NOTE TO USERS

The original manuscript received by UMI contains indistinct, slanted and or light print. All efforts were made to acquire the highest quality manuscript from the author or school. Microfilmed as received.

This reproduction is the best copy available

UMI

AN EXPLORATION OF THE TRADEOFFS BETWEEN REPAIRPERSON STAFFING AND SKILL LEVELS AND THE PROSPECT OF MULTIPLE UNREPAIRED CRITICAL FAULTS

by

GORDON D. BOYER

A Thesis

Submitted to the Faculty of Graduate Studies

In Partial Fulfilment of the Requirements

for the Degree of

Master of Science

Department of Computer Science

University of Manitoba

Winnipeg, Manitoba

© July, 1998

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-32062-6

Canada

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

AN EXPLORATION OF THE TRADEOFFS BETWEEN REPAIRPERSON STAFFING AND SKILL LEVELS AND THE PROSPECT OF MULTIPLE UNREPAIRED CRITICAL FAULTS

BY

GORDON D. BOYER

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Gordon D. Boyer ©1998

Permission has been granted to the Library of The University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film, and to Dissertations Abstracts International to publish an abstract of this thesis/practicum.

The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

ABSTRACT

This thesis consists of a study of Manitoba Telecom Services' Trouble Diagnosis and Repair System (TDRS). The TDRS consists of three levels of service staff responsible for diagnosing and repairing telephone switching equipment faults (troubles). It is modelled as a network flow model with feedback in which service has several probabilistic outcomes and routing is determined by service outcome. There are several sources of heterogeneity including different criticality levels of troubles having different probabilistic distributions of service outcome and the scheduling of different staffing levels over the shifts of the week.

A stochastic logical model of the TDRS is built and then implemented as a stochastic computer simulation program. It is demonstrated that the program produces stable and reasonable measures of the performance of the TDRS. A series of experiments is performed to observe the effect of varying staffing levels on the performance measures of the TDRS, which typifies the sort of questions for which the program may be used to gain insight.

It is also demonstrated that simulation is a useful tool for studying complex systems that may be intractable to analytic methods. In particular the simulation provides insight into the behaviour of the system over daily and weekly periods by providing a graphical interface with dynamic trace outputs.

A utilization analysis of the TDRS is presented that overcomes its non-Markovian nature, in particular its lack of the memoryless property, and its lack of steady state. The utilization analysis has proved to be a useful tool both in verifying the correctness of the computer simulation and in planning experimental work.

TABLE OF CONTENTS

ABSTRACT II

TABLE OF CONTENTS III

ACKNOWLEDGEMENTS VI

TABLE OF FIGURES VIII

TABLE OF TABLES XI

CHAPTER 1: INTRODUCTION 1

1.1: MOTIVATION 1

1.2: OUTLINE OF THE THESIS 8

CHAPTER 2: THE MTS TROUBLE DIAGNOSIS AND REPAIR SYSTEM 11

2.1 SERVICE LEVELS 11

2.2 TROUBLE LEVELS 13

2.3 TROUBLE FLOW 14

2.3.1 Normal Trouble Flow *15*

2.3.2 NOC trouble flow *17*

2.3.3 PNOC trouble flow *18*

2.3.4 DSG trouble flow *18*

2.3.5 Preemption *20*

2.4 TEMPORAL EFFECTS 21

CHAPTER 3: THE LOGICAL MODEL 23

3.1: INTRODUCTION 23

3.2: MODELLING TROUBLES 25

3.3: MODELLING SERVICE LEVELS 26

3.4: MODELLING TROUBLE FLOW 29

3.5: MODELLING TEMPORAL EFFECTS 31

3.5.2: Preemption *32*

3.6: MODELLING STOCHASTIC EXOGENOUS VARIABLES 33

3.6.1: Introduction *33*

3.6.2: Service Time *36*

3.6.3: Trouble Arrivals *38*

3.6.4: Other Exogenous Variables *41*

3.7: MODELLING STOCHASTIC ENDOGENOUS VARIABLES 42

3.7.1: Introduction *42*

3.7.2: Collection of Endogenous Variables *43*

CHAPTER 4: IMPLEMENTATION 45

4.1: INTRODUCTION 45

4.2:CHOICE OF IMPLEMENTATION LANGUAGE 45

4.3: CHOICE OF IMPLEMENTATION PLATFORM 48

4.4: SIMSCRIPT MODELLING CONSTRUCTS 49

4.4.1: The Process *50*

4.4.2: The Set *50*

4.4.3: The Permanent Entity *51*

4.4.4: The Event *52*

4.4.5: The Timing Routine *52*

4.5.6: Define to Means *54*

4.5: IMPLEMENTATION OF THE MODEL USING SIMSCRIPT CONSTRUCTS 54

4.5.1: Implementation of Troubles *54*

4.5.2: Implementation of Service Levels *58*

4.5.3: Implementation of Trouble Flow *61*

4.5.4: Implementation of Temporal Effects *62*

4.6: COLLECTION OF MODEL OUTPUTS 65

4.6.1: SIMSCRIPT Output Collection Statements *65*

4.6.2: Forming Means and Confidence Intervals *66*

4.7: VERIFICATION OF THE IMPLEMENTATION 72

4.7.1: Description of the Trace Output *73*

4.7.1.1: Trace Type I 74

4.7.1.2: Trace Type II 75

4.7.1.2: Trace Type III 75

4.7.2: Verification of Correct Mechanism *76*

4.7.2.1: Trouble Flow in an Uncongested System 76

4.7.2.2: Trouble Flow in a Congested System 78

4.7.2.3: Preemption of a Trouble 79

4.7.2.4: Obtaining Level 2 Staff by Call-out 80

4.7.2.5: Cooperative Service 81

4.7.2.6: Start of a New Shift 82

4.7.3: Verification of Correct Instrumentation *83*

4.8: THE GRAPHICAL INTERFACE 85

4.8.1: The Dynamic Display *86*

4.8.2: Input Forms *88*

CHAPTER 5: DATA REQUIREMENTS AND VALIDATION 92

5.1: INTRODUCTION 92

5.2: THE BASELINE SYSTEM 92

5.2.1:Baseline Model Inputs *95*

5.2.1.1: Trouble Arrivals 95

5.2.1.2: Staff Levels 96

5.2.1.3: Trouble Resolution 97

5.2.1.4: Trouble Routing 98

5.2.1.5: Study Times 98

5.2.1.6: Repair Times 100

5.2.2:Baseline Model Trace Outputs *102*

5.2.2.1: Number of Critical Troubles in System 103

5.2.2.2: Number of Major Troubles in System 104
5.2.2.3: Number of Customer Troubles in System 105
5.2.2.4: Number of Minor Troubles in System 106
5.2.2.5: Troubles Queued for Level 1 Service 107
5.2.2.6: Troubles Queued for Level 2 Service 108
5.2.2.7: Troubles Queued for Technician Service 109
5.2.2.8: Ticketed Troubles 110
5.2.3: Baseline Model Mean Outputs 110
5.2.3.1: Receipt-to-Close Times 111
5.2.3.2: True Queue Lengths 115
5.2.3.3: number of Troubles in the System 117
5.2.3.4: Utilizations 118
5.2.3.5: Call-out Hours 119
5.3 ANALYTIC SOLUTIONS TO THE TDRS 119
5.3.1: Introduction 120
5.3.2: Calculation of Throughputs using the Balance Property 121
5.3.3: Calculation of Utilizations 129
CHAPTER 6: EXPERIMENTAL RESULTS 134
6.1: INTRODUCTION 134
6.2: EXPERIMENTAL PRECISION 134
6.2.1: Variance Reduction 134
6.2.2: Determination of Experimental Conditions 136
6.3: EXPERIMENTAL RESULTS 141
6.3.1: Experimentation with Level 2 (DSG) staffing levels 141
CHAPTER 7: CONCLUSIONS AND FUTURE WORK 147
7.1: CONCLUSIONS 147
7.2: FUTURE WORK 152
LIST OF ACRONYMS 153

BIBLIOGRAPHY 154

ACKNOWLEDGEMENTS

I first wish to thank the people who conceived of this project and provided me the opportunity to work on it. Mr. Barry Gordon, now retired from Manitoba Telecom Services (formerly Manitoba Telephone System) put forth the original research proposal to the Institute of Industrial Mathematical Sciences at the University of Manitoba. Dr. P. N. Shivakumar of the Institute approached my supervisor, Dr. A. N. Arnason of the Department of Computer Science with the hope of finding a student interested in working on developing and implementing the modelling aspect of the project. When it was offered to me I found it to be a very interesting project and accepted.

Secondly I wish to thank those groups that provided me with financial support during my studies. The Natural Sciences and Engineering Research Council of Canada awarded me a Postgraduate Scholarship (PGS1), which was instrumental in my decision to pursue graduate studies. I wish to thank MTS, through the Institute of Industrial Mathematical Sciences for their generous support. Lastly I wish to thank Manitoba Hydro for providing me with employment support while I completed writing this thesis.

I wish to thank the staff of the Network Operations Centre and Provincial Network Operations Centre at MTS for the tour of their facilities and especially Mr. Brent McNeill for his guidance in bringing this work to its current form. Brent provided much valuable feedback during formulation of the logical model and also provided access to MTS documentation describing the operations of both the NOC and PNOC.

Finally to all the people who helped me survive the process of finishing this thesis, I owe a great big thanks. Thank you to my committee members Dr. Randall Peters of Computer Science and Dr. Attahiru Alfa of the Department of Mechanical and Civil

Engineering for their valuable suggestions. Thanks to Tom and Gilbert, for sharing countless lunches and never tiring of my inane technical questions. To fellow students Myrna and Joel, thanks for always being ready to lend a sympathetic ear when it seemed that writing just one more sentence would be the death of me. To my loving wife Marianne, you have been a rock of stability for me during this seemingly never-ending process. Lastly, thanks to my supervisor Neil, from whom I have learned so much. Always the teacher, you gave me just enough guidance to keep my faltering steps moving in the right direction and allowed me the satisfaction of arriving at the end of this journey "on my own".

TABLE OF FIGURES

Figure 2.1: Normal Trouble Flow 16

Figure 2.2: NOC trouble flow 17

Figure 2.3: PNOC trouble flow 18

Figure 2.4: DSG trouble flow 20

Figure 3.1: NOC/PNOC trouble flow 28

Figure 4.1: The Timing Routine 54

Figure 4.2: Trouble Attributes 55

Figure 4.3: Trouble Process Routine 56

Figure 4.4: Generator Process 56

Figure 4.5: Dispatcher Routine 58

Figure 4.6: Exit Routine 59

Figure 4.7: Service Permanent Entity 59

Figure 4.8a: Worker routine part I 62

Figure 4.8b: Worker routine part II 62

Figure 4.9: New shift event 65

Figure 4.10: TALLY and ACCUMULATE statements 66

Figure 4.11: Statistics Collection 73

Figure 4.12: Trace Type I. 75

Figure 4.13: Trouble Flow in an Uncongested System 78

Figure 4.14: Trouble Flow in a Congested System 80

Figure 4.15: Preemption of a Trouble 81

Figure 4.16: Level 2 Call-out 82

Figure 4.17: Cooperative Service 82
Figure 4.18 New Shift: 84
Figure 4.19: Event-actuated Statistics from trace run 85
Figure 4.20: The Dynamic Display 87
Figure 4.21: Trouble Arrival Input Form 90
Figure 4.22: Study Matrix Input Form 91
Figure 4.23: Run Control Input Form 92
Figure 5.1: Baseline Trouble Arrivals 96
Figure 5.2: Baseline Staff Levels 97
Figure 5.3: Baseline Trouble Resolution 99
Figure 5.4: Study Matrix 99
Figure 5.5: Beta Parameters for Trouble Study 101
Figure 5.6: Beta Parameters for Trouble Repair 102
Figure 5.7: Baseline Trace for Critical Troubles in System 104
Figure 5.8: Baseline Trace for Major Troubles in System 105
Figure 5.9: Baseline Trace for Customer Troubles in System 106
Figure 5.10: Baseline Trace for Minor Troubles in System 107
Figure 5.11: Baseline Trace for Troubles Queued for Level 1 Service 108
Figure 5.12: Baseline Trace for Troubles Queued for Level 2 Service 109
Figure 5.13: Baseline Trace for Troubles Queued for technician Service 110
Figure 5.14: Baseline Queue Length Trace for Ticketed Troubles 111
Figure 5.15a: Baseline Mean Receipt-to-close Time and 95% c.i. – Minor troubles 113
Figure 5.15b: Baseline Mean Receipt-to-close Time and 95% c.i. – Customer troubles 114

Figure 5.15c: Baseline Mean Receipt-to-close Time and 95% c.i. – Major troubles..... 115

Figure 5.15d: Baseline Mean Receipt-to-close Time and 95% c.i. – Critical troubles... 116

Figure 5.16: Baseline True Queue Lengths 117

Figure 5.17: Baseline Total Number of Troubles in System 118

Figure 5.18: Baseline Utilizations..... 119

Figure 5.19: Baseline Call-out Hours 120

Figure 5.20: TDRS Throughput model..... 125

Figure 5.21: TDRS second-order Throughput model 127

Figure 5.22: Spreadsheet calculation of Routing Matrix 129

Figure 5.23: Routing Matrix from Simulation..... 129

Figure 5.24a: Analytic Utilizations..... 134

Figure 5.24b: Observed Utilizations 134

Figure 6.1: Observed Critical Troubles in System for 5 run-up lengths..... 139

Figure 6.2a: Log-linear Regression for Confidence Interval half-width – Mean Call-out hours..... 141

Figure 6.2b: Log-linear Regression for Confidence Interval half-width – Mean Receipt-to-Close time for Critical troubles. 142

Figure 6.3: Effect of Level 2 Staffing Scenarios on Call-out Hours..... 145

Figure 6.4: Effect of Level 2 Staffing Scenarios on Utilization. 146

Figure 6.5: Effect of Level 2 Staffing Scenarios on Critical Receipt-to-close time..... 147

TABLE OF TABLES

Table 3.1: TDRS Stochastic Variables 24

Table 3.2: Trouble Attributes 25

Table 3.3: Discrete Exogenous Variables 41

Table 4.1: Simulation Software Categories 45

Table 4.2: Simulation Software Advantages and Disadvantages 46

Table 4.3: Output statistics files 84

Table 4.4: Event-actuated statistics from Spreadsheet 86

Table 6.1: Run-up Experiments 139

Table 6.2: Nominal Staffing Levels 146

CHAPTER 1: INTRODUCTION

1.1: MOTIVATION

The deregulation of the telecommunications industry has created increased competition among the providers of these services, which in turn has created intense pressure for them to find ways to cut costs to maintain profitability, as stated by Regnier and Cameron (1990). Increased competition also means the customer expects a highly reliable service. The objectives of high reliability and low cost conflict; in order to increase the reliability of the network, more money must be spent on such things as better equipment and increased maintenance. Thus the challenge for managers of a telecommunications network is to try to maintain high reliability while at the same time lowering cost.

Manitoba Telecom Services (MTS) was, until 1997, a Crown (i.e. government-owned) Corporation. It is the sole provider of local telephone service to the residents of Manitoba. The lucrative long-distance market has been open to competition for several years, though, so MTS has found itself exposed to the same sort of competitive pressure that has motivated other telecommunications companies to cut costs. One of the areas they examined in the hope of finding some costs savings is the network fault diagnosis and repair process.

During the normal course of operations some network components will fail or otherwise function unsatisfactorily. These failures (called troubles) must be detected, diagnosed and repaired. Of course these troubles have an impact on network reliability, so although an equilibrium of unrepaired troubles exists at most times, it is desirable to

have low numbers of these unrepaired troubles. Having a large number of available repairpersons will certainly help to keep the number of unrepaired troubles low, but it is an unwarranted expense. Providing more training for the personnel who diagnose the troubles should also help to decrease the number of unrepaired troubles by enabling them to repair troubles more quickly, but is itself another expense. So the question the MTS managers want answered is: What is the best mix of personnel staffing levels and training that will result in minimum cost and still maintain the number of unrepaired troubles in the network at an acceptable level?

MTS has several levels of support staff who are responsible for analyzing and resolving troubles. Digital Support Group (DSG) staff have the highest level of training and experience. Network Operations Centre (NOC) staff handle the bulk of the trouble resolution process for the Winnipeg area and Provincial Network Operations Centre (PNOC) are responsible for the province-wide network as well as handling Winnipeg when NOC staff are not available. Both NOC and PNOC call upon DSG staff when needed. Craftspersons perform testing and repair under the direction of NOC/PNOC and DSG staff. In addition to the MTS staff, equipment manufacturers also maintain their own troubleshooting teams who are called upon when critical network components fail.

Trouble reports are received at NOC/PNOC (except for the customer-reported ones, which are received by service staff). The staff analyze the trouble by performing automated diagnostics from the Centre. If the cause of the trouble can be determined, action can be taken to correct it ranging from rebooting switching software to dispatching a Craftsperson to a remote site to repair damaged equipment. If the NOC/PNOC staff cannot determine the cause of the trouble, it will be passed to the DSG staff, and if they

are not available and/or the trouble is serious enough, the manufacturer's troubleshooting staff will also be called.

Troubles are prioritized into classes according to their potential for causing disrupted or degraded service. They can be classified in decreasing order of criticality as:

- critical troubles
- major troubles
- customer-reported troubles
- minor troubles

Critical troubles are ones that result in immediate loss or degradation of service to customers, while major troubles have the potential to do so; they are given immediate attention. The minor and customer-reported troubles are given lower priority and must wait for service until higher-priority troubles are resolved. Accumulations of the lower priority troubles are not desirable, though, since they may be symptomatic of a fault that could disrupt service if left unresolved.

Because troubles are classified according to their criticality, the notion of maintaining the number of unrepaired troubles at an acceptable level must take into account the criticality of the troubles. Having a large number of critical troubles awaiting service is obviously not acceptable, whereas having a large number of minor troubles awaiting service may be. One way to take this into account is to keep track of the numbers of each criticality class separately, and to choose threshold values for each class. If the number of troubles in a certain criticality class exceeds its threshold, then the level of unrepaired troubles is said to be unacceptable.

How can the managers develop and evaluate staffing strategies, and more importantly, how can they determine their effect on network reliability? Law and McComas (1996) describe three possible approaches:

- experimentation with the network itself
- use mathematical methods to develop an analytic solution
- use a simulation model

Experimentation with the network itself is not feasible, since it could lead to disruption of customer service. An analytic solution gives exact answers to the questions of interest and has been used for years in the study of telecommunications networks. However, developing an analytic solution for a complex system is difficult and requires a high degree of mathematical sophistication. Analytic solutions are most easily derived for a homogeneous system, i.e. one in which exogenous inputs such as arrival rates and service delays are constant. Such systems, when run for a sufficient length of time, reach steady state, at which point the distributions of the performance measures remain constant. A heterogeneous system such as the TDRS, in which inputs vary with time, never reaches steady state, meaning that performance measure distributions do not become constant but vary with time. Finding analytic solutions to these transient distributions is much harder than the steady state case, but are of interest to the network manager who wants to know of the potential for large accumulations of failures at certain times.

The limitations of the other two methods make the use of a simulation model an attractive alternative for network analysis. Carson (1996) describes several stages in the development and use of a simulation model:

- *Problem formulation*: what should the simulation model tell us?

- *Model design*: develop a logical model of the elements of interest in the system and their interactions.
- *Data collection*: identify and collect the data needed as input to the model.
- *Model development*: implement the logical model in a programming language.
- *Verification and validation*: does the model work as intended? Do its results correspond to the actual system?
- *Experimentation and analysis*: use the model to help provide insight to the questions of interest.

With a simulation model, various scenarios can be evaluated without the fear of causing disruptions to the network. Law and McComas (1996) state that while a simulation model does not give exact answers for performance measures, it does give estimates and confidence intervals on those estimates, and simulation analysis is applicable to systems of almost any level of complexity.

Given simulation as the technique of choice to investigate this problem, the first task is to abstract the essential elements and their interactions from the actual system into a logical model. Because troubles are prioritized according to their criticality, with lower priority troubles waiting for service until higher priority ones are repaired, a queuing model using priority queues is natural. Each level of service will have its own priority queue, with troubles being serviced in first-in-first-out (FIFO) order.

When formulating a logical model, the modeller must decide which aspects of the system being modelled are important to the questions being asked, and which are not. The model should include all the important aspects but none of the unimportant ones. One of the characterizing aspects of a network such as a telephone network is its

topology, i.e. the interconnections between the various parts of the network. Network topology is usually modelled as a *graph*. A graph consists of a set of nodes and a set of edges. Nodes represent objects of interest in the network being modelled and edges represent connections between two nodes. In a telephone network nodes represent switches and edges represent transport (i.e. the cable or microwave connections that carry the calls from switch to switch).

Topology seems to be an essential aspect of any network; nonetheless, it is not represented in the logical model of the diagnosis and repair process. There are two reasons for this. Firstly the repair process is not concerned with how the network is connected; its goal is to repair troubles as quickly as possible, with no priority given to any particular areas of the network. Secondly, advances in technology have enabled the diagnosis and repair process to be centralized, which gives rise to a logical model where troubles "arrive" at a central operations Centre where they receive service.

Fagerstrom and Healy (1993) note that with the advent of digital switching technology, many telephone central offices now contain a single large digital switch. All aspects of the switch's performance can be monitored from a single remote location, where network personnel can request diagnostic information from the switch or perform tests on individual lines or switch components. Corrective measures can even be performed remotely, such as reinitialization of switch software.

Fagerstrom and Healy (1993) note that network reliability depends on three factors:

- the reliability of individual components
- the repair process

- the topology of the network

The focus of this investigation is on the effect of the repair process on the reliability of a telephone network. In this model of the network the topology is not modelled; the network is considered to consist of a collection of network elements, each of which is capable of failing and being repaired. This may seem to be an oversimplification, but it is justified for two reasons. Firstly, the network being modelled is a metropolitan area network, the Winnipeg area of Manitoba Telecom Services (MTS) and the monitoring of network faults is done from one central location rather than at the central offices housing the switches. This makes the process of diagnosing and repairing faults independent of the topology of the network. Secondly, the repair process itself is not concerned with the network topology; rather, the aim of the repair process is to repair faults as quickly as possible while making efficient use of manpower. Thus the model adopted here is one in which network faults (called *troubles*) "arrive" at the central monitoring location where they are diagnosed and appropriate action taken.

As stated earlier the network operator is faced with a tradeoff between network reliability and the cost of the repair process. Some measure of reliability is needed in order to answer the question "How reliable is reliable enough?" Typically reliability measures for telephone networks are expressed from the customer's point of view. Fagerstrom and Healy (1993) propose two measures of telephone network reliability, namely availability (the probability the network is available when a user at a random time attempts to make a call) and the probability a customer does not experience an outage longer than 5 minutes in a year.

This model requires some different measures of reliability than the ones discussed above. Troubles may or may not cause disruptions in service, which was what the two quantities above were concerned with. Since the repair process is concerned with clearing troubles as quickly as possible, measuring the average time to clear troubles from the network, broken down by criticality level, is a good measure of the efficiency of the repair process. Chen *et al* (1988) report this measure, which they call *receipt-to-close time*, in their simulation study of one aspect of the telephone network repair process. Although it is only an indirect measure of the reliability of the network, one would expect that as receipt-to-close time increases, network reliability decreases.

The second measure reported by Chen *et al* (1988) is queue lengths of troubles awaiting service. Long queues of troubles awaiting service mean that there are more network elements not functioning properly, increasing the probability that a customer will experience a disruption or degradation in service. Observing the behaviour of the queues over time can also reveal the sort of transient effects mentioned earlier, such as certain times of the day or week when queues are large. These sorts of observations are difficult to obtain by mean value analysis.

1.2: Outline of the Thesis

Chapter two is a discussion those aspects of the Manitoba Telecom Services fault diagnosis and repair process that are important to the development of a logical model of the process. It begins with a description of the three skill levels of service personnel and the single pool of repair craftspersons. Next is described the four criticality levels of troubles. Following is a description of the general trouble flow through the process and

the modifications to the general flow for various combinations of trouble criticality level and skill level of available personnel. Lastly is a description of temporal factors.

Chapter three describes the logical model developed from the TDRS described in Chapter two. First is a description of the modelling of the service levels and the craftperson pool. Next is a description of the modelling of troubles. Following is a description of the modelling of trouble flow using an open network queueing model. Following is a description of the modelling of temporal effects. This is followed by a discussion of the exogenous variables in the model, i.e. the service time distributions and other delays that are determined *a priori*. This discussion will include the selection of appropriate distributions and simplifying assumptions.

Chapter four is a discussion of the implementation of the logical model described in Chapter three. It begins with a discussion of the choice of SIMSCRIPT II.5 as the implementation language and UNIX as the implementation platform. Next is a description of the SIMSCRIPT modelling constructs used in the implementation. Following is a description of the implementation of the troubles, service levels, trouble flow and temporal effects. Next is a discussion of the performance measures collected by the implementation. This discussion will include a description of which measures are collected and the use of independent replications to obtain estimates of the mean and confidence intervals for the performance measures. Next is a discussion of the verification procedures for the implementation. The types of trace output produced by the implementation are first described, followed by a description of the use of the trace output to verify the implementation's mechanism. Finally the graphical interface is described.

Chapter five presents a set of model inputs that produces a stable system called the baseline system. A detailed analysis of the system's dynamic and statistical outputs demonstrates that the behaviour of the simulation program is typical of what can be expected when experimenting with real-world data. An analytic solution is developed that yields utilizations for the service levels and it is demonstrated that its answers agree with the simulation output.

Chapter six presents experimental results. First is a discussion of factors that affect the precision of the results produced by the simulation including the use of the variance reduction technique of common random numbers and the choice of parameters for the independent replication method of producing independent observations. Following is a discussion of the results of experiments to analyze the effect of staffing changes on the system. It is shown that staffing changes can both improve staff utilization and improve network reliability as measured by decreased receipt-to-close time.

Chapter seven presents conclusions and a discussion of possible future work.

CHAPTER 2: THE MTS TROUBLE DIAGNOSIS AND REPAIR SYSTEM

This chapter describes the aspects of Manitoba Telecom Services' Trouble Diagnosis and Repair System (TDRS) that are important with respect to the logical model from which the simulation program is designed. First the service levels are described. Next the trouble levels are described. Then the trouble flow is described. Finally temporal effects are described.

2.1 SERVICE LEVELS

There are three levels of repair personnel available to diagnose and correct troubles in the MTS network. In addition there is a separate pool of technicians who repair hardware problems. Generally all troubles are initially diagnosed by level 1 personnel, who work to determine the cause of the trouble, and then clear the trouble themselves or dispatch a technician to repair a hardware problem. If they cannot determine the cause, the trouble is escalated to the next level of support, where the process is repeated. The third level of support, if it is required, is actually provided by the equipment manufacturers. High-priority troubles are immediately elevated to level 2 support. A description of the service levels follows.

Level 1 support (NOC/PNOC): Level 1 support is provided by two areas: the Network Operations Centre (NOC) and the Provincial Network Operations Centre (PNOC). NOC monitors the Winnipeg area network during business hours, 8 a.m. to 4:30 p.m. Monday to Friday. PNOC monitors the province-wide network on a 24-hour basis, and provides level 1 support for the Winnipeg area during off-business hours. Level 1 support handles the bulk of the trouble repair process, i.e. they handle routine troubles

and are also involved with more serious troubles. Some of the duties they perform include:

- determining the severity of an incoming trouble
- dispatching technicians to unmanned sites
- creating and updating an Activity Call Report (ACR)
- provide ongoing assistance to technical and Level 2 as required

Level 2 Support *(*DSG*):* The Digital Support Group (DSG) consists of more experienced personnel who are available during business hours and who also may be called in during off-business hours if a service-threatening situation arises. These people are experts in the digital switching equipment used by MTS. Some of the duties they perform include:

- provide assistance to all personnel working on digital switch equipment
- maintaining a database of troubles and actions taken
- working with Level 3 support to resolve troubles
- training of Level 1 personnel

Level 3 Support (OSO): All manufacturers of digital switching equipment provide emergency technical assistance, referred to here as an Outside Support Organization (OSO). Some examples include Northern Telecom's Emergency Technical Assistance Service (ETAS) and Stromberg-Carlson's Assistance Team (SCAT). The decision to call in Level 3 support is based on such factors as problem severity, availability of Level 2 support and elapsed time. Although Level 3 support is usually called when severe, service-threatening troubles arise, they can be consulted on less

severe problems. These typically involve switching software, and the suggested repair may be to wait until the next software upgrade is ready.

Technician (Craft): A pool of technicians, or Craftspersons, handles all hardware repairs. Central offices in the Winnipeg area are staffed by Craft on a 24-hour basis; remote locations are usually unmanned however.

2.2 Trouble Levels

Incoming troubles are assigned a criticality level according to their potential for causing a disruption in service. In this model there are four criticality levels, with 1 being the least critical and 4 being the most critical. Level 3 and 4 troubles always receive immediate service (providing personnel are available) while Level 1 and 2 troubles may be "ticketed" for service on the next business day. A description of the criticality levels follows.

Level 1 troubles (NS): Level 1 troubles are also called Non-Service Affecting (NS). These are troubles, which of themselves do not affect customer service, but may indicate some more serious trouble in the future. Thus, while action on these troubles may be deferred without affecting customer service, they cannot be left indefinitely because they could develop into more serious troubles. Examples of Level 1 troubles include non-service affecting software inconsistencies, peripheral equipment diagnostic failures and test equipment failures for which a backup is available.

Level 2 troubles (S1/S2): Level 2 troubles include Service Affecting (S1) and Intermittently Service Affecting (S2). Also added to this category are Customer-reported

troubles. These troubles affect service to only one or a small number of customers; alternately they may intermittently affect service to a class of customers. While prompt response to these troubles is not critical, an accumulation of Level 2 troubles is not desirable. Examples of Level 2 troubles include warm switch reinitializations, magnetic tape/disk drive problems, and any hardware or software problems resulting in lost or degraded service to one or several customers.

Level 3 troubles (E2): Level 3 troubles are troubles that are threatening Potential Degradation and/or Outage (E2). Thus while they may not have caused a disruption in service, they could potentially cause much more disruption than Level 2 troubles. These troubles require immediate attention until the trouble is resolved. Examples of Level 3 troubles include inoperative essential standby equipment, essential maintenance terminals out of service and an excessive number of redundant control units out of service.

Level 4 troubles (E1): Level 4 troubles are troubles that cause Degradation and/or Outage (E1). They are the most serious and least common troubles, and cause lost or degraded service to a large number of customers. These troubles require immediate attention until the trouble is resolved. Examples of Level 4 troubles include a switch ceasing operation, multiple switch restarts or reinitializations and an Emergency Services trunk group (e.g. 911) out of service.

2.3 TROUBLE FLOW

This section begins with a description of the normal flow of a trouble through the TDRS. Following is a discussion of factors that alter the normal flow and discussions of

the trouble flows for NOC, PNOC and DSG. Following that is a description of how this flow may be altered by preemption caused by an arriving high priority trouble.

2.3.1 Normal Trouble Flow

The following describes the normal flow for resolution of a trouble. It is first analyzed by NOC/PNOC, where there are three possible outcomes: resolution of the trouble, determination of a hardware fault or failure to resolve the trouble. In the first case, the trouble is cleared and leaves the system. In the second case, the trouble is passed onto Craft for hardware repairs. In the third case the trouble is referred to DSG for analysis. Troubles referred to DSG follow the same flow, except unresolved troubles are referred to OSO. When a trouble is passed onto Craft, there are two possible outcomes: the hardware fault is repaired, in which case the trouble leaves the system, or the fault is not repaired, in which case the trouble is referred back to the previous support level for further analysis. Figure 2.1 describes this normal trouble flow.

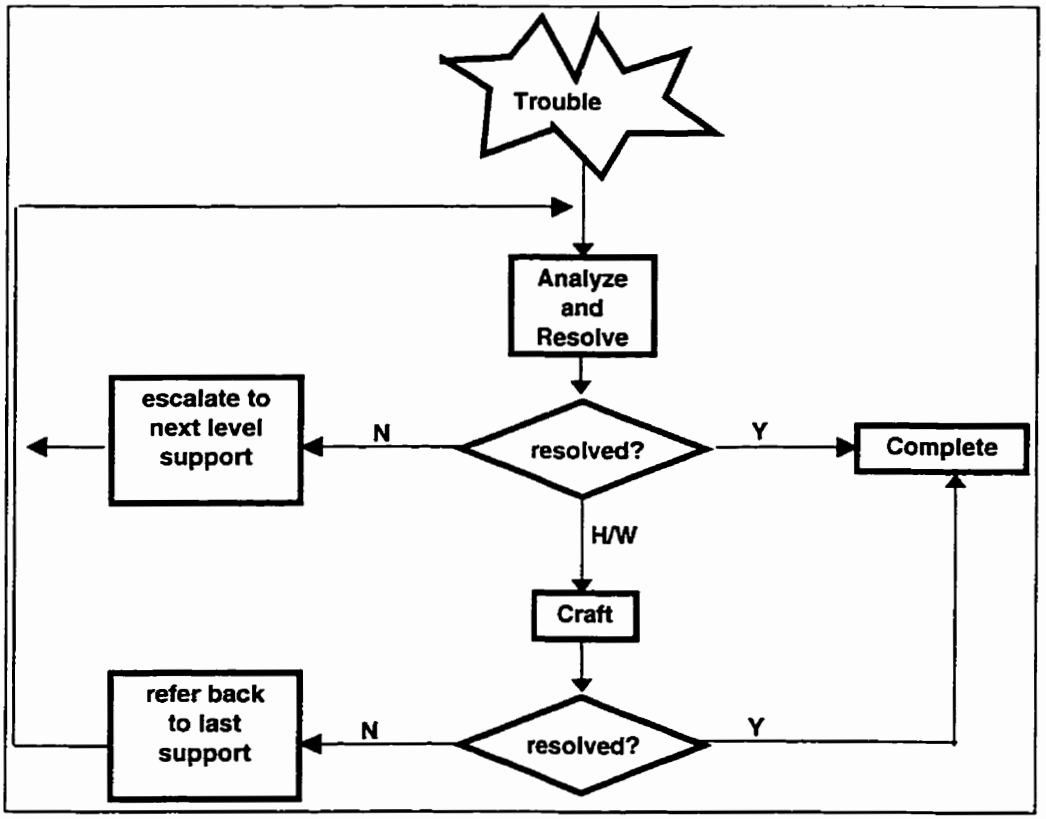

Figure 2.1: Normal Trouble Flow.

This normal trouble flow can be modified by bypassing the Analyze and Resolve stage and directly escalating to the next level support. The trouble flow will be modified according to three factors:

1. support level involved
2. criticality level of trouble
3. time of occurrence (i.e. business or off-business hours)

Discussions of how the specific trouble flows for NOC, PNOC and DSG differ from the normal flow follow.

2.3.2 NOC TROUBLE FLOW

Figure 2.2 details the trouble flow for NOC staff. The only difference from the normal trouble flow is that level 4 (E1) troubles are referred immediately to the next level support (DSG). Since both NOC and DSG work during business hours, and in fact are located in the same office, this is a reasonable modification. The stage labelled "Trouble will be handled locally" encompasses the *Analyze and Resolve* stage of the normal trouble flow, and the decision "Cleared in reasonable length of time?" encompasses both the *Resolved* and *Hardware* outcomes in the "Y" branch. Note also that for level 1 and 2 troubles the length of time for the *Analyze and Resolve* stage is not specified (reasonable length of time) but for level 3 troubles this stage is limited to 30 minutes.

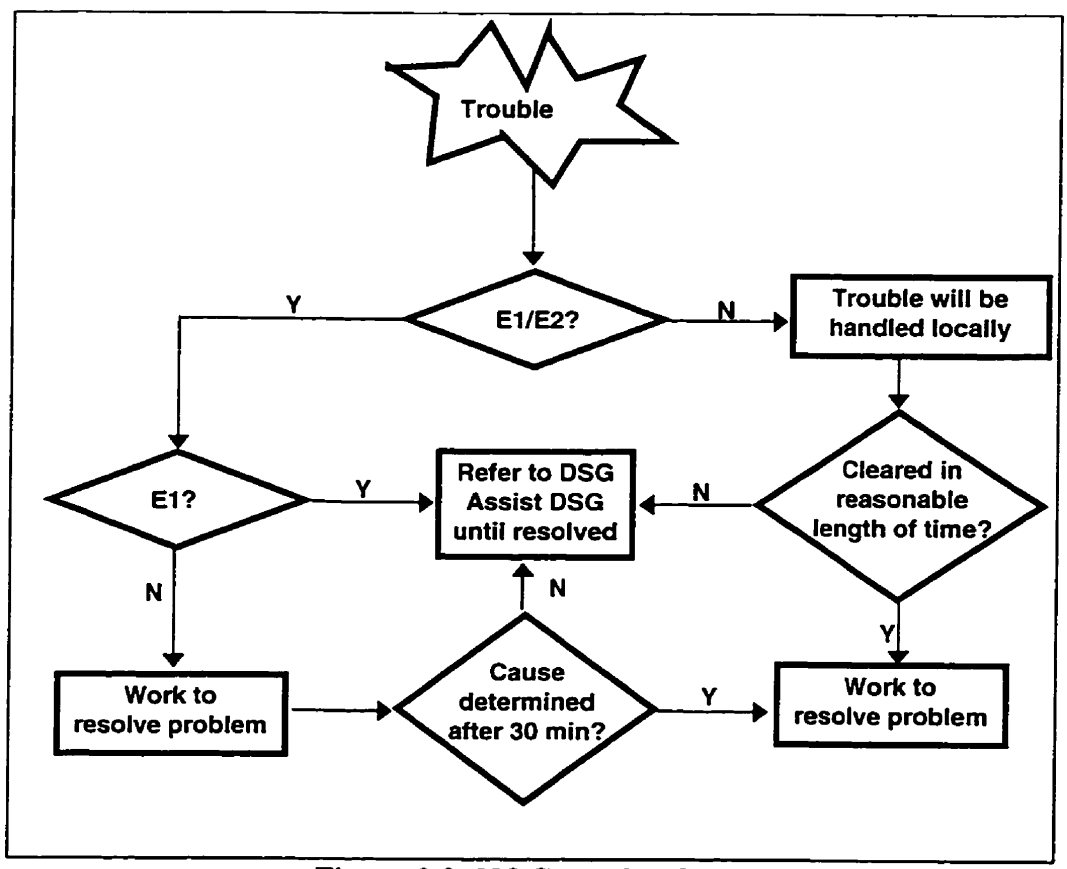

Figure 2.2: NOC trouble flow.

2.3.3 PNOC TROUBLE FLOW

Figure 2.3 details the trouble flow for PNOC staff. PNOC provides level 1 support during non-business hours when DSG support may not be available, so this results in a modification to the normal trouble flow so that level 4 troubles will be referred directly to OSO support if DSG cannot be obtained. The other modification to note is that PNOC generally do not deal with level 1 troubles; instead they are "ticketed" for service during business hours.

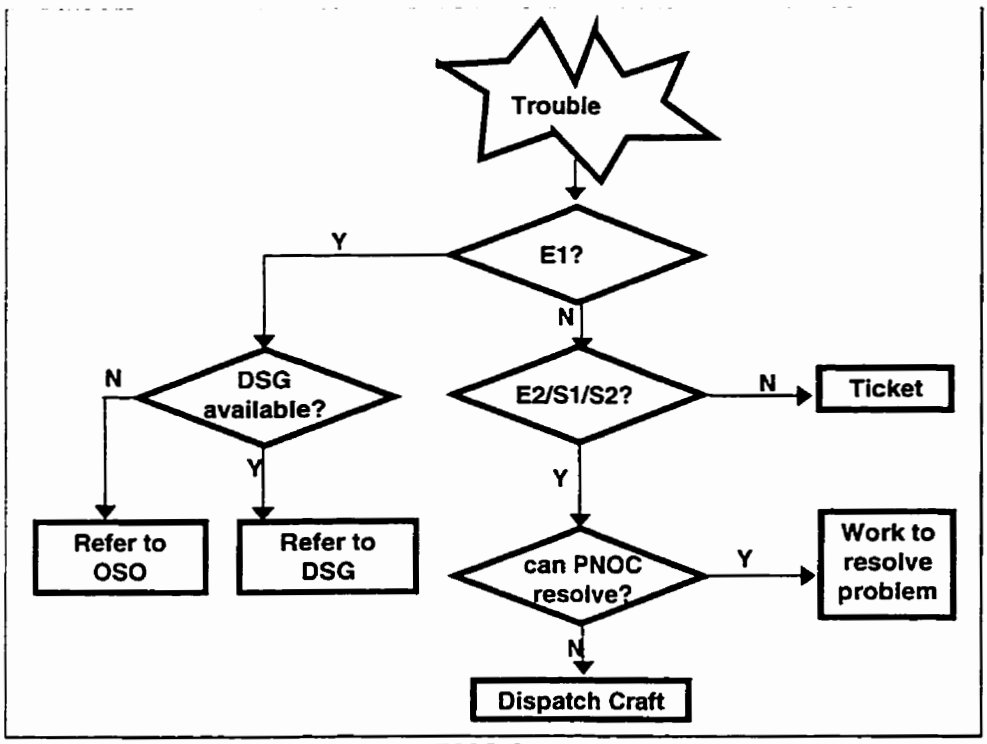

Figure 2.3: PNOC trouble flow.

2.3.4 DSG TROUBLE FLOW

Figure 2.4 details the trouble flow for DSG staff. Since all troubles are received initially by level 1 (NOC/PNOC) staff, DSG staff do not respond directly to a trouble, but rather to a request for assistance from level 1 staff. DSG staff work business hours but are

available for level 3 and 4 troubles on an emergency call-out basis during non-business hours; this results in two branches in the flow diagram. This results in level 3 and 4 troubles being handled differently during business hours and non-business hours. During business hours DSG will contact OSO immediately for a level 4 trouble, but will analyze the trouble first before contacting OSO during non-business hours. For level 3 troubles the analysis phase is longer during non-business hours than during business hours (60 min. *vs.* 30 min.). For level 1 and 2 troubles no specific times are set, and escalation to OSO occurs on a consultation basis.

DSG are normally called on to provide expert assistance to other staff and this is reflected in the flow diagram in such phrases as "Work with tech to determine cause". While this may involve the two staff working continuously together, it can also involve the DSG staff filling a consultative role with only intermittent involvement with other staff.

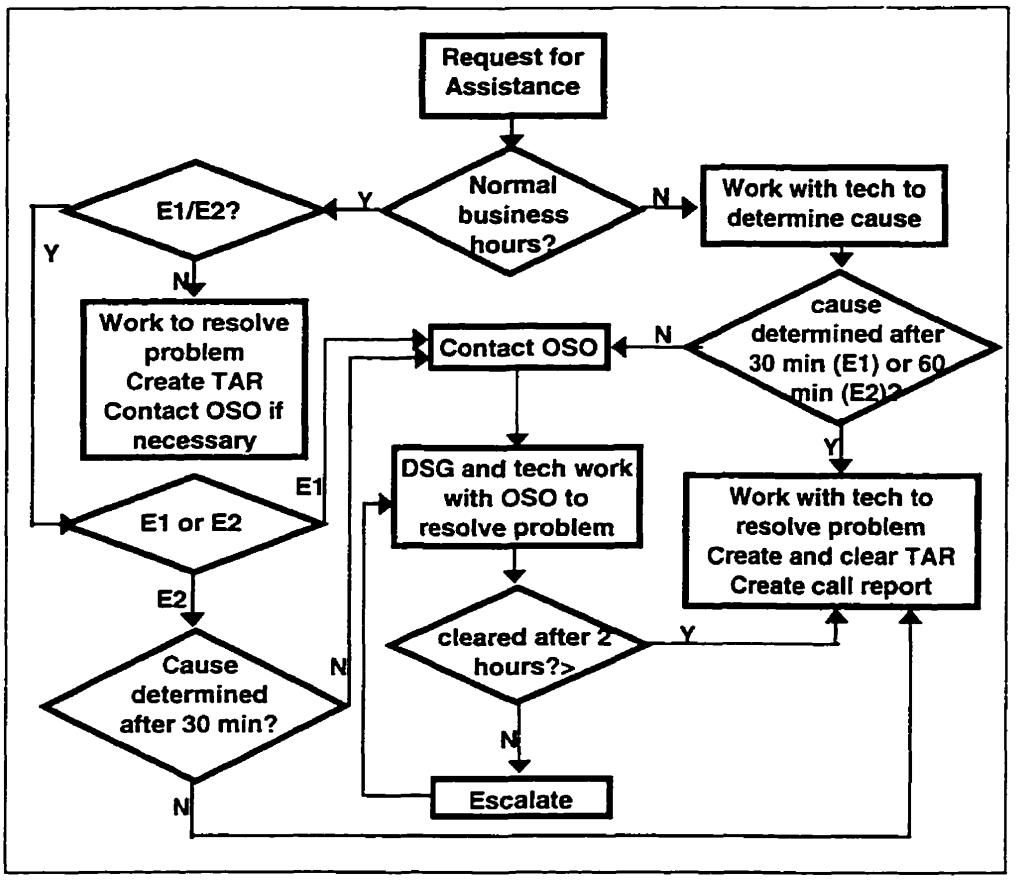

Figure 2.4: DSG trouble flow.

2.3.5 PREEMPTION

As noted in section 2.2 above, level 3 and 4 troubles always receive immediate service, either when they arrive in the system or when they move to a new service level according to the normal trouble flow. If such a trouble requires service at a time when all the support staff are busy with troubles of lower criticality, work on one of these lower criticality troubles will have to be delayed in order to free up staff to work on the higher criticality trouble. This is called *preemption* of the lower criticality trouble. Once the higher priority trouble is cleared, work can resume on the preempted trouble.

2.4 Temporal Effects

The behaviour of the TDRS is not homogeneous with respect to time. There are both time-of-day effects and day-of-week effects. These effects are largely attributable to variations in staffing levels that occur on a daily and weekly basis. For example, both DSG and NOC are staffed during business hours (8:00 a.m. to 4:30 p.m. Monday to Friday) only. Consequently there is less staff available during non-business hours, so accumulations of troubles occur during the evenings and on weekends. These accumulations are then reduced when staff is available during business hours, although it may take several days for the weekend accumulation to be reduced.

If a level 3 or 4 trouble occurs during off-business hours an attempt will be made to obtain DSG staff on a call-out basis; however, there is no guarantee this will be successful. The likelihood of obtaining on-call staff varies with time-of-day and day-of-week. This on-call staff will work on the level 3 or 4 trouble until it is resolved and then return home, so their presence will not affect the accumulation of troubles during off-hours.

Staff who are still working on a trouble when the end of their shift comes will generally remain working on it until the service is complete. This results in more than the normal number of staff working for a brief period at the beginning of a new shift. Exceptions to this policy would be made if the trouble requires a long period of further service; in this case the staff working on the trouble on the old shift would familiarize the staff taking over the work before leaving.

The arrival rate of the troubles themselves is typically free from temporal effects. The reliability of digital switching equipment is such that there is no corresponding

increase in non-customer-reported trouble arrivals during times of increased network usage. Customer-reported troubles will increase during business hours since this is the time when most customers will notice them. This does not have a significant impact on the overall arrival rate since the major source for trouble arrivals is network monitoring activity as reported by Chen *et al* (1988).

CHAPTER 3: THE LOGICAL MODEL

3.1: INTRODUCTION

This chapter contains a discussion of the logical model developed to represent the TDRS. Troubles are modelled as entities requiring service in the model, and the service levels are the providers of this service. Service is provided in three stages: the trouble is studied to attempt to determine the cause, work is done to resolve the trouble and finally the trouble is routed to another service level or it leaves the system. A queuing model is used to represent the system, because if a trouble requires service and there is no staff free to provide it, the trouble must wait in a queue until staff becomes free. Trouble flow in the model is the movement of troubles from one service level to another and eventually leaving the model. Temporal effects are caused by varying staff strength levels during the day.

The events that occur in this model, such as the arrival of the next trouble or the length of time a particular service will take are probabilistic or *stochastic* in nature, meaning it is not known beforehand what particular sequence of events will occur or what their outcomes will be. This probabilistic nature must be represented in the model. There are several stochastic variables in this model, which can be divided into two groups: exogenous and endogenous. Exogenous variables are those that originate from outside the model and are not affected by any scenario represented in the model. These include the trouble arrival rate, the mixture of trouble criticality levels, trouble service times, the outcome of trouble service and the chance of obtaining service personnel on a call-out basis. Exogenous variables can be thought of as the *inputs* to the model.

Endogenous variables are random variables that are affected by the particular scenario represented by the model and include the performance measures, which will be used to evaluate a particular scenario. These include queue lengths, staff utilization and receipt-to-close time. Endogenous variables can be thought of as the *outputs* of the model.

Table 3.1 below summarizes the stochastic variables in the TDRS. Continuous variables may take any value within a specified range. Discrete variables may take on a value from a finite set of values, each with its associated probability of occurrence. Bivariate variables have exactly two possible values (e.g. success/failure) and multivariate variables have more than two possible values. True service queues contain those troubles waiting for service but not those receiving service. Mean troubles in system include both troubles waiting for service and those receiving service.

Variable	Type	Description
Trouble arrival rate	Exogenous	Continuous
Trouble criticality mix	Exogenous	Discrete multivariate
Trouble study time	Exogenous	Continuous
Trouble repair time	Exogenous	Continuous
Trouble service outcome	Exogenous	Discrete multivariate
Call-out success probability	Exogenous	Discrete bivariate
Mean true queues (by service)	Endogenous	Continuous
Mean troubles-in-system (by criticality)	Endogenous	Continuous
Utilization	Endogenous	Continuous
Receipt-to-close time	Endogenous	Continuous

Table 3.1: TDRS Stochastic Variables

In this chapter the term *delay* refers to any passage of time with an identifiable beginning and end. A delay may result from a trouble arriving in the system and waiting for service staff to become available, from staff working on a trouble, etc. The term *server* is used to describe any provider of a service; in this case level 1 and level 2 service staff and the craftperson pool. The servers provide a *service* to the troubles.

3.2: Modelling Troubles

Troubles are modelled identically as entities requiring service in the system. The criticality level of the trouble is modelled as an attribute. The fault causing the trouble is not modelled; i.e. it is not determined when the trouble arrives in the system whether it is caused by a hardware or software problem or other type of fault. Whether or not the trouble is caused by a hardware fault is determined as it receives its service.

In addition to the criticality level, the trouble possesses other attributes needed to determine routing and collect statistics. They are summarized in Table 3.2 below.

attribute	description
id number	unique number to identify the trouble
entry time	entry time into the system - for statistics gathering
criticality level	controls trouble flow
status	which service level is required
shift begun	which shift the current service was begun - used for temporal effects

Table 3.2: Trouble Attributes

3.3: Modelling Service Levels

The TDRS has three levels of support as well as the craftperson pool. The logical model I have developed models only the first two levels and the craftperson pool; level 3 support (OSO) is not modelled. It is not modelled because level 3 support is provided by the equipment manufacturers and is thus not directly under the control of MTS management. Escalation of a critical or major trouble to level 3 support is an exceptional occurrence. Minor and customer troubles may also be referred to level 3 support, but this is usually done in the form of a query about some aspect of the operation of the equipment, not in the context of a service-affecting problem. After discussion with MTS managers it was decided that the additional complexity of modelling level 3 support would not improve the model's ability to answer questions about the day-to-day operations of the repair process.

Level 1 and level 2 support and the craftperson pool are modelled identically as three resources providing service to incoming troubles. Each resource provides service in three stages:

1. Study the trouble.
2. Work on the trouble.
3. Route to another resource or clear the trouble.

The first stage, studying the trouble, models the initial phase of the repair process in which the staff attempts to determine the cause of the trouble and the necessary action to resolve it. This stage has three possible outcomes:

- The trouble can be resolved immediately.

- The trouble is caused by a hardware fault.
- The trouble is unresolved.

Trouble study may or may not result in a delay. If the trouble is to be studied, then a delay will occur. However, trouble study may also simply be a decision to escalate the trouble to a higher service level or to ticket it, which does not result in a delay.

The second stage, working on the trouble, is performed only if the outcome of studying the trouble is that the trouble can be resolved immediately (i). The other two outcomes of the study stage will result in service proceeding immediately to stage 3, routing the trouble to another resource for further analysis or hardware repair. If the result of studying the trouble is that it can be resolved immediately, then the service in stage two results in a delay while the trouble is resolved. After the delay is complete the trouble proceeds to stage three where it is cleared.

The third stage of service routes the trouble to the appropriate queue for the next service required, or marks it as cleared as appropriate. Once marked as cleared, the trouble leaves the system. Since level 3 support is not modelled, those troubles requiring level 3 support also leave the system. This has the effect of biasing receipt-to-close times for those troubles requiring level 3 service low, since in reality they do not leave the system but they do leave the system in the logical model. Utilizations are also biased low since staff are still involved in the service of troubles receiving level 3 service.

The situation where staff from two service levels work together on a trouble is not explicitly modelled. Rather, a trouble may return back to a previous service level, say from level 2 to level 1 or craftperson to level 2, in order to approximate the type of intermittent consultation that occurs when two service levels work on a trouble. Since the

model never has two staff actually working on a trouble simultaneously, this approximation will bias low the utilizations of the staff involved.

Figure 3.1 shows the modelling of level 1 service, which includes both NOC and PNOC workers. Note that the study phase is bypassed for level 4 (critical) troubles at all times, as well as for level 1 (minor) troubles during non-business hours. Minor and customer troubles are ticketed, which means that they are set aside until business hours.

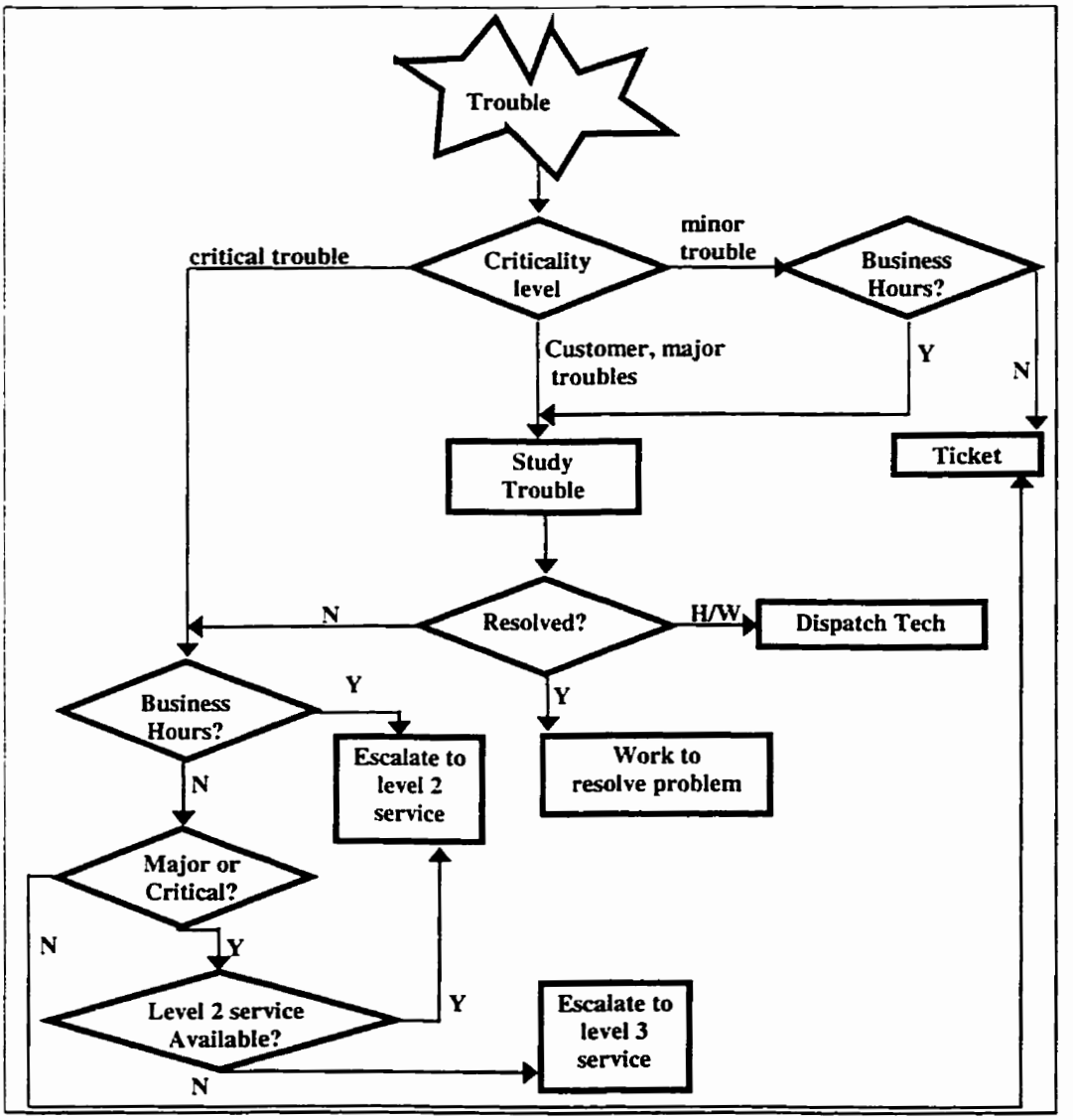

Figure 3.1: NOC/PNOC trouble flow.

3.4: Modelling Trouble Flow

A queuing model is used to model the flow of troubles in the logical model. Each of the level 1, level 2 and craftperson service levels has a queue into which troubles requiring service are entered. The queues are priority queues, which means that troubles are entered into the queue according to their criticality level; i.e. all critical troubles are at the front of the queue, followed by major troubles, then customer troubles and finally minor troubles. Within each criticality class troubles are entered in First-In-First-Out (FIFO) order. Troubles arriving into the system have a criticality level associated with them so that they can be prioritized at the level 1 queue without having been seen by staff.

Trouble flow in the system is summarized in Figure 3.2. Troubles entering the system are first placed in the level 1 queue. When a level 1 server becomes free, the trouble at the head of the level 1 queue is removed and begins its service. Depending on the outcome of the service, the trouble can be routed to the level 2 queue, the craftperson queue, or be cleared and leave the system. A trouble being served by a level 2 server can be routed to the craftperson queue, be cleared or be escalated to level 3 support, in which case it also leaves the system. A trouble being served by a craftperson server can be routed to either the level 1 or level 2 queues (whichever one had provided the most recent service before the craftperson) or is cleared.

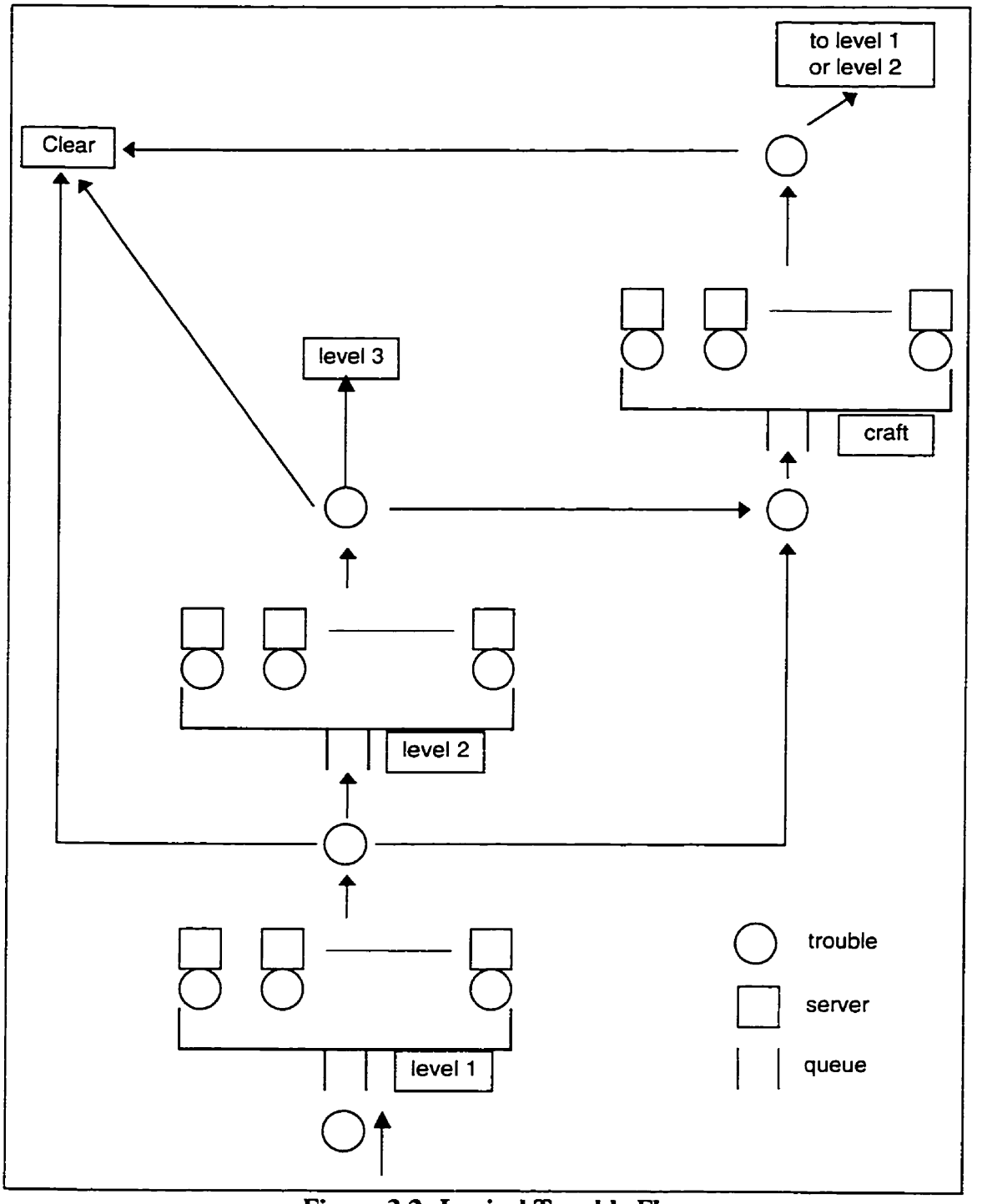

Figure 3.2: Logical Trouble Flow

3.5: MODELLING TEMPORAL EFFECTS

Temporal effects occur on both a daily and weekly basis. They are caused by changing staff levels during the three weekday shifts and reduced staff on the weekend shifts. In the logical model shift changes are accomplished by updating the number of available (i.e. non-busy) servers to equal the new shift strength at shift-change time. This is equivalent to all the non-busy staff leaving at quitting time and the new shift's staff arriving. Note that the servers that are busy at shift-change time remain working on the trouble until its service is completed, which is exactly what happens in the MTS repair process.

If there are troubles waiting for service at shift-change time, they immediately begin service with the arriving servers. If the system is congested with troubles this can result in more than the shift strength of servers working (all the new servers and those continuing a service from the previous shift) for some time after the beginning of a shift.

Generally, level 2 staff work during business hours only. However, critical and major troubles may be escalated from level 1 to level 2 service immediately. If such a trouble arrives on a weekend or during off-hours, it would have to wait some time before it could obtain service from a level 2 server. When a critical or major trouble requires level 2 service at a time when none are available, the model will attempt to provide the level 2 service on a call-out basis. Obtaining level 2 service during off-hours is probabilistic; if a level 2 server is obtained, service begins immediately, and when it is finished, the level 2 server leaves the model. If a level 2 server is not obtained, the trouble must wait until the next shift change. If the shift is a business hours shift then level 2 servers will be available, else the probabilistic process is repeated. The probability of

obtaining off-hours level 2 service depends on both the criticality of the trouble and the current shift.

A simplifying assumption is made regarding the arrival of customer troubles. These are troubles that are reported to MTS by its customers, and as such they are more likely to be received during business hours. It is assumed that the arrival distribution for customer troubles is homogeneous with respect to time in the logical model. The major source of heterogeneity for trouble arrivals is customer troubles, which tend to occur in bunches at the start of business days or as the result of a common fault such as a cable being cut. Chen *et al* (1988) report that customer-reported troubles make up only a small fraction of all trouble reports, the majority of them resulting from routine diagnostics and alarms. After consultation with MTS it was felt that this assumption would not significantly affect the logical model.

3.5.2: PREEMPTION

If a Major or Critical trouble requires service at a time when all the servers are busy with troubles of lower criticality, one of these lower criticality troubles will have to be preempted in order to free up staff to work on the higher criticality trouble. Preemption is implemented in the logical model as follows: a Major or Critical trouble requiring service and finding no available staff will cause the preemption of service for the minor or customer trouble whose service was started most recently. The preempted trouble will return to the appropriate queue and will be placed before all other troubles with the same priority. For example, a preempted customer trouble will be placed in the queue ahead of all other customer and minor troubles, but behind the Critical and Major

troubles. If more than one trouble of the same criticality level is preempted, they are returned to the queue in FIFO order, but ahead of non-preempted troubles with the same criticality. A Critical trouble may preempt the service of a Major trouble if there are no minor or customer troubles to preempt.

Preemption in the TDRS is described as preemptive-resume, i.e. the total service delay of a trouble is not affected by preemption. When a preempted trouble resumes service, its service time is reduced by the amount received before preemption.

3.6: Modelling Stochastic Exogenous Variables

3.6.1: Introduction

In the real-world system there are variables, such as the arrival rate of troubles, which are independent of the way the system operates. These variables originate from outside the system and are termed *exogenous* variables. Typically these variables are the input for the logical model.

Random or *stochastic* variables are defined by Trivedi (1982 section 1.2) as phenomena whose future behaviour is not predictable in a deterministic fashion. That is, given the current value of a stochastic variable, such as a service time, it is impossible to predict the next value it will take on. This unpredictability presents a problem when using stochastic variables in a computer simulation, since the computer that will run the simulation is certainly deterministic. However, the behaviour of the stochastic variable as it takes on many values can be characterized by a mathematical function, called a probability mass function for discrete variables and a probability density function for

continuous variables. A probability mass function can be specified by a histogram with one bar for each discrete value the function may take on. The height of the bar directly gives the probability of the function taking on that value. The probability density function is a continuous function f(x) where x is defined over the range of values the function may take on. The value of f(x) does *not* correspond to the probability of the function taking on the value x. Rather, one can determine the probability of the function taking on a value within a given range x_1 to x_2; this is the area under the curve f(x) between x_1 and x_2.

Using stochastic variables in a simulation requires two things:

1. Choosing the appropriate input function to characterize the stochastic variable.
2. Choosing random samples from the input function to give values of the stochastic variable.

A crucial step in the development of a model is choosing the proper input function to represent the behaviour of a particular stochastic variable. If it is known a stochastic variable follows some input function, then the model can generate random values from that function to represent that variable. "Knowing" which is the correct input function is not a straightforward matter however.

The most common approach used to determine input distributions is to collect data on the stochastic variable of interest in the real-world system and use the data to determine the appropriate distribution. Law and Kelton (1983 section 5.1) describe two ways to use this data:

1. Use statistical inference techniques to fit a theoretical input function to the data and perform hypothesis tests to determine the goodness of fit.

2. Use the data itself to construct an empirical input function and sample directly from it.

In the absence of data collected from the real-world system, another approach must be used. One approach is to use knowledge of the real-world system to suggest that a particular stochastic variable would be well suited to being modelled by a particular function. Another approach is suggested by Law and Kelton (1982 section 5.6), which is to use the beta function, which can take on many "shapes" depending on the values of the two shape parameters, α_1 and α_2, given to it. When using the beta function to model a delay, experts within the real-world domain provide estimates of the minimum and maximum times, a and b respectively, the delay is expected to take. If nothing else is known about the delay, using shape parameter values of $\alpha_1 = \alpha_2 = 1$ will give a uniform function over the interval [a, b]. Shape parameter values of $\alpha_1 > \alpha_2 > 1$ will give a function which is skewed to the right, which is typical of delays involving the time taken to complete a task.

Choosing values for the shape parameters α_1 and α_2 to give the desired shape to the function is not easy, but if the experts provide estimates of the most likely, or *mode* (labelled m) time for the delay and the average, or mean (labelled μ) delay, values for α_1 and α_2 can be calculated to give the desired shape parameters. Law and Kelton (1982 section 5.6) give the following formulae for calculating the shape parameters:

$$\alpha_1 = \frac{(\mu - a)(2m - a - b)}{(m - \mu)(b - a)} \qquad \alpha_2 = \frac{(b - \mu)\alpha_1}{\mu - a}$$

The following sections describe the exogenous variables in the logical model. The choice of input functions is explained for each one.

3.6.2: SERVICE TIME

Data for service times was not available, and the system itself did not suggest any obvious choice for an input distribution, so it was decided to model service times using the beta distribution. Law and Kelton (1982 section 5.6) state that in their experience with real-world data, functions for the time taken to perform some task often have a beta distribution which is skewed to the right.

Several researchers have used simulation to study hospital emergency departments. Emergency departments and the TDRS share several characteristics, namely:

- Patients (troubles) arrive for treatment (repair).
- Acuity (criticality) is assessed at arrival, which is used to prioritize treatment
- Patients (troubles) are diagnosed and appropriate treatment (repairs) done.

Because of these common characteristics, it is felt that some aspects of the logical models of emergency departments can be transferred to the logical model of the TDRS. In his simulation of a hospital emergency room, McGuire (1994) provides a good explanation for using the beta distribution for modelling patient treatment times:

> Treatment times (task times) have certain characteristics that cause the distribution of these times to "look" like a [beta] distribution. The first characteristic is that there is a minimum amount of time the necessary treatment will take. This time is usually not far from the time the treatment will most likely take. This is because most practitioners are competent at what they do, so unless unexpected delays occur, the elapsed time will probably be much closer to the minimum time than it will be to the

maximum time. The average time will usually be larger than the most likely time because of occasional long delays that can occur, inflating the overall average time. This results in a distribution that looks similar to the following:

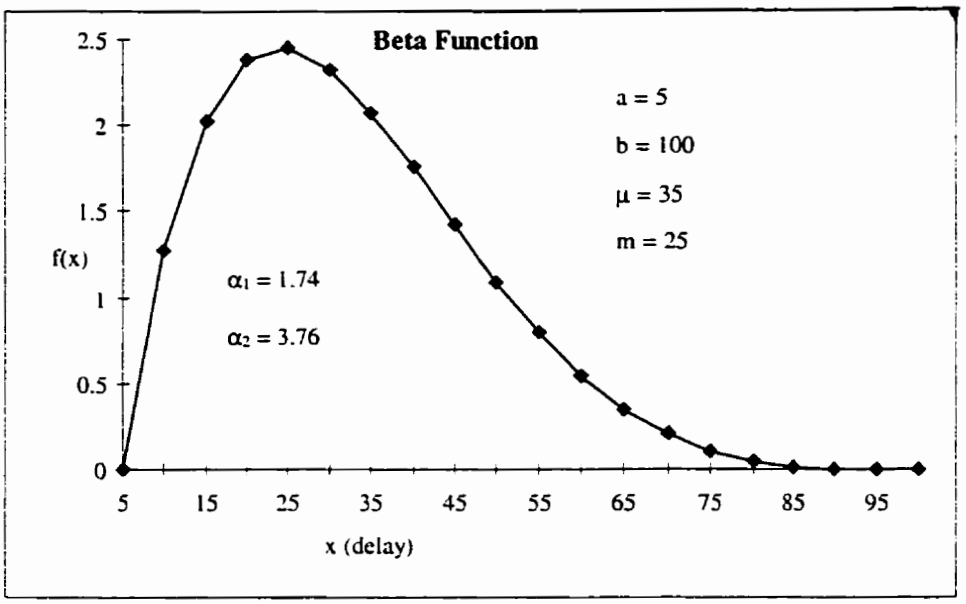

Figure 3.2: Beta Function

The same reasons that make the beta function appropriate for modelling patient treatment times apply to service times, so I have chosen the beta function to model service times.

3.6.3: TROUBLE ARRIVALS

Again, no data for trouble arrival rates was available. In the case of trouble arrivals, some assumptions about the nature of failures for digital switching equipment have been used to guide the choice of the input function. These are:

1. The faults that cause troubles are independent.
2. Each fault gives rise to a single trouble. Thus, from 1) and 2), troubles are independent.
3. The trouble arrival rate does not show any temporal effects. That is, the rate is constant.

Assumption 1 is made because this has been the experience of MTS during the normal day-to-day operation of the network. Gordon (1995) states that "Failure...distributions can be safely assumed to be Poisson", where the Poisson distribution gives the number of events that occur in an interval of time when the events are independent. Stated another way, faults encountered in the normal course of operations do not tend to cause other faults. This seems reasonable for software faults; a software problem is unlikely to generate other problems. As for hardware, failure of individual switch components generally does not cause failure of other components.

It is not hard to imagine faults that are capable of causing other faults. Loss of power to a building, cutting an underground cable and loss of communication and control of a switch would all be capable of causing other faults. These are considered to be exceptional circumstances however, and are not included in this logical model.

Assumption 2 arises from the fact that the network monitoring and testing process detects most faults. This process detects a fault, which then becomes a trouble to be dealt with. The situation where one fault can cause several troubles is found with customer-reported troubles, because a fault affecting several customers is likely to be reported several times. All customer-reported troubles caused by a single fault are "stapled", i.e. they are associated with the fault causing the trouble, in effect becoming a single trouble. The process of stapling all the troubles related to a single fault is essentially administrative in nature and as such is not central to the process of repairing troubles.

Assumption 3 is reasonable for all types of troubles except customer-reported troubles. As discussed in section 3.5, these troubles make up a minor portion of the total

number of troubles, so assuming that their arrival rate is constant as well will not significantly affect the model.

An arrival process is termed a *Poisson process* if the following properties hold, according to Law and Kelton (1982 section 5.7.1):

1. Arrivals occur one at a time.
2. The number of arrivals in a given interval is independent of the number of previous arrivals and also of the times at which they occur.
3. The number of arrivals in a given interval is independent of when the interval occurs.

Assumptions 1 and 2 satisfy properties 1 and 2 for the model, i.e. troubles do not arrive in "batches", and the arrival of new troubles is independent of any previous trouble arrivals. Assumption 3 satisfies property 3 for the model, i.e. the number of arrivals does not show any temporal effects. Thus with the given assumptions the trouble arrival process can be modelled as a Poisson process.

Since troubles arrive one at a time, rather than model the rate of arrival it is more convenient to model the length of time between successive arrivals. In this way the model need only be concerned with the arrival of one trouble occurring at some future time. The time between successive arrivals is called the *interarrival time*. Law and Kelton (1982 section 5.7.1) state a simple relationship between a Poisson arrival process and its corresponding interarrival times: the interarrival times of a Poisson arrival process with rate λ are exponential random variables with parameter $\frac{1}{\lambda}$. Thus I have chosen to model

trouble arrivals by modelling the interarrival time between successive troubles using an exponential function with parameter equal to the reciprocal of the arrival rate.

The newly arriving trouble is assigned a criticality value. I have chosen to model this by selecting the criticality from a multivariate discrete random variable (see Table 3.1). There are four possible values of the random variable, corresponding to the four criticality levels. The probability of any one of the four values being chosen corresponds to the proportion of troubles of that criticality expected in the system.

3.6.4: Other Exogenous Variables

Other exogenous variables in the model have been modelled using discrete random variables. Level 2 service is call-out is bivariate and is used to determine if a level 2 worker can be obtained during non-business hours. Study result is multivariate and is used to determine the result of studying a trouble. Table 3.3 lists these variables and the values they may take on.

Variable	Values
level 2 service call-out	unsuccessful successful
study result	trouble resolved hardware fault trouble unresolved

Table 3.3: Discrete Exogenous Variables

3.7: Modelling Stochastic Endogenous Variables

3.7.1: Introduction

Endogenous variables are ones whose values are affected by changes to the model; i.e. they arise from within the model. As with exogenous variables, they are also stochastic in nature, and just as exogenous variables can be thought of as inputs to the model, endogenous variables can be thought of as outputs from the model. Choosing which variables to include in the model is determined by their usefulness in providing insight to the questions of interest in the real-world system. In the TDRS, these questions are:

1. How quickly are troubles being cleared?
2. How many troubles waiting for a particular service accumulate in the system? How many troubles of each criticality level are accumulating in the system?
3. How efficiently is staff being used?

The endogenous variables that can help answer these questions are, respectively:

1. Receipt-to-Close time (time elapsed from the moment a trouble arrives in the system until it leaves).
2. Queue lengths. Each service level has its own queue of troubles awaiting service. Troubles awaiting service can also be logically organized into queues corresponding to the criticality levels so the second part of the question can be answered.
3. Utilization (the proportion of time a service level was busy providing service).

3.7.2: COLLECTION OF ENDOGENOUS VARIABLES

In order to extract these endogenous variables from the model, the events that give rise to the variables must be noted and recording of appropriate data performed. Figure 3.3 below summarizes these events and the statistics associated with them.

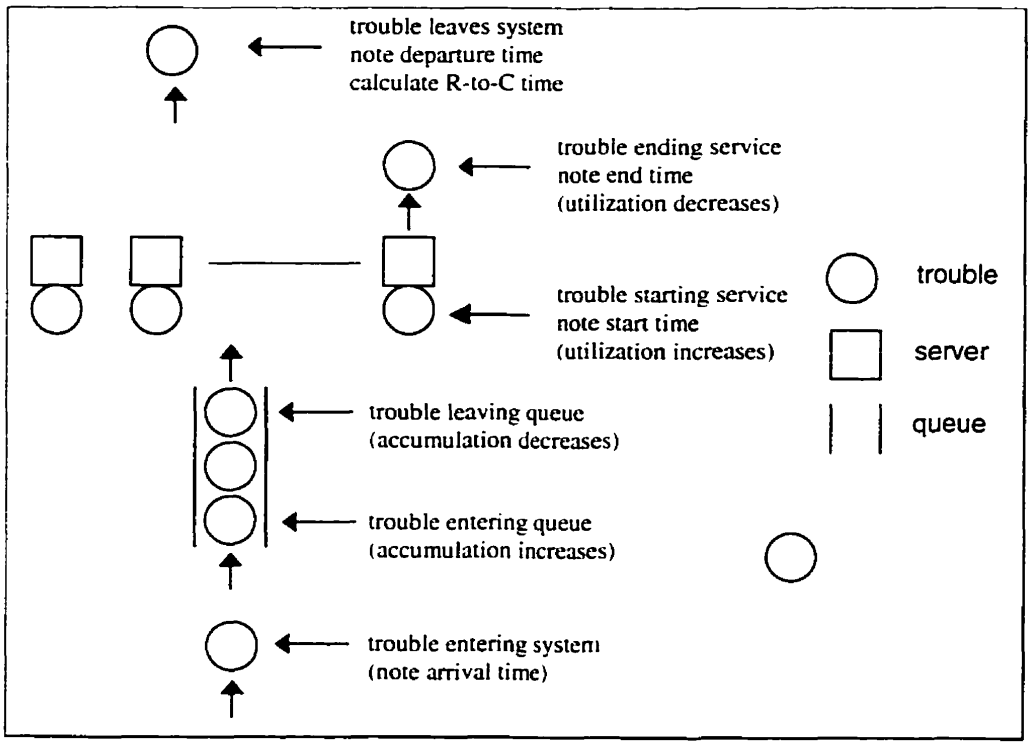

Figure 3.3: Critical Events for Endogenous Variables

Endogenous variables are classified into two categories: event-actuated statistics and time-integral statistics. Event-actuated, or point statistics are generated by events and give rise to a single value. In the TDRS delays like receipt-to-close time and service delays are the event-actuated statistics gathered. Time-integral statistics are collected on variables whose value can vary over time. These variables are called *state variables* because if the system is frozen at some point in time, the values of the state variables characterize the state of the system. Events in the system that change a state variable do

not give rise to a point value but change the value of the state variable. Queue lengths are examples of state variables in the TDRS.

Forming means of event-actuated statistics simply involves finding the average value of all the point values gathered. Forming the mean of a time-integral statistic involves finding the area under the curve of the state variable value over time and dividing by the length of time over which the statistic is gathered, hence the name time-integral.

The heterogeneity of the system with respect to service delays and staffing levels causes a problem with the collection of both types of statistics. Some choice has to be made about the length of the time period over which the statistic is to be collected and the mean value reported. Statistics may also be collected for each criticality level and service level or they may be aggregated as well. The issue of deciding the *granularity*, or fineness, of the time period has no correct answer and really depends on which features of the model's performance are to be studied. For instance, receipt-to-close data is collected with very fine granularity, collecting statistics over each of the 21 shifts during the week (the shift of arrival) and for each of the 4 criticality levels. This was done because it was felt that shift of arrival would have an effect on receipt-to-close time, so this fine level of granularity was needed to observe it. In comparing alternative scenarios, receipt-to-close time can be collected weekly for each criticality level, making it easier to compare results for each scenario.

CHAPTER 4: IMPLEMENTATION

4.1: INTRODUCTION

This chapter discusses the implementation of the logical model described in chapter Three. Section 4.2 discusses the choice of SIMSCRIPT II.5 as the implementation language and Section 4.3 discusses UNIX as the implementation platform. Section 4.4 discusses the modelling constructs provided by SIMSCRIPT II.5 and Section 4.5 discusses the use of these constructs in modelling service levels, troubles and trouble flow. Section 4.6 discusses the collection of model outputs. Section 4.7 discusses the verification procedures for the implementation. Section 4.8 gives a description of the graphical interface.

4.2:CHOICE OF IMPLEMENTATION LANGUAGE

Law and McComas (1996) state that software for simulation falls into several categories that are summarized in Table 4.1.

Category	examples
1) General-purpose programming languages	FORTRAN, C
2) General-purpose simulation languages	SIMSCRIPT II.5
3) Application-specific simulation languages	MedModel
4) Application-specific simulators	NETWORK II.5

Table 4.1: Simulation Software Categories

Software in each category has its advantages and disadvantages. In general, as one moves from category 1 to 4 the speed of programming a model increases but the flexibility in building the model decreases. Table 4.2 summarizes the advantages and disadvantages of each category.

Category	Advantages	Disadvantages
1	Readily available on all systems, so no added expense No need to learn a new language	Must provide all required elements of a simulation Increased programming time
2	Most common elements for model building and statistics gathering are provided	Need to learn a new language Still must develop all aspects of the logical model
3	Provides high-level model constructs specific to a particular domain	May not be able to model certain unique aspects of a logical model
4	"Drag and drop" model building involving no programming	Least flexible environment

Table 4.2: Simulation Software Advantages and Disadvantages

The following factors were considered when making the choice of an implementation language:

1. Availability
2. Flexibility to model unique aspects of the logical model
3. Previous experience with the language
4. Provision of common modelling and statistics gathering elements

5. Graphical interface capabilities

6. Cost

Several languages from categories 1 and 2 were available for this implementation. The general-purpose languages FORTRAN, Pascal, C and C++ and Java are available on both the UNIX and PC platforms. The general-purpose simulation language SIMSCRIPT II.5 is available on the UNIX platform, and a library of C routines in MacDougall (1987) called SMPL that provides basic support for programming simulations is also available. Languages from categories 3 and 4 are not presently available in the Computer Science department, and thus were not considered further.

All languages from categories 1 and 2 provide flexibility in modelling. Previous experience with the language is important since learning a new language adds to the time needed to develop the model implementation. I have extensive experience programming in FORTRAN, Pascal and C, limited experience with C++ and none with Java. In category 2 I have extensive experience with SIMSCRIPT II.5 but none with SMPL.

Provision of common simulation modelling constructs and statistics gathering elements is important because it reduces the time and programming effort needed to implement the model and increases the likelihood the model will operate correctly. Since none of the category 1 languages provide these, the choice of implementation languages is narrowed to one, namely SIMSCRIPT II.5.

The availability of graphical interface capabilities is important for several reasons:

- Observation of graphical output from the model gives the model's users a more immediate, intuitive grasp of the model's behaviour. This is very helpful during the verification phase of the implementation.

- Graphical displays of network behaviour are commonly used in network trouble management; this makes the system more familiar to the people involved with the real-life system and makes it easier to obtain feedback.
- Graphical model input gives the implementation a "user-friendly" feel and increases the likelihood it will actually be used.

SIMSCRIPT II.5 provides a package for building a graphical interface called SIMGRAPHICS. This package allows the modeller to add graphic display elements to dynamically display the model outputs. These elements include bar graphs, dials, clocks, textual information, X-Y plots and animated icons. The modeller can also provide a graphical interface for model input using forms. The forms include such standard graphical elements as text boxes, check boxes, radio buttons and predefined lists.

Finally, there are no costs associated with choosing SIMSCRIPT II.5 as the implementation language, since the software and documentation are already available. Thus SIMSCRIPT II.5 is the implementation language that best meets the stated criteria and is chosen to implement the logical model.

4.3: Choice of Implementation Platform

UNIX was chosen as the platform for implementation of the model. UNIX was chosen because:

1. SIMSCRIPT II.5 is currently available on the UNIX platform in the Computer Science department.
2. MTS personnel have UNIX workstations that can be used to run a compiled version of the implementation.

3. The SIMSCRIPT and SIMGRAPHICS code itself is platform-independent.

The third point is particularly important since it means the implementation can be done on one platform and the model used on another platform without having to make platform-dependent changes to the implementation. The independence of the SIMGRAPHICS graphical code is especially important considering the large differences that exist between graphical environments on different platforms, for example the UNIX X-Windows environment and the PC Windows environment. If at a later time it is decided to port the implementation to a PC platform, the same SIMSCRIPT code can be used without changes. (This would require the purchase of a PC version of SIMSCRIPT II.5)

4.4: SIMSCRIPT Modelling Constructs

SIMSCRIPT II.5 provides constructs that automatically provide the features common to all discrete-event simulation models. In addition the syntax of SIMSCRIPT is English-like and utilizes simulation-oriented terms, which leads to concise, self-documenting code that is understandable not only by the modeller but the user as well. This section discusses the constructs that have been used in the implementation so that the reader is familiar with their use in building a discrete-event simulation model and will understand how they are used to implement the logical model. The use of SIMSCRIPT in building simulation models is discussed in Russell (1983).

4.4.1: THE PROCESS

A process represents a dynamic object in the real-world system and the sequence of events it experiences while in the model. A process is created by the ACTIVATE statement. It has parameters that describe a particular instance of the real-world object. It also has a process routine that describes its behaviour in the model. The process routine is a sequence of related events separated by delays. The delays can be determinate in length, such as service times, or indeterminate, such as waiting for a limited resource to become available. Determinate delays are implemented by the WORK and WAIT statements, which specify the length of the delay. Indeterminate delays are implemented in two ways: indirectly when a process requests a resource that is not available, or directly via the SUSPEND statement. In the former case the process is placed in a set of processes waiting for the resource to become available and in the latter case it must be explicitly placed in a set (the set construct is discussed below). An indeterminate delay is terminated by some event that occurs outside the process routine. If the delay is indirect, the system will automatically resume the process when the required resource becomes available and when the delay is direct the process is explicitly resumed by some event using the RESUME statement.

4.4.2: THE SET

A set is an ordered collection of like entities such as instances of a process. Sets are used to implement queues in the logical model. Sets support the most common queuing disciplines, namely

- First In First Out (FIFO): Members are added to the end of the queue and removed from the front of the queue.
- Last In First Out (LIFO): Members are added and removed from the front of the queue.
- Priority: Members are added to the queue in order of one or more priority attributes and removed from the front of the queue.

Sets are implemented as linear linked lists and are automatically maintained by SIMSCRIPT. A set has an owner and a declared member entity that are assigned pointers to implement the set. A set is declared by naming it and its owner, as follows:

```
THE SYSTEM OWNS THE ticket.queue
```

In this example only one queue is needed so the owner is declared to be the system. By default sets use the FIFO discipline; another discipline can be specified by a DEFINE statement; the following example defines a priority queue:

```
DEFINE ticket.queue AS A FIFO SET RANKED BY HIGH crit.lvl
```

Members of a set are specified by the BELONG statement, as follows.

```
EVERY text.display MAY BELONG TO THE text.queue
```

Adding and removing entities from a set can be handled by the system using the FILE and REMOVE statements. Alternately, if a more complex queuing discipline is needed, the modeller can manage sets by direct manipulation of the linked list.

4.4.3: The Permanent Entity

The permanent entity is the SIMSCRIPT construct that the modeller uses to create data structures to manage the more complex aspects of the logical model. A permanent entity can be thought of as a singly or multiply dimensioned array of records. The record

attributes can include set ownership and the permanent entity can itself belong to a set. The size of the array is set using the CREATE statement.

In this implementation the permanent entity is used for two purposes. The first is to model the service levels. SIMSCRIPT does in fact have a RESOURCE construct which extends the concept of the permanent entity to include automatic declaration and management of working and waiting queues for a resource (see Figure 4.5 below). This construct does not allow for preemption however. The second purpose is to create "bins" for the collection of model outputs. The example below declares and creates a singly dimensioned permanent entity to collect queue length data collected over all the shifts in the working week.

```
EVERY ticket.shift.bin HAS A ticket.queue.length
CREATE EVERY ticket.shift.bin(6)
```

4.4.4: The Event

The SIMSCRIPT event is similar to the process in that it also has a routine that describes its behaviour and that is scheduled to occur at some future time in the model. It differs from the process in that it occurs instantaneously in time i.e. there is no way to model delays within an event. Events are used to schedule tasks such as statistics collection or changing model parameters.

4.4.5: The Timing Routine

The timing routine is at the heart of a discrete-event simulation. A discrete-event simulation consists of a series of future events, which can include events and the re-activation of a process at the end of a work delay or wait delay, that are scheduled to

occur in some point in simulated time. The implementation maintains a list of all these future events; at least one such event must be scheduled prior to beginning the timing routine or the simulation will terminate immediately. When the simulation starts, the timing routine examines the list and chooses the future event with the earliest pending time. The event is removed from the list and the event or process routine associated with it is executed; this can cause other events to be added to the list. Execution proceeds in this fashion until the list contains no more pending events. SIMSCRIPT automates the timing routine for the modeller. The timing routine is summarized in Figure 4.1.

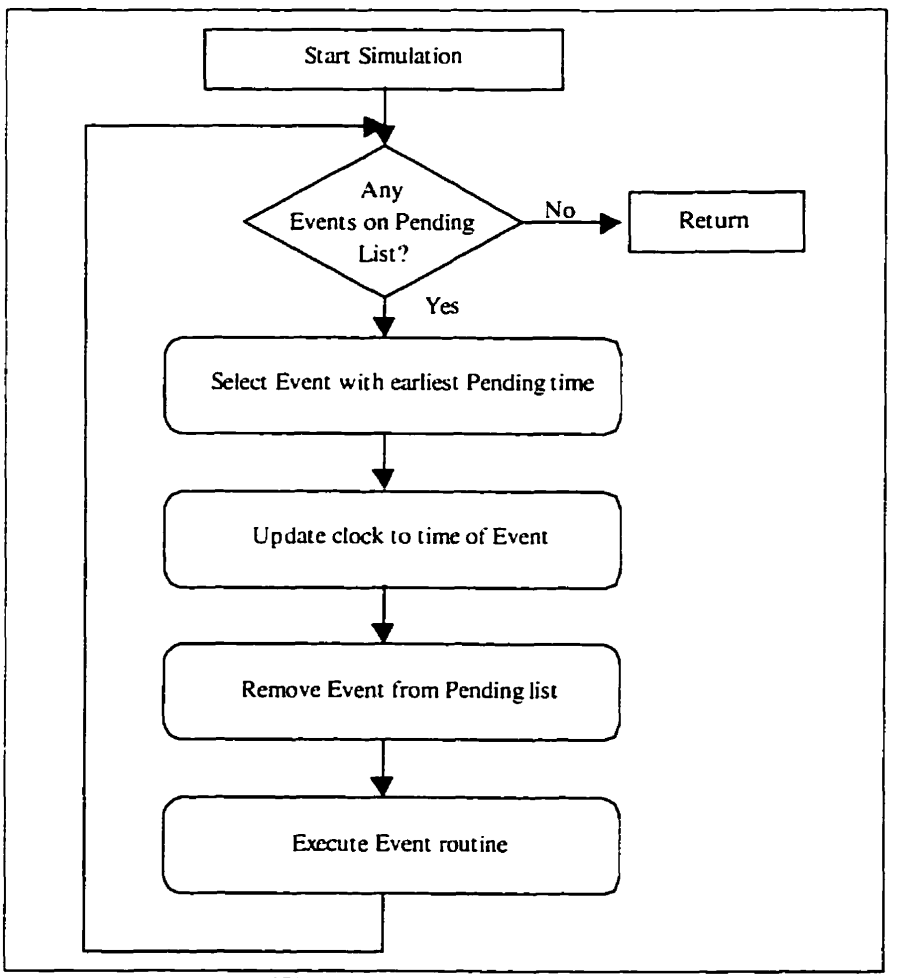

Figure 4.1: The Timing Routine.

4.5.6: DEFINE TO MEANS

SIMSCRIPT allows for string substitution through the use of the DEFINE TO MEAN statement. The most common use of this statement is to allow the use of meaningful symbolic names instead of numeric constants, thus improving the readability and self-documenting nature of the implementation. Some examples are shown below. Following the convention adopted by Russell (1983), DEFINE TO MEAN substitutions begin with a period.

```
DEFINE .minor TO MEAN 1        '' criticality levels
DEFINE .customer TO MEAN 2
DEFINE .major TO MEAN 3
DEFINE .critical TO MEAN 4
```

4.5: IMPLEMENTATION OF THE MODEL USING SIMSCRIPT CONSTRUCTS

4.5.1: IMPLEMENTATION OF TROUBLES

Troubles are modelled as processes because they experience delays while in the system of both determinate (service delays) and indeterminate (waiting for service) length. Figure 4.2 describes the attributes of a trouble. The *number* attribute uniquely identifies the trouble in trace output and the *trace* attribute allows a descriptive message to be printed in the trace. The *entry.time* attribute is used in statistics gathering. The *discard.flag* attribute is used to implement independent replications in the model i.e. at the beginning of a replication all troubles in the system must be discarded. The remaining six attributes are used to determine trouble flow for the trouble. The final three lines declare the three queues to which a trouble may belong.

```
EVERY        trouble
HAS A        number                '' identifier
AND AN       entry.time            '' time trouble was generated
AND AN       entry.shift           '' shift of the week it was generated
AND A        crit.lvl              '' criticality level
AND A        status                '' which service is needed next
AND A        trace                 '' used for trace output
AND A        shift.begun           '' shift service begun/resumed
AND A        last.serv.recd        '' level1, level2 or technician
AND AN       int.flag              '' T if trouble was preempted
AND A        callout.flag, '' T if call-out staff obtained
AND A        discard.flag          '' T if trouble to be discarded
AND A        ticket.flag           '' T if trouble is ticketed
AND MAY BELONG TO A waiting.queue
AND MAY BELONG TO A working.queue
AND MAY BELONG TO A ticket.queue
```

Figure 4.2: Trouble Attributes.

The trouble process routine is shown in Figure 4.3 below. It is quite simple, consisting of a call to the *initialize.trouble* routine and a loop, delimited by the UNTIL and LOOP statements. The initialization routine assigns an identification number and a criticality level to the trouble and initializes the variables needed for trouble flow. The loop requests service until the trouble is resolved or needs level 3 (OSO) service, after which the trouble leaves the system. The *dispatcher* routine determines if a worker of the appropriate service level is available. If not, the trouble experiences an indeterminate delay, implemented by the SUSPEND statement. If so, the trouble experiences a determinate service delay, implemented by the *worker* routine. Note the use of a DEFINE TO MEAN in the Boolean test of the loop. The string *.trouble.is.done* substitutes for the Boolean test 'status(trouble) GE 4'; the DEFINE TO MEAN expresses the real-world meaning of the test, thus improving the readability of the routine.

```
PROCESS trouble
    DEFINE available     '' true if worker is available
    AS AN INTEGER VARIABLE

    CALL initialize.trouble GIVING trouble
    UNTIL .trouble.is.done, DO
      CALL dispatcher GIVING trouble YIELDING available
      IF available EQ .false, SUSPEND ALWAYS
      CALL worker GIVING trouble
    LOOP
END'' trouble
```

Figure 4.3: Trouble Process Routine.

A separate process called the generator implements trouble arrivals. The generator is activated at the beginning of the simulation and it creates new troubles with exponential interarrival times. The generator is shown in Figure 4.4 below.

```
PROCESS generator
       IF graphics EQ .true, CALL initialize.text ALWAYS
       UNTIL TIME.V GE end.time, DO
         ACTIVATE A trouble NOW
         LET itime = EXPONENTIAL.F(inter.time, stream(.arrivals))
         WAIT itime MINUTES
       LOOP
END    '' generator
```

Figure 4.4: Generator Process.

The *dispatcher* routine does not correspond to any single real-world entity but embodies the process of obtaining service staff to work on a trouble. Its algorithm is summarised as follows:

1. If there is idle service staff, assign the staff to work on the trouble.
2. Else if the trouble is major or critical:
    i. If level 2 staff is required, and none are currently scheduled, try to obtain staff on call-out.
    ii. Look for a trouble currently in service that can be preempted.

The dispatcher routine is shown in Figure 4.5 below. Note in the second half of the routine that if no resource is available, the trouble is placed in a queue explicitly, as described in section 4.4.1 for a process executing the SUSPEND statement.

```
ROUTINE dispatcher GIVEN tr YIELDING result
       DEFINE tr,     '' trouble wanting resource
              result '' true if resource is available
       AS INTEGER VARIABLES

       LET result = .false
       IF U(status(tr)) GT 0,              '' worker available
         LET result = .true
       ELSE                                '' all busy/no workers on this shift
         IF crit.lvl(tr) GE .major,
           IF (status(tr) EQ .level2) AND (staff(status(tr), shift) EQ 0),
             CALL call.level2 GIVING tr YIELDING result
           ELSE
             CALL preempt.trouble GIVING tr YIELDING result
           ALWAYS
         ALWAYS
       ALWAYS

       IF result EQ .true,          '' give the resource to tr
         LET trace(tr) = .TR.start.serv
         CALL file.working.queue GIVING tr
         SUBTRACT 1 FROM U(status(tr))
       ELSE                         '' tr must wait
         IF trace(tr) EQ .TR.enter.system AND crit.lvl(tr) EQ .minor,
           LET trace(tr) = .TR.ticket
           FILE tr IN THE ticket.queue
         ELSE
           LET trace(tr) = .TR.queued
           CALL file.waiting.queue GIVING tr
         ALWAYS
       ALWAYS
END    '' dispatcher
```

Figure 4.5: Dispatcher Routine.

A trouble that suspends itself must be resumed by another event. There are two events that result in a suspended trouble being resumed. One is the completion of the service of another trouble, thereby freeing a worker, and the other is the beginning of a new shift, adding new workers to the system. The *exit* routine is responsible for resuming a suspended trouble in both cases, the difference being from which location the exit routine is called. Before the exit routine is called, the number of idle staff is increased as appropriate, and the routine "matches" idle staff with troubles waiting for service. For each match it makes, it resumes the trouble. In the case of a trouble completing service, the *worker* routine calls the *end.service* routine which adds the freed worker to the pool of idle staff if appropriate (i.e. if the worker isn't held over from the previous shift or a

call-out worker) which then calls the exit routine. In the case of a new shift the exit routine is called directly by the *new.shift* event. The exit routine is shown in Figure 4.6 below.

```
ROUTINE exit GIVEN lvl
       DEFINE lvl,    '' service level
              tx      '' trouble starting service
       AS INTEGER VARIABLES

       CALL match GIVING lvl YIELDING tx
       UNTIL tx EQ .null, DO
         IF int.flag(tx) EQ .true,
           LET trace(tx) = .TR.resume.serv
           LET int.flag(tx) = .false
         ELSE
           LET trace(tx) = .TR.start.serv
         ALWAYS
         LET shift.begun(tx) = scount
         CALL file.working.queue GIVING tx
         RESUME trouble CALLED tx
         SUBTRACT 1 FROM U(lvl)
         CALL match GIVING lvl YIELDING tx
       LOOP
END    '' exit
```

Figure 4.6: Exit Routine.

4.5.2: Implementation of Service Levels

Service levels are implemented as a permanent entity. The definition of the permanent entity is shown in Figure 4.7 below. It is dimensioned to size three to represent the three levels of service in the model. The variable *U* stores the number of workers available during the shift. Each service level also owns two queues, one for troubles being worked on and one for troubles waiting for service. The variables *nstaff* and *util* are used for graphical display purposes.

```
       EVERY  server
       HAS A         U                    '' # of workers available
       AND A         nstaff               '' display variable
       AND A         util                 '' display variable
       AND OWNS A    waiting.queue        '' troubles waiting for service
       AND A         working.queue        '' troubles being served
```

Figure 4.7: Service Permanent Entity.

The actual trouble repair process is implemented in the *worker* routine. The worker routine is shown in Figures 4.8a and 4.8b. In Figure 4.8a the decision is made whether the trouble should be studied, ticketed or escalated to the next service level. If the trouble is not to be studied, no further action is taken with the trouble and execution proceeds to the second part of the routine. If it is to be studied, the length of the study delay is determined using the beta function and the outcome of the study (resolved, hardware, unresolved) is also determined. If the outcome of the study phase is resolved, the length of the repair delay is also determined.

Figure 4.8b shows the second part of the worker routine where the study and repair delays (if any) are implemented using the WORK statement. After the delays are finished, the trouble attributes are updated and execution resumes in the trouble process routine

```
ROUTINE worker GIVEN tr
      DEFINE tr,                  '' the trouble being worked on
             decision,                '' how to handle the trouble
             i2,i3,min,max '' beta parm variables
      AS INTEGER VARIABLES
      DEFINE study.time,          '' time to study trouble
             repair.time,  '' time to resolve trouble
             total.time,              '' total time to service trouble
             res                      '' RV to determine outcome of study
      AS DOUBLE VARIABLES
      DEFINE cut1, cut2,          '' cutoffs for result of study
             alpha1, alpha2,          '' beta distribution parameters
      AS REAL VARIABLES
      LET shift.begun(tr) = scount
      LET i2 = status(tr)
      IF ((shift EQ .day)AND(day LE .friday)),       '' business hours
        LET i3 = crit.lvl(tr)
      ELSE
        LET i3 = 4 + crit.lvl(tr)
      ALWAYS
      LET decision = study.matrix(i2, i3)
      IF decision EQ .study,      '' study trouble
        LET alpha1 = alpha.1(.study, i2, i3)
        LET alpha2 = alpha.2(.study, i2, i3)
        LET min = betaparm(.study, i2, i3, .min)
        LET max = betaparm(.study, i2, i3, .max)
        LET study.time = BETA.F(alpha1, alpha2, stream(.studytime))
        LET study.time = min + (max - min) * study.time
        LET res = UNIFORM.F(0, 1, stream(.resolution))
        LET cut1 = resmat(status(tr), 2 * crit.lvl(tr) - 1)
        LET cut2 = resmat(status(tr), 2 * crit.lvl(tr))
        IF res LT cut1,
          LET decision = .resolved
        ELSE IF res LT cut2,
          LET decision = .hardware
        ELSE
          LET decision = .unresolved
        ALWAYS ALWAYS
        IF status(tr) EQ .technician AND decision EQ .hardware,
          LET decision EQ .unresolved
        ALWAYS
        IF decision EQ .resolved,
          LET alpha1 = alpha.1(.repair, i2, i3)
          LET alpha2 = alpha.2(.repair, i2, i3)
          LET min = betaparm(.repair, i2, i3, .min)
          LET max = betaparm(.repair, i2, i3, .max)
          LET repair.time = BETA.F(alpha1, alpha2, stream(.service))
          LET repair.time = min + (max - min) * repair.time
        ELSE
          LET repair.time = 0.0
        ALWAYS
      ELSE                        '' escalate or ticket
        LET study.time = 0.0
        LET repair.time = 0.0
      ALWAYS
      IF callout.flag(tr) EQ .true,
        IF steady.state EQ .true,
          LET callouts(shift) = callouts(shift) + 1.0
          ADD (study.time+repair.time) TO callout.time(shift)
        ALWAYS
        LET callout.flag(tr) = .false
      ALWAYS
```

Figure 4.8a: Worker routine part I.

```
        LET total.time = study.time + repair.time
        IF study.time GT 0,
          WORK study.time MINUTES
          LET trace(tr) = .TR.finish.study
          IF debug EQ .true,
            NOW print.trouble.trace GIVING tr, study.time
          ALWAYS
        ALWAYS
        IF repair.time GT 0,
          WORK repair.time MINUTES
          LET trace(tr) = .TR.finish.repair
          LET servtime(status(tr), shift) = total.time
          IF debug EQ .true,
            NOW print.trouble.trace GIVING tr, -repair.time
          ALWAYS
        ALWAYS
        REMOVE tr FROM THE working.queue(status(tr))
        LET last.serv.recd(tr) = status(tr)
        LET trace(tr) = decision
        IF debug EQ .true, NOW print.trouble.trace GIVING tr, dzero ALWAYS
        LET status(tr) = service.result.matrix(status(tr), decision)
        IF discard.flag(tr) NE .true,
          CALL end.service GIVING tr
        ELSE
          LET status(tr) = .discard
        ALWAYS
END     '' worker
```

Figure 4.8b: Worker routine part II.

4.5.3: IMPLEMENTATION OF TROUBLE FLOW

Two matrices, called *study.matrix* and *service.result.matrix* implement trouble flow. Both are deterministic matrices used to control routing. The study matrix is used by the first part of the worker routine to determine whether to study, ticket or escalate the trouble. The values in this matrix are accessible by the user through the graphical interface so that various trouble flow scenarios can be examined. The service result matrix is used by the second part of the worker routine to update the trouble parameter *status* so that the trouble will be routed to the appropriate service level or leave the system if it is resolved. This matrix is read when the simulation begins; its values are not accessible through the graphical interface because it is not expected that the routing of

troubles to the appropriate service level will need to be changed. For example, once a trouble is resolved, it will always leave the system and if a hardware fault is determined it will always be routed to the technician pool. If for some reason a change needs to be made to this matrix it can be done by editing the text file containing the model parameters.

4.5.4: Implementation of Temporal Effects

Temporal effects are time-of-day and day-of-week effects caused by varying staff levels. The workday is divided into three eight-hour shifts: the night shift from midnight to eight a.m., the day shift from eight a.m. to four p.m. and the evening shift from four p.m. until midnight. The workweek is divided into weekdays (Monday to Friday) and weekends (Saturday and Sunday). Staff level changes are implemented by the *new.shift* event, shown in Figure 4.9 below.

The first task of the new shift is to update the shift count, the current shift and the day if necessary. Next the shift strengths are updated. Shift strengths are stored in the *staff* matrix, which is specified by the user through the interface. If there are troubles waiting for service, they are assigned to the newly arrived staff by the *exit* routine.

After the assignment of troubles to newly arrived staff is complete, there may still be critical or major troubles waiting for level 2 service. For each of these troubles an attempt is made to obtain level 2 service on a call-out basis with the *call.level2* routine. The call-out for critical and major troubles at the beginning of each shift is justified because:

- Critical and major troubles must receive service as soon as possible.

- The large time between the Friday and Monday day shifts results in an accumulation of critical and major troubles if only one call-out attempt is made (when the trouble arrives).
- The probability of obtaining level 2 service on call-out may change with each shift.

```
EVENT new.shift
      DEFINE i,                              '' loop counter
                  tr,next,result             '' level 2 on-call search vars
      AS AN INTEGER VARIABLE

      ADD 1 to scount                        '' increment shift count
      IF shift EQ .evening OR shift EQ .wknd.evening,       '' new day
        IF day LT .sunday,
          ADD 1 to day
        ELSE
          LET day = .monday
        ALWAYS
        IF day LE .friday,
          LET shift = .night
        ELSE
          LET shift = .wknd.night
        ALWAYS
      ELSE
        ADD 1 to shift
      ALWAYS
      FOR i = .level1 TO .technician, DO      '' update shift strengths
        LET U(i) = staff(i, shift)
        LET nstaff(i) = staff(i, shift)
        CALL exit GIVING i        '' assign workers to waiting troubles
      LOOP
'' This loop attempts to find level 2 workers on call-out basis
'' for queued level 3, 4 troubles at the start of the shift.
      IF N.waiting.queue(.level2) GT 0,
        LET tr = F.waiting.queue(.level2)
        WHILE tr NE .null AND crit.lvl(tr) GE .major, DO
          LET next = S.waiting.queue(tr)
          CALL call.level2 GIVING tr YIELDING result
          IF result EQ .true,
            LET trace(tr) = .TR.oncall
            REMOVE tr FROM the waiting.queue(.level2)
            LET shift.begun(tr) = scount
            CALL file.working.queue GIVING tr
            RESUME trouble CALLED tr
            SUBTRACT 1 FROM U(.level2)
          ALWAYS
          LET tr = next
        LOOP
      ALWAYS
      IF shift EQ .evening OR shift EQ .wknd.evening,
        SCHEDULE A collect.shift.data AT TRUNC.F(TIME.V + 1)
        SCHEDULE A new.shift AT TRUNC.F(TIME.V + 1)
      ELSE
        SCHEDULE A collect.shift.data IN shift.length HOURS
        SCHEDULE A new.shift IN shift.length HOURS
      ALWAYS
END   '' new.shift
```

Figure 4.9: New shift event.

The final task of the new shift event is to schedule the next new shift. The initial new shift event is scheduled before entering into the timing loop. Having the current event schedule the next one means that only one new shift is in the pending events list at

any time. Thus whether the simulation runs to completion or is terminated early there is only one pending new shift event which must be removed using the DESTROY statement.

4.6: Collection of Model Outputs

This section first gives a brief description of the SIMSCRIPT constructs for collection of model outputs. Next it explains the techniques used for obtaining estimates of the model outputs.

4.6.1: SIMSCRIPT Output Collection Statements

SIMSCRIPT provides commands for the automatic collection of model outputs using the TALLY and ACCUMULATE statements. The TALLY statement is used to collect event-actuated statistics such as wait times or service times. The ACCUMULATE statement is used to collect time-integral statistics on a state variable such as the length of a queue. The use of the TALLY or ACCUMULATE statements are illustrated in Figure 4.10. The statements declare a new variable that receives the collected statistic, the type of statistic collected and the name of the variable being monitored. Whenever the monitored variable changes value the statistics are automatically updated by SIMSCRIPT.

```
TALLY  RtoC.time.num        AS THE NUMBER,
       RtoC.time.avg        AS THE MEAN,
       RtoC.time.std        AS THE STD.DEV OF RtoC.time
ACCUMULATE avg.q AS THE AVERAGE OF N.waiting.queue
ACCUMULATE avg.x AS THE AVERAGE OF N.working.queue
ACCUMULATE avg.t AS THE AVERAGE OF N.ticket.queue
ACCUMULATE avg.c AS THE AVERAGE OF number.in.sys
```

Figure 4.10: TALLY and ACCUMULATE statements.

4.6.2: Forming Means and Confidence Intervals

Estimates of model outputs produced by a simulation are random samples that may have a high variance. Because of this, the estimate obtained from a single run may differ greatly from the "true" answer, that is the expected value of the quantity. What is needed are:

1. An estimator of the expected value of the quantity being measured, and;
2. A way to assess the precision of this estimator.

Classical statistical analysis can be used to form an estimate of the true mean μ and the true variance σ^2. A number of observations, say n, of the random variable X are made such that the $X_i, i = 1, n$ are independent and identically distributed (IID). Now, the sample mean $\overline{X}(n)$ and sample variance $s^2(n)$ can be formed and will be unbiased estimators of μ and σ^2. The sample mean provides the estimator described in 1 above.

The usual approach to assessing the precision of an estimator is to form a confidence interval for the true measure. The Central Limit Theorem states that as n increases, the distribution of $\overline{X}(n)$ approaches the normal distribution with mean μ and variance σ^2/n. The properties of the normal distribution can be used to construct a confidence interval $\left[\overline{X}(n) - z\sqrt{\frac{\sigma^2}{n}}, \overline{X}(n) + z\sqrt{\frac{\sigma^2}{n}}\right]$. The value z is the upper critical point from the standard normal distribution chosen to give the desired confidence level. For example, for a confidence level of 95%, z = 1.960. This confidence interval is interpreted to mean that if one constructed a large number of these confidence intervals, 95% of them would cover (contain) the true mean.

Since the true value of σ^2 is not known, its estimator $s^2(n)$ must be used to form the confidence interval. When this change is made the mean $\overline{X}(n)$ has a t distribution with n-1 degrees of freedom. The confidence interval now becomes $\left[\overline{X}(n)-t\sqrt{\frac{s^2(n)}{n}}, \overline{X}(n)+t\sqrt{\frac{s^2(n)}{n}}\right]$, where t is the upper critical point from the t distribution with n-1 degrees of freedom chosen to give the desired confidence level. For example, for a confidence level of 95% and n=10, t = 2.262. As n goes to infinity, the t distribution approaches the normal distribution, meaning the critical points from the t distribution are always larger than the corresponding points from the normal distribution. Thus the confidence intervals produced by the latter version are also larger, meaning they will generally have coverage closer to the specified confidence level than the former version.

Two problems arise when applying classical statistics to the output from a simulation. The first is deciding what "sufficiently large n" means. The second is that observations from a simulation are rarely independent.

There is no simple answer to the question of how many observations are necessary to form a valid confidence interval. In general the more non-normal or skewed the distribution function from which the observations are being drawn the higher the number of observations necessary. Law and Kelton (1982 section 8.5.1) state that 10 observations allow the formation of a confidence interval with close to the expected coverage for all but the most highly skewed distribution functions. Even the MM1 system, with exponential interarrival and service times gives a confidence interval with the expected coverage using 10 observations. The beta function used in this

implementation is much less skewed than the exponential, so it is concluded that 10 independent observations are sufficient to form a valid confidence interval.

If the observations are not IID, any conclusions drawn from the mean and confidence interval may be misleading or wrong. Observations from a simulation, for example delay times, are often positively correlated; if a long delay in a queue is observed, it is likely the next delay from the same queue will also be long. Positive correlation decreases the variance of the observations, decreasing the range of the confidence interval, meaning the interval will not cover the true mean with the desired confidence level.

There are several ways of approximating IID observations from a simulation. The method chosen depends on whether the real-world system is physically terminating or non-terminating. A terminating system has some well-defined event that determines an interval of time over which statistics are gathered, for example a bank that stops accepting new customers after 5 p.m. A non-terminating system has no terminating event, for example a hospital emergency room that accepts patients round the clock.

A terminating simulation is implemented for a physically terminating system. The simulation runs until the terminating event occurs. Approximate IID observations are obtained by performing replications using the same initial conditions and different random seeds for each.

For a physically non-terminating system a steady-state simulation is implemented. Steady-state means that the system has run long enough that the distribution of the random variable of interest becomes constant. MacDougall (1987 section 4.6) discusses two methods for obtaining approximate IID observations from a steady-state simulation

are the batch means method and the independent replication method. Batch means simulations obtain their observations from a single simulation run. Independent replication simulations obtain their observations either from a series of runs or from a single run where the system is reset to the initial conditions several times to obtain the replications.

In the batch means method one simulation run is made and the observations are divided into batches each containing an equal number of observations. Sample means are formed from each batch and a grand mean and confidence interval is constructed from the batch means. Because the individual observations are not independent, there will be correlation between batches and the batch means will not be IID. However if the batch size is made large enough, they will be approximately uncorrelated.

In the independent replications method the observations are IID but because the system is restarted for each replication, the system must be run long enough to reach steady state before the observations are made. This initial period of system operation when no observations are made is called the run-up period.

Both methods have their strengths and weaknesses and require the modeller to make some arbitrary choices. The batch means method avoids the run-up problem since the simulation is started only once, but the batch size must be set large enough to approximate IID observations. The independent replication method gives IID observations but the run-up period must be long enough to ensure steady state is reached. Which method is chosen is a matter of personal preference. My personal preference is the independent replication method using reset points during a single simulation to obtain the independent replications.

The TDRS has characteristics of both a terminating and a non-terminating system. It runs on a continuous basis so it is non-terminating in that regard, but each shift defines an interval over which an observation can be made, so it is terminating in that regard. Thus the gathering of model outputs reflects the terminating and non-terminating nature of the system. Its non-terminating nature means that the system should have a run-up period before statistics gathering occurs so that it will be in a steady state. The effect of staff changes during each shift means many of the model outputs will have different values during the different shifts so that these outputs should be collected for each shift. The model outputs that are collected over each shift are average queue lengths of troubles waiting for service, average number of troubles of each criticality level in the system and staff utilization.

Collection of receipt-to-close data is handled differently. A trouble arriving during non-business hours can expect to have a longer wait for service, particularly for level 2 service. Thus a shift-dependent effect is expected for receipt-to-close times, but one corresponding to the shift of *arrival*, not the shift of completion of service. In order to observe this effect, receipt-to-close data is collected over the entire steady-state period, not just at the end of each shift. When a trouble completes its service, its receipt-to-close time is added to the statistical counters for its shift of arrival. In order to observe the carry-over effect of the reduced availability of weekend staff on the early days of the week, receipt-to-close data is gathered over all 21 shifts of the week.

At the end of a replication means and standard deviations are formed for the statistics. As discussed earlier the observations used to form the means are likely positively correlated, meaning they do not have the IID property necessary for forming a

confidence interval. However, the means for the same statistic collected over different replications do form a set of IID observations. Thus grand means of these means over all the replications can be formed, with one observation generated per replication. Grand means of the standard deviations can also be formed. Confidence intervals can be formed for both grand means. The grand mean of the means gives a measure of the central tendency of the statistic and the grand mean of the standard deviations gives a measure of the variability of the statistic.

The formation of grand means from the previous paragraph is perhaps better understood symbolically. Each replication of an experiment yields a mean $\overline{X}$ and a standard deviation S. Now for R replications over the course of the experiment two sets of measurements $\overline{X}_1, \overline{X}_2 \ldots \overline{X}_R$ and $S_1, S_2 \ldots S_R$ are obtained. From these two grand means $\overline{\overline{X}}$ and $\overline{S}$ are formed along with two standard deviations $S_{\overline{X}}$ and S_S. From these two confidence intervals $\overline{\overline{X}} \pm \frac{t_{R-1} S_{\overline{X}}}{\sqrt{R}}$ and $\overline{S} \pm \frac{t_{R-1} S_S}{\sqrt{R}}$ can be formed.

Figure 4.11 below summarises the structure of the program for statistics gathering. After the simulation starts there is a run-up period during which no statistics gathering occurs. At the end of the run-up the statistics counters are reset. Statistics gathering occurs during the steady-state period. Mean queue length and utilization statistics are gathered for each of the six different shifts (night, day and evening for weekdays and weekends). Receipt-to-close data are gathered as troubles finish their service. At the end of the steady-state period the means and standard deviations of the collected statistics are collected into the grand mean statistics and troubles remaining in the system are discarded. This completes one replication. If more replications are

remaining the simulation enters the run-up period again. When all replications are performed the simulation finishes and a report is generated.

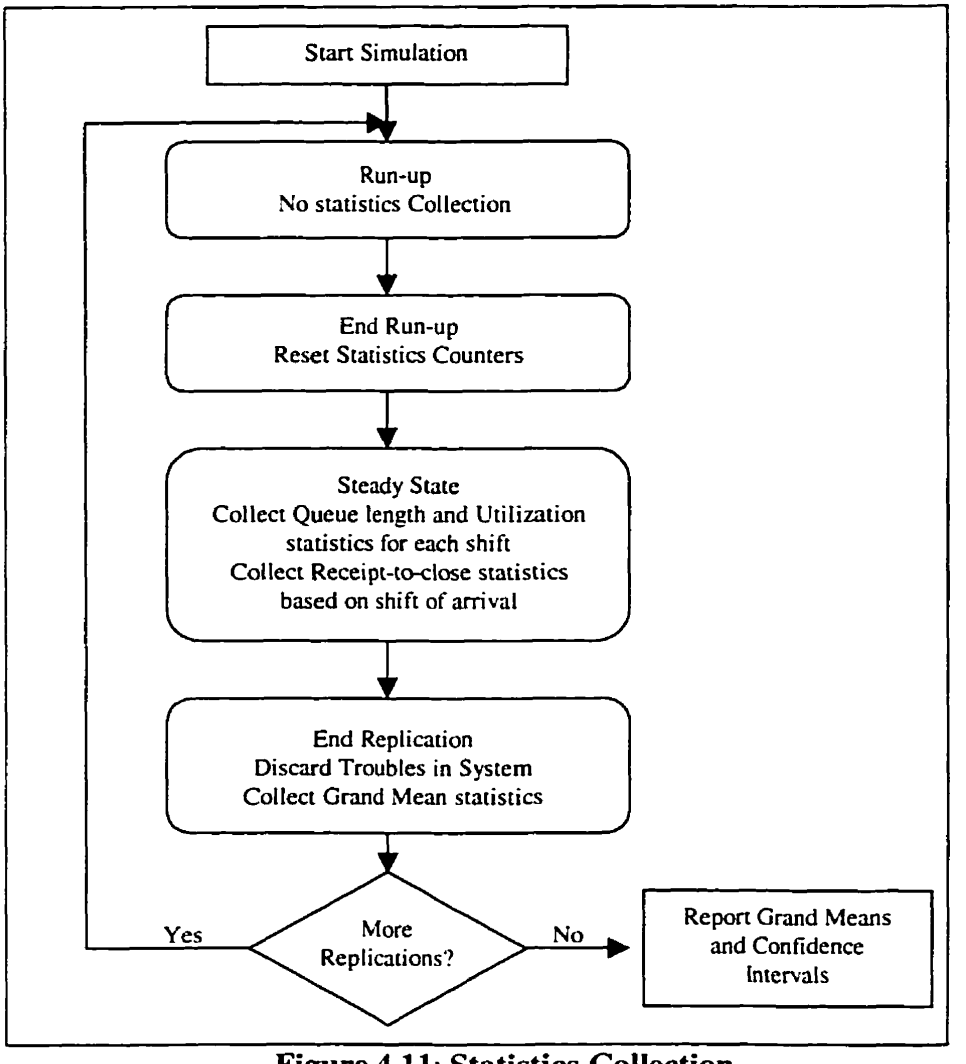

Figure 4.11: Statistics Collection.

4.7: VERIFICATION OF THE IMPLEMENTATION

Verification seeks to determine if the implementation correctly implements the logical model. Correct implementation implies two things: correct mechanism and correct instrumentation. Correct mechanism means the sequence of events produced by the implementation is correct and correct instrumentation means the model outputs are

gathered and reported correctly. Verifying the correctness of the mechanism is accomplished by producing trace output that gives a concise snapshot of the system as various events occur and inspecting it to ensure the logical model is followed. Verifying the correctness of the instrumentation is accomplished by recording the values of event-actuated observations so that outputs can be calculated and compared to the outputs produced by the implementation.

4.7.1: Description of the Trace Output

Because there are many events occurring during even a short simulation run, care must be taken to make the output as brief as possible while still including enough information to do the verification. The trace information should provide the following information:

1. Time of the event.
2. Description of the event.
3. Which entity/entities are involved.
4. Value of state variables (e.g. queue lengths).
5. Value of event-actuated quantities (e.g. delays).

The trace output in my implementation produces one line of text each time an event of interest occurs during the simulation. Several different lines are produced depending on the event. Following are examples of the types of trace lines produced, a description of the information included in the trace line and the events that produce the trace line.

4.7.1.1: TRACE TYPE I

Figure 4.12 below is an example of trace type I. It is the most common trace line and is produced when an event that affects the flow of a trouble through the system occurs. Trouble flow results in changes to state information such as queue lengths and number of busy staff so state information is included in this trace. The first two lines are column headings printed at the beginning of the trace and the last is the actual trace line.

```
Time      Trouble                        service staff info (idle, working, queued)
hh:mm:ss id #  level        EVENT       level  NOC       DSG      tech     tkt
--------|-----|------|-----------------|----|--------|--------|--------|----
08:40:48  1193 major   starting service lvl1(1  1  0)(0  1  9)(2  0  0)(  0)
```

Figure 4.12: Trace Type I.

The information fields displayed, from left to right, are as follows:

1. Time of day in hour, minute, second format (hh:mm:ss).
2. Trouble identifier number.
3. Trouble criticality.
4. Event description.
5. Service level currently servicing the trouble.
6. State of level 1 service (a triple consisting of number of staff idle, number of staff working and number of troubles queued for service).
7. State of level 2 service (as 6 above).
8. State of technician service (as 6 above).
9. Number of ticketed troubles.

The following events will produce a trace type I line:

- Trouble entering system.
- Trouble queued for service.

- Trouble starting service.
- Trouble leaving system.
- Trouble service preempted.
- Trouble resuming preempted service.
- Escalation to next level of service.
- Ticketing a trouble.
- Discarding a trouble.

4.7.1.2: TRACE TYPE II

Below is an example of trace type II. This trace line is produced when a trouble finishes a service delay (either a study delay or a repair delay). The state information from trace type I is replaced by the length of the delay.

```
10:29:46  1198 minor   trouble hardware lvl1  study time (hh:mm:ss) 00:08:59
```

The first five fields of information are identical to trace type I, with the exception of field four, which displays the outcome of the trouble study. The remaining information describes the type of delay and its duration.

4.7.1.2: TRACE TYPE III

Below is an example of trace type III. It is produced when a new shift begins and displays the number of staff arriving for the new shift and the number of currently busy staff. This information is useful when observing the system over a shift change to verify that staff working over a shift change are released when their current service finishes.

```
7.00000 new shift:   168 hrs lvl1( 2, 2) lvl2( 0, 0) tech( 2, 2)
```

The information fields displayed, from left to right, are as follows:

1. Simulation time in days
2. The string "new shift".
3. Simulation time in hours.
4. Staff information for level 1 service (a double consisting of number of staff for the new shift and number of currently working staff).
5. Staff information for level 2 service (as 4 above).
6. Staff information for technician service (as 4 above).

4.7.2: VERIFICATION OF CORRECT MECHANISM

The volume of information produced by the trace output for even a short simulation is quite large; a simulation running two weeks and producing 1354 troubles yields about 12,000 lines of trace output. Sorting through this much information line by line would be tedious and time-consuming. The approach taken is to identify some scenarios and the correct trouble flow that should occur for each scenario. Then examples of these scenarios are found within the trace output and examine them for correctness. Following is a list of these scenarios and a demonstration of correctness using excerpts from trace output.

4.7.2.1: TROUBLE FLOW IN AN UNCONGESTED SYSTEM

When a trouble arriving for service at a service level finds staff available, its service should begin immediately. The service received depends on the trouble flow for the particular criticality level of the trouble and whether or not it arrives during business hours or non-business hours, but it will still be performed immediately.

Time	Trouble			service staff info (idle, working, queued)										
hh:mm:ss	id #	level	EVENT	level	NOC			DSG			tech			tkt
00:00:00	1	cust	entering system		(2	0	0)	(0	0	0)	(2	0	0)	(0)
00:00:00	1	cust	starting service	lvl1	(1	1	0)	(0	0	0)	(2	0	0)	(0)
00:01:10	2	minor	entering system		(1	1	0)	(0	0	0)	(2	0	0)	(0)
00:01:10	2	minor	starting service	lvl1	(0	2	0)	(0	0	0)	(2	0	0)	(0)
00:05:06	2	minor	trouble hardware	lvl1	study time (hh:mm:ss)									00:03:56
00:05:06	2	minor	trouble hardware	lvl1	(0	1	0)	(0	0	0)	(2	0	0)	(0)
00:05:06	2	minor	starting service	tech	(1	1	0)	(0	0	0)	(1	1	0)	(0)
00:08:23	1	cust	trouble resolved	lvl1	study time (hh:mm:ss)									00:08:23
00:15:40	2	minor	trouble resolved	tech	study time (hh:mm:ss)									00:10:33
00:16:51	1	cust	trouble resolved	lvl1	repair time (hh:mm:ss)									00:08:28
00:16:51	1	cust	trouble resolved	lvl1	(1	1	0)	(0	0	0)	(1	1	0)	(0)
00:16:51	1	cust *	service complete		(2	0	0)	(0	0	0)	(1	1	0)	(0)
00:25:14	2	minor	trouble resolved	tech	repair time (hh:mm:ss)									00:09:33
00:25:14	2	minor	trouble resolved	tech	(2	0	0)	(0	0	0)	(1	1	0)	(0)
00:25:14	2	minor *	service complete		(2	0	0)	(0	0	0)	(2	0	0)	(0)

Figure 4.13: Trouble Flow in an Uncongested System.

Obvious choices to examine are the first troubles arriving into the system. There will certainly be level 1 staff available, so it should be serviced immediately. Figure 4.13 shown above traces the first two troubles from their entry to exit from the system. Trouble 1 enters the system at time 0, shown in trace line 1. Note the state information reflects the state of the system before any service is begun for trouble 1. It shows 2 level 1 staff, 0 level 2 staff and 2 technicians available. Trace line 2 shows trouble 1 starting the study phase of its service at time 0 as it should. Now the state of the level 1 service shows 1 staff available and 1 working. During the study phase for trouble 1, trouble 2 arrives into the system. Although it is a minor trouble and it would otherwise be ticketed during non-business hours, since there are no other troubles in the system it also starts its service immediately.

Trouble 2 completes its study phase first, with the result that a hardware fault is diagnosed, shown in trace line 5 (at time 00:05:06). Thus the repair phase of its level 1 service is not performed. It proceeds to the technician service that also has available staff so it begins receiving service immediately.

Trouble 1 completes its study phase at time 00:08:23 (trace line 8). The result of the study phase is that the trouble can be resolved, so the repair phase begins. This phase is complete at time 00:16:51 (trace line 10). Line 11 shows the system's state just as trouble 1 finishes its level 1 service; line 12 shows the state as the trouble leaves the system.

During the repair phase of the level 1 service for trouble 1, trouble 2 completes the study phase of its technician service at time 00:15:40 (trace line 9). The trouble can be resolved during this service so its repair phase begins immediately. Trace lines 13 to 15 (time 00:25:14) show the completion of the repair phase and the exit of trouble 2 from the system.

4.7.2.2: TROUBLE FLOW IN A CONGESTED SYSTEM

As troubles arriving for service find no available staff, they must join a queue until staff becomes available. In Figure 4.14 below, trouble 822 completes the study phase of its level 2 service with a hardware fault being the outcome. Trace line 2 shows that there are 0 technicians available, 2 technicians working and 1 trouble already queued for service. Line 3 shows trouble 822 joining the queue for technician service. Since it is a minor trouble, it is placed at the end of the queue, and the other trouble currently first in the queue will begins its service first. At time 12:23:19 (trace line 9) trouble 821 finishes its technician service and the first trouble in the technician queue (trouble 820) resumes its preempted service. At time 12:23:42 (trace line 12) trouble 823 finishes its technician service, allowing trouble 822 to begin its technician service.

```
Time      Trouble                    service staff info (idle, working, queued)
hh:mm:ss id #  level        EVENT      level  NOC       DSG      tech     tkt
--------|-----|------|-----------------|----|--------|--------|--------|----
12:02:28   822 minor   trouble hardware lvl2  study time (hh:mm:ss) 00:21:13
12:02:28   822 minor   trouble hardware lvl2(3  1  0)(1  0  0)(0  2  1)(  0)
12:02:28   822 minor queued for service tech(3  1  0)(2  0  0)(0  2  2)(  0)
12:08:46   821 major   trouble resolved tech  study time (hh:mm:ss) 00:10:34
12:08:57   824 major   trouble resolved lvl1  study time (hh:mm:ss) 00:15:36
12:18:18   823 cust    trouble resolved tech  study time (hh:mm:ss) 00:19:46
12:23:19   821 major   trouble resolved tech repair time (hh:mm:ss) 00:14:33
12:23:19   821 major   trouble resolved tech(3  1  0)(2  0  0)(0  1  2)(  0)
12:23:19   821 major * service complete     (3  1  0)(2  0  0)(1  1  2)(  0)
12:23:19   820 minor   resuming service tech(3  1  0)(2  0  0)(0  2  1)(  0)
12:23:42   823 cust    trouble resolved tech repair time (hh:mm:ss) 00:05:23
12:23:42   823 cust    trouble resolved tech(3  1  0)(2  0  0)(0  1  1)(  0)
12:23:42   823 cust  * service complete     (3  1  0)(2  0  0)(1  1  1)(  0)
12:23:42   822 minor   starting service tech(3  1  0)(2  0  0)(0  2  0)(  0)
```

Figure 4.14: Trouble Flow in a Congested System.

4.7.2.3: PREEMPTION OF A TROUBLE

A level 3 or 4 trouble arriving for service and finding no available staff can preempt the service of a level 1 or 2 trouble. The preempted trouble rejoins the service queue but will resume its service before any queued troubles of the same criticality level. In Figure 4.15 below trouble 148 arrives for technician service but finds none available with 2 working and 3 queued. Minor trouble 89 has its service preempted in trace line 2, making one technician staff available and increasing the queue to 4. Trouble 148 immediately begins its service in trace line 3. In trace line 4 trouble 83 finishes the repair phase of its technician service, and in trace line 6 it completes its service. Trouble 89 resumes its preempted service in trace line 7, even though it is a minor trouble and there are 3 other troubles waiting for technician service. This is correct only if the other 3 troubles are also minor. Checking backwards through the trace output reveals the last 3 troubles queued for technician service are numbers 110, 109 and 146, all minor troubles

```
Time      Trouble                        service staff info (idle, working, queued)
hh:mm:ss id #  level        EVENT       level  NOC       DSG       tech      tkt
--------|-----|------|-----------------|----|--------|--------|--------|----
10:19:01   148 major    trouble hardware lvl1(2  1  0)(1  1  0)(0  2  3)(  0)
10:19:01    89 minor  -trouble preempted tech(3  1  0)(1  1  0)(1  1  4)(  0)
10:19:01   148 major    starting service tech(3  1  0)(1  1  0)(0  2  4)(  0)
10:19:26    83 minor    trouble resolved tech repair time (hh:mm:ss) 00:17:54
10:19:26    83 minor    trouble resolved tech(3  1  0)(1  1  0)(0  1  4)(  0)
10:19:26    83 minor  * service complete     (3  1  0)(1  1  0)(1  1  4)(  0)
10:19:26    89 minor    resuming service tech(3  1  0)(1  1  0)(0  2  3)(  0)
```

Figure 4.15: Preemption of a Trouble.

4.7.2.4: OBTAINING LEVEL 2 STAFF BY CALL-OUT

When a major or critical trouble arrives for level 2 service during non-business hours, an attempt is made to obtain level 2 staff on a call-out basis. If this is successful, service begins immediately; otherwise the trouble joins the service queue and must wait until the next shift change until another call-out is made or level 2 staff are scheduled to work. Once the call-out service is complete, the level 2 staff is not made available to any other troubles waiting for level 2 service. Figure 4.16 below illustrates this. The entire trace sequence is not shown due to the large number of trace lines in the sequence.

Trace line 1 shows major trouble 231 needing level 2 service with none available. Line 2 shows that the call-out was successful, so service begins immediately. In trace line 4 the study phase of the service is complete, with the result that the trouble can be resolved, so the repair phase begins. During the repair phase a new shift begins with 2 level 2 staff beginning work. Troubles 210 and 170 immediately begin their level 2 service with the result that a total of 3 level 2 staff are working (trace line 7). When the repair phase of the service for trouble 231 is complete, the trouble leaves the system, but the staff that was servicing it does not remain in the system. Thus the number of busy staff remains at 2, which is the staff level specified for the shift.

```
Time      Trouble                    service staff info (idle, working, queued)
hh:mm:ss id #  level        EVENT      level  NOC       DSG       tech     tkt
--------|-----|------|----------------|----|--------|--------|--------|----
06:40:14   231 major trouble unresolved lvl1(0  1  2)(0  0  6)(2  0  0)(  0)
06:40:14   231 major lvl2 on-call found     (0  2  1)(1  0  6)(2  0  0)(  0)
06:40:14   231 major   starting service lvl2(0  2  1)(0  1  6)(2  0  0)(  0)
07:15:50   231 major   trouble resolved lvl2  study time (hh:mm:ss) 00:35:35
2.33333 new shift:    56 hrs lvl1( 4, 2) lvl2( 2, 1) tech( 2, 0)
08:00:00   210 crit    starting service lvl2(3  3  0)(1  2  7)(2  0  0)(  0)
08:00:00   170 cust    starting service lvl2(3  3  0)(0  3  6)(2  0  0)(  0)
08:15:47   231 major   trouble resolved lvl2 repair time (hh:mm:ss) 00:59:57
08:15:47   231 major   trouble resolved lvl2(1  3  0)(0  2  8)(2  0  0)(  0)
08:15:47   231 major * service complete     (1  3  0)(0  2  8)(2  0  0)(  0)
```

Figure 4.16: Level 2 Call-out.

4.7.2.5: COOPERATIVE SERVICE

Staff from two different service levels may work together on a trouble, usually with the senior staff providing assistance intermittently until the trouble is resolved. This is modelled by returning a trouble to a previous level of service. In Figure 4.17 below trouble 7 begins with level 1 service where a hardware fault is determined. It proceeds to technician service where the fault is unresolved. This causes it to go back for level 1 service again, where this time the trouble is resolved and its service is complete.

```
Time      Trouble                    service staff info (idle, working, queued)
hh:mm:ss id #  level        EVENT      level  NOC       DSG       tech     tkt
--------|-----|------|----------------|----|--------|--------|--------|----
01:12:02     7 minor    entering system     (0  2  0)(0  0  0)(1  1  0)(  0)
01:12:02     7 minor ^ trouble ticketed lvl1(0  2  0)(0  0  0)(1  1  0)(  1)
01:13:50     7 minor   starting service lvl1(0  2  0)(0  0  0)(1  1  0)(  0)
01:18:21     7 minor   trouble hardware lvl1  study time (hh:mm:ss) 00:04:31
01:18:21     7 minor   trouble hardware lvl1(0  1  0)(0  0  0)(1  1  0)(  1)
01:18:21     7 minor   starting service tech(0  2  0)(0  0  0)(0  2  0)(  0)
01:36:34     7 minor trouble unresolved tech  study time (hh:mm:ss) 00:18:12
01:36:34     7 minor trouble unresolved tech(0  2  0)(0  0  0)(0  1  1)(  0)
01:36:34     7 minor queued for service lvl1(0  2  1)(0  0  0)(0  2  0)(  0)
01:43:43     7 minor   starting service lvl1(0  2  0)(0  0  0)(1  1  0)(  0)
01:51:55     7 minor   trouble resolved lvl1  study time (hh:mm:ss) 00:08:12
02:01:48     7 minor   trouble resolved lvl1 repair time (hh:mm:ss) 00:09:53
02:01:48     7 minor   trouble resolved lvl1(0  1  0)(0  0  0)(2  0  0)(  0)
02:01:48     7 minor * service complete     (1  1  0)(0  0  0)(2  0  0)(  0)
```

Figure 4.17: Cooperative Service

4.7.2.6: START OF A NEW SHIFT

The start of a new shift sees new workers arrive and ones previously on duty leave. The correct mechanism is to assign the new workers immediately to waiting troubles and for workers previously working on troubles to finish the service and leave the system. Figure 4.18 below illustrates this, for a day shift in this case. The first trace line (time 06:40:14) shows a successful call-out to obtain level 2 staff. The second trace line shows the trouble beginning service and the state information for level 2 shows 0 idle, 1 working and 6 troubles queued for service. The third trace line (time 07:15:50) shows the completion of the study phase of the service with the result that the trouble can be resolved. The fourth trace line (time 07:40:40) shows the state of the system just before the new shift. It shows 0 level 1 staff idle, 2 working and no troubles queued for level 1 service, 0 level 2 staff idle, 1 working (the call-out staff) and 8 troubles queued for level 2 service, 2 technicians idle, 0 working and no troubles queued for technician service, and 1 ticketed trouble. Trace line 4 shows the new shift with 4 new level 1 staff, 2 level 2 staff and 2 technicians. The next 3 lines show the new staff starting service on waiting troubles, the first from the ticket queue and the next two from the level 2 queue. The third of these lines shows 6 level 1 staff (4 new and 2 from the last shift) and 3 level 2 staff (2 new and 1 from the last shift). At trace line 9 (time 08:15:47) trouble 231 finishes its call-out service. The last line shows trouble 231 leaving the system. It shows the number of level 2 staff now at 2, demonstrating that the call-out staff has left the system. Note also that the level 1 staff is now at 4, indicating that the two troubles being served from the last shift have also completed their service.

```
Time      Trouble                    service staff info (idle, working, queued)
hh:mm:ss id #  level       EVENT      level  NOC       DSG       tech     tkt
--------|-----|------|----------------|----|--------|--------|--------|----
06:40:14   231 major lvl2 on-call found     (0  2  1)(1  0  6)(2  0  0)(  0)
06:40:14   231 major   starting service lvl2(0  2  1)(0  1  6)(2  0  0)(  0)
07:15:50   231 major   trouble resolved lvl2  study time (hh:mm:ss) 00:35:35
07:40:40   237 cust    starting service lvl1(0  2  0)(0  1  8)(2  0  0)(  1)
2.33333 new shift:    56 hrs lvl1( 4, 2) lvl2( 2, 1) tech( 2, 0)
08:00:00   238 minor   starting service lvl1(3  3  0)(0  1  8)(2  0  0)(  0)
08:00:00   210 crit    starting service lvl2(3  3  0)(1  2  7)(2  0  0)(  0)
08:00:00   170 cust    starting service lvl2(3  3  0)(0  3  6)(2  0  0)(  0)
08:15:47   231 major   trouble resolved lvl2 repair time (hh:mm:ss) 00:59:57
08:15:47   231 major   trouble resolved lvl2(1  3  0)(0  2  8)(2  0  0)(  0)
08:15:47   231 major * service complete     (1  3  0)(0  2  8)(2  0  0)(  0)
```

Figure 4.18: New Shift

4.7.3: VERIFICATION OF CORRECT INSTRUMENTATION

In addition to trace output, the implementation also produces output files containing the values of event-actuated statistics that can be imported to a spreadsheet program. The spreadsheet can be used to determine the count, mean and standard deviation of the statistics for comparison to the outputs reported by the implementation. Both endogenous and exogenous variables are checked in this manner. The implementation produces six such output files, described in Table 4.3 below.

File	Contents
Arrival.out	Interarrival times for all troubles
Rtoc-1.out	Receipt-to-close times for level 1 troubles (steady state)
Rtoc-2.out	Receipt-to-close times for level 2 troubles (steady state)
Rtoc-3.out	Receipt-to-close times for level 3 troubles (steady state)
Rtoc-4.out	Receipt-to-close times for level 4 troubles (steady state)
Rtoc-5.out	Receipt-to-close times for all troubles (steady state)

Table 4.3: Output statistics files

The same trace output used for the mechanism verification also produced the output files used for instrumentation verification. Figure 4.19 below shows the event-actuated statistics reported by the implementation and Table 4.4 shows the statistics calculated by the spreadsheet. Comparison of the results shows agreement in both the count (base) and mean (average) columns. The standard deviation column shows some difference between the receipt-to-close data ranging from about 0.10% for the overall values to about 1.70% for the critical values. This is likely due to loss of precision in calculation of the standard deviation. Receipt-to-close times are highly variable; for example the receipt-to-close times for critical troubles range from 44 to 1054 minutes. Calculation of the standard deviation involves calculating the sum of squares of the individual values, so the contribution of the square of a small number would be insignificant compared to the contribution of the square of a large number.

Exogenous Variables

	Base	Average	Std.Dev.
Interarrivals	1354	14.9025370 (min)	14.6381126 (min)

Receipt-to-Close Data

criticality	count	average (min)	s.d.
minor	343	146.204	350.027
customer	167	151.398	437.047
major	139	112.206	197.433
critical	30	203.239	218.149
overall	679	143.042	345.440

Figure 4.19: Event-actuated Statistics from trace run

Quantity	Count	Mean (min)	Std. Dev. (min)
Interarrival times	1354	14.90	14.64
Receipt-to-close level 1	343	146.20	350.54
Receipt-to-close level 2	167	151.40	438.36
Receipt-to-close level 3	139	112.20	198.15
Receipt-to-close level 4	30	203.24	221.88
Receipt-to-close overall	679	143.04	345.70

Table 4.4: Event-actuated Statistics from Spreadsheet

4.8: THE GRAPHICAL INTERFACE

The SIMGRAPHICS graphical interface language allows the building of a graphical interface within the implementation without requiring knowledge of programming for a graphical environment. The modeller uses the SIMDRAW graphical editor to create graphic objects like forms for input and output and dynamic display objects like meters and plots. The editor itself is a graphical tool so that no programming is required to create the graphic objects. The objects are saved into a separate file. The modeller can then access the forms through the use of function calls within the implementation. Dynamic display objects are associated with implementation variables by declaring them as DISPLAY variables and associating them with a display object using the SHOW statement. The following sections give a representative sample of the graphical interface.

4.8.1: THE DYNAMIC DISPLAY

The dynamic display is intended to give the user clear, concise feedback on the progress of the simulation. The dynamic display is also useful for diagnosing problems with the implementation that might otherwise be hard to find. Figure 4.20 shows the dynamic display.

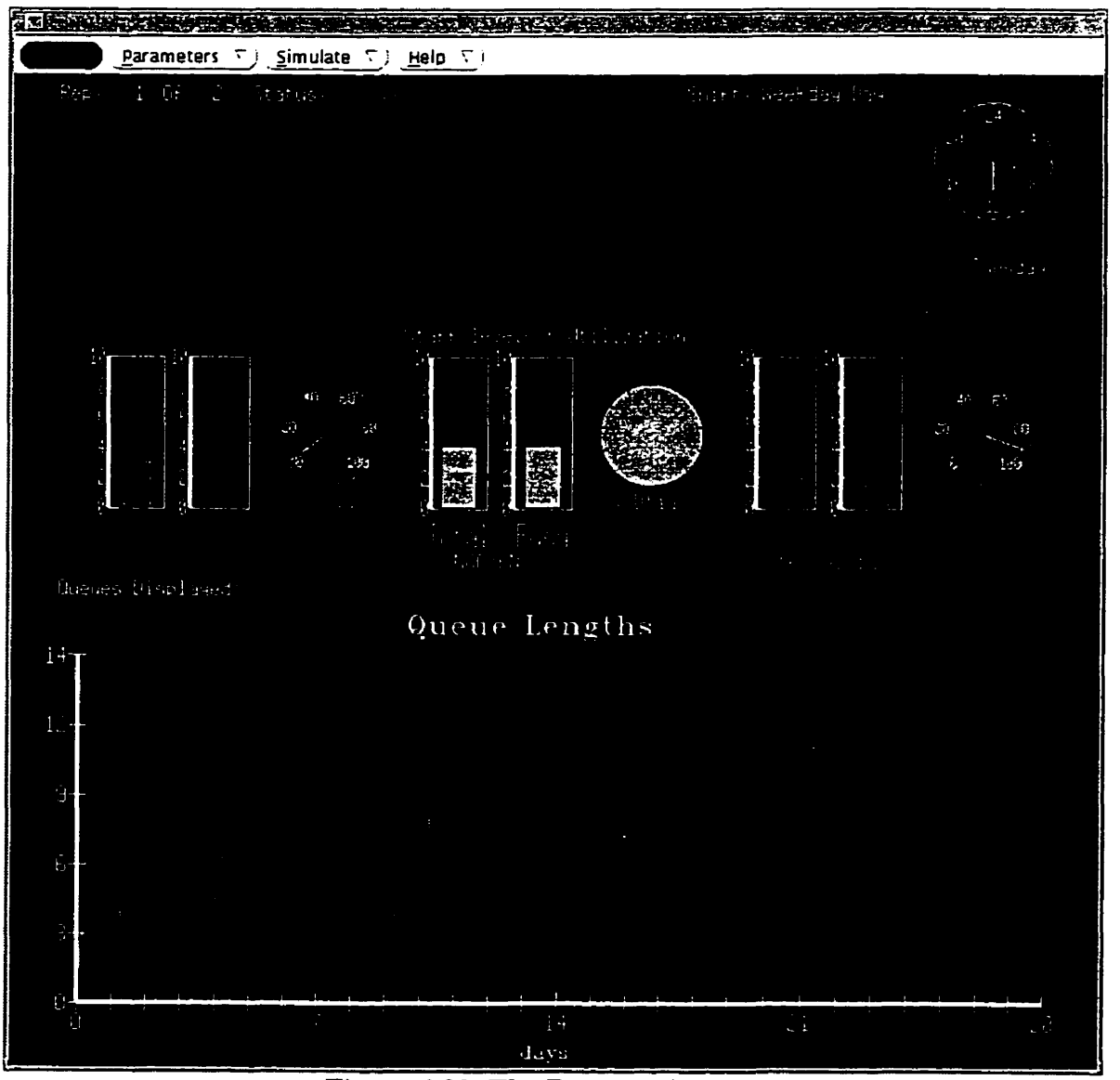

Figure 4.20: The Dynamic Display.

The dynamic display is divided into several sections. Along the top of the display is the menu bar. The File menu contains commands usually found here such as saving the current model input and quitting. The Parameter menu allows the user to change the model parameters. The Simulate menu contains the Run command and a no-graphics Run command when the user wants to simply collect model output data. Also under the Simulate menu are commands to let the user select which queue plots are displayed and to select the speed of the dynamic display. The Help menu allows the user to access text descriptions of the implementation.

Below the menu bar is some text information describing the status of the independent replications. To the right of this is a 24-hour clock displaying the simulation time and text information describing the current shift and day of week.

The middle part of the display contains a series of bar graphs and dials describing the status of the service levels. Each of the level 1, level 2 and technician levels (labelled NOC/PNOC, DSG and technician respectively) has a set of two bar graphs and one dial. The bar graphs display the current number of staff available and working. The dial displays the current utilization of the service level.

The bottom part of the display contains a trace plot of queue lengths. Up to four plots can be displayed at once. Immediately above the trace plot is a legend indicating the plot or plots displayed. The user can choose to display plots by service or criticality. If service is chosen, the plots display the true queue for each service, i.e. troubles waiting for service but not those being served. If criticality is chosen, the plots display the total queue (number in system) for each criticality, i.e. both troubles waiting for service and those being served.

The trace plots that can be displayed for each category of trace plots are:

- service type (level 1, level 2, technician, ticketed)
- criticality (level 1, level 2, level 3, level 4)

The user can choose to display any subset of the four plots.

4.8.2: Input Forms

This section gives some representative examples of input forms showing the use of standard graphical input objects. It should be noted that the user may specify all the model inputs using such forms. The complete set of user inputs is described in the next chapter where the determination of a reasonable set of baseline parameters is described.

Figure 4.21 shows the form used to input model input for trouble arrivals. The pie chart above the form is separate from the form and is used to give the user graphical feedback about the current mix of criticality levels chosen. This form uses text boxes to accept typed data from the user; text boxes are also used to provide category labels on the form. Buttons are used to accept input, cancel the changes or to update the pie chart display.

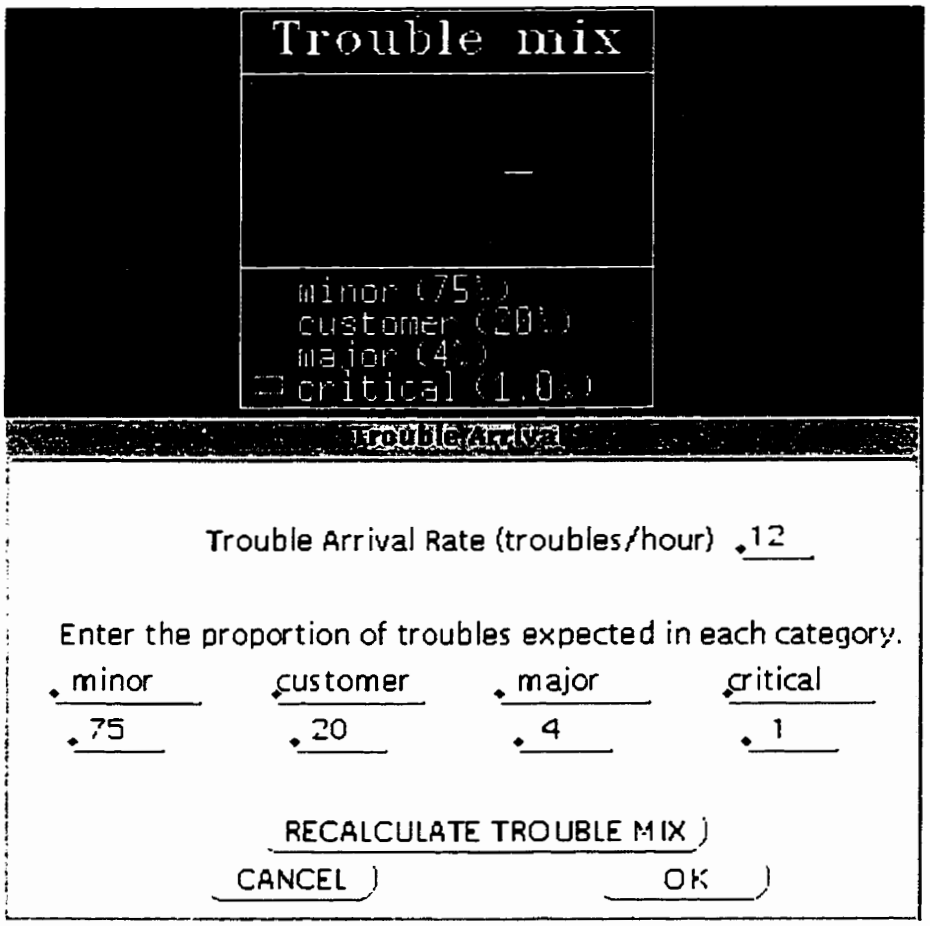

Figure 4.21: Trouble Arrival Input Form.

Figure 4.22 shows the input form for the study matrix described in Section 4.5.3. The study matrix stores the routing information for troubles based on criticality level, service level and business/non-business shift; the three choices, study, ticket and escalate are stored as integers. Because it is an integer matrix, values could be typed in using text boxes, but since there are only three permissible values, corresponding to the DEFINE TO MEANs for study, ticket and escalate, a more user-friendly and error-proof way to accept input is to allow the user to input only those three values using radio buttons. Text boxes are again used for headings.

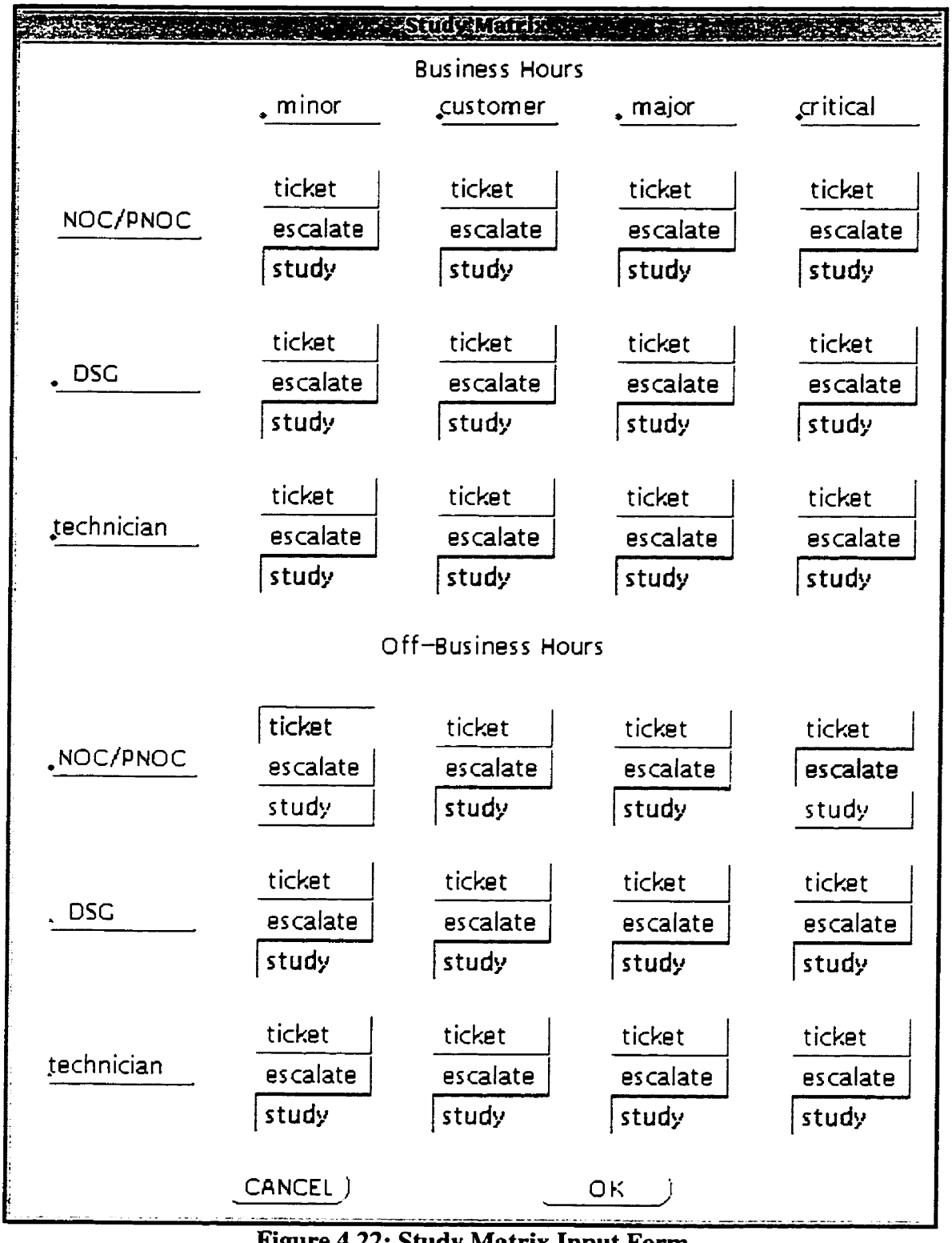

Figure 4.22: Study Matrix Input Form.

Figure 4.23 shows the input form for Run control, i.e. the number and length of replications. This form also allows the user to turn on the debug trace information. This is done using a check box.

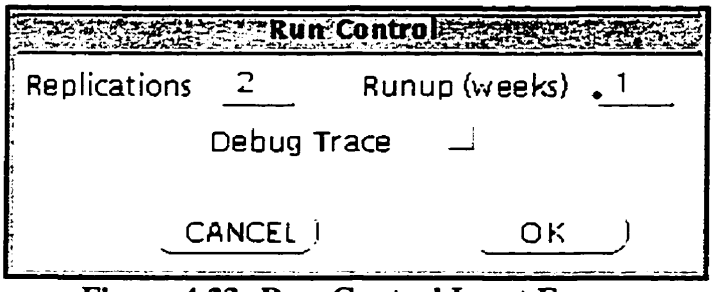

Figure 4.23: Run Control Input Form.

CHAPTER 5: DATA REQUIREMENTS AND VALIDATION

5.1: INTRODUCTION

The data requirements for the TDRS fall into two distinct categories:

- Input data for characterizing the exogenous variables.
- Output data for validation of the endogenous variables.

The first section of this chapter focuses on the former. It describes a set of model inputs derived using a set of qualitative guidelines derived from various sources. It then examines in detail the output produced and demonstrates that the inputs lead to reasonable system behaviour and a stable system.

The second section of this chapter describes the problems encountered and solutions found to derive a deterministic analysis of utilization in stable systems and demonstrate its accuracy. This has proved to be a valuable tool both for planning experiments with the TDRS simulation and for verifying the correct implementation of the logical model.

5.2: THE BASELINE SYSTEM

The first step in performing experiments with the system is to establish a set of model inputs that generate stable model outputs that can be used as a basis for comparison with the outputs from experiments. This set of inputs should be representative of the real-world system and the outputs should correspond to those observed in the real-world system. This set of inputs and the outputs generated from it are

called the *baseline system*. Ideally the baseline system is determined in close consultation with the people involved with the real-world system. The discussion of the baseline system illustrates the sort of data that is required to configure a base system from real-world data.

Typically the establishment of the baseline system is accomplished as a by-product of the process of validation. The validation phase of a simulation project typically occurs immediately following or concurrent with the verification process. Validation seeks to ensure that the implementation produces outputs that resemble those that occur in the real-world system. It is typically an iterative process involving the modeller and the people involved with the real-world system. Ideally there are real-world data available such as observed delays and throughputs that can be used to produce a quantitative measure of the closeness of the implementation and the real-world system. If such data do not exist, then validation must be qualitative, with experts in the real-world system offering opinions about the "closeness" of the implementation's outputs to the behaviour of the real-world system. The expert opinions are then incorporated into the implementation and the validation process continues until all are satisfied that the implementation produces believable outputs.

In the case of the TDRS it was initially anticipated that real-world data would be available for comparison to the implementation output. However several factors combined to make it impossible to obtain this data. Firstly it was found that MTS did indeed have large volumes of data regarding troubles and the time taken to repair them, but no mean value data was available and MTS did not have the manpower available to assemble it. Secondly, shortly after this project was begun, MTS was privatized, resulting

in several management changes. One of the changes was the retirement of the original proponent of this project. The person who inherited responsibility for it was very helpful in providing information relating to the operations of the TDRS, including internal MTS documentation describing trouble flow, a tour of the NOC and PNOC and meetings and e-mail correspondence during the initial formulation of the logical model. However the transition from Crown Corporation to publicly traded one demanded more of his time and he was transferred to other duties. Consequently I have not been able to get feedback from the people involved with the TDRS on the current implementation.

In the absence of quantitative data and qualitative feedback from experts, it was decided to establish a set of qualitative guidelines for use in determining the baseline system. These guidelines are drawn from several sources, including internal MTS documentation, feedback from a demonstration of an early implementation to NOC staff, a simulation of a Customer Trouble Reporting Process by Chen *et al* (1988) and common sense. The guidelines adopted are as follows:

1. The higher the criticality level of the trouble, the less frequent it should be.
2. The higher the criticality level of the trouble, the longer its service should be.
3. Generally, more staff may be involved in servicing high criticality troubles.
4. Level 2 staff should be more "knowledgeable" than level 1 staff, which is reflected in being able to resolve a higher proportion of troubles and in shorter service times for troubles.
5. Queue lengths should be stable on a week-to-week basis.
6. The number of critical troubles in the system at any time should be low. Having more than 2 in the system should be a rare occurrence.

7. Staff utilizations should be reasonable, say within a range of 50-75%.

5.2.1:BASELINE MODEL INPUTS

The first three of the above guidelines are directly related to model inputs. The last three, while directly related to model outputs, have an indirect effect on model inputs since it is the inputs that must be manipulated in order to follow the guidelines. The following sections describe the baseline model inputs and the use of the guidelines in determining them.

5.2.1.1: TROUBLE ARRIVALS

Figure 5.1 below shows the trouble arrival rate and the distribution of trouble criticality levels. The choice of the arrival rate is arbitrary. Other model inputs such as staffing levels, service times and resolution proportions are adjusted given a particular arrival rate. The proportion of troubles in each criticality level is consistent with guideline 1. The proportion of customer-reported troubles is small compared to minor troubles, which is consistent with the assumption that they make up a small proportion of the total troubles (see Section 2.4, Temporal Effects).

Mean Arrival Rate 8.00 (/hr)

criticality	% of total
minor	75.00
customer	15.00
major	9.00
critical	1.00

Figure 5.1: Baseline Trouble Arrivals.

5.2.1.2: STAFF LEVELS

Figure 5.2 below shows the staff levels for the baseline model. Level 1 (NOC/PNOC) service is staffed on all shifts, with an increase in staff during business hours (weekday day shift). This reflects the fact that the NOC is staffed on a round-the-clock basis, while PNOC is staffed during business hours. Level 2 (DSG) service is staffed only during business hours. Technician service is also staffed round-the-clock, with reduced staff on the weekends, because there are no DSG staff dispatching troubles to the technicians on the weekends.

The final two columns of Figure 5.2 show the likelihood of getting level 2 staff on call-out. The likelihood is higher for critical troubles than for major troubles. This is reasonable because major troubles are described as potentially service affecting (Section 2.2.3) and may result in an "over the phone" determination that the trouble can wait for regularly scheduled level 2 staff. The likelihood is zero during business hours, meaning a call-out won't be done when all the level 2 staff are busy during business hours (a common sense assumption). The likelihood is lower on weekends than on weekdays. This choice is somewhat arbitrary but it is believed that it reflects the real world fact that people are less likely to be available on weekends.

				off-hours availability DSG (%)	
shift	NOC/PNOC	DSG	technician	major	critical
weekday night	4	0	3	50	75
weekday day	6	2	3	0	0
weekday evening	4	0	3	50	75
weekend night	4	0	2	40	50
weekend day	4	0	2	40	60
weekend evening	4	0	2	40	50

Figure 5.2: Baseline Staff Levels.

5.2.1.3: TROUBLE RESOLUTION

Figure 5.3 below shows the likelihood of the possible outcomes during the study phase of service (resolved, hardware fault, unresolved). Level 1 service is able to deal with the large majority of minor and customer level troubles either immediately or by dispatching a technician; only 1 in 20 needs to be referred to level 2 service. However, a larger proportion of major and critical troubles is unresolved by level 1 service, thus involving level 2 staff (guideline 3).

The likelihood of resolving troubles at all criticality levels is higher for level 2 service than for level 1 service, reflecting their higher level of skill and experience (guideline 4). As with level 1 service, as the criticality level increases, a higher proportion of troubles are unresolved, although the proportion is smaller than for level 1 service.

The outcome of the study phase of the technician's service is that the trouble is either resolved or unresolved. The staff that dispatched the technician has already determined a hardware fault, so it makes no sense in the real-world system for the technician to determine this as well. This doesn't stop the user from entering some non-zero probability of a technician determining a hardware fault, so the *worker* routine checks for this and changes the result to unresolved (see Figure 4.8a). A determination of unresolved causes the referral of the trouble back to the previous service level (see Co-operative Service, section 4.7.2.5).

NOC/PNOC	minor	customer	major	critical
% resolved	60.00	50.00	40.00	30.00
% hardware	35.00	45.00	40.00	40.00
% unresolved	5.00	5.00	20.00	30.00
DSG	minor	customer	major	critical
% resolved	75.00	75.00	55.00	50.00
% hardware	24.00	20.00	35.00	30.00
% unresolved	1.00	5.00	10.00	20.00
technician	minor	customer	major	critical
% resolved	95.00	95.00	90.00	80.00
% hardware	0.	0.	0.	0.
% unresolved	5.00	5.00	10.00	20.00

Figure 5.3: Baseline Trouble Resolution.

5.2.1.4: TROUBLE ROUTING

Figure 5.4 below shows the study matrix for the baseline system (see Section 4.5.3). It implements the trouble flow discussed in Chapter 4. In particular note the fourth column, which implements trouble for the PNOC during non-business hours (see Figure 2.3). Minor troubles are ticketed and critical troubles are escalated to DSG or OSO.

	regular hours (M-F, 8am-4pm)			off-hours		
criticality	NOC/PNOC	DSG	technician	NOC/PNOC	DSG	technician
minor	study	study	study	ticket	study	study
customer	study	study	study	study	study	study
major	study	study	study	study	study	study
critical	study	study	study	escalate	study	study

Figure 5.4: Study Matrix.

5.2.1.5: STUDY TIMES

Figure 5.5 below shows the estimates that define the beta parameters for the study phase of service. Study times increase with increasing criticality level. Study times are also longer for level 2 staff than level 1 staff, reflecting the fact they tend to see more complex troubles. Study times are longer during non-business hours for major and critical troubles for both level 1 and level 2 staff. This is warranted since higher level staff is

either not available or must be obtained by call-out, so staff currently working on the trouble would be inclined to make an extra effort to resolve the trouble without obtaining additional staff. For the technicians, the study phase of service consists of diagnostics to confirm or localize the fault causing the trouble, which is assumed to take the same amount of time for any criticality level. Thus study times for technicians are constant for all criticality levels.

NOC/PNOC Trouble Study times (min.)

regular hours (M-F, 8am-4pm)

estimate	minor	customer	major	critical
min	3.0	3.0	10.0	10.0
mode	5.0	5.0	15.0	15.0
mean	10.0	10.0	20.0	20.0
max	15.0	15.0	30.0	30.0
off-hours				
min	3.0	3.0	10.0	10.0
mode	5.0	5.0	30.0	30.0
mean	10.0	10.0	45.0	45.0
max	15.0	15.0	60.0	60.0

DSG Trouble Study times (min.)

regular hours (M-F, 8am-4pm)

estimate	minor	customer	major	critical
min	10.0	10.0	15.0	15.0
mode	15.0	15.0	20.0	20.0
mean	20.0	20.0	30.0	30.0
max	45.0	45.0	45.0	45.0
off-hours				
min	10.0	10.0	20.0	20.0
mode	20.0	20.0	30.0	30.0
mean	30.0	30.0	45.0	45.0
max	45.0	45.0	60.0	60.0

technician Trouble Study times (min.)

regular hours (M-F, 8am-4pm)

estimate	minor	customer	major	critical
min	5.0	5.0	5.0	5.0
mode	10.0	10.0	10.0	10.0
mean	15.0	15.0	15.0	15.0
max	20.0	20.0	20.0	20.0
off-hours				
min	5.0	5.0	5.0	5.0
mode	10.0	10.0	10.0	10.0
mean	15.0	15.0	15.0	15.0
max	20.0	20.0	20.0	20.0

Figure 5.5: Beta Parameters for Trouble Study.

5.2.1.6: REPAIR TIMES

Figure 5.6 below shows the estimates that define the beta parameters for the repair phase of service. As with study times, repair times increase with increasing criticality level. For major and critical troubles repair times are longer for level 1 staff than for level

2 staff, which is opposite from the study times. This reflects the greater experience of the level 2 staff. Repair times for critical troubles are longer during non-business hours for all staff levels. In particular the maximum time for level 2 repair is doubled, since the call-out will probably require travel time by the staff involved.

PNOC Trouble Repair times (min.)

	regular hours (M-F, 8am-4pm)			
estimate	minor	customer	major	critical
min	5.0	5.0	30.0	30.0
mode	10.0	12.0	45.0	75.0
mean	15.0	18.0	60.0	100.0
max	30.0	30.0	90.0	240.0
		off-hours		
min	5.0	5.0	30.0	30.0
mode	10.0	12.0	45.0	90.0
mean	15.0	18.0	60.0	120.0
max	30.0	30.0	90.0	240.0

DSG Trouble Repair times (min.)

	regular hours (M-F, 8am-4pm)			
estimate	minor	customer	major	critical
min	5.0	5.0	20.0	30.0
mode	10.0	12.0	30.0	45.0
mean	15.0	18.0	45.0	75.0
max	30.0	30.0	60.0	120.0
		off-hours		
min	5.0	5.0	20.0	30.0
mode	10.0	12.0	30.0	60.0
mean	15.0	18.0	45.0	90.0
max	30.0	30.0	60.0	240.0

technician Trouble Repair times (min.)

	regular hours (M-F, 8am-4pm)			
estimate	minor	customer	major	critical
min	5.0	5.0	10.0	45.0
mode	10.0	12.0	20.0	75.0
mean	15.0	18.0	30.0	100.0
max	30.0	30.0	45.0	200.0
		off-hours		
min	5.0	5.0	10.0	60.0
mode	10.0	12.0	20.0	90.0
mean	15.0	18.0	30.0	120.0
max	30.0	30.0	45.0	240.0

Figure 5.6: Beta Parameters for Trouble Repair.

5.2.2:BASELINE MODEL TRACE OUTPUTS

Trace outputs are produced by the dynamic display and show how the state variables change with time. The state variables that can be displayed are troubles queued for service (level 1, level 2, technician and ticketed troubles) and total number of troubles in the system (minor, customer, major and critical). The following sections describe each of the above traces produced by the baseline system.

5.2.2.1: NUMBER OF CRITICAL TROUBLES IN SYSTEM

Figure 5.7 below shows the trace for the total number of critical troubles in the system over a six-week period. Only once does the number of critical troubles in the system exceed two, on day 28 (a Sunday). There appears to be no daily effect visible in the trace. Some evidence of a weekly effect is present with a slight build-up of troubles on weekends (around day 6, day 27 and day 34). It is not a pronounced effect however and is not present every weekend. This near absence of temporal effects is to be expected with critical troubles since they are service-affecting troubles and can't remain unserviced for any length of time.

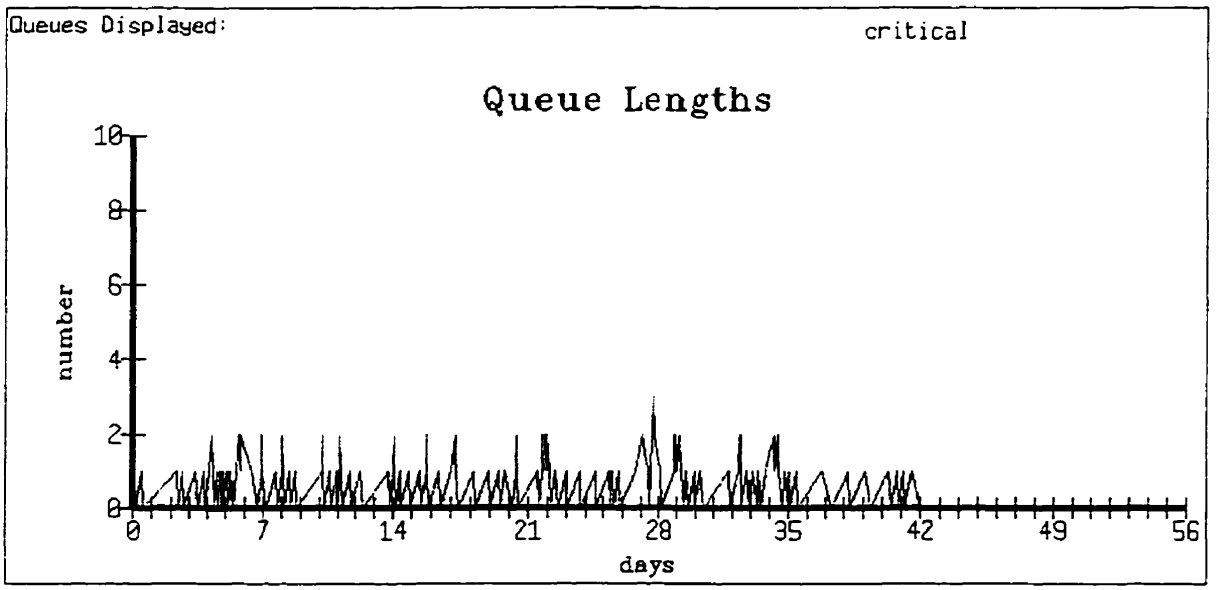

Figure 5.7: Baseline Trace for Critical Troubles in System.

It should be noted that critical troubles are a near-daily occurrence in the baseline system. In the real-world system critical troubles are an exceptional occurrence, possibly associated with some outside event like a power failure or a cable being cut. Troubles of that nature are not modelled here, and the arrival rate of critical troubles is higher than

would be expected in practice so that trends in the service of critical troubles are more easily observed.

5.2.2.2: NUMBER OF MAJOR TROUBLES IN SYSTEM

Figure 5.8 below shows the trace for the total number of major troubles in the system over a six-week period. A weekly effect is evident on the weekends where there are periods of at least one day in length in which the number of troubles in the system is never zero. This is to be expected since major troubles that require level 2 service on the weekends are less likely to obtain service by call-out and must wait until the next week. No clear daily weekday effect is visible. The number of troubles in the system is more variable than for critical troubles, which is to be expected since they occur more frequently than critical ones.

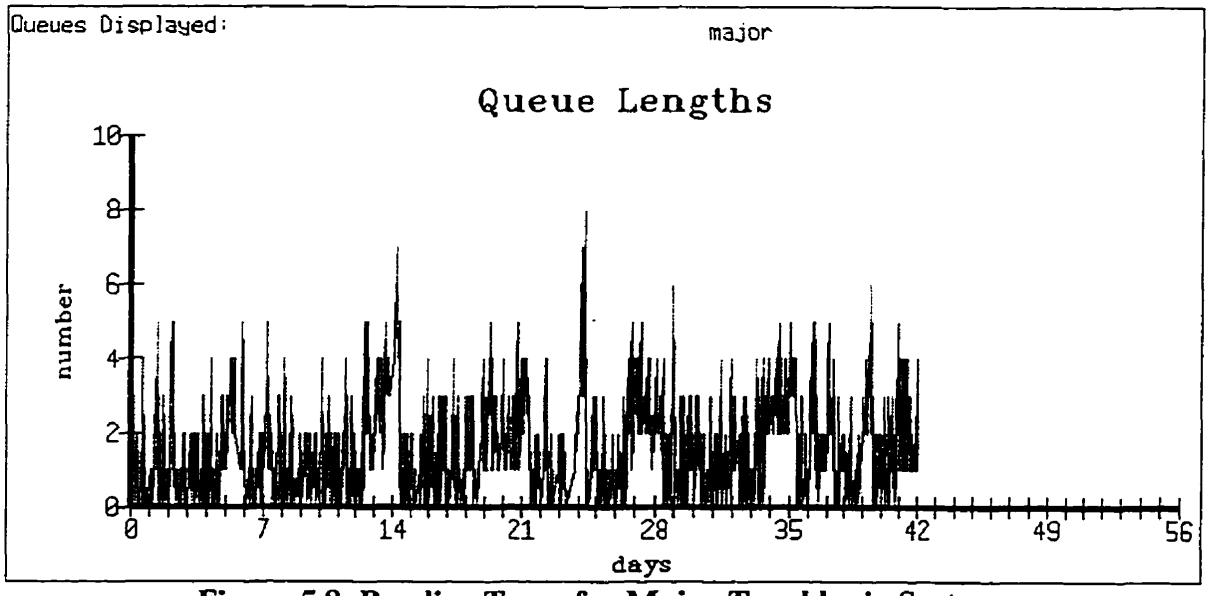

Figure 5.8: Baseline Trace for Major Troubles in System.

5.2.2.3: NUMBER OF CUSTOMER TROUBLES IN SYSTEM

Figure 5.9 below shows the trace for the total number of customer troubles in the system over a six-week period. The weekend effect observed for major troubles is present for customer troubles as well. A daily weekday effect is observed in week 6 (35 – 40 days). If one observes the tick marks for days 36 – 39 (corresponding to midnight Monday, Tuesday, Wednesday and Thursday), there are distinct areas where the trace plot does not return to 0. This effect is not clearly observed in the other weeks. The variability is similar to that for major troubles, which is consistent with their similar arrival rates.

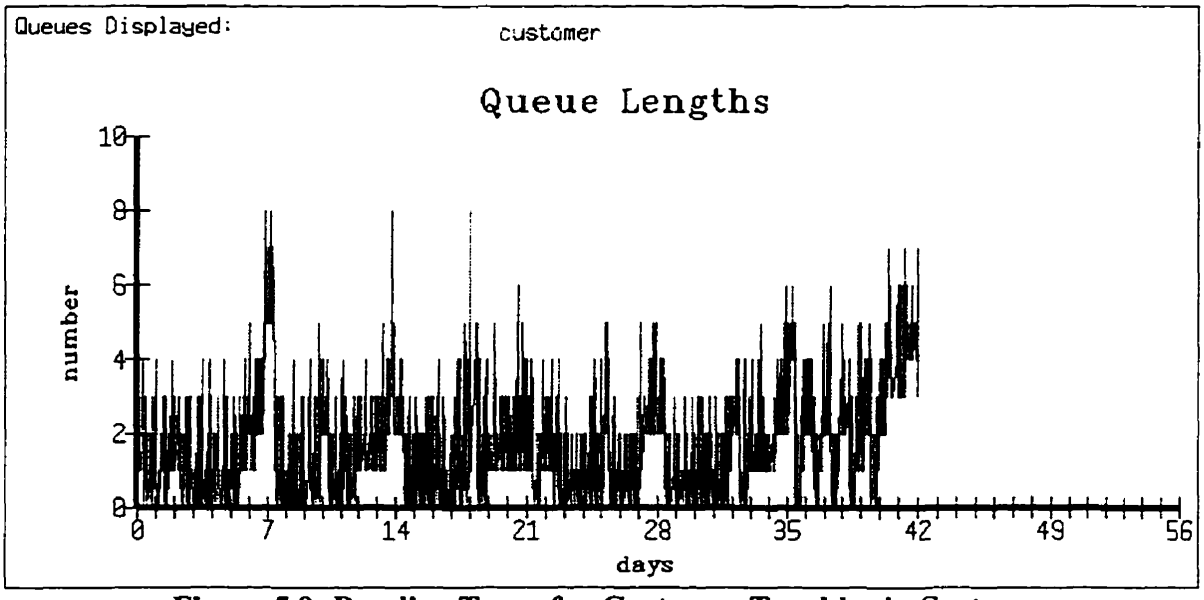

Figure 5.9: Baseline Trace for Customer Troubles in System.

5.2.2.4: NUMBER OF MINOR TROUBLES IN SYSTEM

Figure 5.10 below shows the trace for the total number of minor troubles in the system over a six-week period. Both the daily weekday and weekend effects can be clearly observed in the trace. The weekday effect sees the number of troubles dropping in the middle of the day, corresponding to the day shift when more staff are available, and rising at the beginning and end of the day, corresponding to the night and evening shifts when less staff are available. The weekend effect sees a large build-up of troubles on Saturday and Sunday, which is cleared towards the end on Monday. Both effects are caused by troubles waiting for level 2 service, which is not obtained by call-out for minor troubles; consequently they must wait for the next period of business hours to receive level 2 service. The number of minor troubles in the system is the most variable of the four criticality classes. This is to be expected since they are the most common troubles.

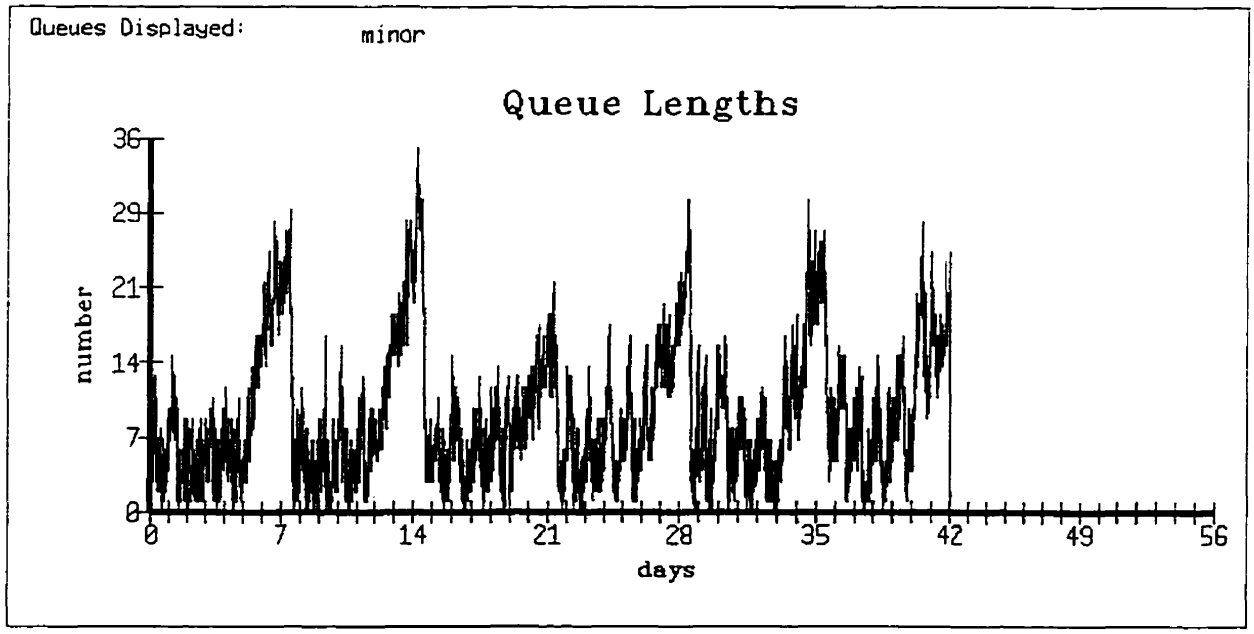

Figure 5.10: Baseline Trace for Minor Troubles in System.

5.2.2.5: TROUBLES QUEUED FOR LEVEL 1 SERVICE

Figure 5.11 below shows the trace for troubles queued for level 1 service in the system over a four-week period. Level 1 service is available on a 24-hour basis from PNOC with extra NOC staff during business hours. Around the clock availability should minimize both the weekday and the weekend effect. The largest peaks of 3 and 4 troubles do occur on the weekends, particularly for the third week (days 20 and 21), indicating a slight weekend effect. This is accounted for by the fact that the extra NOC staff does not work on the weekends. During the weekdays the larger peaks of height 2 are generally found towards the beginning or end of the day, which is also consistent with NOC staff working only business hours.

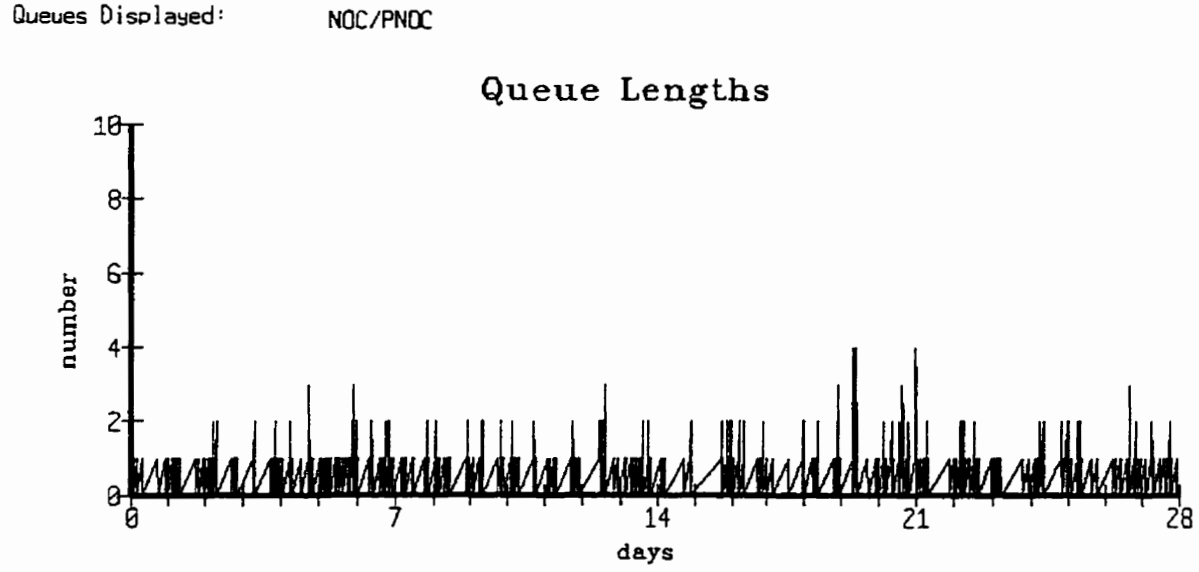

Figure 5.11: Baseline Trace for Troubles Queued for Level 1 Service.

5.2.2.6: TROUBLES QUEUED FOR LEVEL 2 SERVICE

Figure 6.12 below shows the trace for troubles queued for level 2 service in the system over a four-week period. Both weekday and weekend effects are clearly visible, caused by the fact that level 2 staff are regularly scheduled to work during business hours only. During weekdays the number of queued troubles are lowest at the middle of the week, corresponding to business hours when level 2 staff are working, then rising during the evening and night shifts. During the weekend a large build-up of troubles occurs, composed mostly of minor and customer troubles that have to wait until Monday for service. An interesting feature of the weekend effect is the occurrence of spikes as the queue builds up during the weekend. These represent the times when a callout has been successful for a major or critical trouble.

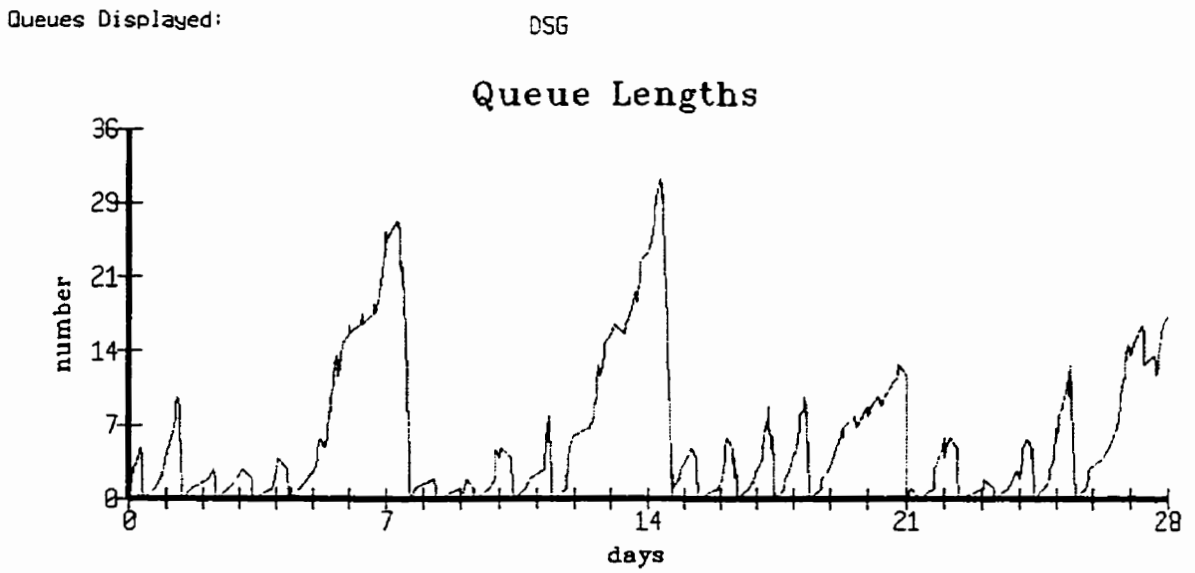

Figure 5.12: Baseline Trace for Troubles Queued for Level 2 Service.

5.2.2.7: TROUBLES QUEUED FOR TECHNICIAN SERVICE

Figure 5.13 below shows the trace for troubles queued for technician service in the system over a four-week period. Three of four weekends show a weekend effect, accounted for by the fact that there are fewer staff working on weekends. Fewer staff on weekends is justified since there are fewer level 1 and level 2 staff on weekends to assign troubles to the technicians. A weekday effect is not clearly visible but if it could be observed it is expected to be opposite the one observed for level 2 staff, i.e. the queue for technician service would build during business hours when more staff are assigning troubles to technicians.

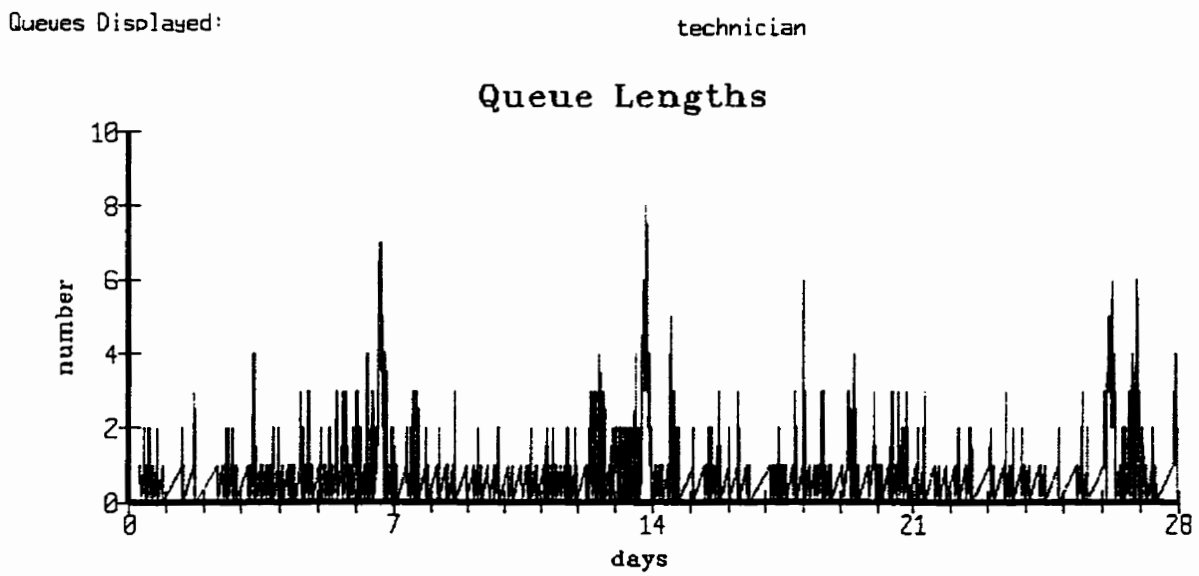

Figure 5.13: Baseline Trace for Troubles Queued for technician Service.

5.2.2.8: TICKETED TROUBLES

Figure 5.14 below shows the trace for ticketed troubles in the system over a four-week period. A strong weekday effect can be observed as minor troubles entering the system during non-business hours are ticketed and are removed from the ticket queue primarily during business hours. No weekend effect is apparent as troubles are ticketed throughout the weekend and removed from the ticket queue when no other troubles requiring service are present.

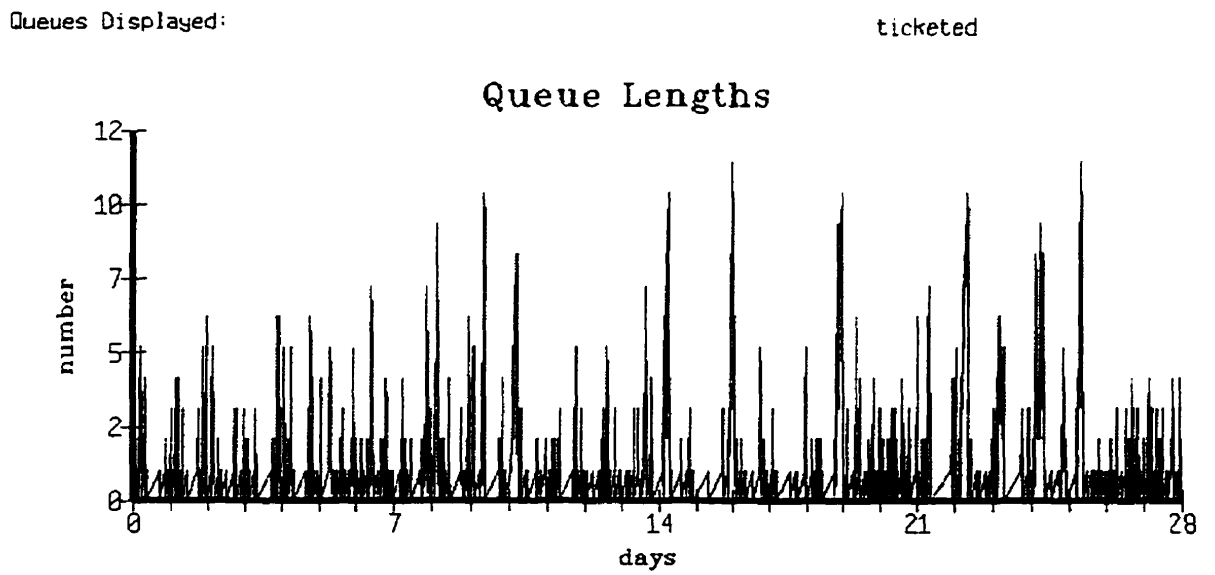

Figure 5.14: Baseline Queue Length Trace for Ticketed Troubles.

5.2.3: BASELINE MODEL MEAN OUTPUTS

Mean value outputs are produced at the end of the simulation in the form of a text report. The mean values and the 95% confidence intervals over the number of replications are both reported. The mean values reported are receipt-to-close times, true queue lengths of troubles waiting for service and total number of troubles in the system.

5.2.3.1: RECEIPT-TO-CLOSE TIMES

Figures 5.15a to 5.15d below show the baseline system mean value receipt-to-close times and 95% confidence intervals for minor, customer, major and critical troubles respectively, averaged over the shift of the week the trouble arrived. The shifts are numbered consecutively starting with the Monday night shift. In this numbering scheme the five business hours shifts (numbered 2, 5, 8, 11 and 14) are marked with the letter "B".

As expected there is a pronounced shift of arrival effect on receipt-to-close times for all criticality classes. Two effects are apparent. During business hours receipt-to-close times are lower and generally have a smaller confidence interval. On weekends receipt-to-close times are higher and have larger confidence intervals.

Figure 5.15a shows receipt-to-close times for minor troubles. The means and confidence intervals for business hour shifts are lower for every weekday except Monday. This effect is due to staff "catching up" with higher-priority troubles from the preceding weekend. This catch-up effect has disappeared by Tuesday. The weekend effect starts with the Friday evening shift, which has the highest mean and largest confidence interval. The means and confidence intervals get progressively smaller as the weekend progresses. This effect is caused by minor troubles requiring level 2 service that must wait until the next week for service. The Friday evening shift is the furthest away from the next business hours shift, so minor troubles arriving then have the longest wait.

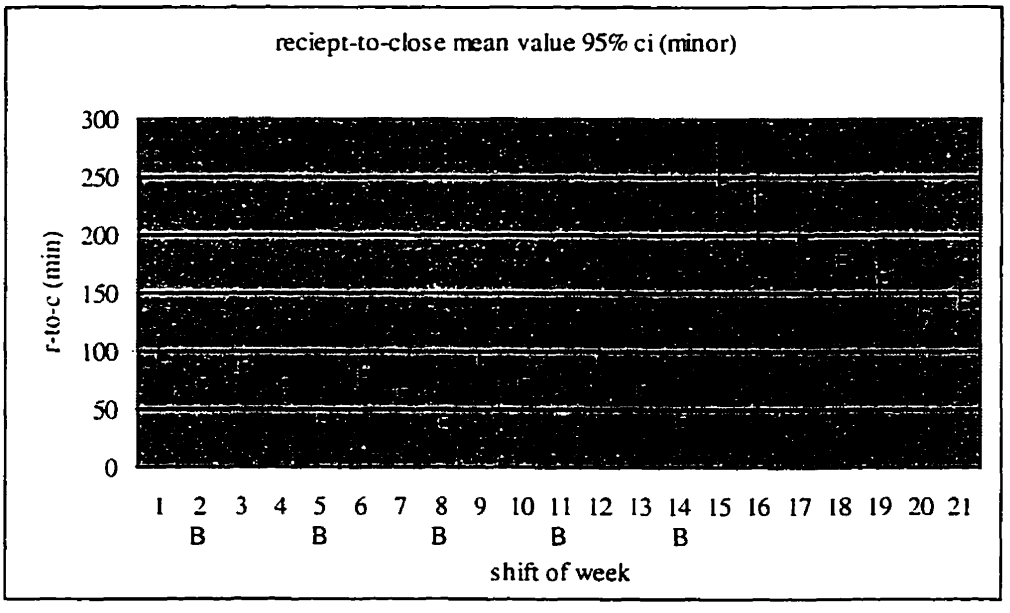

Figure 5.15a: Baseline Mean Receipt-to-close Time and 95% c.i. – Minor troubles.

Figure 5.15b shows receipt-to-close times for customer troubles. There is no catch-up effect apparent on Monday, but the same weekend effect that was observed for minor troubles is present here. Also evident is an evening effect during the weekdays, with troubles arriving on the evening shifts from Monday to Thursday having a higher mean and larger confidence interval. This effect is due to troubles requiring level 2 service having to wait for two shifts and is more pronounced here than for minor troubles because a higher proportion of customer troubles require level 2 service.

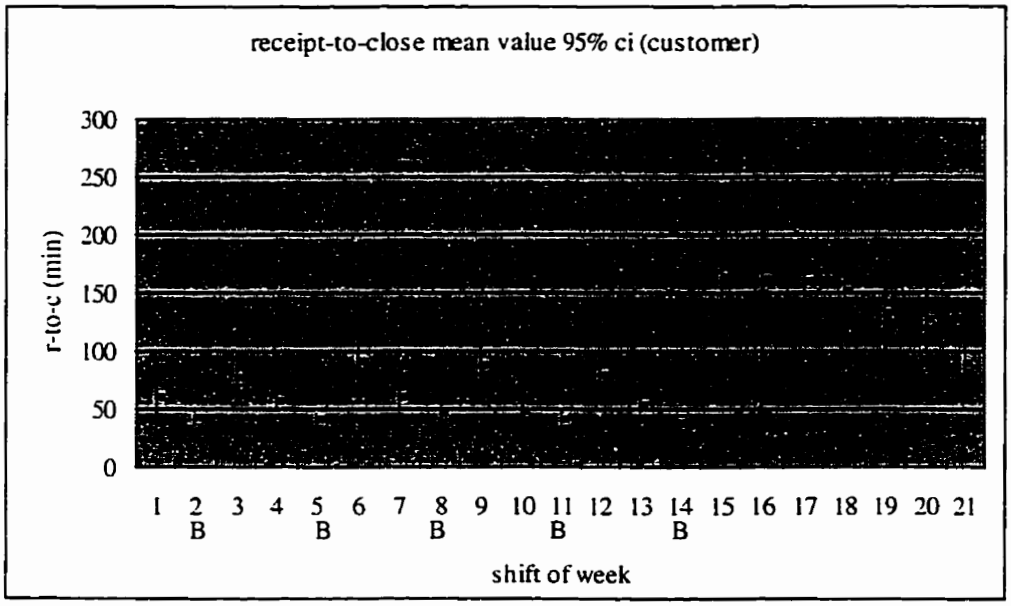

Figure 5.15b: Baseline Mean Receipt-to-close Time and 95% c.i. – Customer troubles.

Figure 5.15c shows receipt-to-close times for major troubles. The catch-up effect on Monday is minor, with a slightly larger confidence interval during business hours. Friday's business hours confidence interval is the largest, possibly to those troubles requiring co-operative service carrying over to the following week. The evening shift effect observed for customer troubles is also evident here. The weekend effect is also present, although the downward trend of receipt-to-close mean values through the weekend shifts is not pronounced, due to the fact that major troubles can receive level 2 service by call-out.

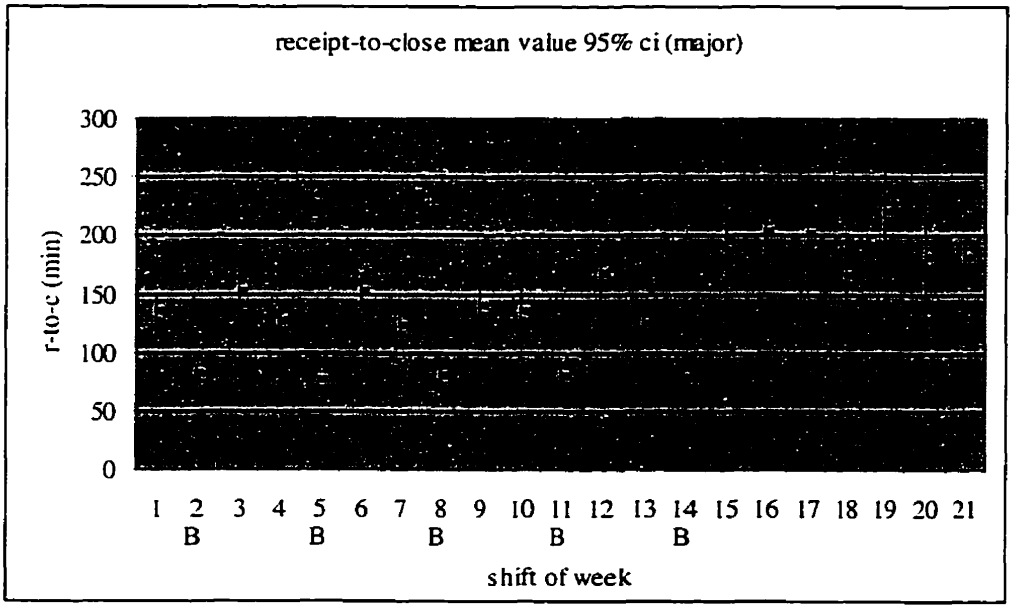

Figure 5.15c: Baseline Mean Receipt-to-close Time and 95% c.i. – Major troubles.

Figure 5.15d shows receipt-to-close times for critical troubles. No catch-up effect is visible. The Monday to Friday shifts do not show large relative differences in their mean values between business and non-business hours, although the evening shift effect is still apparent. The weekend effect shows a large increase in both the means and confidence intervals, with no downward trend evident except for the Sunday afternoon shift (shift 21).

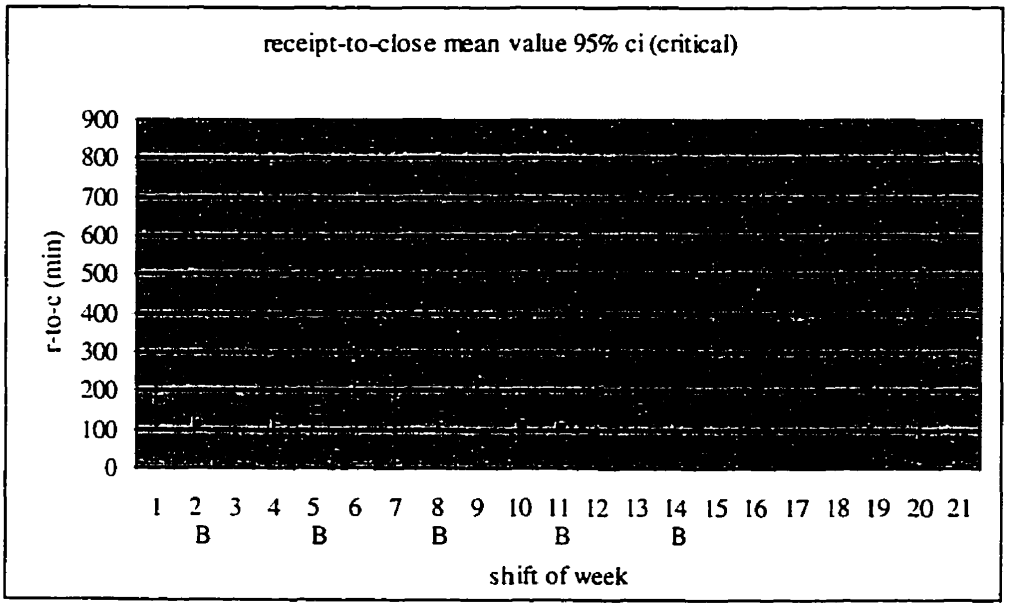

Figure 5.15d: Baseline Mean Receipt-to-close Time and 95% c.i. – Critical troubles.

5.2.3.2: TRUE QUEUE LENGTHS

Figure 5.16 below shows the baseline system mean queue lengths of troubles waiting for service for each service level and ticketed troubles. Short queues are evident for level 1 (NOC/PNOC) service, which is to be expected since level 1 service is staffed around the clock. The technician queue is also short but shows more variability than level 1. Weekend queues are longer for technician service due to lower staffing levels. The ticket queue is lowest during business hours when all levels of staff are available. The

level 2 (DSG) queue shows the most variability, with the lowest mean level occurring on the weekday evening shift. Although this shift is not staffed with level 2 workers, it occurs immediately after business hours when level 2 staff are available. Thus level 2 staff have spent the previous shift decreasing the backlog of troubles waiting for level 2 service, so the level 2 queue should be at its shortest at the beginning of the weekday evening shift. The level 2 queue gets increasingly longer throughout the weekend as shown by the increasing mean queue lengths over the weekend shifts.

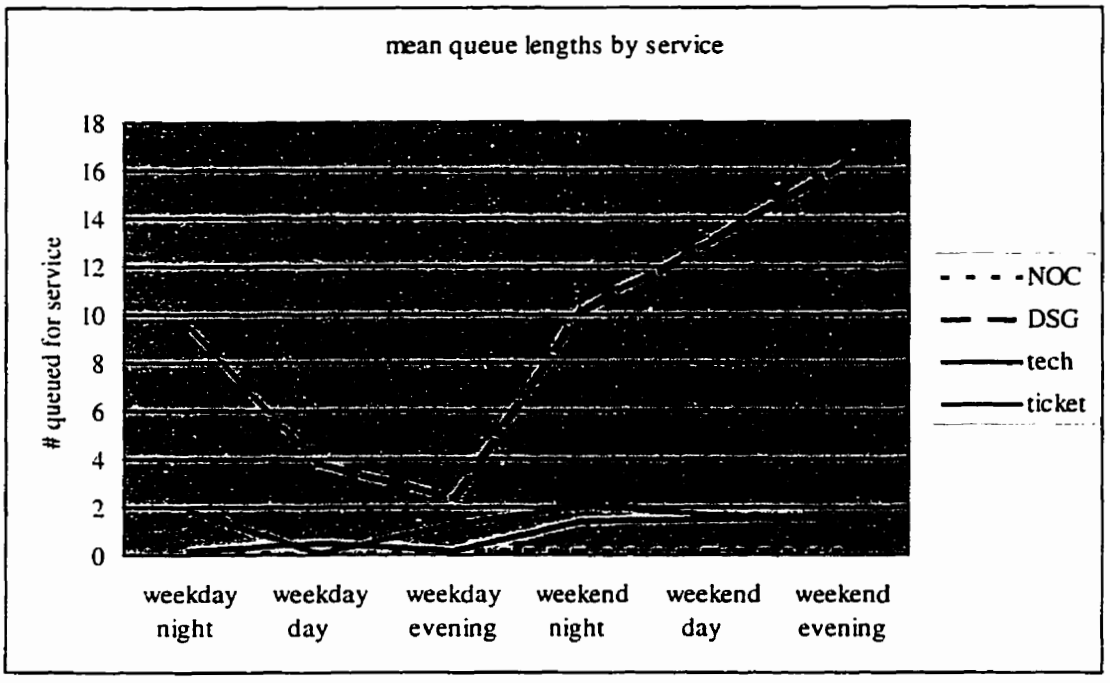

Figure 5.16: Baseline True Queue Lengths.

5.2.3.3: NUMBER OF TROUBLES IN THE SYSTEM

Figure 5.17 below shows the baseline system mean total number of troubles in the system (also known as the total queue) for each criticality level. As expected, critical troubles have the lowest mean number in the system, never rising above 1 throughout the week. Customer and major troubles have nearly identical mean queue lengths throughout the week, reflecting their similar arrival rates. The number of minor troubles in the system is the most variable, showing the same pattern as the number of troubles waiting for level 2 service in Figure 5.16 above. This reflects the fact that minor troubles must wait for regularly scheduled level 2 staff to obtain level 2 service.

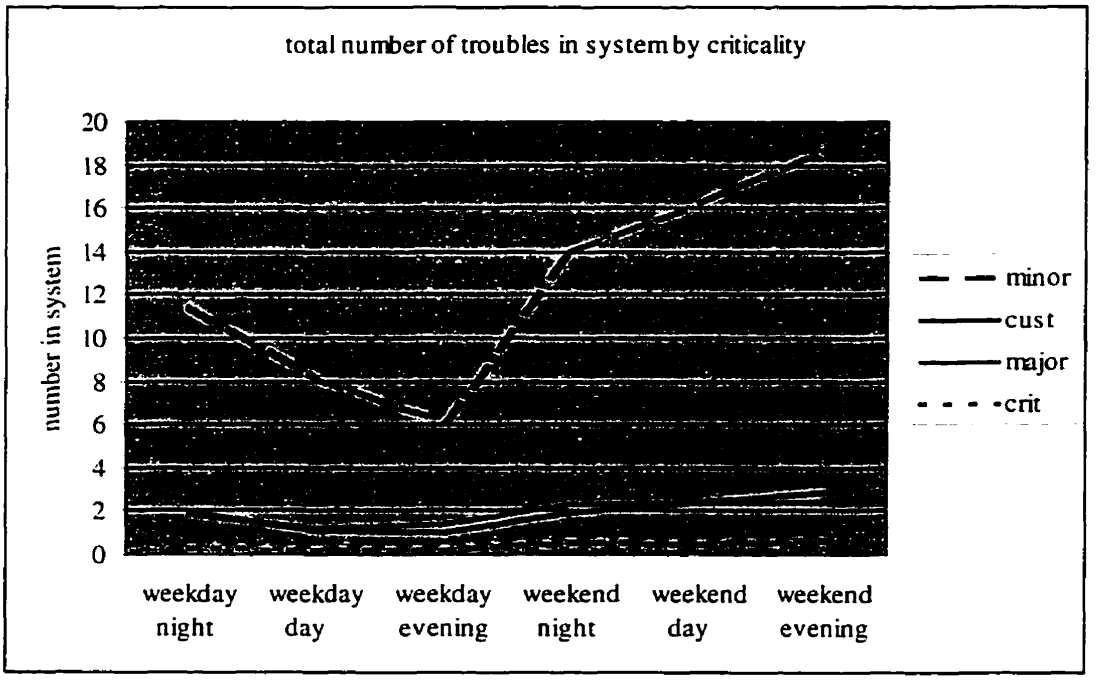

Figure 5.17: Baseline Total Number of Troubles in System.

5.2.3.4: UTILIZATIONS

Figure 5.18 below shows the baseline system utilizations for each service level. Level 1 utilizations are consistent throughout the week except for business hours (weekday day shift) when more staff are available causing utilization to drop. Technician utilizations are higher on weekends due to lower staff levels on weekends. During business hours technician utilization is higher than the other weekday shifts even though the staffing level is constant during the week. This is caused by the fact that more level 1 and level 2 staff are available to assign troubles to technicians during business hours.

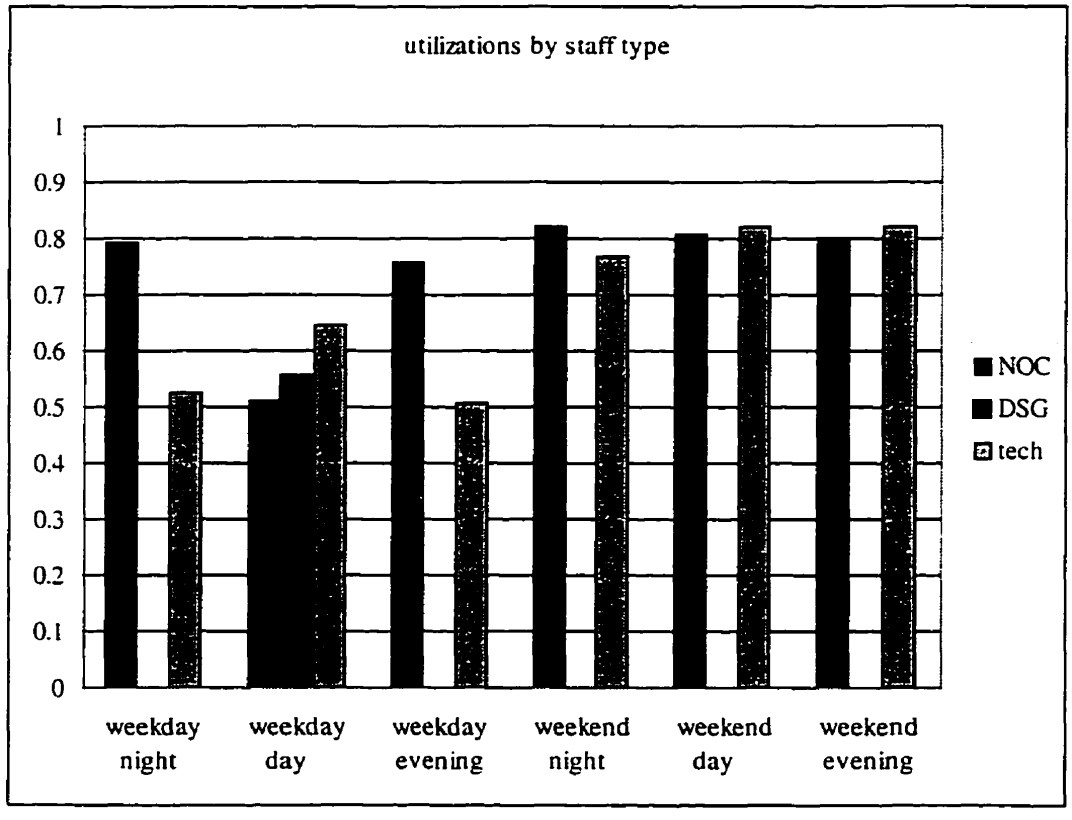

Figure 5.18: Baseline Utilizations.

5.2.3.5: CALL-OUT HOURS

Data on call-outs is an important factor to consider when evaluating various scenarios since call-outs increase manpower costs. Figure 5.19 below shows the output produced for level 2 call-outs. The average number of call-outs and the average length of a call-out are reported, broken down by shift. The average total number of call-out hours worked per week is also reported.

Callout Data over 10 Replications

shift	number of callouts	95% ci		avg length of callout (min)	95% ci (min)	
weekday night	1.6600,	1.5471,	1.7729,	125.8302,	115.1087,	136.5516
weekday day	0. ,	0. ,	0. ,	0. ,	0. ,	0.
weekday evening	1.1400,	1.0103,	1.2697,	91.5097,	76.1341,	106.8852
weekend night	1.3250,	1.0683,	1.5817,	93.2217,	74.4277,	112.0158
weekend day	1.7625,	1.4959,	2.0291,	136.4346,	113.9446,	158.9247
weekend evening	1.7750,	1.4292,	2.1208,	135.1690,	102.0931,	168.2450

Total Callout Time per week over 10 Replications

mean (hr.)	95% ci (hr.)	
30.273	28.535	32.010

Figure 5.19: Baseline Call-out Hours.

5.3 ANALYTIC SOLUTIONS TO THE TDRS

This section addresses the question of analytical solutions to the TDRS. While solutions to systems like the TDRS are generally not possible, a method has been found for obtaining a utilization analysis for each of the three service classes. Two problems arise in this analysis: the lack of memoryless property and the heterogeneity of the system. Both these problems are overcome and their solutions are described in this section.

5.3.1: INTRODUCTION

The TDRS is what is described in queuing theory as an open network of queues. It is a network because the output from one queue (service level) becomes the input to another queue. It is an open network because customers (troubles) can enter the system from the outside world and can leave the system. By comparison, a closed network is one where customers can neither enter nor leave the system, but continually circulate within the system.

According to Molloy (1987 section 7.1) the problem of finding analytic solutions to a network for quantities such as mean queue lengths for each queue and the expected delay in the system for a customer is hard in general. For certain classes of networks such solutions are possible. Such networks are ones whose queues all have the $M \Rightarrow M$ property. A queue possesses the $M \Rightarrow M$ property if, given a Markovian (memoryless or Poisson) input process, its output process is also Markovian. Examples of such queues are FIFO with exponential service times and LIFO preemptive-resume with service times having a Coxian distribution.

The queues in the TDRS do not have the $M \Rightarrow M$ property. The queues are priority with partial preemptive-resume. (Recall that a customer trouble cannot preempt a minor trouble) The service times are mixture distributions composed of a study time and possibly a repair time, both of which are beta random variables. In fact the word *memoryless* implies that only the current state determines (stochastically) what a customer does next. Different past histories have no effect on the distribution of service times or the distribution of transitions made on service completion. The memoryless property is violated in several ways in the TDRS. Consider for instance the case where it

is determined that a trouble can be resolved within the current service. This fact determines that the trouble's service time will include both study time and repair time. Consider also the case where a trouble is unresolved by technician service. If it had completed level 1 service prior to technician service, it is routed back to level 1, but if its prior service was level 2, it is routed back to level 2. The first case uses routing knowledge to determine service time and the second case uses knowledge of past service to determine routing, both violations of the memoryless property.

Thus finding an analytic solution for expected delays and queue sizes in the system is likely intractable, or at best a long and tedious exercise. Even if such a solution were possible, it would still only provide answers for a homogeneous system. The TDRS has two sources of heterogeneity; the first is caused by staffing changes and the second is caused by different trouble criticalities having different service times and routings. The next section shows how an analytic analysis of utilization can be achieved despite the two sources of heterogeneity.

5.3.2: Calculation of Throughputs using the Balance Property

A throughput analysis of a system without the $M \Rightarrow M$ property can still be performed because the throughput of the system does not depend on any particular queuing discipline, arrival distribution or service distribution. This is so simply because of the fact that if the system is to be stable, i.e. if queue lengths do not keep growing forever, then the total throughput (output) of each queue must balance the total arrival rate at each queue. MacDougall (1987 section 1.4) states the *utilization law*, a special case of Little's Law but with equal generality, relates throughput with utilization given

mean service times and can be applied to calculate the utilizations of the servers. Given the throughput of a queue λ, the expected service time $E[S]$ and the number of servers N, the utilization law calculates utilization as $\rho = \frac{\lambda E[S]}{N}$.

The first problem to be considered is the lack of memorylessness. For the time being heterogeneity will be ignored and the system consisting of a single trouble criticality and a single, continuous "shift" is considered.

Throughput analyses can be performed on both open and closed networks. In order to do a throughput analysis of an open system, the outside world is considered another queue supplying customers into the system and accepting customers from the system. The throughput of one queue is routed to one or more queues based on some discrete probability distribution. Now since the sum of the inputs to a queue must be balanced by the throughput of the queue if the system is to be stable, each queue generates a balance equation involving throughputs. This system of linear equations can be described by a matrix equation of the form $\bar{\lambda}R = \bar{\lambda}$, where $\bar{\lambda}$ is the vector of throughputs for each queue (including the "queue" for the outside world) and R is called the *routing matrix*. Each row of the routing matrix corresponds to a queue in the network and specifies the discrete probability distribution that determines the proportion of the queue's throughput going to each queue in the network. (Note that a queue's output can be routed back to itself.) Since the rows are probability distributions, each row must sum to 1. This system can be solved for either an open or closed network to give throughputs of all the queues relative to one queue. Since the throughput of one queue in an open system is known absolutely (the outside world), absolute throughputs can be found for

the queues in an open network. Examples of specifying the routing matrix and solving for $\bar{\lambda}$ can be found in Molloy (1989 Chapter 7).

Figure 5.20 below models the TDRS in terms of throughputs between queues and the outside world. Throughputs are labelled λ_O, λ_1, λ_2 and λ_t from the outside world, level 1 service, level 2 service and technician service respectively. For those queues whose throughput is distributed to two or more queues, the proportion of the throughput going to each queue is labelled. For example, the throughput from level 1 service goes to three queues, with the proportions labelled P_{1R} (resolved troubles going to the outside world), P_{1U} (unresolved troubles going to level 2 service) and P_{1H} (hardware troubles going to technician service). The use of the subscripts R, U and H correspond to the way these probability distributions are specified as inputs to the TDRS. Note the branch of λ_t labelled P_{tU}, the troubles unresolved at technician service, splits into two branches. One branch is labelled LSR = 1 and represents the path taken by troubles that had level 1 service before technician service. The other branch is labelled LSR = 2 and represents the path taken by troubles that had level 2 service before technician service. That is, troubles that proceed from level 1 service to technician service and are not resolved return to level 1 service, similarly troubles that proceed from level 2 service to technician service return to level 2 service.

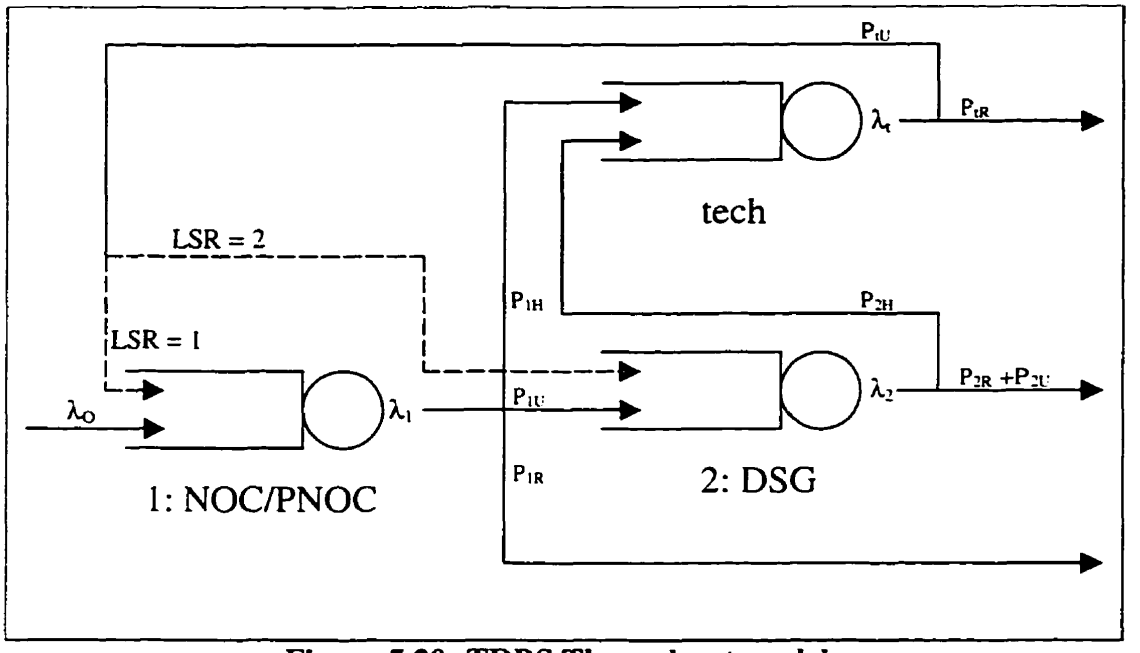

Figure 5.20: TDRS Throughput model.

Now the problem caused by the violation of the memoryless property can be seen in Figure 5.20. It is not known *a priori* the proportion of troubles leaving the technician queue that are routed to level 1 and the proportion routed to level 2. Only the sum of the two proportions, P_{tU}, is known. This is a consequence of the fact that the routing of unresolved troubles from the technician queue depends on the memory of the queue from which the trouble arrived at the technician queue.

In order to be able to carry out the throughput analysis, a way is needed to determine the routing probabilities for the technician queue. One approach would be to attempt to model the TDRS as a first-order Markov chain. If this is possible, then all the routing probabilities can be determined beforehand. According to Trivedi (1982 section 7.1), such models are termed memoryless, a consequence of the Markov property. Again, the memoryless property means that routing depends only on which queue the trouble is currently located at and not on the past history of the trouble. This is not the case with the

routing of troubles leaving the technician queue however. Thus a first-order Markov chain model of the TDRS is not possible.

Thus the solution must be to extract the required proportions using information contained in the current model. First of all, it is observed that P_{tR} and P_{tU} are memoryless, or first-order, in that the choice of either of these two paths does not depend on the past history of the trouble. So one expects both these streams to have the same proportion of troubles whose previous service was level 1, designated P_t^1. Now this proportion must also be equal to the proportion of troubles arriving at the technician queue from level 1 service. But P_t^1 is also exactly the proportion of P_{tU} that is routed back to level 1 service. Given this proportion the second-order memory effects can be eliminated, while preserving throughputs that are adjusted to be equivalent to those of the second-order system, giving the throughput diagram shown in Figure 5.21 below.

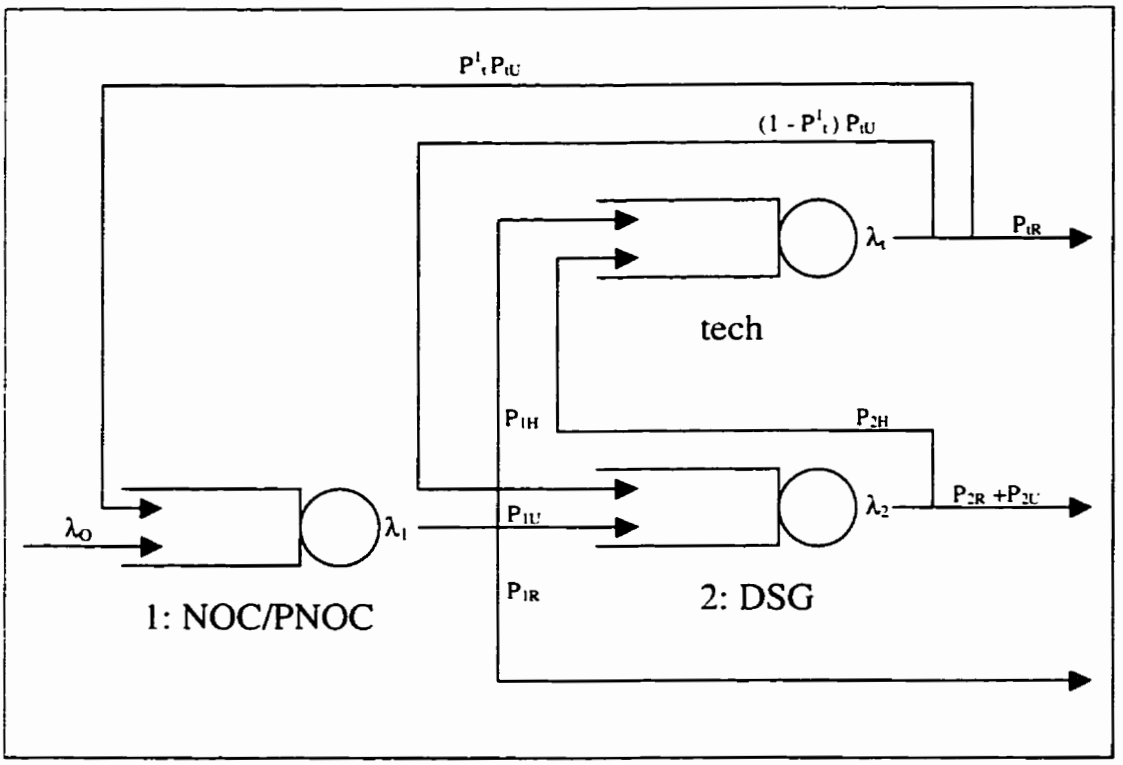

Figure 5.21: TDRS second-order Throughput model.

Now the balance property can be used to obtain P_t^1. It is the proportion of troubles arriving at the technician queue from level 1 service and is equal to $P_t^1 = \frac{P_{1H}\lambda_1}{P_{1H}\lambda_1 + P_{2H}\lambda_2}(1)$. But λ_1 and λ_2 are not known *a priori* because of the feedback from the technician queue to both the level 1 and level 2 queues. This feedback adds to both λ_1 and λ_2. However the system can be solved by an iterative technique described below. As a first guess, feedback can be ignored which gives $\lambda_1 = \lambda_O$ and $\lambda_2 = P_{1U}\lambda_O$. This gives an initial guess for $P_t^1 = \frac{P_{1H}\lambda_O}{P_{1H}\lambda_O + P_{1U}P_{2H}\lambda_O}(2)$. This initial guess can be put into the routing matrix and the system can be solved for the throughputs. This will produce new values for λ_1 and λ_2, and a new guess for P_t^1 can be calculated using equation (1) above. Once again the system is solved, obtaining new values for λ_1 and λ_2,

which are again used to obtain a new value for P_t^I. This procedure is continued until the difference between successive values of P_t^I becomes small enough that its value is considered fixed. This fixed-point iteration on P_t^I allows the calculation of equivalent first-order probability distributions for the routing matrix and thus allows the throughputs to be determined. Equation (3) below shows the linear system for the TDRS.

$$[\lambda_O, \lambda_1, \lambda_2, \lambda_t] \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ P_{1R} & 0 & P_{1U} & P_{1H} \\ P_{2R} + P_{2U} & 0 & 0 & P_{2H} \\ P_{tR} & P_t^I P_{tU} & (1 - P_t^I) P_{tU} & 0 \end{bmatrix} = [\lambda_O, \lambda_1, \lambda_2, \lambda_t] \quad (3)$$

This calculation is ideally suited to a spreadsheet application. A spreadsheet has been developed in Microsoft Excel, which calculates the required parameters from the same input data used by the TDRS, performs the fixed-point iteration to determine P_t^I and calculates the throughputs of each queue for each criticality class. Figure 5.22 below shows the calculation of the routing matrix for critical troubles. Lambda is the throughput from the outside world, equal to the arrival rate of critical troubles. P is the guess for P_t^I and Pnext is the calculated value using equation (1) above. The value labelled RHS is the result of solving the balance equation for the queue representing the outside world. It should be equal to lambda and is used as a check, as is the rowsum column, which should contain 1's. In practice convergence is very rapid, usually achieving 5 significant figure accuracy in 3 or 4 iterations.

Critical Troubles

Throughputs	
P	0.754098
Lambda	0.08
Lambda1	0.086957
Lambda2	0.028355
Lambdat	0.046125
RHS	0.08
Pnext	0.754098

Routing Matrix

	outside	NOC	DSG	tech	rowsum
outside	0	1	0	0	1
NOC	0.3	0	0.3	0.4	1
DSG	0.7	0	0	0.3	1
tech	0.8	0.15082	0.04918	0	1

Figure 5.22: Spreadsheet calculation of Routing Matrix.

Iteration is implemented in a spreadsheet by creating formulas with circular references. In Figure 5.22, the proportion P refers to the proportion Pnext, which refers to the (4,2) and (4,3) positions in the routing matrix, which refer back to P.

As an additional check on using this method to determine the routing matrices, the simulation was modified so it would count the number of troubles routed from each queue to the other queues. This allows the determination of the proportion of troubles routed to each queue; in other words, the rows of the routing matrix. If the input data specifies only 1 replication, the routing matrix is printed instead of a regular report. The input data must also specify arrivals of only one criticality class so the matrix produced will correspond directly to one of the calculated routing matrices. Figure 5.23 below shows the routing matrix obtained for critical troubles, which corresponds to the analytic results shown in Figure 5.22 above, showing good agreement with the analytic values. In fact the simulation produced output for all four routing matrices that closely matched the analytic results produced by the spreadsheet.

Routing Matrix

service	base	outside	NOC	DSG	tech
NOC/PNOC	145530	.299904	0.	.298042	.402055
DSG	46079	.702489	0.	0.	.297511
technician	72186	.804103	.158383	.037514	0.

Figure 5.23: Routing Matrix from Simulation.

Now that the problem caused by the lack of the memoryless property has been resolved by determining an equivalent first-order system, the two sources of heterogeneity can be dealt with in a straightforward manner. The heterogeneity caused by the different criticality levels is dealt with first.

Each criticality class of trouble has its own probability distribution to determine routing after it finishes service (see section 4.5.3). Thus each criticality class is considered separately and produces its own unique routing matrix. For each system the throughput from the outside world is calculated as the overall arrival rate of troubles multiplied by the proportion of troubles expected at the given criticality. Now the overall throughput for a server class is simply the sum of the throughputs from each routing matrix for that server class.

The second source of heterogeneity is the changing staffing levels over different shifts. Interestingly, this source of heterogeneity does not affect throughput, but it does affect the balance property, both on the input side (expected service times change with changing shifts) and on the output side (staffing levels change with changing shifts). This heterogeneity is also dealt with by performing summations, which is described in the next section.

5.3.3: Calculation of Utilizations

The concept of stability is expressed in a single measure called the *utilization*. Utilization, designated ρ, is defined as the proportion of time a server is busy; being a proportion, it cannot have a value greater than 1. However, the utilization can also be thought of as the ratio of the service hours demanded to service hours available. If this

ratio is less than 1 it is equal to the utilization and the system is stable. If this ratio is greater than 1 then more service is demanded than available and the system becomes unstable.

Recall from Section 5.3.1 the utilization law that allows the calculation of utilization as $\rho = \frac{\lambda E[S]}{N}$. The numerator can be thought of as the number of servers required and the denominator is the number of servers available. In the calculation of throughput from the previous section, the service times and number of servers are not involved at all. Instead it is assumed that there is sufficient server capacity available to serve the throughput, i.e. the system is stable. By calculating utilizations this assumption can be put to the test.

As noted above the service times in this model are mixture distributions consisting of a study time and possibly a repair time. All services involve study time, and the proportion of those that will have an additional repair time is known; it is the proportion of troubles that are resolved. So the expected service time is calculated as $E[S] = E[study] + P_R E[repair]$. Now the throughputs can be combined with the study and repair times and staffing levels to calculate utilizations for each service level.

The TDRS is not homogeneous with respect to time because staffing levels vary throughout the week. However, staffing levels are recurrent on a weekly cycle. This means that utilizations may not be stable for a particular shift; this has already been observed in the baseline system with queue lengths growing over certain shifts. However, excess server demand can be carried over to other shifts during the week that have excess

server availability. Thus if utilization is to be used to define system stability it must be determined on a weekly basis.

Determining analytic utilizations now becomes a matter of summing the total server time demanded over a week and dividing it by the total server time available over a week. This is a bookkeeping process suited to the spreadsheet, but it is worth noting the various factors that must be accounted for in determining the total server time demanded. First the expected service times for each criticality class are calculated from the expected study time, expected repair time, and proportion of troubles resolved. Next, the throughputs for each criticality level are multiplied by the expected service times during each shift to obtain the service demanded. Finally the service demanded for each service level is summed over all shifts, yielding the weekly service demanded for each service level. The calculation of weekly service available is more straightforward, involving multiplying the number of staff available per shift by the number of shifts and summing. Finally the ratio of service demanded to service available gives the desired analytic utilization.

These analytic utilizations are valuable for two reasons. Firstly it is useful as a tool for planning experiments with the TDRS. A set of model inputs can be checked using the spreadsheet to determine if it defines a stable system. If all the utilizations are less than 1, then the system is stable and the experiment will provide meaningful results. If one or more utilizations are greater than 1, then the system is unstable and not a realistic representation of a long-term real-world system. Thus the user of the model has a tool that will quickly tell him if the model inputs are realistic without having to infer this from running the model.

Secondly, analytic utilizations provide a result that can be checked against the utilizations produced by the model itself. This is a valuable tool for the modeller, since if the results agree then this provides evidence that the model has been implemented correctly, and if the results disagree then the modeller should strongly suspect that model has not been implemented correctly. In the case of the TDRS, "the modeller" did indeed discover an error in the implementation using the utilization results from the spreadsheet! The utilization analysis and simulation results agreed for the technician service, but not for level 1 and level 2 service. The simulation reported low results for level 2 and high results for level 1. After examining the logic controlling trouble routing, it was discovered that all troubles unresolved at technician service were being routed back to level 1 service, rather than back to the previous service. This problem could have been observed in the trace output, but finding logic errors from trace output is a tedious job and things can be missed. The results of the utilization analysis clearly illustrate a problem and give at least some insight into where the problem might be located.

Two factors that are not accounted for in the analytic utilizations can cause them to differ from the results observed in a simulation. The first and most obvious one is the effect of call-outs. Troubles that are serviced by call-out do not place a demand on regularly scheduled level 2 staff, which lowers the apparent service demanded and has the effect of lowering the observed utilization. The call-out effect is exclusive to level 2 service. The second effect is caused by the fact that the same server completes a service that carries over a shift change. Thus the portion of service that carries over to the next shift is provided by an "extra" server not accounted for in the service available. This also has the effect of lowering the apparent service demanded and the observed utilization.

This of course is not exclusive to level 2 service, but is more pronounced for level 2 service since they spend a greater proportion of time working on major and critical troubles with longer service times that will result in longer carryover hours.

When comparing the observed utilizations against the analytic utilizations the model was chosen so as to minimize the effects described above. The probabilities of obtaining level 2 staff by call-out were set to 0. The expected study and repair times for major and critical troubles were also lowered to minimize the carryover effect. Figures 5.24a and 5.24b show the analytic utilizations and the observed results from the corresponding simulation respectively. Analytic utilizations for all three servers are contained within the 95% confidence intervals obtained from the simulation. The observed utilizations are all slightly lower than the analytic utilizations, however. This could be due to chance or it could be due to the carryover effect noted above.

	Total Hours		
service	demanded	available	**Utilization**
NOC	540.5196	752	0.718776
DSG	75.6107	168	0.450064
tech	287.4166	456	0.6303

Figure 5.24a: Analytic Utilizations.

```
   Server Utilization Data over  10 Replications
   ==========================================
     service        mean            95% ci
 NOC/PNOC ,         .7162,       .7080,      .7244
   DSG    ,         .4433,       .4206,      .4660
technician,         .6252,       .6166,      .6338
```

Figure 5.24b: Observed Utilizations.

CHAPTER 6: EXPERIMENTAL RESULTS

6.1: INTRODUCTION

This chapter discusses aspects of the TDRS simulation that affect the precision of its outputs as measured by the confidence interval width. The variance reduction technique of common random numbers is discussed first. Next the determination of the experimental conditions to obtain an acceptable level of precision is discussed. Then some experimental results which are representative of the kind of information network managers can expect from using the TDRS simulation are presented.

6.2: EXPERIMENTAL PRECISION

6.2.1: VARIANCE REDUCTION

A simulation that has stochastic input will produce output that is itself stochastic. In a simulation like the TDRS the user is interested in comparing the performance of the system under various scenarios, and wants to be confident that the observed differences in model outputs are due to the differences in the scenarios and not just due to randomness in the experimental conditions. For instance, if a certain scenario produces smaller receipt-to-close times than another does, it may be that the service times were, by chance, shorter for the former rather than it being a better scenario. One way to increase confidence in the differences in model outputs being "real" rather than due to their stochastic nature is to reduce the variance of the outputs. The reduction in variance

increases the precision, producing a narrower confidence interval for the same simulation effort.

There are several variance reduction techniques, as discussed in Law and Kelton (Chapter 11); the one used in this implementation is the use of common random numbers. In this technique, the same series, or stream, of random numbers is used for the same purpose in each of the scenarios under study. For example, if the same random number stream is used to generate interarrival times, each of the alternative scenarios will experience the same interarrival times. Put into intuitive terms, common random numbers allows the user to test alternative scenarios under identical experimental conditions, increasing the likelihood that observed differences are due to the different scenarios. In terms of reducing variance, common random numbers work if the observed outputs show similar changes in the various scenarios given changes in the random numbers. For example, small interarrival times would be expected to produce large receipt-to-close times in all the scenarios.

Using common random numbers means writing the implementation such that the same random numbers are used for the same purpose in each scenario. SIMSCRIPT provides support for this, providing 10 random number "streams" (actually the same random number generator with 10 different seeds). Thus by using separate streams for generating stochastic model variables the modeller can ensure the same random number is used to produce the same stochastic variable in each scenario.

Even with language support for the use of common random numbers, it may not be possible to completely implement its use for a complex model. For example, in the TDRS, the order in which troubles begin their service may be quite different under

different scenarios. This means that even though the random numbers used to generate study times may be taken from a separate stream, the order of the troubles for which they are used will be different. However, even the partial implementation of common random numbers is useful in reduction of variance.

Common random numbers has been partially implemented in this model of the TDRS. Separate streams are used to generate the interarrival times and the criticality level of the trouble arrivals. Thus all the scenarios will be evaluated with the same series of incoming troubles. That is to say that each scenario will see exactly the same number of arrivals arriving at exactly the same times with the same criticality class. Thus the 10000^{th} trouble arrives at the same time and is always the same criticality class, but it may follow different paths through the system in the different scenarios.

6.2.2: Determination of Experimental Conditions

The TDRS uses the independent replication method to obtain independent observations of the quantities of interest (see Section 4.6.2). This section discusses the determination of the experimental parameters that affect the independent observations produced by the TDRS simulation. These parameters are the run-up and steady-state times within an independent replication and the number of replications.

Each replication involves a run-up period during which no statistics are collected and a steady-state period during which statistics collection occurs. Because the TDRS is recurrent on a weekly cycle, both the run-up and steady-state times are specified in weeks. The weekly cycle also means that the TDRS never truly reaches a "steady state", meaning that the system has run long enough for the distributions for measures like queue

lengths and delays to become stable. This meaning of steady state implies a time homogeneous system. In the case of the TDRS, steady state means that the distribution of troubles among the service queues is "typical" at any given point in the weekly cycle because initial effects (i.e. starting with no troubles in the system) have been allowed to wear off.

There is no generally accepted procedure for determining the length of the run-up period according to Law and Kelton (section 8.6.3). Since expected service times for troubles are much less than a week, it is expected that the run-up period need not be long in order to get a typical distribution of troubles in the system. In order to test this assumption and quantify the run-up period, the baseline system was run using five different run-up times, outlined in Table 6.1 below. The observations from these experiments can be studied to determine if run-up time affects them in a systematic way. The focus was on the mean number of critical troubles in the system. Critical troubles tend to spend the longest time in the system, so any run-up effect is most easily seen by studying them.

Experiment	Run-up (weeks)	Steady-state (weeks)
P1-4	1	4
P2-4	2	4
P3-4	3	4
P4-4	4	4
P6-4	4	6

Table 6.1: Run-up Experiments.

Figure 6.1 below compares the mean number of critical troubles in the system for the five run-up experiments. All five experiments show similar results for the weekday.

The weekend results show what initially looks like a run-up effect, as the number in the system increase with run-up time. However, further increases in run-up time lead to decreased observations. It appears that the variations in the observations are random and not due to any run-up effect. The apparent rise in the weekend observations is explained by the correlation between the observations. A higher observation on the weekend night shift will lead to higher ones for the day and evening shifts. Thus there is no need to perform long run-ups and a run-up period of 2 weeks was chosen.

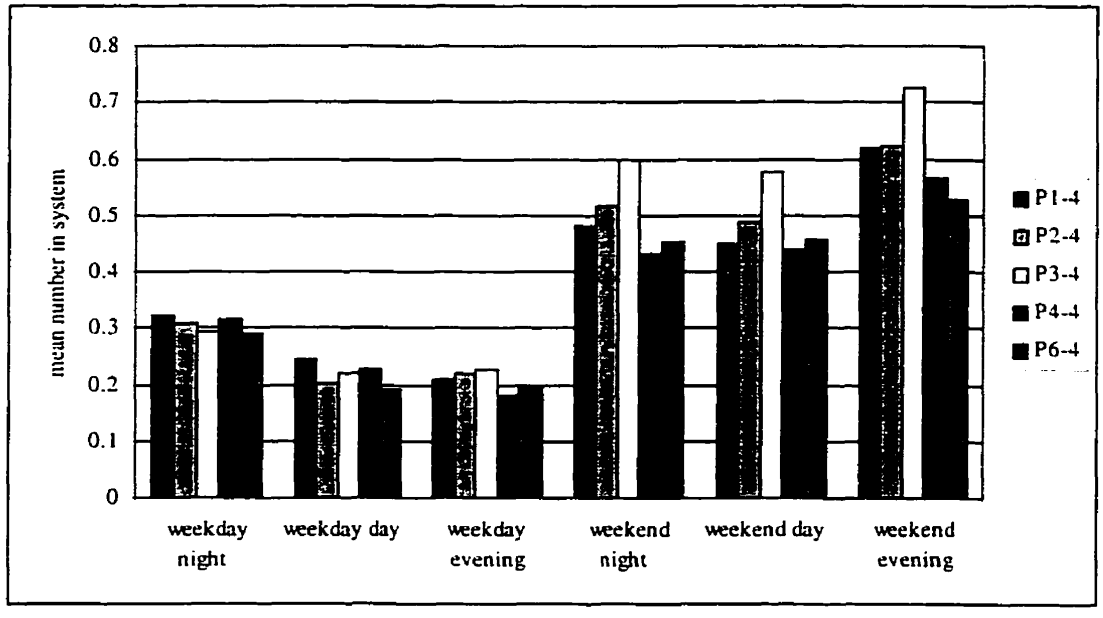

Figure 6.1: Observed Critical Troubles in System for 5 run-up lengths.

The length of the steady-state time and the number of replications both affect the precision of the observations collected by the simulation. As both are increased, precision also increases as reflected by a narrower confidence interval, with a corresponding increase in computing time. A minimum of about 10 replications is needed to form a confidence interval with the expected coverage (Section 4.6.2) and this is the value that has been chosen.

In order to determine the effect of steady-state time on the confidence interval and choose an appropriate steady-state time, a series of experiments was performed to measure its effect on the confidence interval half-width of two outputs, the weekly mean receipt-to-close time and the weekly mean call-out hours. 6 experiments were performed with the baseline system, each with 10 replications and a run-up time of 2 weeks and with steady-state times of 2, 4, 6, 8, 12 and 20 weeks, designated P2-2, P2-4, P2-6, P2-8, P2-12 and P2-20 respectively. For each experiment, the 95% confidence interval half-width is calculated, expressed as a percentage of the mean call-out hours.

For a fixed number of replications, the size of the confidence interval is determined by the standard error of the mean, which has in its denominator the square root of N, the sample size. Thus, for a fixed number of replications and IID observations, the confidence interval width is inversely proportional to the square root of the sample size. Since sample size is proportional to the length of the steady-state time, a log-linear regression of the log of the confidence interval on the log of the steady-state time is expected, with an expected slope of -0.5.

Figures 6.2a and 6.2b below show the log-linear plots for the weekly mean call-out hours and weekly mean receipt-to-close time for critical troubles respectively. The simulated results are plotted and the least-squares line determined by the simulated results is shown along with the equation of the line.

The slopes of both lines are indeed quite close to the expected value of -0.5. This is somewhat surprising since results from a simulation are often positively correlated (Section 4.6.2). While this is certainly true if one were to observe event-actuated statistics such as individual trouble delay times, it appears that statistics averaged over the entire

week are not correlated. This is a result of the system being stable and recurrent on a weekly cycle; the results of the previous week do not affect the results of the current week.

From the equations of the two lines one can calculate the length of the steady-state time needed to give an expected confidence interval half-width of 10%. This is the value of x that gives $y = \ln(10) = 2.302$. For the weekly mean call-out hours the value is $y = 3.25$ and for the weekly mean receipt-to-close time for critical troubles the value is $y = 3.70$. Thus a steady-state time of 4 weeks is sufficient to give an expected confidence interval half-width of 10%. This means that observed differences of 10% or more in the outputs of various experimental scenarios is very likely due to the experimental conditions and not to chance. Differences of less than 10% may also be significant because variance reduction has been used in the simulation; in this case experiments using more replications can be used to determine if the result is significant.

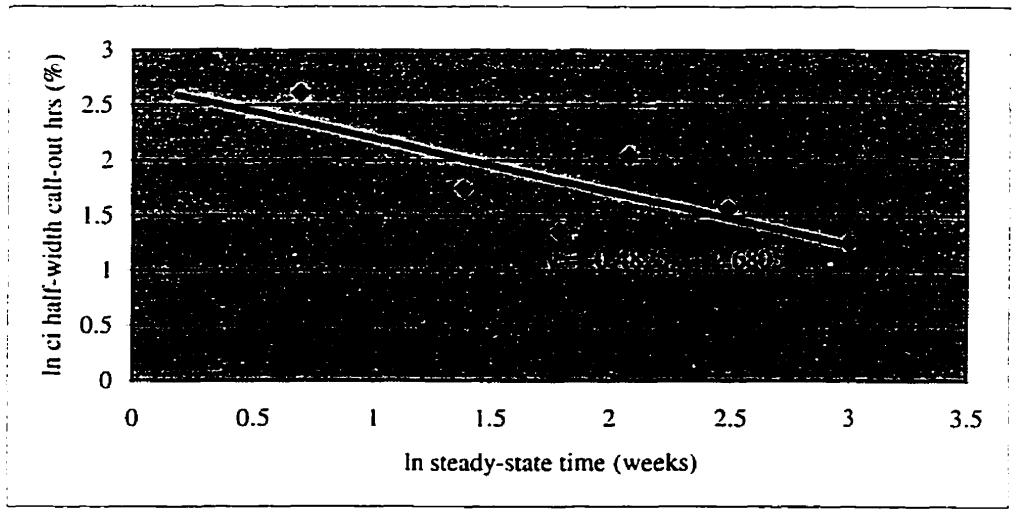

Figure 6.2a: Log-linear Regression for Confidence Interval half-width – Mean Call-out hours.

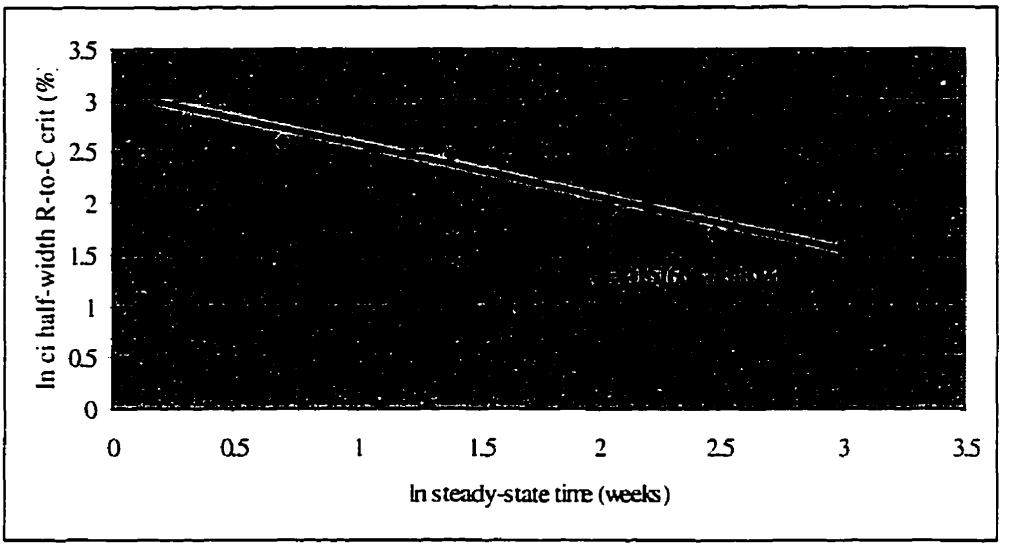

Figure 6.2b: Log-linear Regression for Confidence Interval half-width – Mean Receipt-to-Close time for Critical troubles.

6.3: Experimental Results

A limited number of experiments have been performed with the TDRS. This is because the baseline system presented in Chapter 5, while a stable system with reasonable outputs, probably is not representative of the real-world system. Thus extensive experimentation in order to find "answers" to the sort of questions that network managers may have would not be meaningful unless real-world inputs can be used. Thus the experiments performed is intended to be representative of the type of experiments that can be performed. The aspect of the model that has been chosen to explore experimentally is its sensitivity of the system to staffing levels.

6.3.1: Experimentation with Level 2 (DSG) staffing levels

This section describes some experimentation involving various level 2 (DSG) staffing scenarios and their impact on both the cost of providing level 2 service and the

reliability of the network. As stated in Chapter 1, decisions about the operation of the TDRS always involve a trade-off between staffing levels and the reliability of the network. Nowhere is that more apparent than with the level 2 staff. Being the most knowledgeable and experienced staff, they are able to resolve a wider range of troubles in less time than the less experienced level 1 staff. Thus having more level 2 staff available leads to a more reliable network. However, the costs of providing more level 2 staff, in salary and training, are substantially higher than for other levels.

In this series of experiments level 2 staffing is manipulated and the effect both on the cost of providing the service and on network reliability is observed. Staffing is manipulated both in the number of available workers and in the shifts that they are scheduled to work. Level 2 service is unique in that the cost of providing the service cannot be measured simply from the nominal staffing levels. The effect of call-outs must also be considered, since there is also a cost associated with it that is (presumably) higher on an hourly basis than the cost of regularly scheduled staff. Thus when comparing scenarios with the same nominal staffing level any change in the number of call-out hours must also be considered.

Receipt-to-close time for critical troubles is chosen as a measure of network reliability. These are troubles that are affecting service to customers so they are a direct indication of the reliability of the network.

A series of 5 experiments was performed which varied the level 2 staffing; these are compared against the baseline system. There are 6 shifts during the week: the weekday night, weekday day, weekday evening, weekend night, weekend day and weekend evening shifts. Each experiment varied the level 2 staffing level for each shift

and is designated by the letter "d" followed by 6 digits corresponding to the level 2 staffing level for each of the 6 shifts in the order given above. For example, the baseline system would have the designation "d020000". The other experiments are designated "d011000", "d020010", "d011010", "d010010" and "d111111". The experiments can be grouped according to their nominal staffing requirements. The baseline and d011000 have equal staffing. The experiments d020101 and d011010 have equal staffing that is higher than the baseline. The experiment d010010 has staffing that is lower than the baseline because one staff has been moved from a weekday shift to a weekend shift. The experiment d111111 has much higher staffing than the baseline and is included to illustrate the cost of eliminating call-outs. It should be noted that the utilization analysis of the experiment d010010 yields a utilization for DSG of 1.36, meaning that it is not analytically stable in the absence of call-outs. However, observation of the trace plots for the system when run on the simulation indicates that it is indeed stable, with call-outs picking up the extra demand. Table 6.2 below summarizes the nominal staffing levels of the experiments.

Experiment	Nominal Staffing (hr/week)
d020000 (baseline)	80
d011000	80
d020010	96
d011010	96
d010010	56
d111111	168

Table 6.2: Nominal Staffing Levels.

Figure 6.3 below shows the mean call-out hours per week for the 6 staffing scenarios, with nominal staffing levels shown below the experiment names. The pairs of columns 1 and 2 and columns 3 and 4 illustrate the effect of keeping the overall staffing level constant but moving staff into a non-business hour shift. The pairs of columns 1 and 3 and columns 2 and 4 illustrate the effect of adding one extra worker to the weekend shift. While both effects cause a decrease of about the same absolute amount in the amount of call-out hours worked, the strategy of moving staff to non-business hours shifts is better in terms of reducing staffing costs because it involves no extra staffing requirements. This is illustrated even more clearly when columns 2 and 3 are compared where it can be seen that adding extra staff on the weekend results in more call-out hours worked than staff had simply been moved to non-business hours.

An interesting effect is observed in column 5, the nominally unstable system. Even though this represents a reduction in staffing levels from the baseline system, it also results in fewer call-out hours being worked. In fact it shows about the same decrease in call-out hours, as did adding weekend staff to the baseline system (column 3).

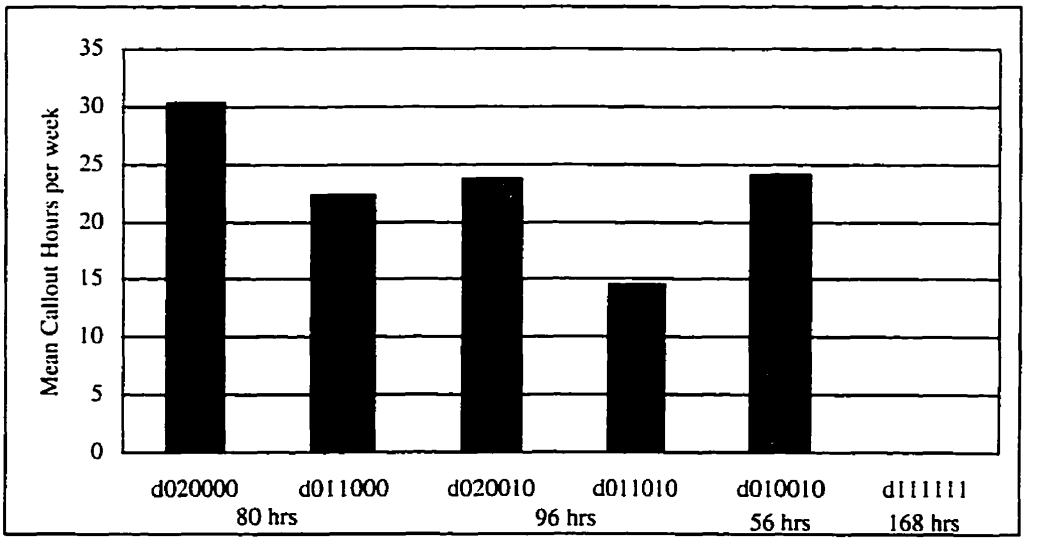

Figure 6.3: Effect of Level 2 Staffing Scenarios on Call-out Hours.

Since in all the staffing scenarios the same stream of troubles is encountered requiring essentially the same amount of service, the explanation for why certain scenarios have reduced staffing requirements must be that the regularly scheduled staff are being utilized more effectively. Figure 6.4 below shows the utilization of the regularly scheduled staff for the staffing scenarios and it clearly shows that the scenarios with reduced call-out hours have increased utilizations. Clearly the effect of moving staffing strength closer to when it is needed has a pronounced effect on their utilization. It is interesting to note that the utilization of the nominally unstable system (column 5) is very nearly 1, meaning that staff are busy all the time in this system. This would be of interest to network managers because it is a situation that can lead to the "burnout" of their most experienced network troubleshooters.

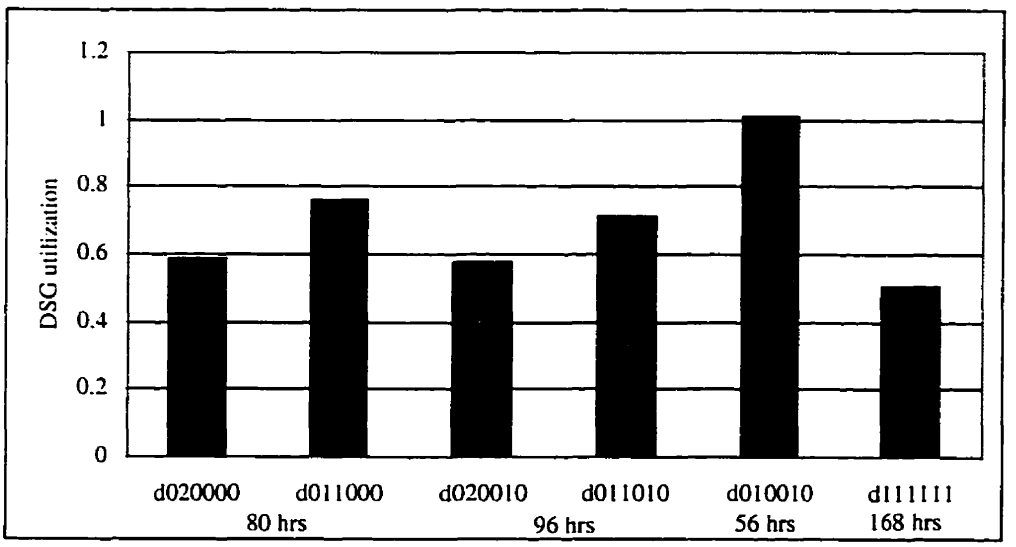

Figure 6.4: Effect of Level 2 Staffing Scenarios on Utilization.

Figure 6.5 below shows the mean receipt-to-close time for critical troubles for each of the staffing scenarios. All of the alternative scenarios perform better than the baseline system. This is to be expected since all of them involve placing regularly

scheduled staff on at least one non-business hours shift, thereby allowing troubles arriving during those shifts to start level 2 service sooner.

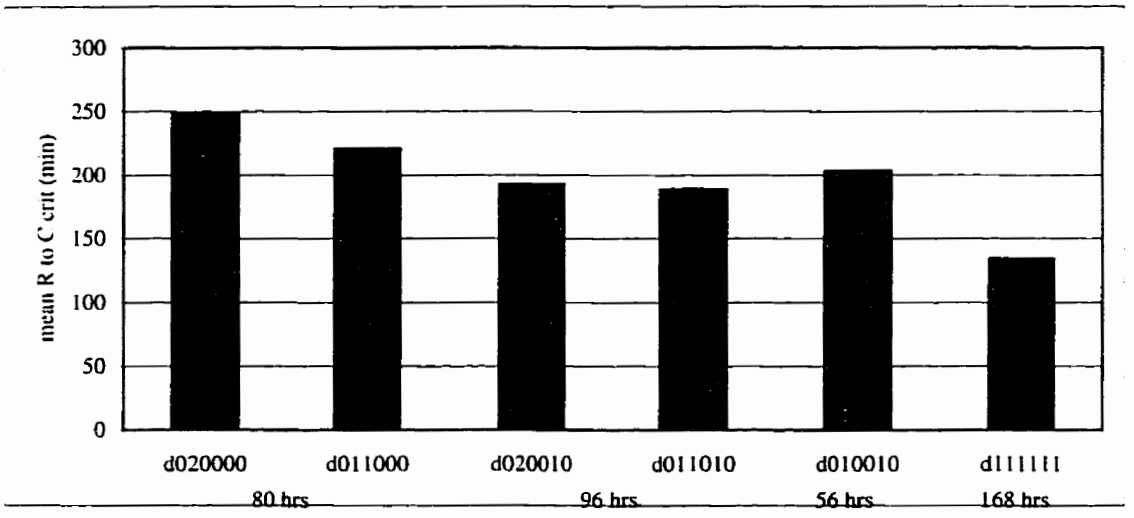

Figure 6.5: Effect of Level 2 Staffing Scenarios on Critical Receipt-to-close time.

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

7.1: CONCLUSIONS

In this work the TRDS has been modelled as a network flow model. This model has applicability to other systems involving diagnosis and repair; in particular it bears close resemblance to models of hospital emergency departments. A utilization analysis of the system has also been performed that overcomes both its non-memoryless and heterogeneous characteristics. It has been shown that a dynamic graphical display is a useful tool for studying a heterogeneous system. It has also been found that the problem of gathering statistics in a heterogeneous system involves a compromise between finer granularity to observe the effects of heterogeneity and coarser granularity to make comparison of experimental systems easier.

The model for the TDRS is a generalization of the job-shop model in that the service of jobs (troubles) has several outcomes that are determined probabilistically. Consequently the routing of jobs in the network is not predetermined but depends on the outcome of service. Routing may also depend on memory of the penultimate service received. It shares many characteristics of models developed for other diagnosis systems, in particular those for hospital emergency departments (ED's). In the TDRS model, troubles are categorized by a single attribute, criticality class, which is used to determine service outcome and service time distributions. In ED models developed by Draeger (1992) and Evans *et al* (1996), patients are characterized by two attributes: case type (e.g. cardiac, trauma, etc.) and class (i.e. acuity or severity). Both these attributes are used to determine the routing of a patient in the system. Another attribute, the patient's mode of

arrival (walkin or squad) determines where the patient begins service. As in the TDRS, the ED model of Evans *et al* (1996) features probabilistic routing. However, routing does not appear to depend on any memory of previous service.

The sources of heterogeneity in the TDRS, namely varying staffing levels and service time distributions, are also present in ED models. The ED models allow for flexibility in determining the starting and ending times of shifts, something the TDRS does not do. One source of heterogeneity present in ED models that is not present in the TDRS and that is a heterogeneous arrival rate for patients. The arrival rate in the ED model of Draeger (1992) varies on both a daily and weekly basis.

Hospital ED models also differ from the TDRS in that they have two distinct groups of resources: staff and treatment areas. Service for a patient can begin only when both the appropriate staff and treatment area become available. This presents scheduling problems for ED models not present in the TDRS. For example the ED model of McGuire (1994) employs an internal waiting room. Once a patient receives one service and is waiting for the next, should he remain in the treatment room, thus making the room unavailable for treatment of another patient, or should he be moved to the internal waiting room to free the treatment room? This is a generalization of what is termed cooperative service in the TDRS, which approximates the intermittent consultation between service levels, but does not involve two resources working on one trouble simultaneously.

Model outputs for both ED models and the TDRS are quite similar. Both employ dynamic graphical displays of model behaviour for model verification and to help the

real-world experts provide feedback on the operation of the model. Statistics collected are also similar: service delays, waiting delays, queue lengths and total delay in the system.

There are two other similarities worth noting. Patients leave an ED model in two ways, either by discharge or by admission to the hospital. This corresponds well to the two ways troubles leave the TDRS, resolution or escalation to OSO support. Draeger (1992) also notes the use of an ED model to explore a staffing scenario involving the use of call-outs to cover periods of high demand.

The similarities and differences between the TDRS and ED models would indicate that is a special case of a more general diagnosis system that is concerned with one type of "patient" (switching equipment) and approximates the use of two simultaneous resources by the use of network feedback with memory.

The utilization analysis has proved to be a powerful tool both for verification of the model implementation and in planning experiments. It uses a fixed-point iteration to determine equivalent first-order routing distributions from a non-memoryless system. It then deals with two sources of heterogeneity by summing throughput balance requirements for each criticality class and using them to sum up service demanded over each time period during the recurrent weekly cycle.

Work on the problem of memory of previous service has been performed by other investigators. De Souza e Silva and Gerla (1991) note that route dependence on past history can be accounted for in an analytical solution for a product-form network by introducing classes. In their terminology, a chain is a homogeneous group of customers, corresponding to a criticality class of troubles. When routing probabilities for a given chain at a particular service centre depend on past history, classes are introduced to the

service centre. Each class may have its own routing probability distribution, and for certain queueing disciplines, its own service demand distributions. They also state that a product-form queueing network with chains having multiple classes can also be solved by an equivalent network having only one class, which they attribute to Lavenberg and Sauer (1983). A reading of the latter work did not yield a technique that could be applied to the problem at hand, and in any case the TDRS does not yield a product-form network.

The techniques used in the utilization analysis can be applied to more general examples of the same problems. The problem created by the memory of the service received prior to the current service determining the routing can be generalized. In the case of the TDRS only 2 servers provided the prior service, but it can be generalized to any number N. The solution must now perform N-1 fixed-point iterations to obtain proportions of customers arriving from these N-1 servers.

The first source of heterogeneity in the TDRS, namely the different criticality classes having different routing probabilities, can also be generalized. In the case of the TDRS only one customer attribute affects routing probabilities, but the analysis can be generalized to handle any number of attributes affecting routing probabilities. The number of routing matrices required is the cross product of the attributes. For example, in an ED model the number of routing matrices is the number of acuity classes times the number of patient types.

The analysis can also be generalized to handle the additional source of heterogeneity present in the ED models, namely heterogeneous arrival rates. In the TDRS only one routing matrix is required for each criticality class because arrival rates are homogeneous. If arrival rates are heterogeneous the analysis requires a different routing

matrix for each different arrival rate over the recurrent cycle. Now the number of routing matrices needed is the cross product of the customer attributes affecting flow and the number of time periods where arrival rates change over the recurrent cycle. For a complex system the bookkeeping may seem daunting but the resulting analysis will be all the more important for such a system as both a check of its correctness and as a planning tool.

The dynamic display is a necessity for studying the behaviour of a heterogeneous system for two reasons. Firstly it provides insights about the system's behaviour over time that are not apparent from mean value outputs. Secondly the people involved in the real-world system are better able to interpret the model's behaviour and will give better feedback than if presented with tables full of numbers. Thus support for dynamic graphics is essential in a simulation package.

The problem of choosing an appropriate level of granularity for collection of model outputs is one I struggled with. Consequently model outputs are reported with levels of granularity ranging from each individual shift of the week for receipt-to-close times, to an aggregate of 6 shift types over the week for queue lengths, to a single value for total call-out hours. My choices were based on which aspect of the system's behaviour I wished to highlight, from shift effects to comparison of alternate scenarios. Because my choices of granularity changed with the questions being asked, it indicates that there is no correct choice for granularity of model outputs, and granularity is an aspect of the simulation the user should be able to select according to the kind of questions he is asking.

7.2: Future Work

Future work in this area includes running the TDRS on real-world data and seeing if its results correspond to real-world outputs. The fact that the simulation program is table-driven, meaning input to the model is read from a file and not hard-coded, will make this easier. This process, along with demonstrating the simulation to real-world experts, would certainly lead to model enhancements. The model enhancements would also include enhancements to the dynamic display. The utilization analysis can also be added to the TDRS itself so that it could operate using the same input data the simulation uses. A longer term project would be generalizing the system into a general diagnosis and treatment system.

LIST OF ACRONYMS

DSG	Digital Support Group
ED	Emergency Department
ETAS	Emergency Technical Assistance Service (Northern Telecom)
FIFO	First-In-First-Out
IID	independent and identically distributed
LIFO	Last-In-First-Out
MTS	Manitoba Telecom Services
NOC	Network Operations Centre
NS	Non-Service Affecting
OSO	Outside Support Organization
PNOC	Provincial Network Operations Centre
SCAT	Stromberg-Carlson's Assistance Team
TDRS	Trouble Diagnosis and Repair System

BIBLIOGRAPHY

Carson, J. S. 1996. So you want to be a simulation consultant. p. 111-113 *In*: J. M. Charnes, D. J. Morrice, D. T. Brunner and J. J. Swain [ed.] Proceedings of the 1996 Winter Simulation Conference.

Chen, A. L. P., E. J. Cameron, G. F. Shuttleworth and E. C. Anderson. 1988. A simulation approach for network operations performance studies. p. 105-112. *In*: G. J. Knafl [ed.] Proceedings COMPSAC 88: The twelfth international computer software and applications conference, Computer Society Press of the IEEE, Washington D. C.

de Souza e Silva, E. and M. Gerla. 1991. Queueing network models for load balancing in distributed systems. Journal of Parallel and Distributed Computing 12: 24-38.

Draeger, M. A. 1992. An emergency department simulation model used to evaluate alternative nurse staffing and patient population scenarios. p. 1057-1064 *In*: J. J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson [ed.] Proceedings of the 1992 Winter Simulation Conference.

Evans, G. W., T. B. Gor, and E. Unger. 1996. A simulation model for evaluating personnel schedules in a hospital emergency department. p. 1205-1209. *In*: J. M. Charnes, D. J. Morrice, D. T. Brunner and J. J. Swain [ed.] Proceedings of the 1996 Winter Simulation Conference.

Fagerstrom, R. and J. Healy. 1993. The reliability of LEC telephone networks. IEEE communications magazine, June 1993, p. 44-48

Gordon, B. 1995. An exploration of the tradeoffs between repairperson staffing and skill levels and the prospect of multiple unrepaired critical faults. Proposed research project, MTS.

Lavenberg, S. S. and C. H. Sauer. 1983. Analytical results for queueing models. *In*: S. S. Lavenberg [ed.] *Computer performance modeling handbook.* Academic Press, N. Y.

Law, A. M. and W. D. Kelton. 1983. *Simulation modeling and analysis.* McGraw-Hill Inc, N. Y.

Law, A. M. and M. G. McComas. 1996. Simulation of communications networks. p. 73-77. *In*: J. M. Charnes, D. J. Morrice, D. T. Brunner and J. J. Swain [ed.] Proceedings of the 1996 Winter Simulation Conference.

MacDougall, M. H. 1987. *Simulating computer systems.* MIT Press, Cambridge, Mass.

McGuire, F. 1994. Using simulation to reduce length of stay in emergency departments. p. 861-867 In: J. D. Tew, S. Manivannan, D. A. Sadowski and A. F. Seila [ed.] Proceedings of the 1994 Winter Simulation Conference, IEEE, N. Y.

Molloy, M. K. 1989. *Fundamentals of performance modeling.* Macmillan Publishing Company, N. Y.

Regnier, J. and W. H. Cameron. 1990. State-dependent dynamic traffic management for telephone networks. IEEE communications magazine, October 1990. p. 52-53.

Russell, E. C. 1983. *Building Simulation Models with SIMSCRIPT II.5.* CACI Inc.-Federal, Los Angeles.

Sargent, R. G. 1996. Verifying and validating simulation models. p. 55-63. *In*: J. M. Charnes, D. J. Morrice, D. T. Brunner and J. J. Swain [ed.] Proceedings of the 1996 Winter Simulation Conference.

Trivedi, K. S. 1982. Probability and statistics with reliability, queuing, and computer science applications. Prentice-Hall Inc., Englewood Cliffs, N. J.

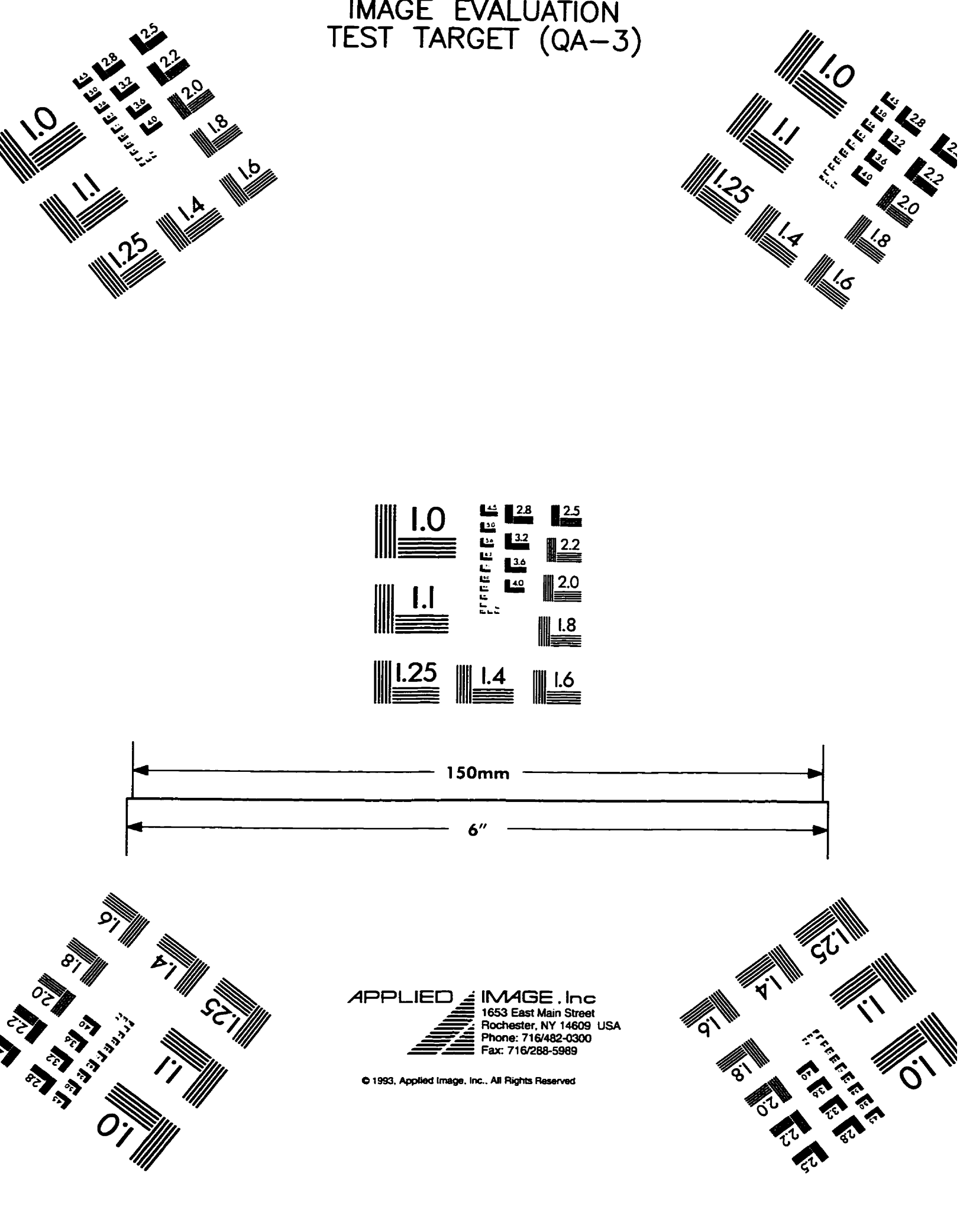
IMAGE EVALUATION
TEST TARGET (QA-3)
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
150mm
6"
APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989
© 1993, Applied Image, Inc., All Rights Reserved