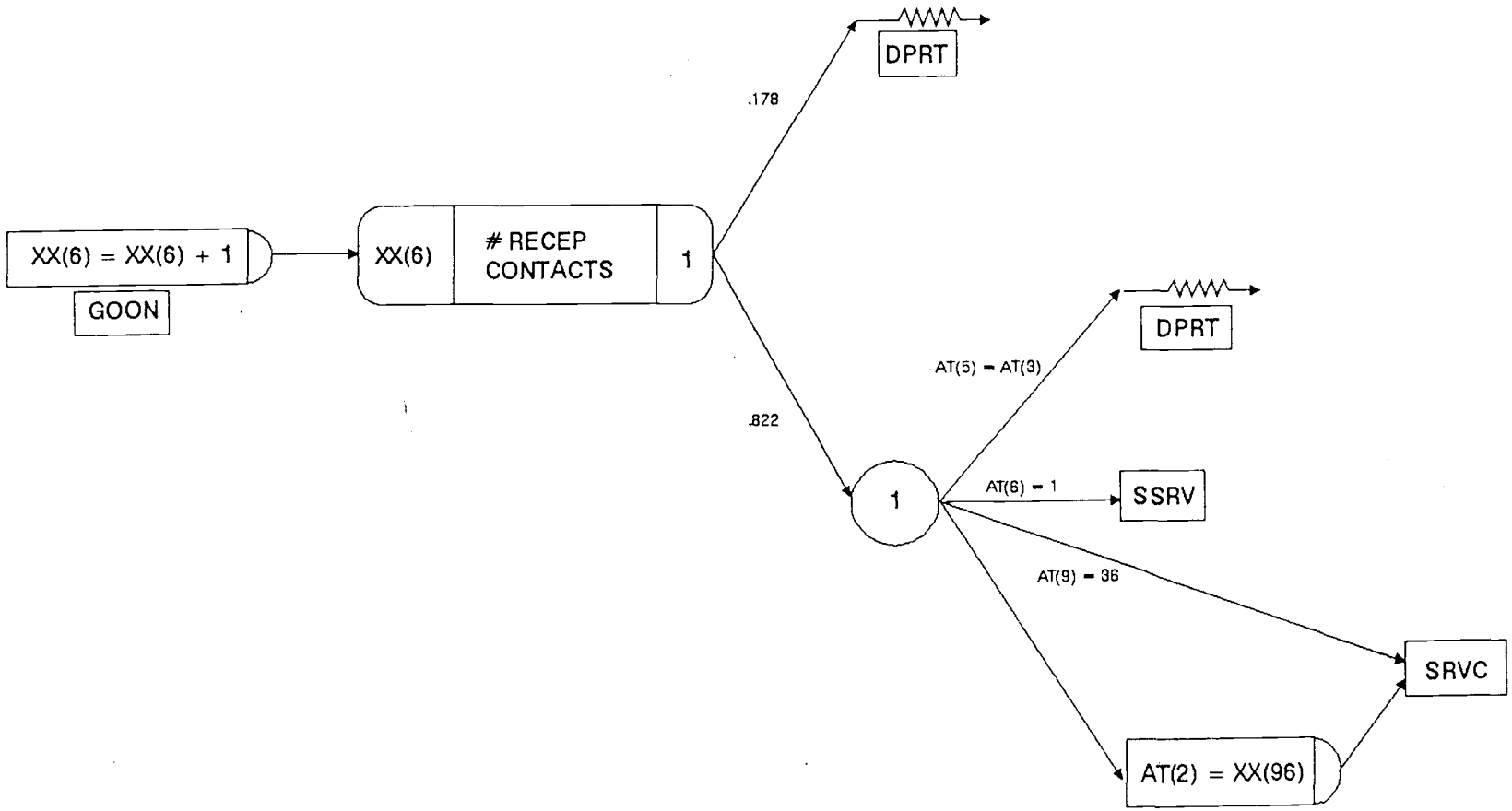


Figure 10. Receptionist Function (continued)



queue for receptionist help. Otherwise, they return to the queue. Attribute eleven keeps track of the waiting time in the receptionist queue and XX(15), queue length.

When a customer leaves after being helped, the receptionist is freed and the queue gate is opened. If its queue is empty and the receptionist function includes serving as an MVR when idle (XX(1)=1, user input provided from the data file) then a customer is removed from the MVR queue and serviced by the receptionist.

The customer who has had contact with the receptionist will either depart the DMV, or the receptionist will send them to self-service or MVR service as appropriate. Under two circumstances will a customer depart the DMV from the receptionist function: if their transactions were completed by the receptionist (receptionist service time = MVR service time) or if they were unprepared (currently 17.8 percent of customers).

A customer going on to MVR service will have their service time coefficient changed from 1.00 to a fraction. This constant coefficient will be multiplied by the MVR service time and represents the effect the receptionist has on lowering MVR service times.

The MVR function models the MVR service counters, queue and the events that take place there. Up to four different types of servers are available (for example, regular transaction servers and express window servers). Each type may have any number of servers working. A

separate queue is used for each type of server and all servers of the same type serve their queue. In the DMV model used for experimental analysis described later all servers are type one. Therefore, the user interface (front end, back end) was designed to accept information only on the receptionist and one type of MVR. See the Recommendations for Future Research section. The SLAM network diagram is shown in Figure 11.

Within this submodel there are three attribute values that play a key role. Attribute four stores the server (resource) type which in turn controls the queue file number and resource number the customer will use. Attribute seven stores the file number for the resource (server) dummy queue. A dummy queue was used to separate the seizing of a resource (an await node) from the actual queue (a gate node). This allows for selective customer jockeying of queues which would otherwise not be allowed after entering the queue. Attribute eight keeps track of the waiting time in the MVR queue.

The first event that takes place is in the queue switching rules. The switching rules are designed to route the customer to their type of server's queue or, if busy, to an available server of a different type. This is accomplished by changing the attribute four value to the appropriate server type based on resource availability (NNRSC). These rules have no effect on the model when only one type of server is specified, as in the DMV model

Figure 11. MVR Function

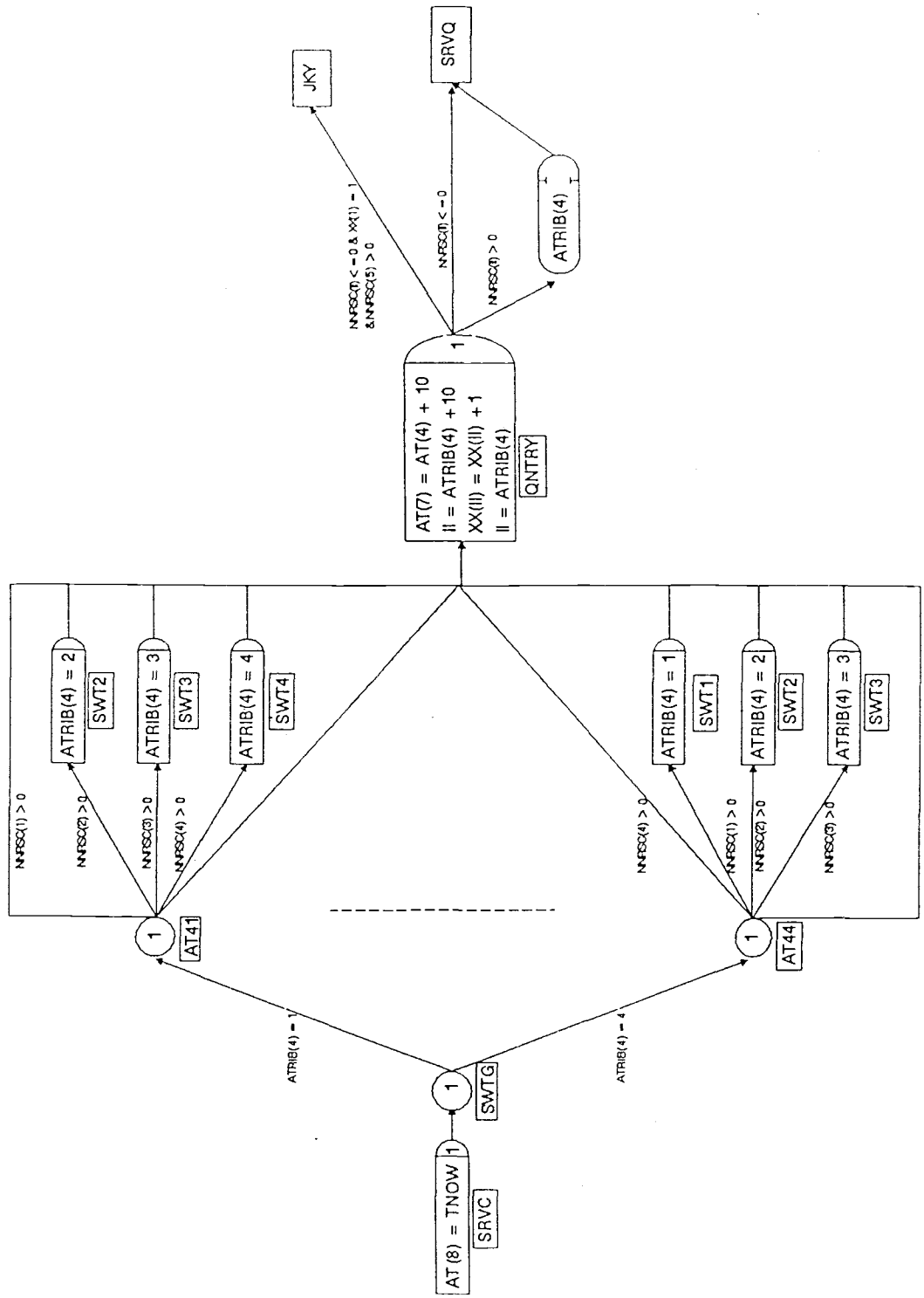


Figure 11. MVR Function (continued)

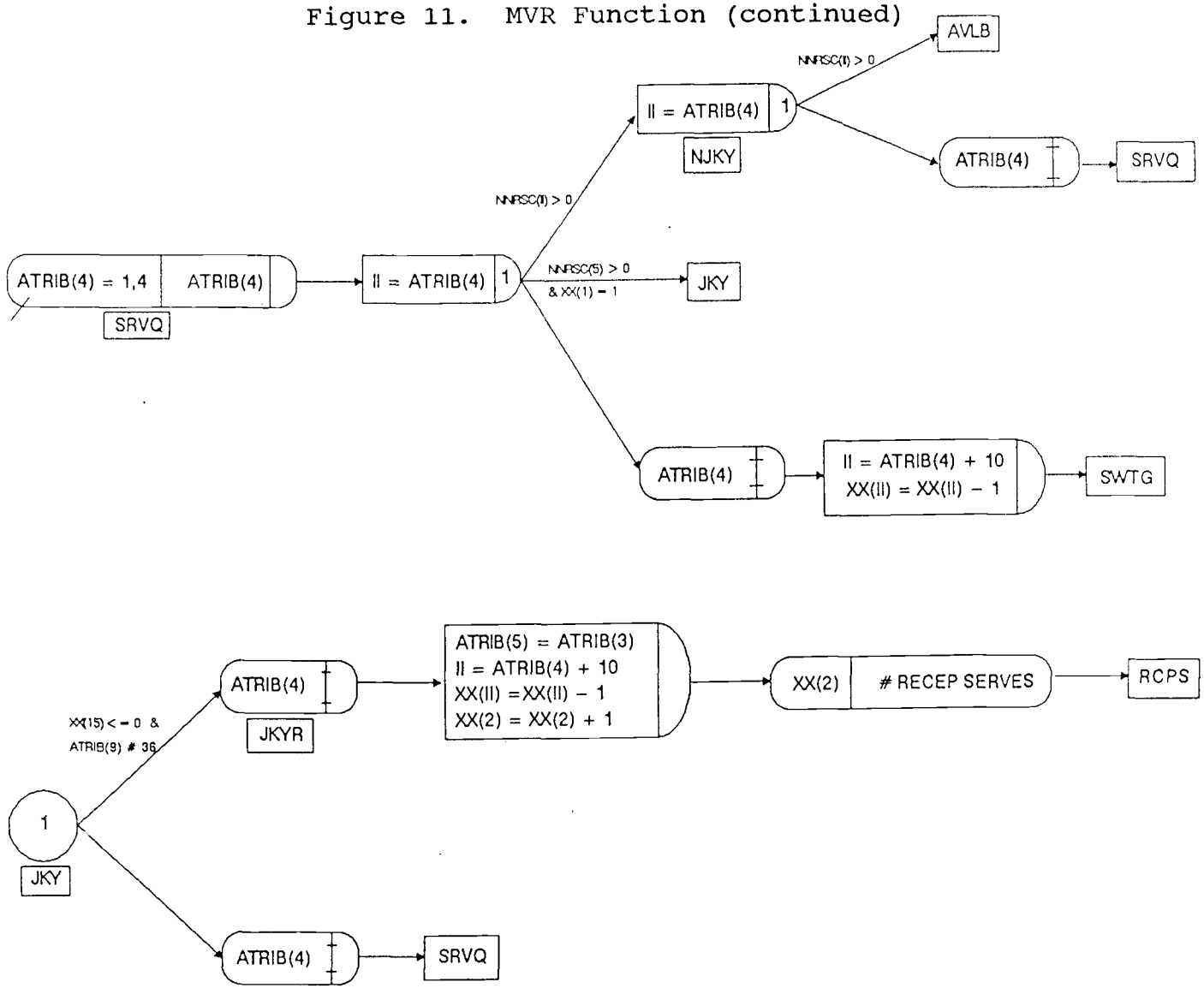
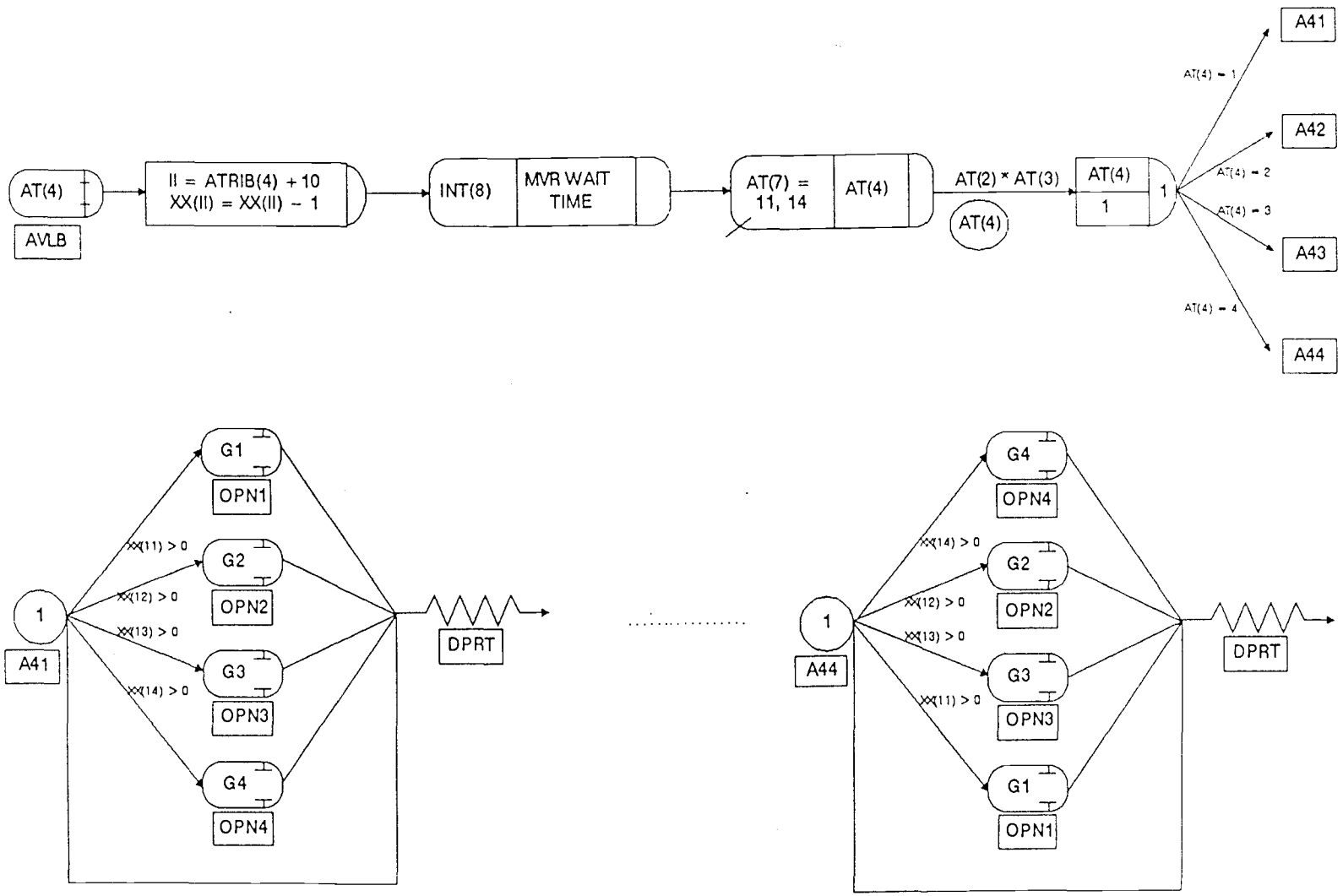


Figure 11. MVR Function (continued)



used for experimental purposes described later.

After determining the appropriate queue to enter, the customer goes through the QNTRY (queue entry) node where variable values change (for example, queue size is increased by one) and then one of three possible paths is taken. If all MVRs are busy, the receptionist is free and may serve as an MVR when idle (XX(1)=1), then the customer may jockey to the receptionist. If jockeying is not possible and all MVRs are busy, the customer waits in the server queue for the gate to open. If an MVR is free then the gate is opened and the customer passes through the queue and goes directly to the MVR.

At the JKY node another check is made to make sure the receptionist is free. If busy, the customer returns to the server queue. Otherwise, the customer leaves the MVR function for the receptionist. A count of these jockeys is made and other variables adjusted.

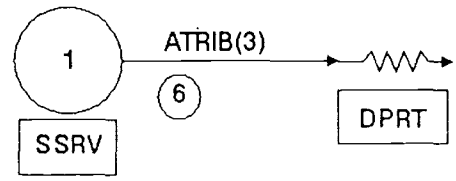
Customers enter a server queue based on their attribute four (server type) value and wait for the gate to be opened. All MVR queues are high value first priority in which drive line customers go to the head of the queue, due to their appointment. A gate is only opened when a receptionist becomes free or when an MVR becomes free. When a gate opens, all of the customers in that queue are released and follow one of three paths. If a server of its type is available (an MVR was freed) the customer does not jockey and is sent to the MVR. If a receptionist

became free and may serve as an MVR, the customer jockeys. Otherwise, the gate was opened after the freeing of an MVR of another type and the customer goes back to the switching rules to change queues.

As the customer goes to the available MVR they pass through the dummy queue and are then serviced. The net service time is calculated by multiplying the gross service time by the service time coefficient, attributes three and two respectively. The service time coefficient starts out as 1.00 and changes after receptionist contact. After service is over the MVR is freed and an MVR queue gate may be opened, with priority given to the server's own queue. If the server's queue is empty another type server gate may be opened. The customer then departs the system.

A customer may arrive at the self-service counter only if their transaction allows for self-service (determined by their attribute six value). This function was included for completeness sake and has no real effect on the present system as there is no waiting line at the self-service counter. This may change and additional constraints or collection of statistics could easily be included in the model. The SLAM network diagram is shown in Figure 12.

Figure 12. Self-Service Function

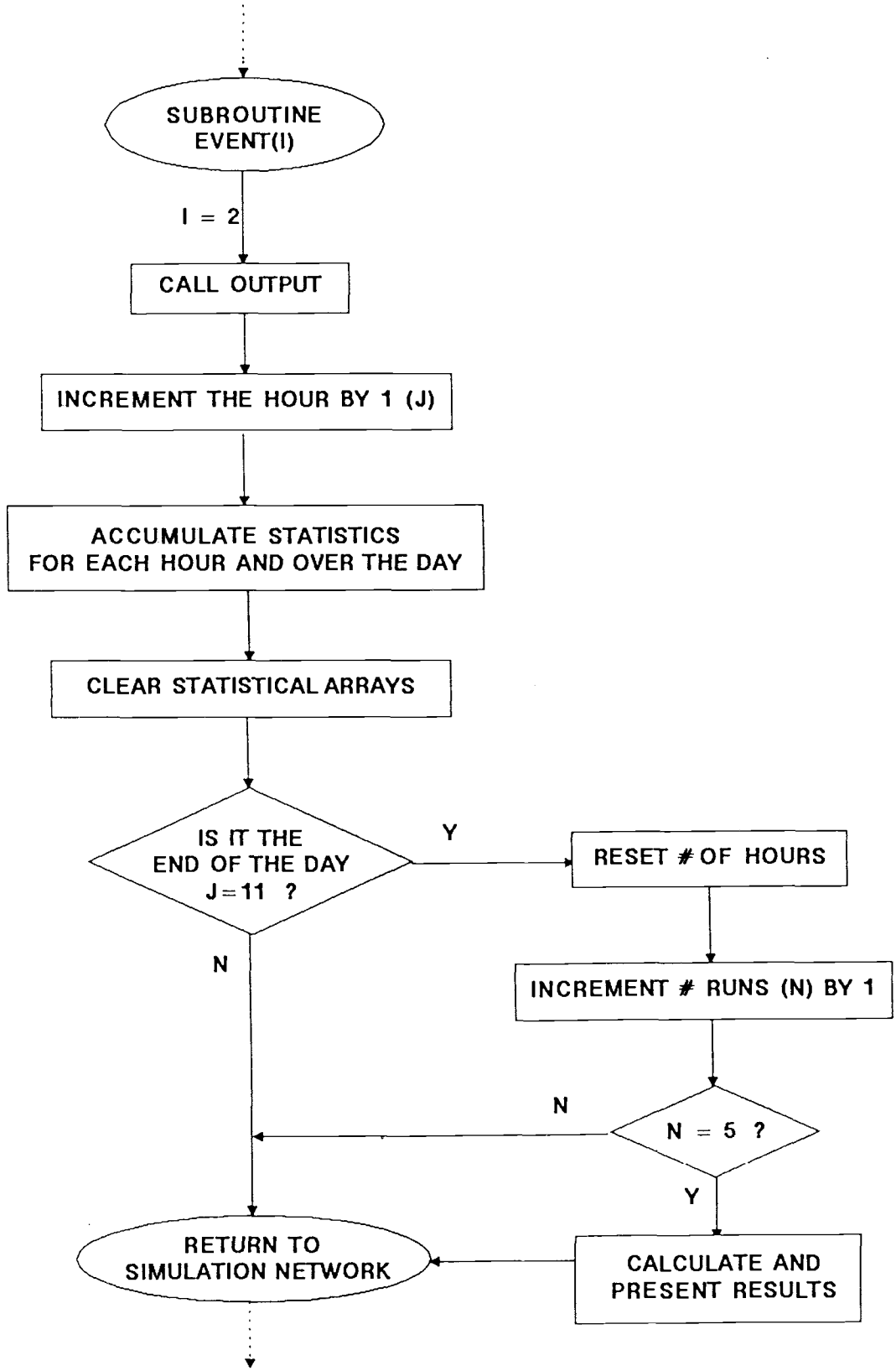


Back End

The back end of the program was designed to collect the statistics at hourly intervals, clear the statistical arrays and present the results of the five simulation runs in a concise, tabular format. Normally, SLAM would require the user to enter output file names after each hour, to execute the OUTPUT module and to enter several commands before a complex set of statistical results are displayed. These results would then have to be consolidated by the user. The back end was designed to do this for the user through FORTRAN subroutines called by EVENT 2. A flow chart of the back end is shown in Figure 13.

EVENT 2 is called every sixty minutes by the network during simulation. The variable J represents the number of hours of completed simulation and is incremented by one. The hourly statistics are then collected for each hour over the five runs. The daily statistics are also accumulated by summing over each hour and over the five runs. The statistical arrays are then cleared which will

Figure 13. Back End



allow the hourly statistics to be collected non-cumulatively. If it is the end of the day then J is reset to zero and the number of runs is incremented by one. If five runs have been completed then the statistics are averaged over the five runs and the results displayed. The program then returns control to the network where the menu subroutine will be called and the menu displayed (N=5). If it is not the end of the day or five runs not yet completed, then the program returns control to the network where the simulation will continue.

A complete listing of the SLAM network program is in Appendix 2. A complete listing of FORTRAN subroutines is in Appendix 3.

Data Collection/Uncertainty Modeling

There were two types of data collected: data that is stored internally within the network and data that is input through the front end. Much of the data collected was available from the Systems and Planning Section of the Motor Vehicles Division.

Data stored within the network includes: relative frequency of transaction types, transaction times and distributions, self-service possibility, self-service utilization, and percent turn-away from the receptionist. The information in Table 3 was provided by the Systems and Planning Section of the DMV.

Table 3. DMV Transactions

TRANSACTION TYPE	RELATIVE FREQUENCY	AVERAGE TIME MINUTES	SELF-SERVE?
additional cash receipts	.000040	3.0955	N
canceled plate/marker	.001455	4.2284	N
defensive driving course payments	.000464	4.9964	N
duplicate instruction permit	.001348	7.4410	N
duplicate titles	.006971	6.8922	N
duplicate titles w/ renewals	.003286	7.3862	N
equipment failure	.001376	1.7720	N
highway permits	.001574	4.7040	N
law test - ATD	.024352	1.2761	N
law test - written	.007470	4.5004	N
look-ups and driving records	.024572	4.4972	N
mileage reports	.002917	2.5372	N
mobile home trip permits	.000313	5.1237	N
motorcycle/moped drive test	.003369	5.0655	N
motorcycle endorsement	.002823	8.8689	N
original class 1,2,3	.004608	8.9649	N
original identification card	.011902	10.5049	N
original instruction permit	.022511	9.4764	N
original oprtr/combnd/moped	.042864	9.9196	N
paper renewals	.081178	4.8926	N
photo duplicate licenses	.054329	7.4410	N
photo renewal license	.101341	6.9306	N
re-exam drive test	.000621	12.0401	N
reinstatement fees	.013864	8.1975	N
replacement plates & stickers	.011273	6.1342	N
snow park permits	.003211	2.5072	N
student or emergency permits	.000050	15.9318	N
title only	.125817	5.1539	N
title with renewals	.163531	6.8655	N
trip permits	.075511	4.5242	N
unable to accommodate	.002344	1.0983	N
validated reminders	.130744	2.2750	N
vehicles address change	.024108	.4399	Y(60%)
vehicle ID marker/temp pass	.006068	5.3894	N
VIN inspection	.041796	2.9413	N
average drive test	n/a	23.1105	n/a

The relative frequency refers to the number of transactions occurring of that type within a year, divided by the total number of transactions for that year. These numbers were used in the network as the probabilities for

determining the type of transaction a customer will need.

The average transaction time is the average time that it takes an MVR to complete the transaction, including the time it takes the customer to reach the counter after their number has been called and for the MVR to greet the customer. These were used to generate service times for the customers arriving to the DMV.

Although the service time distributions had not been determined by the DMV, some actual timings were available with which the distributions were estimated. Using the Chi-Square Goodness of Fit test several common distributions (chi-square, erlang, exponential, gamma, log-normal, normal, student's t, triangular, uniform and weibull) were fit to the data and significance levels determined. Unfortunately, no single distribution fit all the transactions. The only distribution that would not have been rejected at the .05 significance level was the normal distribution for Titles Only which had a significance level of .677. However, the majority of the transactions were definitely not normally distributed being skewed to the right. In addition, Law and Kelton (1982) found "experience indicates that common distributions like the normal or uniform are typically not 'correct' for most simulation applications." As there was no "good" fit to the data, a triangular distribution was assumed for all transactions for several reasons.

First, it was apparent that the distribution of the

timings is not symmetrical in general. The triangular distribution can be specified through its minimum and maximum as skewed in either direction and at any magnitude.

Second, the minimum and maximum for the distribution also sets limits on the service times generated. This is important since the data suggests that there is a minimum service time somewhat larger than zero but less than the mean. This minimum represents the time it takes for the customer to arrive to the counter from the customer waiting area, be greeted by the MVR, and either complete the transaction in the shortest time or be turned away. The data also suggests that the maximum must be somewhat shorter than the work day but greater than the mean. Logically, there is a maximum on how long it takes to fill out a form and complete a transaction. A government office serving the public can not afford to spend an inordinate amount of time with one customer while the queue backs out the door.

Third, Law and Kelton (1982) recommend the use of the triangular distribution "when there are no system data available." Although some data were available, no standard distribution fit the observed data well.

The only transaction that could be completed at the self-service counter was for Vehicle Address Changes. Of this transaction, sixty percent of the customers use the self-service counter instead of an MVR to complete their

transaction.

Percent turn-away from the receptionist is defined as the percent of customer the receptionist would be able to turn-away from further service due to inadequate preparation. These people do not have either the necessary paper work or other requirements. This number was calculated from data in the Field Office Facilities Project (Mills, Jabs and Bouten, 1987) as the average number of persons contacted by the receptionist who left without further service. The average for the six offices studied was 17.80 percent. The range was 4.95 percent in Medford to 34.02 percent in Beaverton. The width of the range is most likely due to inconsistencies in the training, experience and use of the receptionists in each office rather than demographics or sample size. Sensitivity analysis was conducted with these numbers as described in the RESULTS section.

The data collected for input to the program includes: customer arrival rates, typical number of appointments for drive tests, service time coefficient, receptionist service time, and the fraction of customers contacted by the receptionist.

Customer arrival rates varied from office to office and day to day. Although the actual arrival rates were not available, the number of transactions per hour was available from the DRIVE system at the DMV in Salem. These numbers were used to determine "typical" levels of

business such that upper and lower bounds could be set for the simulation runs. For a small office (Tillamook) the number of customers arriving per hour ranged from zero to twenty-two. The average arrivals per hour was approximately eight. For a large office (Beaverton) the range was from twelve to eighty-four. The average arrivals per hour was approximately thirty-seven. The arrival rate distribution was assumed to be exponential because of its memoryless property.

The number of drive tests scheduled per hour is available from the drive test sheets in each field office. For example, the North Salem drive test sheets showed an average of 19.8 tests per day. A "typical" day would have the following drive test appointments:

hour	1	2	3	4	5	6	7	8	9
# tests	2	3	2	3	0	3	3	3	1

The service time coefficient is the multiplying factor for the MVR service times. This represents the effect that the receptionist has on lowering transaction times. This number is as yet unquantifiable; however, a best-guess by DMV personnel would be a three percent reduction. Sensitivity was therefore conducted on this factor. The lower limit was assumed to be zero, the receptionist has no effect and will not raise the service time. The upper limit was estimated to be an "optimistic" ten percent. See the RESULTS section for the sensitivity

analysis.

The receptionist service time represents the time the receptionist spends in contact with a customer. Although some of the offices have receptionists, no work measurement timings are available as yet. In addition, timings of the current situation may not be valid if new guidelines are defined for the receptionist function. Consequently, a best-guess estimate by DMV personnel of forty seconds was used. Sensitivity analysis was conducted on this number as described in the RESULTS section.

The fraction of customers contacted by the receptionist was based on the expected receptionist function guidelines. According to DMV personnel, the purpose of the receptionist would be to greet customers and give them a number for waiting. They expect the receptionist to be able to contact almost all (if not all) of the customers. Therefore, ninety-nine percent was used allowing for the few that might slip by without contact.

RESULTS

Experimental Application

The purpose of developing this simulation program was to study the effects of a receptionist in a DMV field office in order to improve customer satisfaction. Specifically, the questions to be answered are:

1. Is there a cut-off point where an MVR is more effective as a receptionist rather than behind the counter serving customers?
2. Is the cut-off point at one particular arrival rate or does it vary for each office size (number of workers)?
3. Should the receptionist also serve customers out of the MVR queue when idle?

Effectiveness is measured in terms of waiting times, queue length and worker utilization (earned hours/worked hours). By decreasing waiting times and/or decreasing queue length customer satisfaction could be improved. Decreasing worker utilization implies that the workers would be more efficient (same amount of work in less time) at completing their tasks and would also be beneficial.

In order to answer these questions the simulation program was run over a variety of scenarios to represent a wide range of field offices. Field office size and volume of business were varied and compared between three

options: all workers as MVRs and no receptionist; one worker as a receptionist and the rest as MVRs; or one worker as a receptionist who also serves customers out of the MVR queue when idle, and the rest as MVRs. In general, field office size and volume of business are positively correlated with the larger offices having a higher volume of business.

The size of a field office, defined here as the number of MVRs working behind the counter serving customers, was varied from three to seven. Although many field offices have more than seven MVRs, the number who spend the majority of their time behind the counter serving customers is somewhat less than the total on duty. These other MVRs may be doing dealer work or other non-customer activities. The number of MVRs working also varies throughout the day; however, for experimental purposes it was held constant over the day and only varied between simulation runs.

The volume of business is defined as the rate at which customers enter the field office. The arrival rate varies for each office and depends on the time of day and day of week or month. Therefore, for each office size defined above the mean customer arrival rate (customers per hour as entered into the DMV Staffing and Arrivals table) was varied as in Table 4. The lower end of the range was determined when the mean arrival rates entered into the table generated waiting times close to zero. The

upper end of the range was determined by the limits of the system (the number in the system exceeded the maximum allowable by SLAM). Like the number of MVRs, the mean arrival rate was held constant over the day and only varied between runs. This allows for isolating the key factors that affect the queues and determining cut-off points between having a receptionist or not.

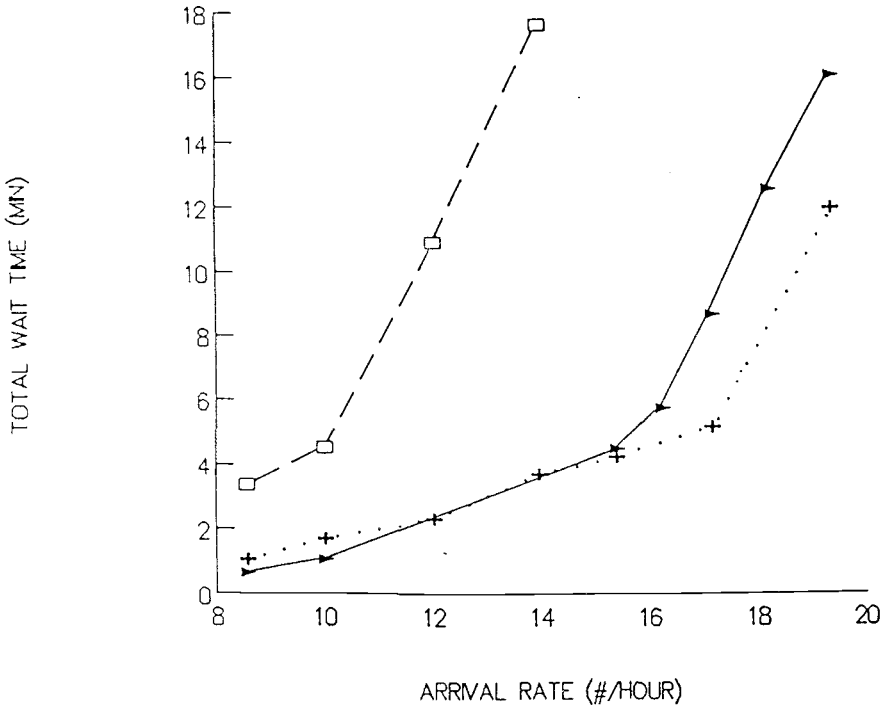
Table 4. Office Sizes and Arrival Rates Simulated

# MVRs	ARRIVAL RATES SIMULATED	
	minimum	maximum
3	8	20
4	14	32
5	23	43
6	28	52
7	40	60

The results of the simulated runs are presented in Figures 14 to 23. Plotted on each graph are three lines which represent the three options in which the workers may be scheduled.

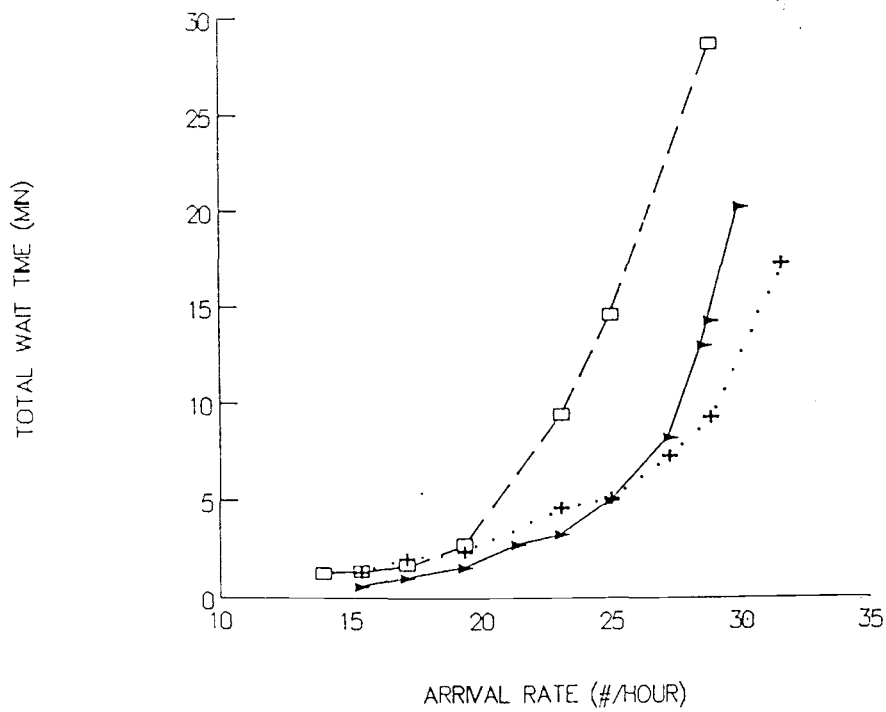
Figures 14 to 18 graph total waiting time versus customer arrival rate for each of the office sizes. Total waiting time is defined as the customer waiting time in the receptionist queue, if present, plus the waiting time in the MVR queue.

Figure 14. Three Workers: Wait Time vs. Arrival Rate



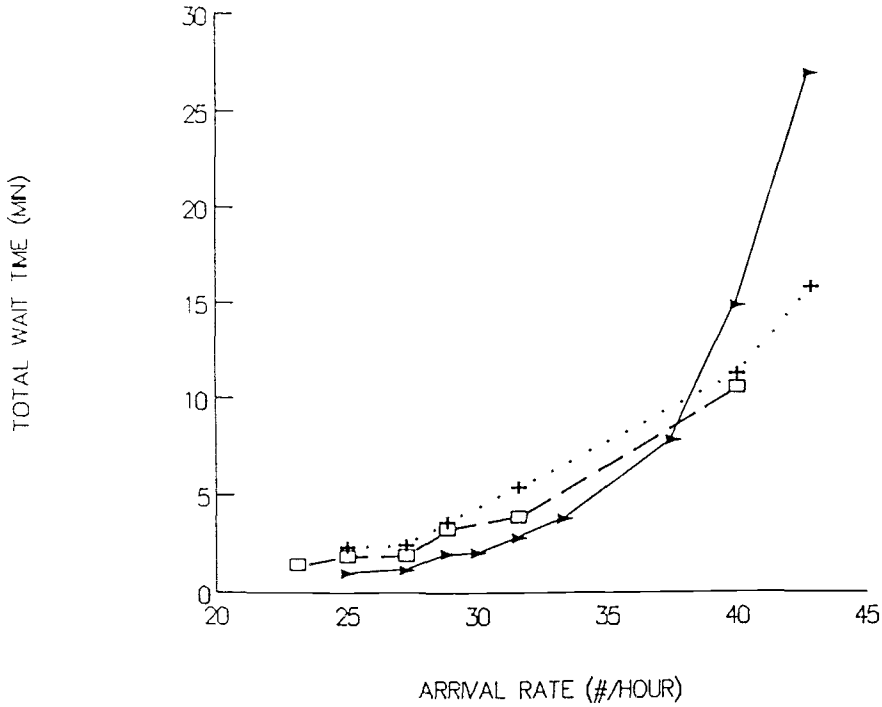
___ 3 MVRs _ _ _ 2 MVRs 1 Recep ... 2 MVRs 1 Recep(mvr)

Figure 15. Four Workers: Wait Time vs. Arrival Rate



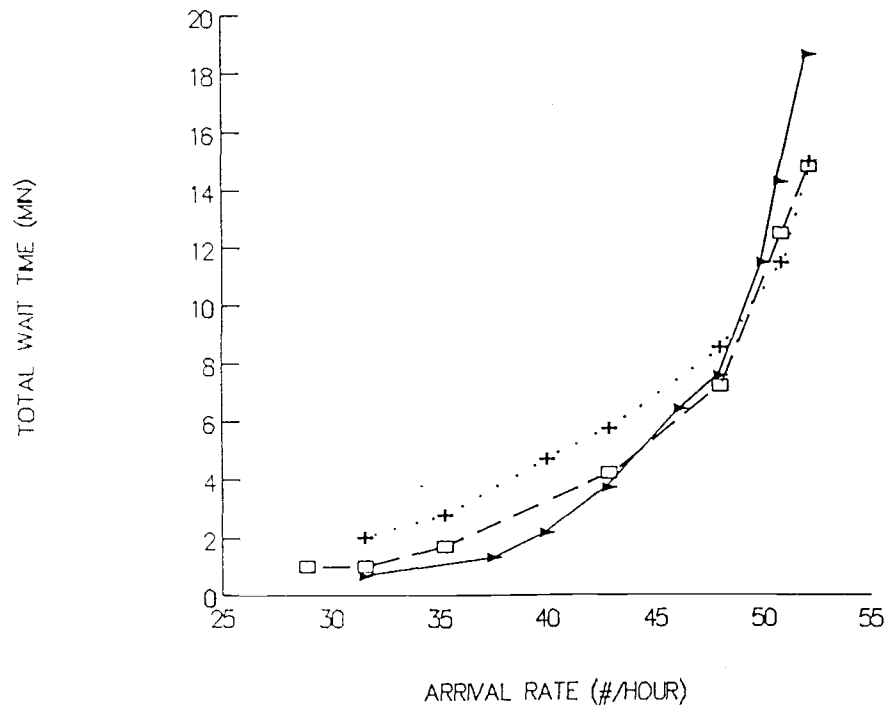
___ 4 MVRs _ _ _ 3 MVRs 1 Recep ... 3 MVRs 1 Recep(mvr)

Figure 16. Five Workers: Wait Time vs. Arrival Rate



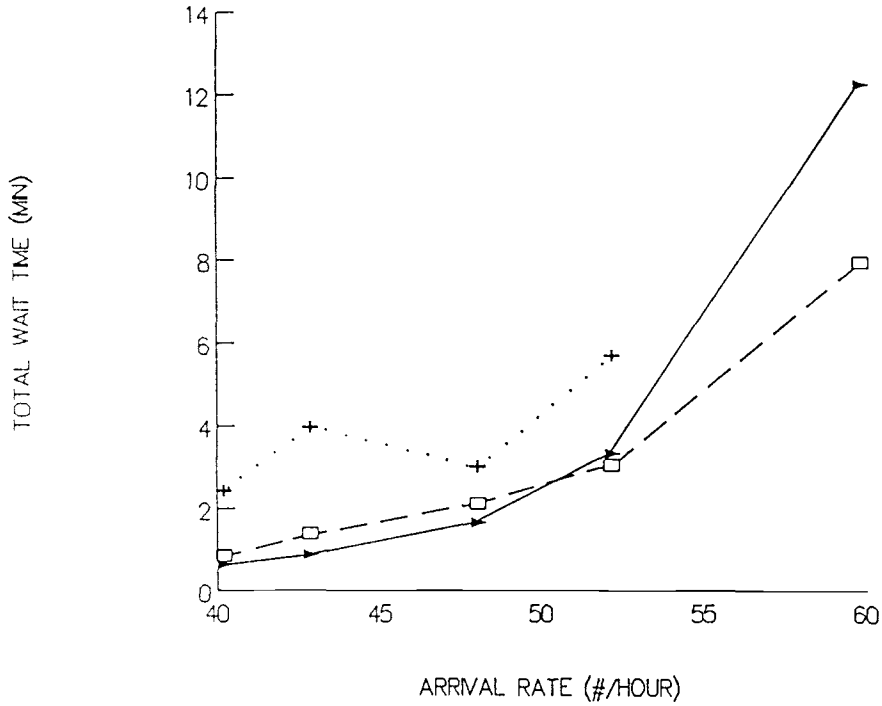
____ 5 MVRs _ _ _ _ 4 MVRs 1 Recep ... 4 MVRs 1 Recep(mvr)

Figure 17. Six Workers: Wait Time vs. Arrival Rate



____ 6 MVRs _ _ _ _ 5 MVRs 1 Recep ... 5 MVRs 1 Recep(mvr)

Figure 18. Seven Workers: Wait Time vs. Arrival Rate



___ 7 MVRs _ _ _ 6 MVRs 1 Recep ... 6 MVRs 1 Recep(mvr)

It is apparent from the five graphs that as customers arrive at a faster rate the amount of time spent waiting will increase for each of the three options.

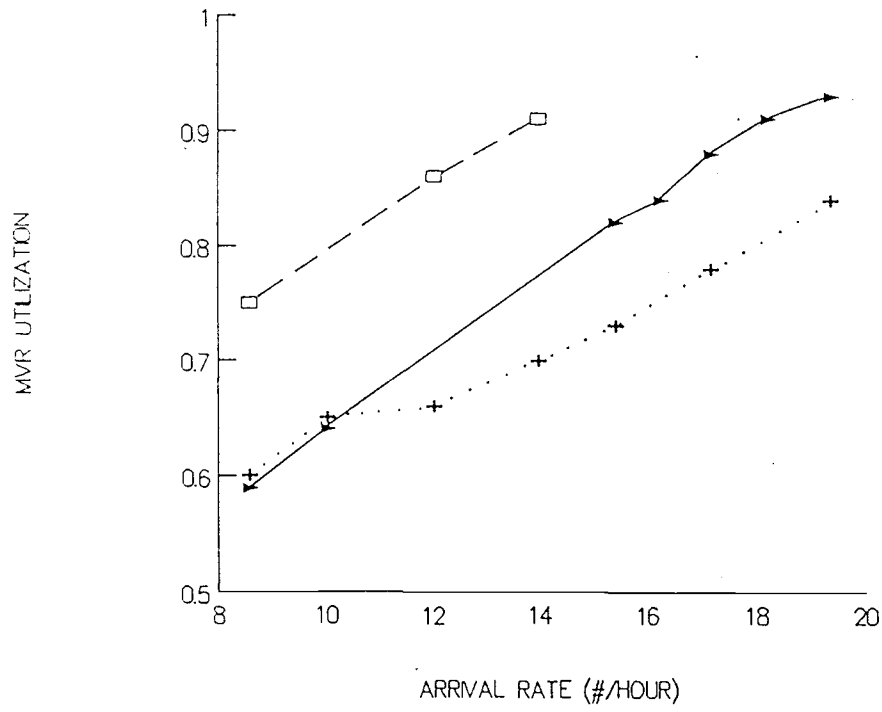
In general, at low arrival rates the option which generates the shortest waiting times is to have all workers act as MVRs and no receptionist. It should also be noted that at these lower arrival rates several of the lines are very close and may be viable alternatives to the minimum.

As the arrival rates increase it becomes beneficial to have a receptionist. In Figures 14 and 15, three and four workers respectively, the option of having one worker as a receptionist who acts as an MVR when idle generates

the lowest waiting times at high arrival rates. In Figures 16 to 18, five, six and seven workers respectively, the option of having one worker as a receptionist only doing receiving duties generates the lowest waiting times at high arrival rates.

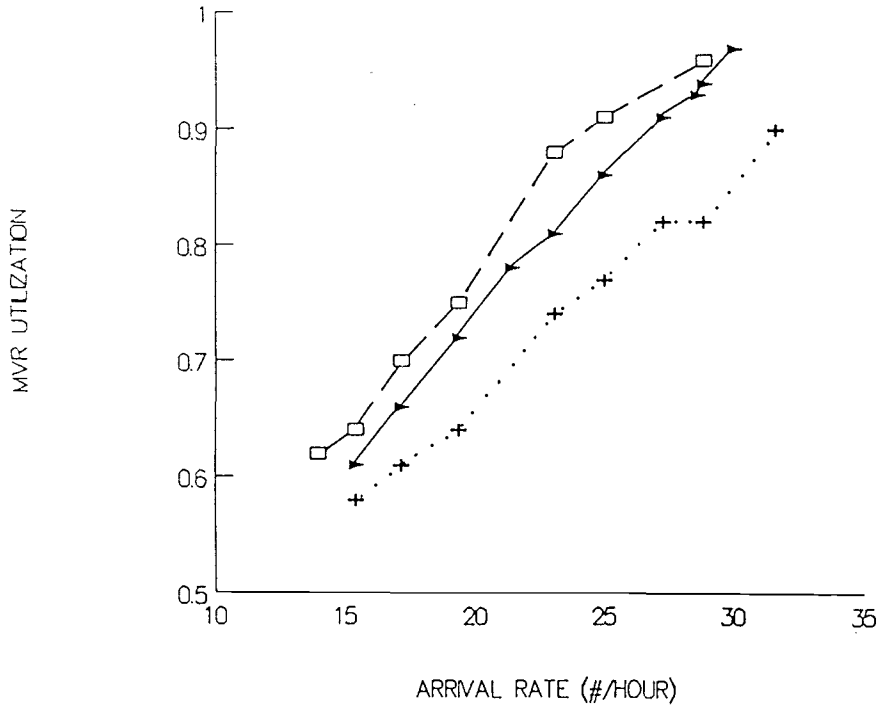
Figures 19 to 23 graph MVR utilization versus arrival rate. MVR utilization is defined as earned hours divided by worked hours for the MVRs behind the counter serving customers. The receptionist is not included in calculating MVR utilization.

Figure 19. Three Workers: Utilization vs. Arrival Rate



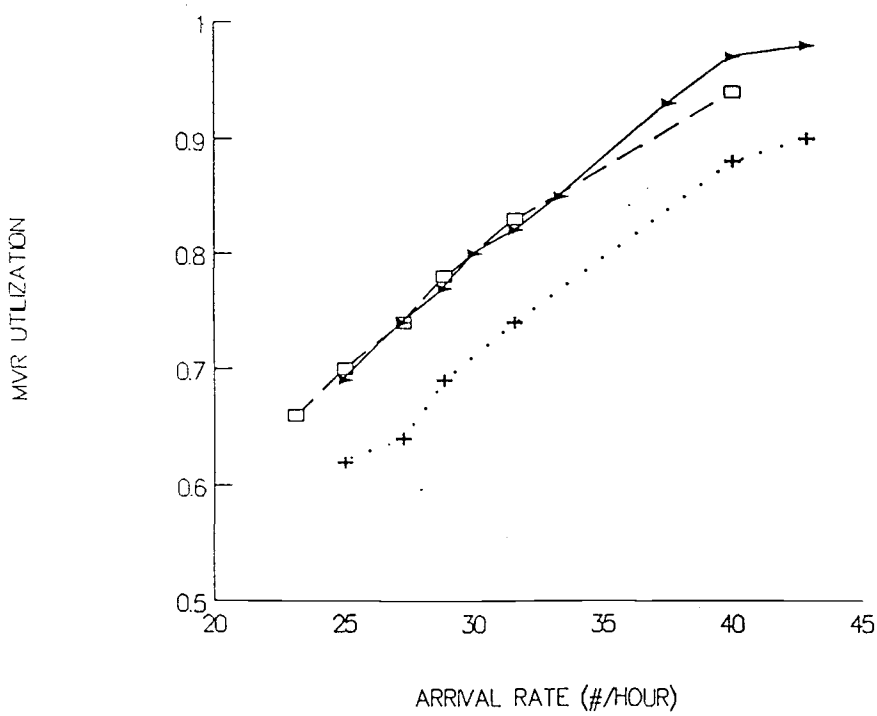
____ 3 MVRs _ _ _ 2 MVRs 1 Recep ... 2 MVRs 1 Recep(mvr)

Figure 20. Four Workers: Utilization vs. Arrival Rate



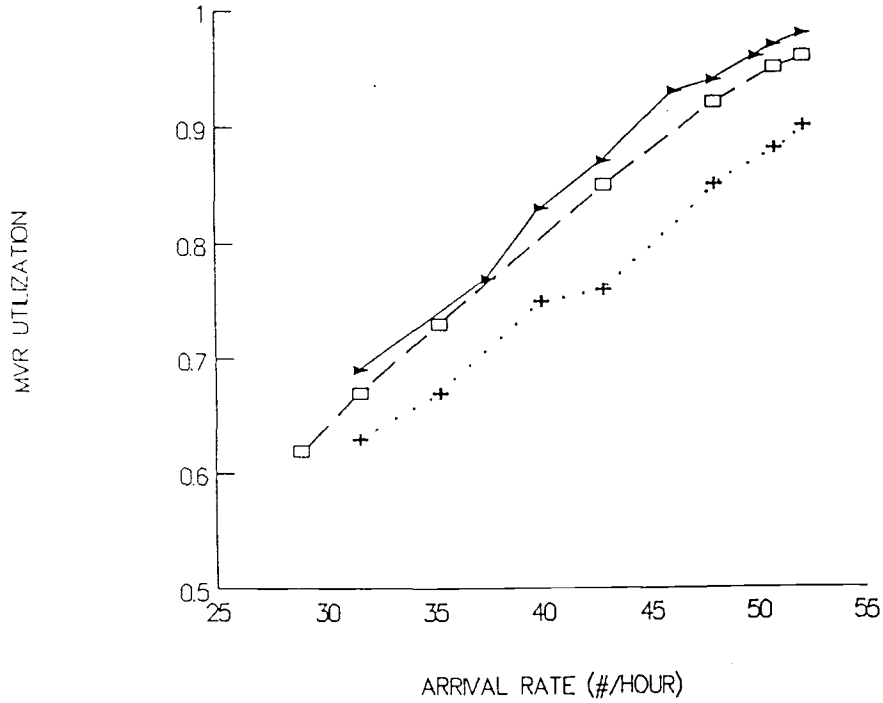
____ 4 MVRs _ _ _ 3 MVRs 1 Recep ... 3 MVRs 1 Recep(mvr)

Figure 21. Five Workers: Utilization vs. Arrival Rate



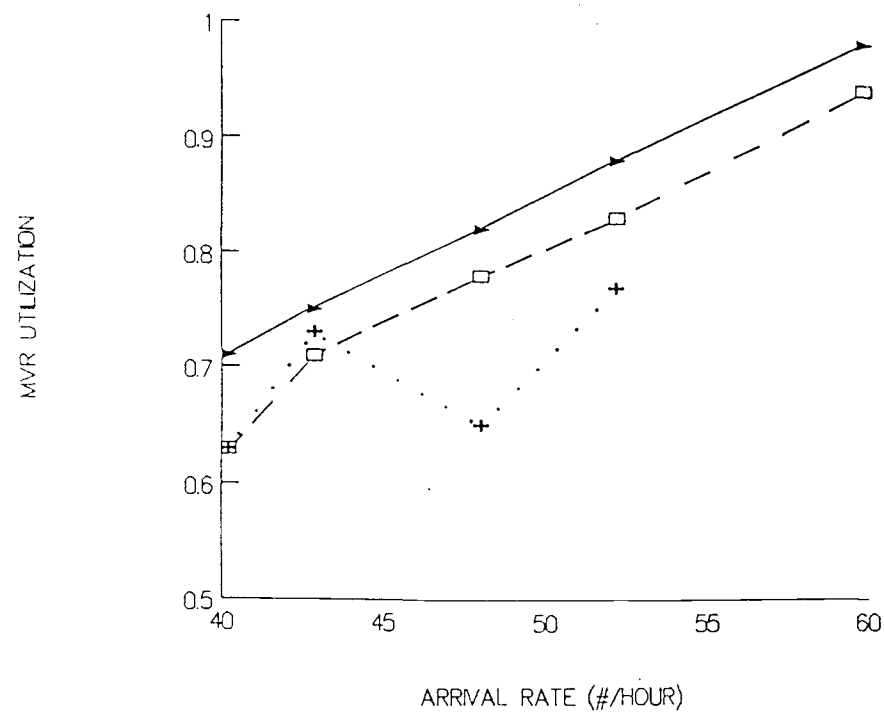
____ 5 MVRs _ _ _ 4 MVRs 1 Recep ... 4 MVRs 1 Recep(mvr)

Figure 22. Six Workers: Utilization vs. Arrival Rate



____ 6 MVRs _ _ _ 5 MVRs 1 Recep ... 5 MVRs 1 Recep(mvr)

Figure 23. Seven Workers: Utilization vs. Arrival Rate



____ 7 MVRs _ _ _ 6 MVRs 1 Recep ... 6 MVRs 1 Recep(mvr)

Again, it is apparent that as the customer arrival rate increases the utilization of the MVRs increases. This is logical because as more customers arrive per hour the MVRs will become busier.

Another noticeable trend is that the option of having one receptionist also acting as an MVR when idle generally has the lowest utilization. While MVR utilization for the no receptionist option seems to be independent of office size, the option with one receptionist drops in utilization as the offices increase in size.

The results from the simulation study are summarized in Table 5. The columns represent the three ways in which the workers may be scheduled: all workers (X) as MVRs and no receptionist [X,0]; one worker as a receptionist and the rest as MVRs [X-1,1]; or one worker as a receptionist serving customers out of the MVR queue when idle, and the rest as MVRs [X-1,1(mvr)]. For each office size (number of workers) the scheduling options which generated the minimum waiting times were determined. A "no" entered in a column means that the option was not minimum among the three options, at any arrival rate. Those options that generated minimum waiting times are denoted by the range of arrival rates in which they were minimum.

The first trend to be noted here is that when the offices are not busy (low customer arrival rates) it is best not to have a receptionist. The effect the

receptionist would have on lowering service time and reducing the number in the MVR queue would not compensate for the extra time spent in the receptionist queue. In addition, the lowering of MVR time and queue size is not needed when the MVRs are already not very busy. However, as the offices get busier it becomes more effective to have one of the MVRs serve as a receptionist.

Table 5. Arrival Rates at Minimum Waiting Times (customers per hour)

	#MVRs, #RECEPTIONISTS		
X workers	X,0	X-1,1	X-1,1(mvr)
3 workers	0 - 14	no	14 +
4 workers	0 - 25	no	25 +
5 workers	0 - 38	38 +	no
6 workers	0 - 45	45 +	no
7 workers	0 - 51	51 +	no

The second trend to be noted is in the duties of the receptionist. In the smaller offices it is necessary for the receptionist to also serve customers out of the MVR queue when not busy with their receiving duties. Because the removal of an MVR from behind the counter to the receptionist function has a large effect on the number serving customers (a decrease of 25-33%), the receptionist must help the MVRs. The smaller offices also have lower arrival rates to begin with and, consequently, the increase in duties does not create a large receptionist queue. In the larger offices the receptionist should only

do receiving duties. The removal of an MVR from behind the counter to the receptionist function has a smaller effect on the number serving customers (a decrease of 14-20%). In addition, the larger offices have higher arrival rates and the receptionist is busier to begin with. The additional duties of serving customers out of the MVR queue would cause the receptionist queue to increase to the point where the decrease in the MVR queue does not compensate for the increase in the receptionist queue.

The table above represents the minimum waiting times obtained by the computer simulations; however, at several times the other scheduling options generated waiting times very close to the minimum. The question then becomes: is there any significant difference between the three options?

Pairwise t-tests were conducted at the .05 significance level with the null hypothesis being that the two population means are equal, and the variances unknown and may or may not be equal. The comparison of waiting times for significant differences between the three options is summarized in Table 6. Many of the arrival rates in this table overlap the three options meaning that there is no significant difference and any of those options could be used. Consequently, the table below is a more flexible tool for planning the allocation of resources and allows for the use of a receptionist at any time of the day, slow or busy.

Table 6. Arrival Rates for Planning Resource Allocation (customers per hour)

	#MVRs,#RECEPTIONISTS		
X workers	X,0	X-1,1	X-1,1(mvr)
3 workers	0 - 17	no	10 +
4 workers	0 - 29	no	24 +
5 workers	0 - 41	31 +	35 +
6 workers	0 - 52	40 +	45 +
7 workers	0 - 54	48 +	no

Sensitivity Analysis

Sensitivity analysis was conducted on three of the simulation model variables: the percent in MVR service time reduction after customer contact with the receptionist, the percent of customers the receptionist would turn away from further service, and the receptionist service time. Because there have been limited work measurement studies done on the receptionist function little is known about the above variables. Either a best-guess estimate was made by DMV personnel or an estimate was made based on related data.

The same model was used for all of the sensitivity analyses. The arrival rate was set at forty customers per hour, and there were four MVRs behind the counter serving customers and one receptionist doing only receiving duties.

Percent Reduction in MVR Service Time

Percent reduction in MVR service time had been estimated by DMV personnel at three percent. A lower limit of zero was assumed, implying that the receptionist has no effect and will not raise the service time. An upper limit was estimated by DMV personnel to be an "optimistic" ten percent.

Figure 24 shows the effect changing the percent in MVR service time reduction has on total waiting time. As the percent in MVR service time reduction was varied from zero to ten percent there was a significant effect on the system. The average total waiting time for customers (time in receptionist queue plus time in MVR queue) decreased by over fifty percent as the percent reduction in service time was increased. In other words, as the service time was decreased the waiting time decreased.

A similar effect on the utilization of MVRs is shown in Figure 25. As the service time reduction is increased from zero to ten percent, the utilization decreases six percent.

Figure 24. Total Wait Time vs. % MVR Time Reduction (4 MVRs, 1 Recep, Arrival Rate = 40/hour)

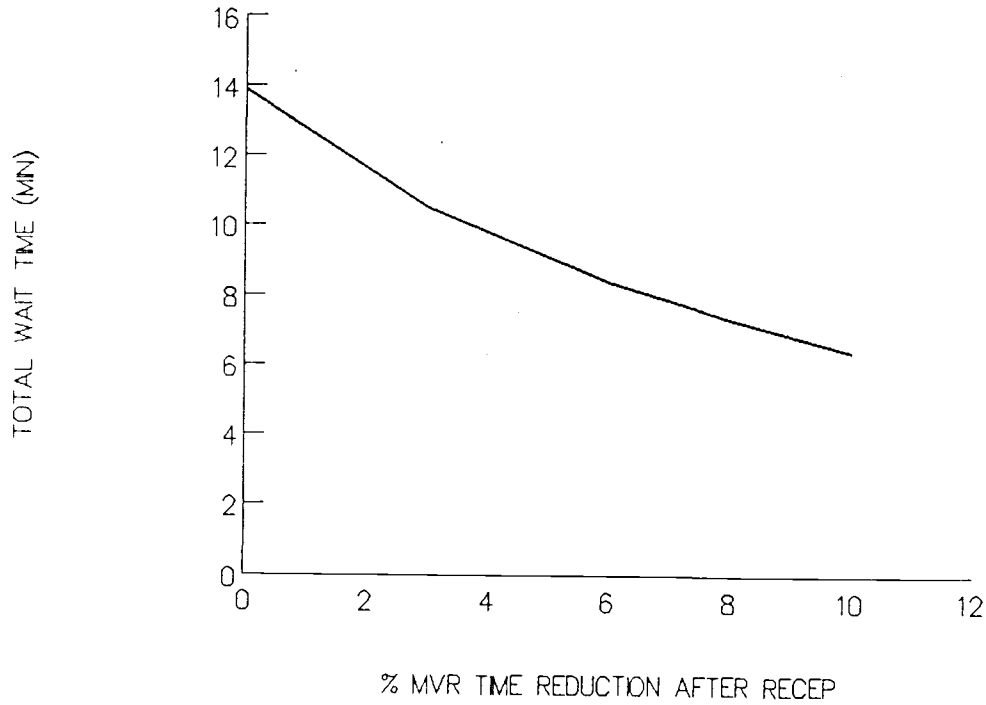
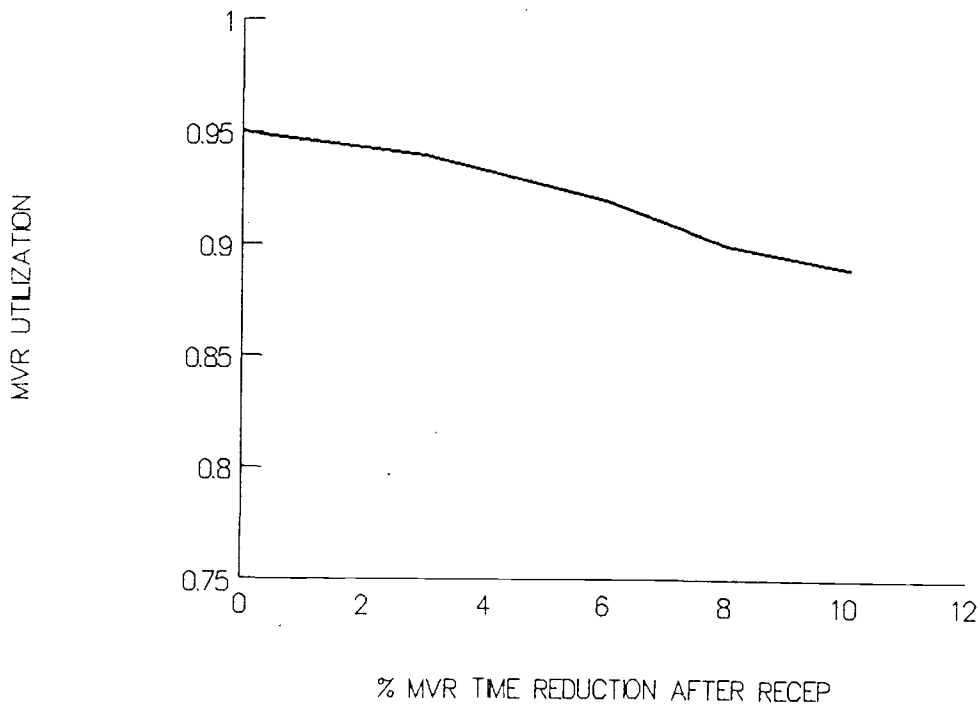


Figure 25. MVR Utilization vs. % MVR Time Reduction (4 MVRs, 1 Recep, Arrival Rate = 40/hour)



Percent Turn Away

The percent of customers the receptionist would be able to turn away from further service was calculated from data in the Field Offices Facilities Project (Mills, Jabs and Bouten, 1987). The average for the six offices with a receptionist studied was 17.8 percent. The values ranged from 4.95 percent to 34.02 percent. Percent turn away was therefore varied from five to thirty-five percent.

Figure 26 shows the effect changing percent turn away has on total waiting time. As the percent of customers turned away from the receptionist was varied from five to 17.8 percent (the average) there was a significant effect on the system. The average total waiting time for customers (time in receptionist queue plus time in MVR queue) decreased by over seventy-five percent. As the percent was changed from 17.8 to thirty-five percent the total wait time decreased by fifty percent.

A similar effect on the utilization of MVRs is shown in Figure 27. As the turn away percent is increased from five to thirty-five percent, the utilization decreases twenty-five percent. In this case the decrease in utilization is due to fewer people entering the MVR queue.

Figure 26. Total Wait Time vs. % Turn Away
(4 MVRs, 1 Recep, Arrival Rate = 40/hour)

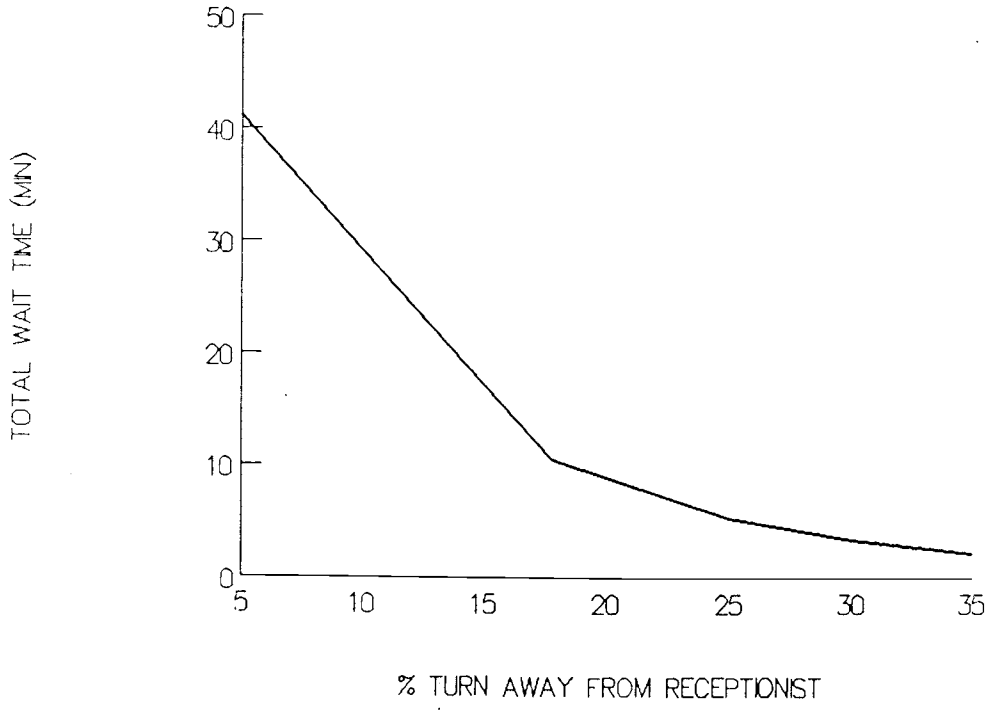
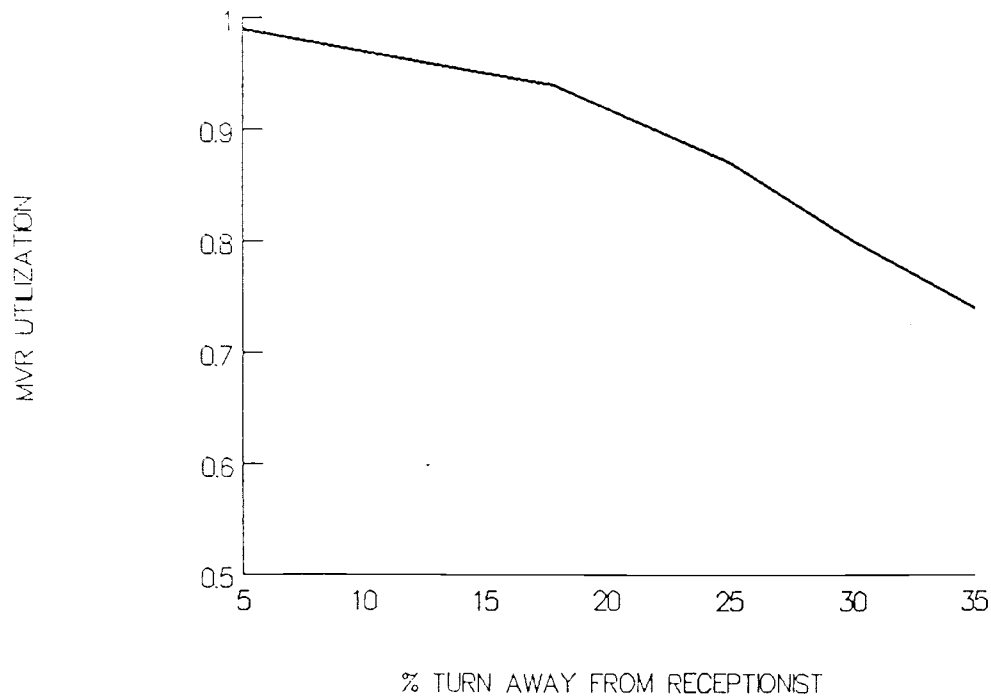


Figure 27. MVR Utilization vs. % Turn Away
(4 MVRs, 1 Recep, Arrival Rate = 40/hour)



Receptionist Service Time

Because receptionist service time is the single biggest factor, outside of arrival rate, that affects the length of the receptionist queue, sensitivity was conducted on this variable to determine its effects on the receptionist function. The receptionist service time had been estimated by DMV personnel at forty seconds. This was varied from thirty to ninety seconds in order to determine any relationships between service time and the queue.

Figure 28 shows the effect changing the receptionist service time has on receptionist queue length for a given customer arrival rate. As the time it takes to service a customer increases, the length of the queue also increases. However, the point at which the queue length begins to grow beyond the capacity of the system will depend on the arrival rate of customers. For this analysis, the arrival rate was forty per hour, or 1.5 minutes between arrivals. As the service time approaches 1.5 minutes the receptionist queue length grows exponentially. Given a goal of limiting the queue to three customers the average service time must be at the most one minute (thirty seconds faster than the time between arrivals).

A similar effect on the waiting time in the queue is shown in Figure 29.

Figure 28. Queue Length vs. Receptionist Service Time (4 MVRs, 1 Recep, Arrival Rate = 40/hour)

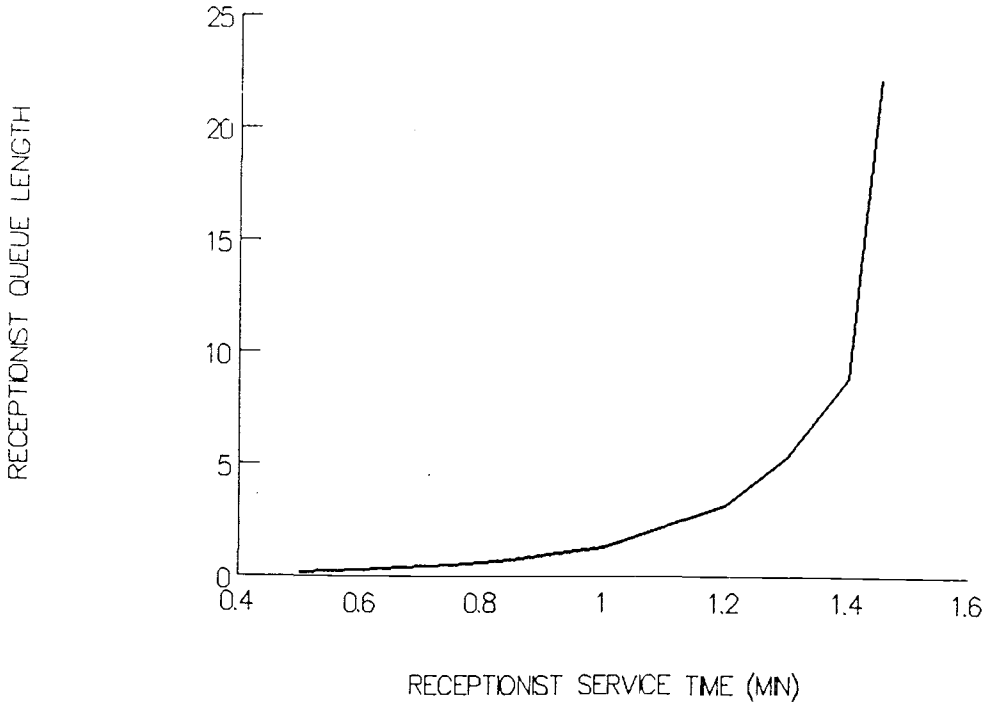
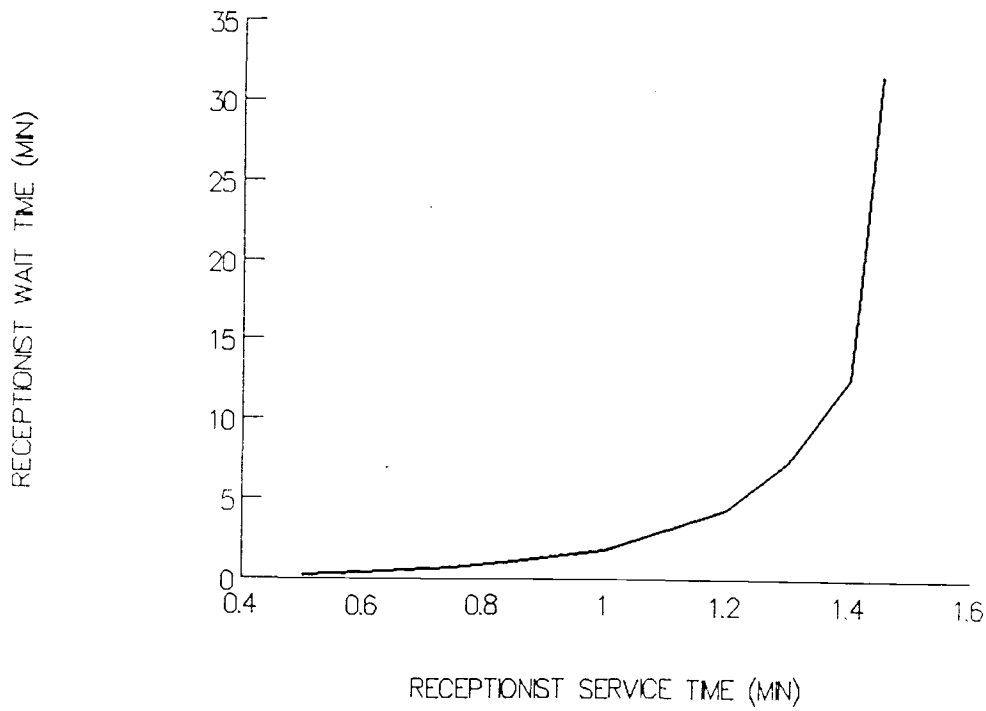
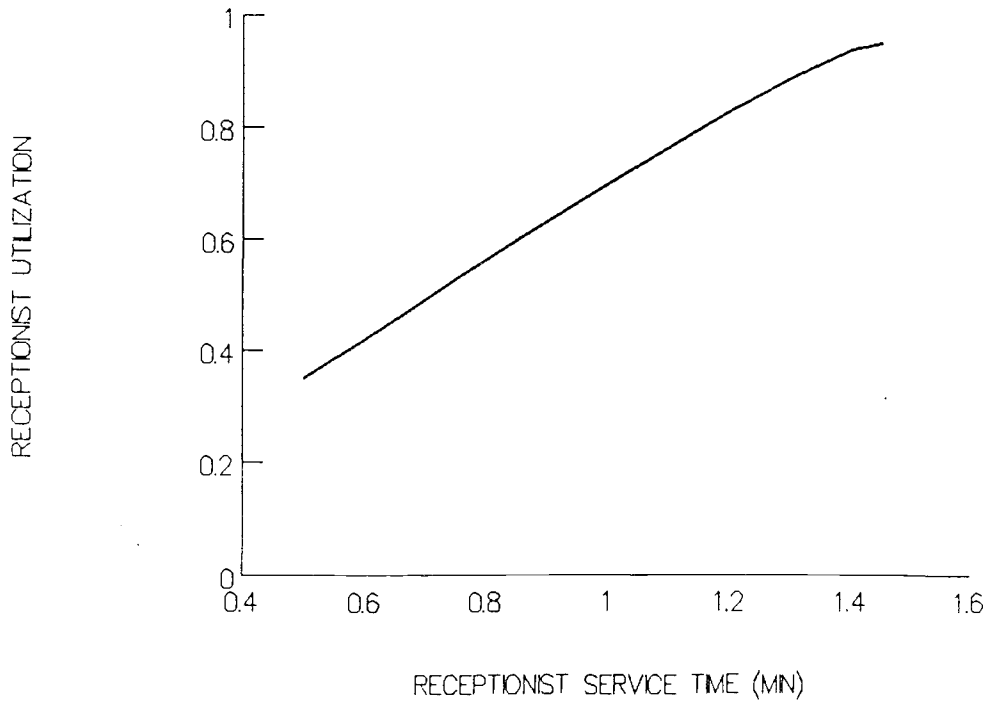


Figure 29. Wait Time vs. Receptionist Service Time (4 MVRs, 1 Recep, Arrival Rate = 40/hour)



The relationship between receptionist utilization and service time is shown in Figure 30. As the time it takes to service a customer increases, the busier the receptionist becomes.

Figure 30. Utilization vs. Receptionist Service Time (4 MVRs, 1 Recep, Arrival Rate = 40/hour)



DISCUSSION AND CONCLUSIONS

Explanation of Results

The results from the experimental application on DMV field offices were presented in the previous section and represent one application of the program developed. As can be seen from the graphs a positive correlation exists between arrival rates and waiting times, and between arrival rates and MVR utilization. This essentially means that as more entities arrive, the servers will be busier and customers will have to wait longer.

Perhaps a relationship that is not as easy to understand is the one between waiting times and server utilization. At first glance it appears that if server utilization goes up they are working harder and the entities should be served faster. However, in order to improve utilization there needs to be more buffer in the queue for the slack times. This in turn means larger average queues and higher average waiting times. Waiting times and worker utilization therefore have a positive correlation; consequently, there is a trade-off between having high utilization and having small queues. This is often an important question in a business that is trying to increase its production by keeping machines constantly busy. They must balance a high production rate (machine utilization) with the cost of work-in-process inventory.

In the DMV field office situation the question is not one of increasing production rate as customers arrive only when it is necessary to do business. Therefore, increasing worker utilization may not be the most desirable objective. In fact the reverse is true. Constant arrival rates (production rate) with increasing utilization means that the workers are doing the same work in a greater amount of time. Under these conditions utilization and efficiency have inverse relationships. When comparing utilization between two different allocations of resources (for example, having a receptionist or not) the one with the lower utilization is more efficient at using their resources. In addition, the positive correlation between waiting times and utilization means that an added benefit would be lower waiting times and increased customer satisfaction.

When discussing the efficiency of the resource allocations from Figures 19 to 23, presented in the previous section, it must be remembered that the utilization graphed does not include the receptionist. In nearly all cases the option of having a receptionist who also acts as an MVR when idle has the lowest utilization. This is most likely due to the receptionist who removes customers from the MVR queue for service and not that they are necessarily any more efficient. Consequently, when comparing results the receptionist utilization must be factored in.

Criticism of Results and Methods

Most of the credibility questions in the model representation of DMV field office activities lies in the following simplifying assumptions:

1. one transaction per customer
2. customers leave after their transaction and do not enter another operation
3. no specific MVR is assigned to the drive line
4. drive line customer appointments are dealt with by putting them at the head of the MVR queue.

The concept of a customer and a transaction were combined to simplify many parts of the model. First, there is no data available on the number of transactions per customer. Second, there may be a correlation between the types of multiple transactions the customer would be doing at that time, also unknown. Therefore, deciding which customers would have multiple transactions, what those transactions might be, and routing the customer between field office operations was eliminated. Third, trying to include these variables would have added uncertainty rather than clarifying the model.

The assumptions about the drive line have very little affect on the system. In reality, MVRs are assigned to the drive line based on scheduled appointments. If there is time between appointments the MVR will help out at the counter, serving customers. This is essentially the same as the MVR being stationed at the counter, and doing the

driving tests as soon as possible. Assuming there was not an over booking of appointments, enough MVRs would be available. Therefore, it is irrelevant as to exactly which MVR was on the drive line and output is not affected. Although drive line customers do not enter the MVR queue, the model routes them that way. By putting them at the head of the queue it allows them to take the first available MVR essentially at their appointed time. Their waiting time would be the same as if they were waiting elsewhere for the MVR to return from another test.

Other assumptions in the model that affect its credibility include the distributions used for generating arrivals and service times. Because of a lack of fit with conventional distributions for transaction times and a lack of available data for arrival rates, they were assumed. As mentioned in the Uncertainty Modeling section the decisions were based on what information was known and based on precedent. It is suggested that further research be conducted on these variables as mentioned in the next section.

Recommendations for Future Research

There are five areas in which additional research could be conducted: redefining some of the model branches to more accurately describe a particular system, other model applications, interfacing the model with an expert

system (ES), extending the user interface, and more accurately defining system variables.

The switching rules (jockeying queues to a free server) is one area in which the branching may be altered. Currently, this does not affect the DMV use of the model; however, if express windows are used that have separate queues the rules will become more important. Presently, the rules allow switching of any customer to any other queue if a server is free. This could be changed to basing the switches on transaction type or server type. Another decision branch that could be changed would be for customers leaving the MVR queue to be served by the receptionist. Currently any transaction other than a drive test may be served by the receptionist. This could be based on transaction time, transaction type or some other criteria.

There are several other areas where this program could be used including: airline ticket counters, other government agency customer offices (for example, immigration and social security), and bank teller queues.

The model could be interfaced with an ES in several ways. The ES could be the decision maker and query the simulation for measurements of performance. Conversely, the simulation could query the ES for altering resources based on the state of the system. An ES could also be developed around the simulation as a training tool for new and old managers to learn dynamics of the system.

While the model handles up to four types of servers (user defines the number of servers of each type) in four queues, the front end and back end was designed to handle only one type based on current DMV practices. However, this could be expanded to allow input data on arrivals to each of four queues and the number of servers for each queue. The back end would also have to be expanded to allow for the output statistics on each queue. Because of the modular design of program subroutines this could be easily accomplished with little network modifications.

Finally, improving the credibility of the model as used by DMV Systems and Planning would be achieved by developing more accurate distributions for arrival rates and service times. Time studies would have to be conducted on arrivals and user defined distributions developed for transactions times. The DMV would then be able to rerun some of the analyses for more current results.

BIBLIOGRAPHY

BENGU, GOLGEN and JORGE HADDOCK, "A Generative Simulation-Optimization System", Computers & Industrial Engineering, Volume 10, Number 4, pp. 301-313, 1986.

HADDOCK, JORGE, "An Expert System Framework Based on a Simulation Generator", Simulation, Volume 48, Number 2, pp. 45-53, February, 1987.

HADDOCK, JORGE and ROBERT P. DAVIS, "Building a Simulation Generator for Manufacturing Cell Design and Control", Institute of Industrial Engineers, 1985 Annual International Industrial Engineering Conference Proceedings, pp. 237-244.

LAW, AVERILL M. and W. DAVID KELTON, Simulation Modeling and Analysis, McGraw-Hill, New York, 1982.

LILEGDON, WILLIAM R. and JEAN J. O'REILLY, SLAM II PC Version User's Manual, Pritsker & Associates, Inc., March, 1986.

MATHEWSON, S.C., "Program Generators", Interactive Systems, European Computing Conference on Interactive Systems, London, pp. 423-439, September, 1975.

MATHEWSON, S.C., "The Application of Program Generator Software and Its Extensions to Discrete Event Simulation Modeling", IIE Transactions, Volume 16, Number 1, pp. 3-18, March, 1984.

MEDEIROS, DEBORAH J. and RANDALL P. SADOWSKI, "Simulation of Robotic Manufacturing Cells: A Modular Approach", Simulation, Volume 40, Number 1, pp. 3-12, January, 1983.

MILLS, NANCY, LORELLE JABS and JEANNE BOUTEN, Field Office Facilities Final Report, Motor Vehicles Division, Systems and Planning Section, Salem, Oregon, September, 1987.

O'KEEFE, ROBERT, "Simulation and Expert Systems - A Taxonomy and Some Examples", Simulation, Volume 46, Number 1, pp. 10-16, January, 1986.

PRITSKER, A.A.B. and C.D. PEGDEN, Introduction to Simulation and SLAM, Halstead Press, John Wiley and Sons, New York, 1979.

RASMUSSEN, E. HART, "Queue Simulation", Byte, pp. 157-174, March, 1984.

SAUER, C.H. and E.A. MACNAIR, "Queueing Network Software for Systems Modelling", Software - Practice and Experience, Volume 9, pp. 369-380, 1979.

SCHROEDER, ROGER G., Operations Management: Decision Making in the Operations Function, McGraw-Hill, Tokyo, Japan, 1982.

STANDRIDGE, CHARLES R., "Performing Simulation Projects with The Extended Simulation System (TESS)", Simulation, Volume 45, Number 6, pp. 283-291, December, 1985.

SUBRAHMANIAN, ESWARAN and ROBERT L. CANNON, "A Generator Program for Models of Discrete-Event Systems", Simulation, pp. 93-101, March, 1981.

APPENDICES

APPENDIX 1. PROGRAM EXECUTION EXAMPLE

The example presented here assumes that the computer has a hard disk (drive C:) containing the files DMVSIM.EXE and DMVSIM.TRA, and the system original is in the A floppy disk drive.

The first step to get the program running is to enter DMVSIM at the prompt. This calls the DMV simulation executable file, containing the FORTRAN subroutines linked with the SLAM executive.

```
C:\SLAM>DMVSIM
```

Next, the computer instructs the user to input the translated model file. The translated model file contains the SLAM network model necessary for the executable file to run the simulation.

```
Enter file name of translated model:DMVSIM.TRA
```

The following instructions are displayed on the screen and the menu is presented.

```
                    **** DMV SIMULATION PROGRAM ****

                          Written by A. Mechling
                          Oregon State University

          This program simulates customers arriving to a
     DMV field office.  The office may have a variable
     number of MVRs, an arrival rate that changes each
     hour, and variable number of receptionists who can
     do only receptionist duties or can serve customers
     out of the MVR queue when they are idle.
          The program is menu driven and no knowledge of
     SLAM or simulation methods is required for the user.
     At times when input is needed from the user either
     a selection of choices will be available or the user
     must input information.  When the user must input
     their own information an example of the input is
     given in parentheses (), or a default value is given
     in brackets [].  The default value can be taken by
     hitting <ENTER> without any input.
          Output may be obtained at any time by hitting
     <PRINT SCREEN>, after the printer is ready.
Pause.
Please press <return> to continue.
```

```
                    **** MENU ****

                    1 - INPUT NEW DATA FILE
                    2 - EDIT/REVIEW OLD DATA FILE
                    3 - RUN THE SIMULATION
                    4 - EXIT

Input your choice -->
```

When a "1" is entered in response to the prompt,

```
Input your choice -->  1
```

the following screens of instructions are displayed.

```
                    **** INPUT NEW DATA FILE ****

          This subroutine will allow you to input new
     data for the model to be simulated on. Data to be
     input for each hour are:

     MVRs SERVING CUSTOMERS - includes all MVRs serving
          customers at the counter or on the drive
          line during that hour - integer input
     RECEPTIONISTS - number of receptionists on duty
          greeting customers during that hour -
          integer input
     CUSTOMER ARRIVAL INTERVAL - the time interval at
          which customers arrive to the DMV in terms
          of average minutes between arrivals - real
          input, must use a decimal
     # DRIVING TESTS SCHEDULED - the total number of
          driving tests that are scheduled during
          that hour - integer input


Pause.
Please press <return> to continue.
```

```
          Starting with the first hour of the day, input
     the data for the row, hit <ENTER> and continue to
     the next hour. When the office closes simply hit
     <ENTER> for the following rows, without any input.
          The four data may be separated by blank spaces
     to center data under headings, or by commas. Some
     examples of input are:

     Valid Input          vs.        Invalid Input
     -----------                     -------------
     5,2,3.5,0                       5 2,3.5,0
     3     1     2.5     4           3.5     1     2.5     4
     6,3,4.0,1                       6,3,4,1
```

The first set of data to input is into the DMV STAFFING AND ARRIVALS table. The empty table is displayed a row at a time and the user may begin to enter the data.

DMV STAFFING AND ARRIVALS

HOUR	MVRs SERVING CUSTOMERS	RECEP-TIONISTS	CUSTOMER ARRIVAL INTERVAL	# DRIVING TESTS SCHEDULED
1				

Each row represents one hour in the day and up to an eleven hour day may be simulated. If there are less than eleven hours then the remaining rows of input may be skipped by hitting <ENTER>.

DMV STAFFING AND ARRIVALS

HOUR	MVRs SERVING CUSTOMERS	RECEP-TIONISTS	CUSTOMER ARRIVAL INTERVAL	# DRIVING TESTS SCHEDULED
1	4	0	2.2	0
2	4	0	2.1	1
3	4	0	1.9	2
4	4	0	2.0	3
5	4	1	1.7	2
6	4	1	1.6	2
7	4	1	2.1	1
8	4	1	2.8	0
9				
10				
11				

In this case there were only eight hours in the day to be simulated; therefore, at rows nine, ten and eleven the <ENTER> key was hit.

Since there was a receptionist scheduled the computer then prompts the user to answer questions about the receptionist function.

```
What percent of the customers are met by the
receptionist (<=100)  --> 99

Can the receptionist serve customers from the
MVR queue when idle (Y/N) [N] -->

What is the average service time for the
receptionist in minutes (e.g. 1.20) -->  .67

By what percent is service time reduced
after receptionist contact (<=100) -->  5
```

At any time during input when there is a default value displayed in brackets [], that value may be taken by either typing it in or by hitting <ENTER>. This was done for the second question.

The user is then asked to enter the name under which the data will be stored for later simulation.

```
Enter the name of the file in which the raw data
will be saved  (e.g. MEDFORD.DAT) -->  SALEM.DAT
```

The data is automatically stored and the menu returns.

```
                    **** MENU ****

                1 - INPUT NEW DATA FILE
                2 - EDIT/REVIEW OLD DATA FILE
                3 - RUN THE SIMULATION
                4 - EXIT

Input your choice --> 2
```

When a "2" is entered in response to the prompt a brief introduction is made and the user is asked to enter a file name.

```
                ***** EDIT/REVIEW OLD DATA FILE *****

          This subroutine will allow review of a pre-
viously stored data set and to change selected
values.
          The edited data may be saved under the same
name (replace the old file) or under a new name
(the old file remains and a new file is created).

Enter the name of the file from which you want to
view the data [DEFAULT.DAT] --> SALEM.DAT
```

In this case the user decided to check the file just created for accuracy. The current contents of the file are displayed and it is discovered that there was an error in row six.

DMV STAFFING AND ARRIVALS

HOUR	MVRs SERVING CUSTOMERS	RECEP-TIONISTS	CUSTOMER ARRIVAL INTERVAL	# DRIVING TESTS SCHEDULED
1	4	0	2.20	0
2	4	0	2.10	1
3	4	0	1.90	2
4	4	0	2.00	3
5	4	1	1.70	2
6	4	1	1.60	2
7	4	1	2.10	1
8	4	1	2.80	0
9	0	0	.00	0
10	0	0	.00	0
11	0	0	.00	0

```
THE RECEPTIONIST CONTACTS  99.0 PERCENT OF THE CUSTOMERS
THE RECEPTIONIST ONLY DOES RECEPTIONIST DUTIES
THE RECEPTIONIST SERVICE TIME IS   .67 MINUTES
SERVICE TIME IS REDUCED BY    5.0 PERCENT AFTER RECEPTIONIST CONTACT
```

The user is then asked if there are any changes to be made in the table. The response here is yes (Y) they want to make a change in row six.

```
Do you want to make changes in the DMV STAFFING AND
ARRIVALS table (Y/N) --> Y
Which row do you want to change (1,2...) --> 6
```

The old row is displayed and the new row input.

DMV STAFFING AND ARRIVALS

	HOUR	MVRs SERVING CUSTOMERS	RECEP-TIONISTS	CUSTOMER ARRIVAL INTERVAL	# DRIVING TESTS SCHEDULED
OLD DATA	6	4	1	1.60	2
NEW DATA	6				

DMV STAFFING AND ARRIVALS

	HOUR	MVRs SERVING CUSTOMERS	RECEP-TIONISTS	CUSTOMER ARRIVAL INTERVAL	# DRIVING TESTS SCHEDULED
OLD DATA	6	4	1	1.60	2
NEW DATA	6	4	1	1.75	1

When the <ENTER> key is hit at the end of the new row then the updated version of the table is displayed.

DMV STAFFING AND ARRIVALS

HOUR	MVRs SERVING CUSTOMERS	RECEP-TIONISTS	CUSTOMER ARRIVAL INTERVAL	# DRIVING TESTS SCHEDULED
1	4	0	2.20	0
2	4	0	2.10	1
3	4	0	1.90	2
4	4	0	2.00	3
5	4	1	1.70	2
6	4	1	1.75	1
7	4	1	2.10	1
8	4	1	2.80	0
9	0	0	.00	0
10	0	0	.00	0
11	0	0	.00	0

THE RECEPTIONIST CONTACTS 99.0 PERCENT OF THE CUSTOMERS
THE RECEPTIONIST ONLY DOES RECEPTIONIST DUTIES
THE RECEPTIONIST SERVICE TIME IS .67 MINUTES
SERVICE TIME IS REDUCED BY 5.0 PERCENT AFTER RECEPTIONIST CONTACT

No more changes are desired and the receptionist information is re-entered.

```
Do you want to make changes in the DMV STAFFING AND
ARRIVALS table (Y/N) --> N

What percent of the customers are met by the
receptionist (   99.0) -->99

Can the receptionist serve customers from the
MVR queue when idle [N] -->

What is the average service time for the
receptionist in minutes ( .67) -->.67

By what percent is service time reduced after
receptionist contact (  5.0) --> 5

Enter the name of the file in which the raw data
will be saved  [SALEM.DAT    ] -->
```

The default values were accepted for the second question about the receptionist as well as the file name. Because the modified data is stored under the same name, the original data is overwritten. If it is desired to save both the original data and the new data then a new name must be entered at the prompt.

Again the menu returns.

```
                    **** MENU ****

                    1 - INPUT NEW DATA FILE
                    2 - EDIT/REVIEW OLD DATA FILE
                    3 - RUN THE SIMULATION
                    4 - EXIT

Input your choice --> 3
```

This time it was desired to execute the simulation for the data file just modified and a "3" was input at the prompt.

Again, a brief introduction is made and the user is asked to enter a file name. Since the simulation may take up to ten or fifteen minutes on some computers, the user is asked to verify the data file before commencing.

```
                    **** RUN A SIMULATION ****


          This subroutine will execute the simulation
     of a previously stored data set.  The statistics
     are then displayed on the screen as output.
          <PRINT SCREEN>  will  cause  the printer to
     make a hard copy  of the data  that is displayed
     on the screen.



Enter the name of the data file from which the model
will be run [DEFAULT.DAT] -->  SALEM.DAT

Is SALEM.DAT     the file you want to run (Y/N) [Y] --> Y
```

As mentioned before, five simulation runs are made before the results are displayed. The following messages are displayed on the screen to inform the user of simulation progress.

```
THE SIMULATION IS RUNNING ON SALEM.DAT
            RUN #     1
```

```
THE SIMULATION IS RUNNING ON SALEM.DAT
            RUN #     2
```

```
                    :
                    :
                    :
```

```
THE SIMULATION IS RUNNING ON SALEM.DAT
            RUN #     5
```

When the simulation is completed the results are displayed on the screen, first for the receptionist and then for the MVRs. It is a good idea to get a hard copy of this by hitting the <PRINT SCREEN> key for both tables of results.

SALEM.DAT STATISTICS ON THE RECEPTIONIST QUEUE

HOUR	AVERAGE WAITING TIME	STD DEV WAITING TIME	AVERAGE QUEUE LENGTH	STD DEV QUEUE LENGTH	RECEP. UTILI-ZATION
1	.00	.00	.00	.00	.00
2	.00	.00	.00	.00	.00
3	.00	.00	.00	.00	.00
4	.00	.00	.00	.00	.00
5	1.14	1.09	.74	.94	.64
6	.87	.93	.55	.83	.63
7	.63	.78	.31	.56	.51
8	.73	.86	.30	.62	.44
9	.00	.00	.00	.00	.00
10	.00	.00	.00	.00	.00
11	.00	.00	.00	.00	.00
DAILY AVERAGE	.84		.48		.55

NUMBER OF CUST RECEP. SERVES AS AN MVR 0.
NUMBER OF CUST RECEP. CONTACTS 123.

Pause.

SALEM.DAT STATISTICS ON THE MVR QUEUE

HOUR	AVERAGE WAITING TIME	STD DEV WAITING TIME	AVERAGE QUEUE LENGTH	STD DEV QUEUE LENGTH	MVR UTILI-ZATION
1	1.32	1.78	.65	1.00	.76
2	2.34	1.99	1.22	1.07	.77
3	6.34	3.23	3.96	2.08	.94
4	11.04	5.08	6.69	1.59	.90
5	12.66	4.89	5.82	1.29	.89
6	5.93	4.90	2.44	2.04	.86
7	2.28	2.14	.89	1.00	.75
8	.71	1.05	.27	.52	.58
9	.00	.00	.00	.00	.00
10	.00	.00	.00	.00	.00
11	.00	.00	.00	.00	.00
DAILY AVERAGE	5.33		2.74		.81

Pause.

After the second <RETURN> (<ENTER>) prompt the menu returns.

```
                    **** MENU ****

                    1 - INPUT NEW DATA FILE
                    2 - EDIT/REVIEW OLD DATA FILE
                    3 - RUN THE SIMULATION
                    4 - EXIT

Input your choice --> 4
```

This time it was desired to exit the program and a "4" was input at the prompt. Program execution is stopped with the following message displayed on the screen.

```
Stop - Program terminated.

C:\SLAM>
```

APPENDIX 2. SLAM NETWORK MODEL

```
GEN,MECHLING,PROTOTYPE ONE,5/26/88,50,N,N,Y,N,N;
LIMITS,15,11,100;
INIT,0,660;
PRIORITY/1,HVF(10)/2,HVF(10)/3,HVF(10)/4,HVF(10);
TIMST,XX(11),MVR1 QL;
TIMST,XX(12),MVR2 QL;
TIMST,XX(13),MVR3 QL;
TIMST,XX(14),MVR4 QL;
TIMST,XX(15),RECEP QL;
NETWORK;
;
;-------------------------------------------------
;*************************************************;
;******* NETWORK MODEL OF DMV FIELD OFFICES *******;
;*************************************************;
;-------------------------------------------------
;
;******* DECLARE GATES, RESOURCES AND THEIR QUEUES *******
;
      GATE/G1,OPEN,1;                          Gate 1, ifile 1, MVR 1 queue
      GATE/G2,OPEN,2;                          Gate 2, ifile 2, MVR 2 queue
      GATE/G3,OPEN,3;                          Gate 3, ifile 3, MVR 3 queue
      GATE/G4,OPEN,4;                          Gate 4, ifile 4, MVR 4 queue
      GATE/G5,OPEN,5;                          Gate 5, ifile 5, Recep queue
      RESOURCE/MVR1(0),11;                     MVR 1, ifile 11
      RESOURCE/MVR2(0),12;                     MVR 2, ifile 12
      RESOURCE/MVR3(0),13;                     MVR 3, ifile 13
      RESOURCE/MVR4(0),14;                     MVR 4, ifile 14
      RESOURCE/RECEP(0),15;                    Recep, ifile 15
;
;********** CALL THE MENU SUBROUTINE **********
;******* AND INITIALIZE STARTING VALUES *******
;
      CREATE,,,,1;                             Create dummy to
      EVENT(1);                                call the menu.
      ALTER,MVR1/XX(50);                       Declare # of MVRs.
      ALTER,MVR2/0;
      ALTER,MVR3/0;
      ALTER,MVR4/0;
      ALTER,RECEP/XX(51);                      Declare # of receps.
      ASSIGN,XX(3)=XX(52);                     Assign TBC for reg cust.
      ASSIGN,XX(4)=XX(20);                     Store # of receps.
      ASSIGN,XX(5)=XX(53);                     Assign TBC for drv tests.
      TERM;                                    Destroy dummy.
;
;******* CUSTOMERS ARRIVE AT THE DMV FIELD OFFICE *******
;
      CREATE,XX(5),,1;                         Create drive line custs.
      ASSIGN,ATRIB(3)=TRIAG(15,23,31),ATRIB(4)=1;Store trans.time,MVR type,
      ASSIGN,ATRIB(6)=0,ATRIB(9)=36;           self serve poss,trans.type,
      ASSIGN,ATRIB(10)=1;                      Q priority in atts 3,4,6,9,10.
      ACT,,,SYST;                              Enter DMV system.
      CREATE,EXPON(XX(3),3),,1;                Create regular customers.
      ASSIGN,ATRIB(10)=0,1;                    Store Q priority in att 10.
;                                              Determine transaction type
;                                              from preset probabilities:
        ACT,,.00004,JB1;                       additional cash receipts
        ACT,,.00146,JB2;                       cancelled plate/marker
        ACT,,.00046,JB3;                       def driving course payments
```

```
          ACT,,.00135,JB4;                                  duplicate instruction permit
          ACT,,.00697,JB5;                                  duplicate titles
          ACT,,.00329,JB6;                                  duplicate titles w/ renewals
          ACT,,.00138,JB7;                                  equipment failure
          ACT,,.00157,JB8;                                  highway permits
          ACT,,.02435,JB9;                                  law test - atd
          ACT,,.00747,JB10;                                 law test - written
          ACT,,.02457,JB11;                                 look-ups & driving records
          ACT,,.00292,JB12;                                 mileage reports
          ACT,,.00031,JB13;                                 mobile home trip permits
          ACT,,.00337,JB14;                                 motorcycle/moped drive test
          ACT,,.00282,JB15;                                 motorcycle endorsement
          ACT,,.00461,JB16;                                 original class 1, 2, 3
          ACT,,.01190,JB17;                                 original id card
          ACT,,.02251,JB18;                                 original instruction permit
          ACT,,.04286,JB19;                                 orig. operator/combined/moped
          ACT,,.08118,JB20;                                 paper renewals
          ACT,,.05433,JB21;                                 photo duplicate licenses
          ACT,,.10134,JB22;                                 photo renewal licenses
          ACT,,.00062,JB23;                                 re-exam drive test
          ACT,,.01386,JB24;                                 reinstatement fees
          ACT,,.01127,JB25;                                 replacement plates & stickers
          ACT,,.00321,JB26;                                 snow park permits
          ACT,,.00005,JB27;                                 student or emergency permits
          ACT,,.12582,JB28;                                 title only
          ACT,,.16353,JB29;                                 title w/ renewals
          ACT,,.07551,JB30;                                 trip permits
          ACT,,.00234,JB31;                                 unable to accomodate
          ACT,,.13074,JB32;                                 validated reminders
          ACT,,.02411,JB33;                                 vehicles address change
          ACT,,.00607,JB34;                                 vehicle id marker/temp pass
          ACT,,.04181,JB35;                                 vin inspection
JB1   ASSIGN,ATRIB(3)=TRIAG(1.5,3.0955,6.2);                Assign statements for
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=1;              storing transaction time,
      ACT,,,SYST;                                           MVR type, self service
JB2   ASSIGN,ATRIB(3)=TRIAG(2.1,4.2284,8.4);                possibility, and trans-
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=2;              action type in attributes
      ACT,,,SYST;                                           3,4,6,9
JB3   ASSIGN,ATRIB(3)=TRIAG(2.5,4.9964,10);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=3;
      ACT,,,SYST;                                           then enter DMV system.
JB4   ASSIGN,ATRIB(3)=TRIAG(3.7,7.4410,14.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=4;
      ACT,,,SYST;
JB5   ASSIGN,ATRIB(3)=TRIAG(3.4,6.8922,13.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=5;
      ACT,,,SYST;
JB6   ASSIGN,ATRIB(3)=TRIAG(3.7,7.3862,14.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=6;
      ACT,,,SYST;
JB7   ASSIGN,ATRIB(3)=TRIAG(.9,1.7720,3.5);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=7;
      ACT,,,SYST;
JB8   ASSIGN,ATRIB(3)=TRIAG(2.4,4.7040,9.4);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=8;
      ACT,,,SYST;
JB9   ASSIGN,ATRIB(3)=TRIAG(.6,1.2761,2.5);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=9;
      ACT,,,SYST;
```

```
JB10  ASSIGN,ATRIB(3)=TRIAG(2.3,4.5004,9.0);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=10;
      ACT,,,SYST;
JB11  ASSIGN,ATRIB(3)=TRIAG(2.3,4.4972,9.0);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=11;
      ACT,,,SYST;
JB12  ASSIGN,ATRIB(3)=TRIAG(1.3,2.5372,5.1);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=12;
      ACT,,,SYST;
JB13  ASSIGN,ATRIB(3)=TRIAG(2.5,5.1237,10.2);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=13;
      ACT,,,SYST;
JB14  ASSIGN,ATRIB(3)=TRIAG(2.5,5.0655,10.1);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=14;
      ACT,,,SYST;
JB15  ASSIGN,ATRIB(3)=TRIAG(4.4,8.8689,17.7);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=15;
      ACT,,,SYST;
JB16  ASSIGN,ATRIB(3)=TRIAG(4.5,8.9649,17.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=16;
      ACT,,,SYST;
JB17  ASSIGN,ATRIB(3)=TRIAG(5.3,10.5049,21.0);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=17;
      ACT,,,SYST;
JB18  ASSIGN,ATRIB(3)=TRIAG(4.7,9.4764,18.9);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=18;
      ACT,,,SYST;
JB19  ASSIGN,ATRIB(3)=TRIAG(5,9.9196,20);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=19;
      ACT,,,SYST;
JB20  ASSIGN,ATRIB(3)=TRIAG(2.4,4.8926,9.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=20;
      ACT,,,SYST;
JB21  ASSIGN,ATRIB(3)=TRIAG(3.7,7.4410,14.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=21;
      ACT,,,SYST;
JB22  ASSIGN,ATRIB(3)=TRIAG(3.5,6.9306,13.8);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=22;
      ACT,,,SYST;
JB23  ASSIGN,ATRIB(3)=TRIAG(6,12.0401,24);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=23;
      ACT,,,SYST;
JB24  ASSIGN,ATRIB(3)=TRIAG(4.1,8.1975,16.4);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=24;
      ACT,,,SYST;
JB25  ASSIGN,ATRIB(3)=TRIAG(3.1,6.1342,12.3);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=25;
      ACT,,,SYST;
JB26  ASSIGN,ATRIB(3)=TRIAG(1.3,2.5072,5);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=26;
      ACT,,,SYST;
JB27  ASSIGN,ATRIB(3)=TRIAG(8,15.9318,32);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=27;
      ACT,,,SYST;
JB28  ASSIGN,ATRIB(3)=TRIAG(2.6,5.1539,10.3);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=28;
      ACT,,,SYST;
JB29  ASSIGN,ATRIB(3)=TRIAG(3.4,6.8655,13.7);
      ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=29;
```

```
       ACT,,,SYST;
JB30   ASSIGN,ATRIB(3)=TRIAG(2.3,4.5242,9);
       ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=30;
       ACT,,,SYST;
JB31   ASSIGN,ATRIB(3)=TRIAG(.5,1.0983,2.2);
       ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=31;
       ACT,,,SYST;
JB32   ASSIGN,ATRIB(3)=TRIAG(1.1,2.2750,4.5);
       ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=32;
       ACT,,,SYST;
JB33   ASSIGN,ATRIB(3)=TRIAG(.22,0.4399,.88);
       ASSIGN,ATRIB(4)=1,ATRIB(6)=1,ATRIB(9)=33;
       ACT,,,SYST;
JB34   ASSIGN,ATRIB(3)=TRIAG(2.7,5.3894,10.8);
       ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=34;
       ACT,,,SYST;
JB35   ASSIGN,ATRIB(3)=TRIAG(1.5,2.9413,5.8);
       ASSIGN,ATRIB(4)=1,ATRIB(6)=0,ATRIB(9)=35;
       ACT,,,SYST;
;
;******* DECISION RULES FOR ROUTING CUSTOMERS *******
;
SYST   ASSIGN,ATRIB(2)=1.0;                          Store transactn time coeff.
       ASSIGN,ATRIB(5)=TRIAG(0.1,XX(95),2.5),1;      Store rcp srvc time.
         ACT,,XX(4).GT.0,RCP;                        Send to recep if on duty or
         ACT,,ATRIB(6).NE.1,SRVC;                    send to MVR if self serve not
         ACT;                                        possible or go on.
       GOON,1;
         ACT,,.600,SSRV;                             60% use self service.
         ACT,,.400,SRVC;                             40% don't use self service.
;
;******* RECEPTIONIST RULES AND SERVICE *******
;
RCP    GOON,1;
         ACT,,XX(94),RCPS;                           XX(94) % use receptionist.
         ACT,,1-XX(94);                              The rest go on
           GOON,1;
             ACT,,ATRIB(6).EQ.1,SSRV;                to self serve if possible
             ACT,,,SRVC;                             or to MVR service.
RCPS   ASSIGN,XX(15)=XX(15)+1,ATRIB(11)=TNOW;        Adjust Q count & store tnow.
RCPQ   AWAIT(5),G5,,1;                               Wait in receptionist queue.
         ACT,,XX(4).EQ.0,BALK;                       Go to MVR if recep off duty
         ACT,,NNRSC(5).GT.0,AWAIT;                   or go to recep if not busy
         ACT;
           CLOSE,G5;                                 or close recep Q gate and
           ACT,,,RCPQ;                               return to recep queue.
BALK   ASSIGN,XX(15)=XX(15)-1;                       Adjust queue count
       ACT,,,SRVC;                                   and send to MVR service.
AWAIT  GOON,1;
         ACT,,NNRSC(5).GT.0,RSRV;                    Go to recep service if not
         ACT;                                        busy or
           CLOSE,G5;                                 close recep queue gate and
           ACT,,,RCPQ;                               return to recep queue.
RSRV   ASSIGN,XX(15)=XX(15)-1;                       Adjust queue count.
       COLCT(11),INT(11),RECEP WAIT TIME;            Collect waiting time.
       AWAIT(15),RECEP;                              Engage receptionist.
       ACT/5,ATRIB(5);                               Receptionist service.
       FREE,RECEP;                                   Free receptionist.
       OPEN,G5,1;                                    Open recep queue gate.
```

```
        ACT,,XX(1).NE.1.OR.XX(15).GT.0,GOON;      Cust leaves recep or
        ACT,,XX(11).GT.0.AND.NNRSC(5).GT.0,OP1;   if recep can serve MVR custs,
        ACT,,XX(12).GT.0.AND.NNRSC(5).GT.0,OP2;   recep is free & MVR Q full
        ACT,,XX(13).GT.0.AND.NNRSC(5).GT.0,OP3;   then open a busy queue.
        ACT,,XX(14).GT.0.AND.NNRSC(5).GT.0,OP4;
        ACT,,,GOON;                               Otherwise, go on.
OP1     OPEN,G1;                                  Open the busy queue and
        ACT,,,GOON;                               go on.
OP2     OPEN,G2;
        ACT,,,GOON;
OP3     OPEN,G3;
        ACT,,,GOON;
OP4     OPEN,G4;
GOON    ASSIGN,XX(6)=XX(6)+1;                     Store # recep contacts.
        COLCT(6),XX(6),# RECEPTIONIST CONTACTS,,1; Collect # recep contacts.
          ACT,,0.178,DPRT;                        17.8% depart DMV after recep.
          ACT,,0.822;                             82.2% go on to MVR.
        GOON,1;
          ACT,,ATRIB(5).EQ.ATRIB(3),DPRT;         If recep serves as MVR, depart
          ACT,,ATRIB(6).EQ.1,SSRV;                or go to self service if poss
          ACT,,ATRIB(9).EQ.36,SRVC;               or go to MVR service if d.tst
          ACT;                                    or other transactions
            ASSIGN,ATRIB(2)=XX(96);               change transaction time coeff
            ACT,,,SRVC;                           and go to MVR service.
;
;******* SELF SERVICE STATION *******
;
SSRV    GOON;                                     Self service station.
        ACT/6,ATRIB(3),,DPRT;                     Self service activity.
;
;******* MVR RULES AND SERVICE *******
;
SRVC    ASSIGN,ATRIB(8)=TNOW;                     MVR station, store tnow.
SWTG    GOON,1;                                   Queue switching rules
          ACT,,ATRIB(4).EQ.1,AT41;                based on incoming attribute
          ACT,,ATRIB(4).EQ.2,AT42;                4 values - server type.
          ACT,,ATRIB(4).EQ.3,AT43;
          ACT,,,AT44;
AT41    GOON,1;
          ACT,,NNRSC(1).GT.0,QNTRY;               If MVR type 1 is free enter Q
          ACT,,NNRSC(4).GT.0,SWT4;                or switch to MVR 4 if free
          ACT,,NNRSC(3).GT.0,SWT3;                or to MVR type 3 if free
          ACT,,NNRSC(2).GT.0,SWT2;                or to MVR type 2 if free
          ACT,,,QNTRY;                            or stay in Q 1 if all busy.
AT42    GOON,1;
          ACT,,NNRSC(2).GT.0,QNTRY;               If MVR type 2 is free enter Q
          ACT,,NNRSC(3).GT.0,SWT3;                or switch to MVR 3 if free
          ACT,,NNRSC(4).GT.0,SWT4;                or to MVR type 4 if free
          ACT,,NNRSC(1).GT.0,SWT1;                or to MVR type 1 if free
          ACT,,,QNTRY;                            or stay in Q 2 if all busy.
AT43    GOON,1;
          ACT,,NNRSC(3).GT.0,QNTRY;               If MVR type 3 is free enter Q
          ACT,,NNRSC(1).GT.0,SWT1;                or switch to MVR 1 if free
          ACT,,NNRSC(2).GT.0,SWT2;                or to MVR type 2 if free
          ACT,,NNRSC(4).GT.0,SWT4;                or to MVR type 4 if free
          ACT,,,QNTRY;                            or stay in Q 3 if all busy.
AT44    GOON,1;
          ACT,,NNRSC(4).GT.0,QNTRY;               If MVR type 4 is free enter Q
          ACT,,NNRSC(2).GT.0,SWT2;                or switch to MVR 2 if free
```

```
        ACT,,NNRSC(1).GT.0,SWT1;                  or to MVR type 1 if free
        ACT,,NNRSC(3).GT.0,SWT3;                  or to MVR type 3 if free
        ACT,,,QNTRY;                              or stay in Q 4 if all busy.
SWT1    ASSIGN,ATRIB(4)=1;                        Switch queues by changing
        ACT,,,QNTRY;                              server type - attribute 4
SWT2    ASSIGN,ATRIB(4)=2;
        ACT,,,QNTRY;                              and enter queue system.
SWT3    ASSIGN,ATRIB(4)=3;
        ACT,,,QNTRY;
SWT4    ASSIGN,ATRIB(4)=4;
QNTRY   ASSIGN,ATRIB(7)=ATRIB(4)+10;              Enter queue system.
        ASSIGN,II=ATRIB(4)+10,XX(II)=XX(II)+1;    Adjust MVR queue count.
        ASSIGN,II=ATRIB(4),1;
          ACT,,NNRSC(II).LE.0.AND.
            NNRSC(5).GT.0.AND.XX(1).EQ.1,JKY;     Go to recep for srvc if allowd
          ACT,,NNRSC(II).LE.0,SRVQ;               or wait in Q if MVR busy
          ACT,,NNRSC(II).GT.0;                    or go to queue and
            OPEN,ATRIB(4);                        open queue gate.
SRVQ    AWAIT(ATRIB(4)=1,4),ATRIB(4);             MVR queue gate.
        ASSIGN,II=ATRIB(4),1;
          ACT,,NNRSC(II).GT.0,NJKY;               Go to MVR if available
          ACT,,NNRSC(5).GT.0.AND.XX(1).EQ.1,JKY;  or go to recep if allowed
          ACT;                                    or go on.
            CLOSE,ATRIB(4);                       Close queue gate,
            ASSIGN,II=ATRIB(4)+10,XX(II)=XX(II)-1; readjust Q count and
            ACT,,,SWTG;                           send to switchng rules.
NJKY    ASSIGN,II=ATRIB(4),1;
          ACT,,NNRSC(II).GT.0,AVLB;               Go to MVR if available
          ACT;
            CLOSE,ATRIB(4);                       or close queue gate and
            ACT,,,SRVQ;                           return to MVR queue.
AVLB    CLOSE,ATRIB(4);                           MVR is available, close gate.
        ASSIGN,II=ATRIB(4)+10,XX(II)=XX(II)-1;    Adjust queue count.
        COLCT(8),INT(8),SRVC WAIT TIME;           Collect MVR waiting time.
MVR     AWAIT(ATRIB(7)=11,14),ATRIB(4);           Engage MVR server.
        ACT/ATRIB(4)=1,4,ATRIB(2)*ATRIB(3);       MVR service.
        FREE,ATRIB(4),1;                          Free MVR server.
          ACT,,ATRIB(4).EQ.1,A41;                 Go to queue gate opening rules
          ACT,,ATRIB(4).EQ.2,A42;                 based on incoming attribute 4
          ACT,,ATRIB(4).EQ.3,A43;                 values - server type.
          ACT,,,A44;
A41     GOON,1;
          ACT,,XX(11).GT.0,OPN1;                  Open the server's Q if busy
          ACT,,XX(14).GT.0,OPN4;                  or queue 4 if busy
          ACT,,XX(13).GT.0,OPN3;                  or queue 3 if busy
          ACT,,XX(12).GT.0,OPN2;                  or queue 2 if busy
          ACT,,,DPRT;                             or go on.
A42     GOON,1;
          ACT,,XX(12).GT.0,OPN2;                  Open the server's Q if busy
          ACT,,XX(13).GT.0,OPN3;                  or queue 3 if busy
          ACT,,XX(14).GT.0,OPN4;                  or queue 4 if busy
          ACT,,XX(11).GT.0,OPN1;                  or queue 1 if busy
          ACT,,,DPRT;                             or go on.
A43     GOON,1;
          ACT,,XX(13).GT.0,OPN3;                  Open the server's Q if busy
          ACT,,XX(11).GT.0,OPN1;                  or queue 1 if busy
          ACT,,XX(12).GT.0,OPN2;                  or queue 2 if busy
          ACT,,XX(14).GT.0,OPN4;                  or queue 4 if busy
          ACT,,,DPRT;                             or go on.
```

```
A44    GOON,1;
         ACT,,XX(14).GT.0,OPN4;                    Open the server's Q if busy
         ACT,,XX(12).GT.0,OPN2;                    or queue 2 if busy
         ACT,,XX(11).GT.0,OPN1;                    or queue 1 if busy
         ACT,,XX(13).GT.0,OPN3;                    or queue 3 if busy
         ACT,,,DPRT;                               or go on.
OPN1   OPEN,G1;
       ACT,,,DPRT;
OPN2   OPEN,G2;
       ACT,,,DPRT;
OPN3   OPEN,G3;
       ACT,,,DPRT;
OPN4   OPEN,G4;
       ACT,,,DPRT;
JKY    GOON,1;
         ACT,,XX(15).LE.0.AND.ATRIB(9).NE.36,JKYR;Go to recep if available
         ACT;
           CLOSE,ATRIB(4);                         or close queue gate and
           ACT,,,SRVQ;                             return to MVR queue.
JKYR   CLOSE,ATRIB(4);                             Close queue gate.
       ASSIGN,ATRIB(5)=ATRIB(3);                   Store new recep service time.
       ASSIGN,II=ATRIB(4)+10,XX(II)=XX(II)-1;      Adjust queue count.
       ASSIGN,XX(2)=XX(2)+1;                       Store # customers jockeying.
       COLCT(2),XX(2),# CUSTS RECEP SERVES;        Collect # custs recep serves.
       ACT,,,RCPS;                                 Go to receptionist.
;
;******* CUSTOMERS LEAVE THE DMV FIELD OFFICE *******
;
DPRT   TERM;                                       Customer departs DMV.
;
;******* ALTER ARRIVAL RATES, MVR AND RECEPTIONIST AVAILABILITY *******
;************ FOR EACH OF 11 POSSIBLE HOURS OUT OF THE DAY ************
;
H1     CREATE,,0,,1;                               Create dummy.
       ACT,60;                                     Delay 60 minutes.
       EVENT(2);                                   Call subrtn for storing stats.
H2     ASSIGN,XX(3)=XX(56),XX(4)=XX(21);           Alter arrival rate, # receps.
       ASSIGN,XX(5)=XX(57);                        Alter drv line arrival rates.
       ALTER,MVR1/XX(54);                          Alter # MVRs of type 1.
       ALTER,MVR2/0;                               Alter # MVRs of type 2.
       ALTER,MVR3/0;                               Alter # MVRs of type 3.
       ALTER,MVR4/0;                               Alter # MVRs of type 4.
       ALTER,RECEP/XX(55);                         Alter # receptionists.
       ACT,60;                                     Delay 60 minutes.
       EVENT(2);                                   Call subrtn for storing stats.
H3     ASSIGN,XX(3)=XX(60),XX(4)=XX(22);
       ASSIGN,XX(5)=XX(61);                        "
       ALTER,MVR1/XX(58);
       ALTER,MVR2/0;                               "
       ALTER,MVR3/0;
       ALTER,MVR4/0;                               "
       ALTER,RECEP/XX(59);
       ACT,60;                                     "
       EVENT(2);
H4     ASSIGN,XX(3)=XX(64),XX(4)=XX(23);
       ASSIGN,XX(5)=XX(65);
       ALTER,MVR1/XX(62);
       ALTER,MVR2/0;
       ALTER,MVR3/0;
```

```
        ALTER,MVR4/0;
        ALTER,RECEP/XX(63);
        ACT,60;
        EVENT(2);
H5      ASSIGN,XX(3)=XX(68),XX(4)=XX(24);
        ASSIGN,XX(5)=XX(69);
        ALTER,MVR1/XX(66);
        ALTER,MVR2/0;
        ALTER,MVR3/0;
        ALTER,MVR4/0;
        ALTER,RECEP/XX(67);
        ACT,60;
        EVENT(2);
H6      ASSIGN,XX(3)=XX(72),XX(4)=XX(25);
        ASSIGN,XX(5)=XX(73);
        ALTER,MVR1/XX(70);
        ALTER,MVR2/0;
        ALTER,MVR3/0;
        ALTER,MVR4/0;
        ALTER,RECEP/XX(71);
        ACT,60;
        EVENT(2);
H7      ASSIGN,XX(3)=XX(76),XX(4)=XX(26);
        ASSIGN,XX(5)=XX(77);
        ALTER,MVR1/XX(74);
        ALTER,MVR2/0;
        ALTER,MVR3/0;
        ALTER,MVR4/0;
        ALTER,RECEP/XX(75);
        ACT,60;
        EVENT(2);
H8      ASSIGN,XX(3)=XX(80),XX(4)=XX(27);
        ASSIGN,XX(5)=XX(81);
        ALTER,MVR1/XX(78);
        ALTER,MVR2/0;
        ALTER,MVR3/0;
        ALTER,MVR4/0;
        ALTER,RECEP/XX(79);
        ACT,60;
        EVENT(2);
H9      ASSIGN,XX(3)=XX(84),XX(4)=XX(28);
        ASSIGN,XX(5)=XX(85);
        ALTER,MVR1/XX(82);
        ALTER,MVR2/0;
        ALTER,MVR3/0;
        ALTER,MVR4/0;
        ALTER,RECEP/XX(83);
        ACT,60;
        EVENT(2);
H10     ASSIGN,XX(3)=XX(88),XX(4)=XX(29);
        ASSIGN,XX(5)=XX(89);
        ALTER,MVR1/XX(86);
        ALTER,MVR2/0;
        ALTER,MVR3/0;
        ALTER,MVR4/0;
        ALTER,RECEP/XX(87);
        ACT,60;
        EVENT(2);
H11     ASSIGN,XX(3)=XX(92),XX(4)=XX(30);
```

```
      ASSIGN,XX(5)=XX(93);
      ALTER,MVR1/XX(90);
      ALTER,MVR2/0;
      ALTER,MVR3/0;
      ALTER,MVR4/0;
      ALTER,RECEP/XX(91);
      ACT,60;
      EVENT(2);
      TERM;
      END;
;
;-------------------------------------------
;*******************************************!
;******* END OF DMV FIELD OFFICE MODEL *******!
;*******************************************!
;-------------------------------------------
;
FIN;
```

APPENDIX 3. FORTRAN SUBROUTINES

```
$NOTSTRICT
$STORAGE:2
$NOTLARGE
$NOFLOATCALLS
*
*===================================================================
*            ********** EVENT SUBROUTINE *********
*
* This subroutine is the link between the network model and the
* user-written FORTRAN subroutines.
* EVENT(1) will cause the menu to be called from the network.
*          This subroutine controls the front end of the program.
* EVENT(2) will cause the output subroutine to be called from the
*          network.  This subroutine controls the back end of the
*          program.
*===================================================================
*
      SUBROUTINE EVENT(I)
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      GO TO (1,2) I
 1    CALL MENU
      RETURN
 2    CALL OUTPUT
      RETURN
      END
*
*===================================================================
*              ******** HEADING SUBROUTINE ********
*
* This subroutine is called by the MENU subroutine for printing
* the DMV STAFFING AND ARRIVALS table heading.
*===================================================================
*
      SUBROUTINE HEADING
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     +,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
 6    FORMAT (14X,A)
      WRITE(*,6) ' '
      WRITE(*,6) ' '
      WRITE(*,6) '                DMV STAFFING AND ARRIVALS                '
      WRITE(*,6) '                -------------------------'
      WRITE(*,6) ' '
      WRITE(*,6) '          MVRs                    CUSTOMER    # DRIVING'
      WRITE(*,6) '        SERVING      RECEP-       ARRIVAL      TESTS'
      WRITE(*,6) ' HOUR  CUSTOMERS    TIONISTS    INTERVAL    SCHEDULED'
      WRITE(*,6) ' ------------------------------------------------------'
      RETURN
      END
```

```
*
*=================================================================
*               ********** MENU SUBROUTINE **********
*
* This subroutine is the front end of the program.  It is the
* user interactive part.  Four choices are available from the
* menu:  input, edit/review, run and exit.
* Subroutines called:
*       INITIALIZE  variables before running simulation
*       HEADING     printed for dmv staffing & arrivals table
*       STOREDATA   stores data in a named data file
* Variables used:
*       CHANGE    for changing (editing) data file
*       CHOICE    selected from menu
*       COEFF     reduction in service time after receptionist
*       FNAME     file name for saving new data under
*       FRACTION  fraction customers contacted by receptionist
*       IDLE      y/n if receptionist acts as mvr when idle
*       IOCHECK   > 0 if input is invalid
*       N         number of simulation runs completed
*       RUN       y/n for running simulation on selected file name
*       SNAME     file name for saving modified data under
*       PERCENT   % customers contacted by receptionist
*       TIME      receptionist service time
* Arrays used:
*       ARR(K)     arrival rate of customers hour K
*       NDRV(K)    number of driving tests scheduled hour K
*       NMVR(K)    number of mvrs scheduled hour K
*       NREC(K)    number of receptionists scheduled hour K
*=================================================================
*
      SUBROUTINE MENU
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     +,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/UCOM2/ NMVR(11),NREC(11),ARR(11),NDRV(11),PERCENT,TIME,
     +COEFF,IDLE
      COMMON/UCOM3/ N,FNAME
      CHARACTER FNAME*12,SNAME*12,CHOICE*1,IDLE*1,RUN*1,CHANGE*1
*
*     print menu if before 1st run or after 5th run
*     otherwise go to another simulation run
*
      IF (N.GT.0.AND.N.LT.5) GO TO 302
*
 6    FORMAT (14X,A)
 7    FORMAT (/////////)
 8    FORMAT (/////)
 11   FORMAT (BN,2I11,F11.2,I10)
 21   FORMAT (BN,14X,I4,I8,I11,F13.2,I10)
 22   FORMAT (3X,A,I6,I8,I11,F13.2,I10)
 23   FORMAT (BN,3X,I10,I12,F11.2,I10)
*
*     print introduction to program
*
```

```
      WRITE(*,7)
      WRITE(*,6) '            **** DMV SIMULATION PROGRAM ****'
      WRITE(*,6) ' '
      WRITE(*,6) '                 Written by A. Mechling'
      WRITE(*,6) '                 Oregon State University'
      WRITE(*,6) ' '
      WRITE(*,6) '      This program simulates customers arriving to a'
      WRITE(*,6) ' DMV field office.  The office may have a variable'
      WRITE(*,6) ' number of MVRs, an arrival rate that changes each'
      WRITE(*,6) ' hour, and variable number of receptionists who can'
      WRITE(*,6) ' do only receptionist duties or can serve customers'
      WRITE(*,6) ' out of the MVR queue when they are idle.'
      WRITE(*,6) '      The program is menu driven and no knowledge of'
      WRITE(*,6) ' SLAM or simulation methods is required for the user.'
      WRITE(*,6) ' At times when input is needed from the user either'
      WRITE(*,6) ' a selection of choices will be available or the user'
      WRITE(*,6) ' must input information.  When the user must input'
      WRITE(*,6) ' their own information an example of the input is'
      WRITE(*,6) ' given in parentheses (), or a default value is given'
      WRITE(*,6) ' in brackets [].  The default value can be taken by'
      WRITE(*,6) ' hitting <ENTER> without any input.'
      WRITE(*,6) '      Output may be obtained at any time by hitting'
      WRITE(*,6) ' <PRINT SCREEN>, after the printer is ready.'
      PAUSE
*
*     print menu
*
 5    WRITE(*,7)
      WRITE(*,6) '  ________________________________________________'
      WRITE(*,6) ' |                                                |'
      WRITE(*,6) ' |                 ****  MENU  ****               |'
      WRITE(*,6) ' |                                                |'
      WRITE(*,6) ' |________________________________________________|'
      WRITE(*,6) ' |                                                |'
      WRITE(*,6) ' |                                                |'
      WRITE(*,6) ' |                                                |'
      WRITE(*,6) ' |            1 - INPUT NEW DATA FILE             |'
      WRITE(*,6) ' |            2 - EDIT/REVIEW OLD DATA FILE       |'
      WRITE(*,6) ' |            3 - RUN THE SIMULATION              |'
      WRITE(*,6) ' |            4 - EXIT                            |'
      WRITE(*,6) ' |                                                |'
      WRITE(*,6) ' |________________________________________________|'
      WRITE(*,8)
*
*     user enters choice from keyboard and option is executed
*
      WRITE(*,'(A\)') ' Input your choice -->  '
      READ(*,'(A)') CHOICE
      IF (CHOICE.EQ.'1') THEN
          GO TO 1
        ELSEIF (CHOICE.EQ.'2') THEN
          GO TO 2
        ELSEIF (CHOICE.EQ.'3') THEN
          GO TO 3
```

```
        ELSEIF (CHOICE.EQ.'4') THEN
          STOP
        ELSE
          GO TO 5
        ENDIF
*
***** MENU OPTION 1, inputting new data file
*
*       print instructions
*
 1      WRITE(*,7)
        WRITE(*,6) '          **** INPUT NEW DATA FILE **** '
        WRITE(*,*) ' '
        WRITE(*,*) ' '
        WRITE(*,6) '      This  subroutine will allow you to  input new'
        WRITE(*,6) ' data for the model to be simulated on.  Data to be'
        WRITE(*,6) ' input for each hour are:'
        WRITE(*,*) ' '
        WRITE(*,6) ' MVRs SERVING CUSTOMERS - includes all MVRs serving'
        WRITE(*,6) '          customers at the counter  or on the drive'
        WRITE(*,6) '          line during that hour - integer input'
        WRITE(*,6) ' RECEPTIONISTS - number of  receptionists  on  duty'
        WRITE(*,6) '          greeting  customers  during  that  hour -'
        WRITE(*,6) '          integer input'
        WRITE(*,6) ' CUSTOMER ARRIVAL INTERVAL - the time  interval  at '
        WRITE(*,6) '          which customers arrive to the DMV in terms '
        WRITE(*,6) '          of average minutes between arrivals - real'
        WRITE(*,6) '          input, must use a decimal'
        WRITE(*,6) ' # DRIVING TESTS SCHEDULED - the  total  number  of '
        WRITE(*,6) '          driving tests  that are scheduled  during'
        WRITE(*,6) '          that hour - integer input'
        WRITE(*,*) ' '
        WRITE(*,*) ' '
        PAUSE
        WRITE(*,7)
        WRITE(*,6) '      Starting with the first hour of the day, input'
        WRITE(*,6) ' the data for the row,  hit <ENTER>  and continue to'
        WRITE(*,6) ' the next hour.   When the office closes  simply hit'
        WRITE(*,6) ' <ENTER> for the following rows, without any input.'
        WRITE(*,6) '      The four data may be separated by blank spaces'
        WRITE(*,6) ' to center data under headings,  or by commas.  Some'
        WRITE(*,6) ' examples of input are:'
        WRITE(*,*) ' '
        WRITE(*,6) ' Valid Input        vs.        Invalid Input'
        WRITE(*,6) ' -----------                   -------------'
        WRITE(*,6) ' 5,2,3.5,0                     5 2,3.5,0'
        WRITE(*,6) ' 3     1     2.5     4       3.5     1     2.5     4'
        WRITE(*,6) ' 6,3,4.0,1                     6,3,4,1  '
*
*       call subroutine to print table heading
*
        CALL HEADING
*
*       for each hour (1-11) read user input from keyboard
```

```
*
      DO 102 I=1,11
 101  WRITE(*,'(14X,I4,3X\)') I
      READ(*,11,IOSTAT=IOCHECK) NMVR(I),NREC(I),ARR(I),NDRV(I)
      IF (IOCHECK.GT.0) THEN
        WRITE(*,*) ' THE ABOVE INPUT IS INVALID, PLEASE RE-ENTER.'
        GO TO 101
      ENDIF
 102  CONTINUE
*
*     if there are any receptionists scheduled, ask
*     questions about the receptionist function
*
      IF (NREC(1).NE.0.OR.NREC(2).NE.0.OR.NREC(3).NE.0.OR.
     +NREC(4).NE.0.OR.NREC(5).NE.0.OR.NREC(6).NE.0.OR.
     +NREC(7).NE.0.OR.NREC(8).NE.0.OR.NREC(9).NE.0.OR.
     +NREC(10).NE.0.OR.NREC(11).NE.0) THEN
 103    WRITE(*,*) ' '
        WRITE(*,*) 'What percent of the customers are met by the'
        WRITE(*,'(A\)') ' receptionist (<=100)  --> '
        READ(*,*,IOSTAT=IOCHECK) PERCENT
        IF (PERCENT.GT.100.OR.IOCHECK.GT.0) THEN
          WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
          GO TO 103
        ENDIF
 104    WRITE(*,*) ' '
        WRITE(*,*) 'Can the receptionist serve customers from the'
        WRITE(*,'(A\)') ' MVR queue when idle (Y/N) [N] -->  '
        READ(*,'(A)',IOSTAT=IOCHECK) IDLE
        IF (IDLE.EQ.' ') IDLE='N'
        IF ((IDLE.NE.'Y'.AND.IDLE.NE.'N').OR.IOCHECK.GT.0) THEN
          WRITE(*,*) IDLE,' IS INVALID INPUT'
          GO TO 104
        ENDIF
 105    WRITE(*,*) ' '
        WRITE(*,*) 'What is the average service time for the '
        WRITE(*,'(A\)') ' receptionist in minutes (e.g. 1.20) -->  '
        READ(*,*,IOSTAT=IOCHECK) TIME
        IF (IOCHECK.GT.0) THEN
          WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
          GO TO 105
        ENDIF
 106    WRITE(*,*) ' '
        WRITE(*,*) 'By what percent is service time reduced '
        WRITE(*,'(A\)') ' after receptionist contact (<=100) -->  '
        READ(*,*,IOSTAT=IOCHECK) COEFF
        IF (IOCHECK.GT.0.OR.COEFF.GT.100) THEN
          WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
          GO TO 106
        ENDIF
*
*     if no receptionist has been scheduled set variables to default
*
      ELSE
```

```
        PERCENT=0.00
        IDLE='N'
        TIME=0.0
        COEFF=0.00
      ENDIF
*
*     read name of file from user (keyboard) in which the
*     data will be stored, open the file
*
      WRITE(*,*) ' '
      WRITE(*,*) 'Enter the name of the file in which the raw data '
      WRITE(*,'(A\)') ' will be saved  (e.g. MEDFORD.DAT) -->  '
      READ(*,'(A)') FNAME
      OPEN(1,FILE=FNAME,STATUS='NEW')
*
*     call subroutine for storing the data in the open file
*
      CALL STOREDATA
*
*     return to the menu
*
      GO TO 5
*
***** MENU OPTION 2, editing/reviewing an old data file
*
 2    WRITE(*,7)
      WRITE(*,6) '       ***** EDIT/REVIEW OLD DATA FILE *****'
      WRITE(*,8)
      WRITE(*,6) '      This subroutine will allow review of a pre-'
      WRITE(*,6) ' viously stored data set  and to change  selected'
      WRITE(*,6) ' values.'
      WRITE(*,6) '      The edited data may be saved under the same'
      WRITE(*,6) ' name (replace the old file)  or under a new name'
      WRITE(*,6) ' (the old file remains and a new file is created).'
 201  WRITE(*,8)
*
*     read name of file from user (keyboard) from which the
*     data will be retrieved, open the file
*
      WRITE(*,*) 'Enter the name of the file from which you want to '
      WRITE(*,'(A\)') ' view the data  [DEFAULT.DAT] -->  '
      READ(*,'(A)') FNAME
      IF (FNAME.EQ.' ') FNAME='DEFAULT.DAT'
      OPEN(2,FILE=FNAME,IOSTAT=IOCHECK)
      IF (IOCHECK.GT.0) THEN
        WRITE (*,*) FNAME,' IS AN INVALID FILE NAME'
        GO TO 201
      ENDIF
*
*     read the file contents into appropriate variables and
*     convert into proper form for displaying in table
*
      REWIND 2
      NMVR(0)=0
```

```
      NREC(0)=0
      DO 202 K=1,11
        READ(2,*) NMVR(K)
        NMVR(K)=NMVR(K-1)+NMVR(K)
        READ(2,*) NREC(K)
        NREC(K)=NREC(K-1)+NREC(K)
        READ(2,*) ARR(K)
        IF (ARR(K).GE.60) ARR(K)=0
        READ(2,*) NDRV(K)
        IF (NDRV(K).GT.60) THEN
          NDRV(K)=0
        ELSE
          NDRV(K)=60/NDRV(K)
        ENDIF
 202  CONTINUE
      READ(2,*) FRACTION
      READ(2,'(A)') IDLE
      READ(2,*) TIME
      READ(2,*) COEFF
      CLOSE(2)
 203  WRITE(*,7)
*
*     call subroutine to print table heading
*
      CALL HEADING
*
*     for each hour (1-11) print data file information into table
*
      DO 204 I=1,11
 204  WRITE(*,21) I,NMVR(I),NREC(I),ARR(I),NDRV(I)
*
*     print information about the receptionist function
*     if a receptionist has been scheduled
*
      WRITE(*,*) ' '
      IF (FRACTION.NE.0.00) THEN
        WRITE(*,'(A,F6.1,A)') ' THE RECEPTIONIST CONTACTS',
     ++(FRACTION*100),' PERCENT OF THE CUSTOMERS'
        IF (IDLE.EQ.'Y') THEN
          WRITE(*,*) 'THE RECEPTIONIST ALSO ACTS AS AN MVR WHEN IDLE'
        ELSE
          WRITE(*,*) 'THE RECEPTIONIST ONLY DOES RECEPTIONIST DUTIES'
        ENDIF
        WRITE(*,'(A,F5.2,A)') ' THE RECEPTIONIST SERVICE TIME IS ',
     +TIME, ' MINUTES'
        WRITE(*,'(A,F6.1,A)') ' SERVICE TIME IS REDUCED BY ',
     ++((1-COEFF)*100),' PERCENT AFTER RECEPTIONIST CONTACT'
      ENDIF
*
*     changes may be made to the table if the user wants
*
 205  WRITE(*,*) 'Do you want to make changes in the DMV STAFFING AND'
      WRITE(*,'(A\)') ' ARRIVALS table (Y/N) --> '
      READ(*,'(A)',IOSTAT=IOCHECK) CHANGE
```

```
      IF ((CHANGE.NE.'Y'.AND.CHANGE.NE.'N').OR.IOCHECK.GT.0) THEN
        WRITE(*,*) CHANGE,' IS INVALID INPUT'
        GO TO 205
      ENDIF
*
*     if a change is desired it is done a row at a time
*
      IF (CHANGE.EQ.'Y') THEN
 206    WRITE(*,'(A\)') ' Which row do you want to change (1,2...) --> '
        READ(*,*,IOSTAT=IOCHECK) I
        IF (IOCHECK.GT.0.OR.I.GT.11.OR.I.LE.0) THEN
          WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
          GO TO 206
        ENDIF
*
*     call subroutine to print table heading
*
        CALL HEADING
*
*     print the old data row and the user inputs the new data row
*
        WRITE(*,22) ' OLD DATA',I,NMVR(I),NREC(I),ARR(I),NDRV(I)
 207    WRITE(*,'(3X,A,I6\)') ' NEW DATA',I
        READ(*,23,IOSTAT=IOCHECK) NMVR(I),NREC(I),ARR(I),NDRV(I)
        IF (IOCHECK.GT.0) THEN
          WRITE(*,*) ' THE ABOVE INPUT IS INVALID, PLEASE RE-ENTER.'
          GO TO 207
        ENDIF
*
*     return to asking the user if more changes in the table
*     are required
*
        GO TO 203
      ENDIF
*
*     if no changes are desired in the table and
*     if there are any receptionists scheduled, ask
*     questions about the receptionist function
*
      IF (NREC(1).NE.0.OR.NREC(2).NE.0.OR.NREC(3).NE.0.OR.
     +NREC(4).NE.0.OR.NREC(5).NE.0.OR.NREC(6).NE.0.OR.
     +NREC(7).NE.0.OR.NREC(8).NE.0.OR.NREC(9).NE.0.OR.
     +NREC(10).NE.0.OR.NREC(11).NE.0) THEN
 208    WRITE(*,*) ' '
        WRITE(*,*) 'What percent of the customers are met by the'
        WRITE(*,'(A,F7.1,A\)') ' receptionist (',+(FRACTION*100),') -->'
        READ(*,*,IOSTAT=IOCHECK) PERCENT
        IF (PERCENT.GT.100.OR.IOCHECK.GT.0) THEN
          WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
          GO TO 208
        ENDIF
 209    WRITE(*,*) ' '
        WRITE(*,*) 'Can the receptionist serve customers from the'
        WRITE(*,'(2A,A\)') ' MVR queue when idle [',IDLE,'] -->  '
```

```
      READ(*,'(A)',IOSTAT=IOCHECK) CHANGE
      IF (CHANGE.NE.' ') IDLE=CHANGE
      IF ((IDLE.NE.'Y'.AND.IDLE.NE.'N').OR.IOCHECK.GT.0) THEN
        WRITE(*,*) IDLE,' IS INVALID INPUT'
        GO TO 209
      ENDIF
210   WRITE(*,*) ' '
      WRITE(*,*) 'What is the average service time for the'
      WRITE(*,'(A,F4.2,A\)') ' receptionist in minutes (',TIME,') -->'
      READ(*,*,IOSTAT=IOCHECK) TIME
      IF (IOCHECK.GT.0) THEN
        WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
        GO TO 210
      ENDIF
211   WRITE(*,*) ' '
      WRITE(*,*) 'By what percent is service time reduced after '
      WRITE(*,'(A,F5.1,A\)') ' receptionist contact (',
     ++((1-COEFF)*100),') --> '
      READ(*,*,IOSTAT=IOCHECK) COEFF
      IF (IOCHECK.GT.0.OR.COEFF.GT.100) THEN
        WRITE(*,*) ' THE ABOVE IS INVALID INPUT'
        GO TO 211
      ENDIF
     ELSE
      PERCENT=0.00
      IDLE='N'
      TIME=0.0
      COEFF=0.00
     ENDIF
     WRITE(*,*) ' '
*
*     the user can save the edited file under the same name
*     (given as the default) or under a new name, open the file
*
      WRITE(*,*) 'Enter the name of the file in which the raw data '
      WRITE(*,'(2A,A\)') ' will be saved  [',FNAME,'] -->  '
      READ(*,'(A)') SNAME
      IF (SNAME.EQ.' ') SNAME=FNAME
      OPEN(1,FILE=SNAME,STATUS='NEW')
*
*     call the subroutine for storing the data in the open file
*
      CALL STOREDATA
*
*     return to the menu
*
      GO TO 5
*
***** MENU OPTION 3, executing five simulation runs
*
 3    WRITE(*,7)
      WRITE(*,6) '             **** RUN A SIMULATION ****'
      WRITE(*,8)
      WRITE(*,6) '      This subroutine will execute the simulation'
```

```
      WRITE(*,6) ' of a previously stored data set.  The statistics'
      WRITE(*,6) ' are then displayed on the screen as output.'
      WRITE(*,6) '      <PRINT SCREEN>  will  cause  the printer to '
      WRITE(*,6) ' make a hard copy  of the data  that is displayed'
      WRITE(*,6) ' on the screen. '
*
*     read name of file from user (keyboard) for running the simulation
*
 301  WRITE(*,8)
      WRITE(*,*) 'Enter the name of the data file from which the model'
      WRITE(*,'(A\)') ' will be run [DEFAULT.DAT] -->  '
      READ(*,'(A)') FNAME
      IF (FNAME.EQ.' ') FNAME='DEFAULT.DAT'
*
*     check with the user before commencing run
*
      WRITE(*,'(/2A,A\)') ' Is ',FNAME,
     +' the file you want to run (Y/N) [Y] --> '
      READ(*,'(A)') RUN
      IF (RUN.NE.'Y'.AND.RUN.NE.' ') GO TO 301
*
*     open the file and read the data into appropriate
*     XX variables for network usage
*
 302  OPEN(3,FILE=FNAME,IOSTAT=IOCHECK)
      IF (IOCHECK.GT.0) THEN
        WRITE (*,*) FNAME,' IS AN INVALID FILE NAME'
        GO TO 301
      ENDIF
      REWIND 3
      READ(3,*) (XX(I),I=50,94)
      READ(3,'(A)') IDLE
      READ(3,*) XX(95)
      READ(3,*) XX(96)
      CLOSE(3)
      IF (IDLE.EQ.'Y') XX(1)=1
      IF (IDLE.NE.'Y') XX(1)=0
*
*     call subroutine to initialize other variables
*
      CALL INITIALIZE
*
*     display message to user on simulation status
*
      WRITE(*,7)
      WRITE(*,8)
      WRITE(*,*) '               THE SIMULATION IS RUNNING ON ',FNAME
      WRITE(*,'(A,I4)') '                                  RUN # ',+(N+1)
      WRITE(*,7)
*
*     return control to the network for Nth simulation run
*
      RETURN
      END
```

```
*
*=====================================================================
*               ******* INITIALIZE SUBROUTINE *******
*
* This subroutine will initialize the appropriate variables
* before each of the 5 simulation runs.
* Variables used:
*       HOURM     number of hours mvrs are on duty
*       HOURR     number on hours receptionists are on duty
*       N         number of simulation runs completed
*       RCONT     number of customers receptionist contacts
*       RSERV     number of customers receptionist serves as mvr
*       SUM1      total number of receptionists on duty
*       SUM2      total number of mvrs on duty
*       TQL1      total of the AQL(1,I)
*       TQL2      total of the AQL(2,I)
*       TUTIL1    total of the UTIL(1,I)
*       TUTIL2    total of the UTIL(2,I)
*       TWT1      total of the AWT(1,I)
*       TWT2      total of the AWT(2,I)
* Arrays used:    (J=1 receptionist, J=2 mvr)
*       AWT(J,I)  sum over 5 runs of average waiting times in hour I
*       AQL(J,I)  sum over 5 runs of average queue lengths in hour I
*       SDWT(J,I) sum over 5 runs of std dev waiting times in hour I
*       SDQL(J,I) sum over 5 runs of std dev queue lengths in hour I
*       UTIL(J,I) sum over 5 runs of utilization in hour I
*       XX(20-30) total number of receptionists in hour I+19
*       XX(31-41) total number of mvrs in hour I+30
*=====================================================================
*
      SUBROUTINE INITIALIZE
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     +,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/UCOM1/RSERV,RCONT,TWT1,TWT2,TQL1,TQL2,TUTIL1,TUTIL2,HOURM,
     +HOURR,SUM1,SUM2,AWT(2,11),AQL(2,11),SDWT(2,11),SDQL(2,11),
     +UTIL(2,11)
      COMMON/UCOM3/ N,FNAME
      CHARACTER FNAME*12
      IF (N.GT.0.AND.N.LT.5) GO TO 30
*
*     initialize these variables if before the first
*     run or after the fifth
*
      N=0
      TWT1=0
      TWT2=0
      TQL1=0
      TQL2=0
      TUTIL1=0
      TUTIL2=0
      HOURM=0
      HOURR=0
      RSERV=0
      RCONT=0
```

```
*
*     initialize these variables every run
*
 30   SUM1=0
      SUM2=0
      DO 20 I=1,11
        SUM1=SUM1+XX(47+I*4)
        XX(I+19)=SUM1
        SUM2=SUM2+XX(46+I*4)
        XX(I+30)=SUM2
        IF (N.GT.0.AND.N.LT.5) GO TO 20
*
*     initialize these variables if before the first
*     run or after the fifth
*
        DO 10 J=1,2
          AWT(J,I)=0
          AQL(J,I)=0
          SDWT(J,I)=0
          SDQL(J,I)=0
          UTIL(J,I)=0
 10     CONTINUE
 20   CONTINUE
      RETURN
      END
```

```
*
*=====================================================================
*                  ********* OUTPUT SUBROUTINE *********
*
* This subroutine is called through EVENT 2 and is the program back
* end.  It collects statistics every hour, then clears the statistical
* arrays.  After 5 runs have been completed for 11 hours each, the
* output is printed.
* Variables used:
*       HOURM     number of hours mvrs are on duty
*       HOURR     number on hours receptionists are on duty
*       N         number of simulation runs completed
*       RCONT     number of customers receptionist contacts
*       RSERV     number of customers receptionist serves as mvr
*       SUM1      total number of receptionists on duty
*       SUM2      total number of mvrs on duty
*       TQL1      total of the AQL(1,I)
*       TQL2      total of the AQL(2,I)
*       TUTIL1    total of the UTIL(1,I)
*       TUTIL2    total of the UTIL(2,I)
*       TWT1      total of the AWT(1,I)
*       TWT2      total of the AWT(2,I)
* Arrays used:    (J=1 receptionist, J=2 mvr)
*       AWT(J,I)  sum over 5 runs of average waiting times in hour I
*       AQL(J,I)  sum over 5 runs of average queue lengths in hour I
*       SDWT(J,I) sum over 5 runs of std dev waiting times in hour I
*       SDQL(J,I) sum over 5 runs of std dev queue lengths in hour I
*       UTIL(J,I) sum over 5 runs of utilization in hour I
*       XX(20-30) total number of receptionists in hour I+19
*       XX(31-41) total number of mvrs in hour I+30
*=====================================================================
*
      SUBROUTINE OUTPUT
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     +,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/UCOM1/RSERV,RCONT,TWT1,TWT2,TQL1,TQL2,TUTIL1,TUTIL2,HOURM,
     +HOURR,SUM1,SUM2,AWT(2,11),AQL(2,11),SDWT(2,11),SDQL(2,11),
     +UTIL(2,11)
      COMMON/UCOM3/ N,FNAME
      CHARACTER FNAME*12
  1   FORMAT (13X,I4,1X,5F9.2)
  2   FORMAT (1X,A,F6.0)
  3   FORMAT (4X,A,F10.2,2(11X,F7.2))
  6   FORMAT (13X,A)
  8   FORMAT (/////)
*
*     increment the hour (just simulated) by 1
*
      J=J+1
*
*     sum the hourly statistics for the Jth hour
*
      AWT(1,J)=CCAVG(11)+AWT(1,J)
      AWT(2,J)=CCAVG(8)+AWT(2,J)
```

```
      SDWT(1,J)=CCSTD(11)+SDWT(1,J)
      SDWT(2,J)=CCSTD(8)+SDWT(2,J)
      AQL(1,J)=TTAVG(5)+AQL(1,J)
      AQL(2,J)=TTAVG(1)+AQL(2,J)
      SDQL(1,J)=TTSTD(5)+SDQL(1,J)
      SDQL(2,J)=TTSTD(1)+SDQL(2,J)
*
*     if there were no receptionists on duty the values are 0
*
      IF (XX(J+19).LE.0) THEN
        UTIL(1,J)=0
        AWT(1,J)=0
        SDWT(1,J)=0
        AQL(1,J)=0
        SDQL(1,J)=0
*
*     if there were receptionists on duty increment the hourr by 1
*     calculate hourly utilization and totals
*
      ELSE
        HOURR=HOURR+1
        UTIL(1,J)=AAAVG(5)/XX(J+19)+UTIL(1,J)
        TUTIL1=TUTIL1+AAAVG(5)/XX(J+19)
        TWT1=TWT1+CCAVG(11)
        TQL1=TQL1+TTAVG(5)
      ENDIF
*
*     if there were no mvrs on duty the values are 0
*
      IF (XX(J+30).LE.0) THEN
        UTIL(2,J)=0
        AWT(2,J)=0
        SDWT(2,J)=0
        AQL(2,J)=0
        SDQL(2,J)=0
*
*     if there were mvrs on duty increment the hourm by 1
*     calculate daily utilization and totals
*
      ELSE
        HOURM=HOURM+1
        UTIL(2,J)=AAAVG(1)/XX(J+30)+UTIL(2,J)
        TUTIL2=TUTIL2+AAAVG(1)/XX(J+30)
        TWT2=TWT2+CCAVG(8)
        TQL2=TQL2+TTAVG(1)
      ENDIF
*
*     store other statistical values and sums
*
      RSERV=RSERV+CCNUM(2)
      RCONT=RCONT+CCNUM(6)
*
*     clear the statistical arrays in SLAM
*
```

```
      CALL CLEAR
*
*     if its the end of the day then reset the hour (J) to 0
*     and increment the day (N) by 1
*
      IF (J.EQ.11) THEN
        N=N+1
        J=0
      ENDIF
*
*     if 5 simulation runs have been completed then
*     calculate the averages over the 5 runs
*
      IF (N.EQ.5) THEN
      DO 30 K=1,11
        DO 40 L=1,2
          AWT(L,K)=AWT(L,K)/N
          AQL(L,K)=AQL(L,K)/N
          SDWT(L,K)=SDWT(L,K)/N
          SDQL(L,K)=SDQL(L,K)/N
          UTIL(L,K)=UTIL(L,K)/N
 40     CONTINUE
 30   CONTINUE
      RSERV=RSERV/N
      RCONT=RCONT/N
      TAWT2=TWT2/HOURM
      TAQL2=TQL2/HOURM
      TAUTL2=TUTIL2/HOURM
      IF (HOURR.LE.0) GO TO 50
*
*     if there was a receptionist at some time do these stats
*
      TAWT1=TWT1/HOURR
      TAQL1=TQL1/HOURR
      TAUTL1=TUTIL1/HOURR
*
*     print the heading for the receptionist
*
      WRITE(*,8)
      WRITE(*,*) '       ',FNAME,'STATISTICS ON THE RECEPTIONIST QUEUE'
      WRITE(*,*) ' '
      WRITE(*,6) '          AVERAGE  STD DEV  AVERAGE  STD DEV  RECEP.'
      WRITE(*,6) '          WAITING  WAITING  QUEUE    QUEUE    UTILI-'
      WRITE(*,6) ' HOUR     TIME     TIME     LENGTH   LENGTH   ZATION'
      WRITE(*,6) ' ------------------------------------------------'
*
*     print the statistics for each hour of the day (1-11)
*
      DO 10 I=1,11
 10     WRITE(*,1) I,AWT(1,I),SDWT(1,I),AQL(1,I),SDQL(1,I),UTIL(1,I)
*
*     print the average over the day
*
      WRITE(*,3) 'DAILY AVERAGE',TAWT1,TAQL1,TAUTL1
```

```
*
*     print other receptionist function stats
*
      WRITE(*,*) ' '
      WRITE(*,2) ' NUMBER OF CUST RECEP. SERVES AS AN MVR ',RSERV
      WRITE(*,2) ' NUMBER OF CUST RECEP. CONTACTS         ',RCONT
      WRITE(*,*) ' '
      PAUSE
*
*     print the heading for the mvr
*
 50   WRITE(*,8)
      WRITE(*,*) '             ',FNAME,' STATISTICS ON THE MVR QUEUE'
      WRITE(*,*) ' '
      WRITE(*,6) '          AVERAGE  STD DEV  AVERAGE  STD DEV  MVR'
      WRITE(*,6) '          WAITING  WAITING  QUEUE    QUEUE    UTILI-'
      WRITE(*,6) ' HOUR     TIME     TIME     LENGTH   LENGTH   ZATION'
      WRITE(*,6) ' ------------------------------------------------'
*
*     print the statistics for each hour of the day (1-11)
*
      DO 20 M=1,11
 20     WRITE(*,1) M,AWT(2,M),SDWT(2,M),AQL(2,M),SDQL(2,M),UTIL(2,M)
*
*     print the average over the day
*
      WRITE(*,3) 'DAILY AVERAGE',TAWT2,TAQL2,TAUTL2
      WRITE(*,'(///)')
      PAUSE
      ENDIF
*
*     return control to the network
*
      RETURN
      END
```

```
*
*=====================================================================
*            ********* STOREDATA SUBROUTINE **********
*
* This subroutine will take the data as input by the user and store
* it in an acceptable form for running the simulation.  It is
* called from the MENU subroutine.
* Variables used:
*       COEFF     reduction in service time after receptionist
*       IDLE      y/n if receptionist acts as mvr when idle
*       PERCENT   % customers contacted by receptionist
*       TIME      receptionist service time
* Arrays used:
*       ARR(K)     arrival rate of customers hour K
*       NDRV(K)    number of driving tests scheduled hour K
*       NMVR(K)    number of mvrs scheduled hour K
*       NREC(K)    number of receptionists scheduled hour K
*=====================================================================
*
      SUBROUTINE STOREDATA
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     +,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
      COMMON/UCOM2/ NMVR(11),NREC(11),ARR(11),NDRV(11),PERCENT,TIME,
     +COEFF,IDLE
      CHARACTER IDLE*1
      NMVR(0)=0
      NREC(0)=0
*
*     for each hour of the day store the data in file 1
*
      DO 108 K=1,11
        X=1
*
*     store the change in the number of receps and mvrs each hour
*     as calculated from the total number scheduled
*
        WRITE(1,*) +(NMVR(K)-NMVR(K-1))
        WRITE(1,*) +(NREC(K)-NREC(K-1))
*
*     store the time between creations as 660 minutes if there
*     are no arrivals in hour K (i.e. the end of the day)
*
        IF (ARR(K).EQ.0.OR.ARR(K).EQ.' ') ARR(K)=660
        WRITE(1,*) ARR(K)
*
*     if there are no driving tests scheduled in hour K then
*     calculate the time until the next arrival and store it
*     as the time between creations
*
        IF (NDRV(K).LE.0) THEN
          DO 106 J=K,11
            IF (NDRV(J).GT.0) X=0
            IF (NDRV(J).LE.0.AND.X.EQ.1) NDRV(K)=NDRV(K)+61
106       CONTINUE
```

```
 107      WRITE(1,*) +(NDRV(K))
        ELSE
*
*     if there are some driving tests scheduled in hour K then
*     calculate the time between creations as:  60 minutes
*     divided by the number of tests
*
          WRITE(1,*) +(60/NDRV(K))
        ENDIF
 108  CONTINUE
*
*     store the percent custs met by the recep as a fraction,
*     recep idle activity, recep service time, and % reduction
*     in mvr service time as a service time coefficient
*
      WRITE(1,*) +(PERCENT/100)
      WRITE(1,'(A)') IDLE
      WRITE(1,*) TIME
      WRITE(1,*) +(1-COEFF/100)
*
*     write an eof on the file, close it and return to the calling
*     position
*
      ENDFILE 1
      CLOSE(1)
      RETURN
      END
```