A SIMULATION MODEL FOR TRAFFIC LOADING ANALYSIS

by

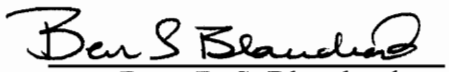
Mark A. Athearn

Project Report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the
degree of

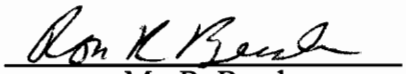
MASTER OF SCIENCE
in
Systems Engineering

APPROVED:

Dr. S. F. Midkiff
(Committee Chairman)



Dean B. S. Blanchard



Mr. R. Bender

May 1996
Blacksburg, Virginia
Traffic Simulation

LD
5655
V251
1996
A844

A Simulation Model for Traffic Loading Analysis

by

Mark A. Athearn

Committee Chairman: Dr. S. F. Midkiff

Systems Engineering

(ABSTRACT)

A simulation model for traffic loading analysis is needed to aid the designers of land-mobile radio communication systems. In the past, our system designers have relied on analytical solutions to perform traffic loading analysis. As our communication system evolved and changed from a voice-only system, to a voice and data system, it became apparent that the assumptions needed for an accurate Erlang C prediction were no longer valid. Thus, the need for a simulation model was realized in order to better predict the radio system's capacity to support a given traffic load.

This paper describes the process by which the simulation model was designed, developed, and validated. The simulation modeling system was developed using systems engineering methodologies. This document describes the need for a traffic analysis tool and the development process followed, from conceptualization through test and evaluation. An analysis package is also presented to show the verification and validation of the simulation model. Recommendations and a summary conclude the discussion.

Dedication

To my wife, Tammy, and our sons, Mitch and Trent

Acknowledgments

I would like to thank the members of the project work team. Without their dedication and efforts this project would not have been possible. Thanks to Tom Shannon, Jason Johnson, Jim Silverstrim, David VanderStaay, and Rick Taylor.

I would also like to thank my advisors, Ron Bender, Dr. Midkiff, and Dean Blanchard for their guidance and support during this project.

A special thanks to my wife, Tammy, and our sons, Mitch and Trent, for enduring the lonely weekends and evenings for the past five years.

Table of Contents

List of Figures

List of Tables

1.0 Introduction

I am presently employed as a Systems Engineer with a company that develops, manufactures, and delivers, land-mobile radio communications systems. These systems provide wireless voice and data communications to end users in a wide range of markets. These markets include the utilities industry, the public sector, such as fire and police departments, the federal government, and international entities. As a systems engineer, my design activities evolve around the following areas:

- Site Selection - Provide optimal RF coverage to the customer's service territory while minimizing the number of sites.
- Traffic Analysis - Provide enough communication capacity to adequately handle the customer's needs, being careful not to over-design due to cost concerns.
- Frequency Planning - Provide a plan to reuse frequencies at sites throughout the customer's service territory to minimize the number of frequencies required, while minimizing adjacent and co-channel interference.
- System Configuration - Provide a properly configured system that meets the customer's needs.

This paper presents a simulation model for traffic loading analysis. The paper will describe the need for the model, its development, and its subsequent analysis.

2.0 Definition of System Requirements

2.1 Definition of Need

Until the recent past, our communications systems handled voice calls only. These voice calls would arrive into the system and if a RF (radio frequency) channel, or server, were available the call would be processed and sent out over-the-air. If a channel were unavailable, the call would be queued until a server became available. To properly design a system to handle a given voice load, the Erlang C *M/M/m* analytical model was employed. The assumptions needed to justify the use of the Erlang C solution were evident. The large number of independent radio users created a stochastic process of arriving calls, known as a Poisson process. One of the properties associated with the Poisson process is that the call interarrival times are exponentially distributed. The service times are also independent of one another and independent of the interarrival times. The Erlang C analytical model worked well for this type of call activity. Then wireless data became a reality.

Our systems are now capable of transmitting and receiving RF data. However, the system does not queue data calls. Outbound data calls, host computer to mobile radio, are retried up to 3 times if a server is unavailable. Inbound data calls are dropped immediately if a channel is unavailable. The data messages are broken into smaller packets for transmission over-the-air. These packets are assembled into discrete 'burst' packets of a certain size for transmission. The Erlang C assumptions are no longer true for data calls. Data calls are not queued and a deterministic service time replaces the exponentially distributed service times of the Erlang model. Furthermore, the Bit Error Rate (BER) of the RF environment will also affect retries, throughput, and channel availability.

The Erlang C analytical model is incorporated into a spreadsheet as a traffic analysis tool. When our systems became data capable, a data table was added to this analysis spreadsheet. The data table allows the system designer to input up to four different call types, each with a unique message length. An estimated transmission time is associated with each message length through the use of a lookup table. The average transmission time is used for the mean value of the exponentially distributed service time. However, because data calls are not queued, the assumptions needed for an Erlang C solution are no longer valid. An Erlang B solution, which assumes blocked calls are denied access to the system and are dropped, is not valid for our combined voice and data system because the voice calls are queued. Given these facts, and the uniqueness of the communication system's design, a simulation model of the system needs to be employed to provide a more accurate traffic analysis.

The simulation model shall be available for use by trained system designers by May 1, 1996. Budgetary constraints for the design, development, test, and deployment of the system dictate that the total system cost shall not exceed $60,000.

2.2 *Feasibility Analysis*

Several senior members of the systems engineering staff began investigating the uses of simulation modeling. I was familiar with the use of models to perform work load simulations for factory automation and process flow control from my prior experience in industrial automation. These factory process models involve topics such as resource availability and queuing theory. I was also aware that communications system theory involves many of the same theoretical components as the factory models. Further investigation showed that simulation modeling is used extensively in communication systems and network loading analysis.

Our communications system has a limited number of servers and each server has a finite service time. Calls arrive into the system, at a given arrival rate, and are acted upon based on the availability of a server. If a server is unavailable the call is either queued or blocked. Simulation modeling is well known for analysis of these types of queuing systems. Literature was sought that pertained to simulation modeling and computer systems performance analysis. A textbook, "The Art of Computer Systems Performance Analysis" [1] was reviewed during this stage. A simulation model software vendor was brought to our offices to demonstrate their simulation modeling software and to answer questions about the uses of the modeling software. At the conclusion of this feasibility analysis, it was determined that a simulation model could be developed which represented our system and would allow the system designer characterize the traffic load.

2.3 System Operational Requirements

2.3.1 Mission Definition

The primary mission of the traffic simulation model is to accurately predict the optimum number of communication channels at a single site to satisfy the radio users' needs. The model shall combine voice, data, and telephone interconnect calls into a single, integrated tool. The model will be able to accept user traffic profiles as an input, and will deliver the probability of queuing or blocking for a given number of working channels as an output. The probability of queuing and blocking is also referred to as the Grade of Service, or GOS, in this document. A sample user profile is shown below in Table 1.

Table 1. Sample Customer Traffic Profile

	Description	Active Users	Msg Size (bytes)	Calls (hr/user)	Duration (secs)
Voice	Group Calls	507	-	2.0	4.4
	Individual Calls	350	-	0.2	4.8
	Interconnect Calls	282	-	0.01	60
Data	Outbound Message	159	200	20	-
	Inbound Message	159	50	10	-
	Automatic Vehicle Locator (AVL)	83	98	15	-

Primarily, the system will be used by communication system designers to aid in determining the correct number of RF repeaters, or channels, needed to adequately handle a proposed traffic load. The design goals are to provide the right number of working channels to produce a 10% or less GOS.

A secondary mission of the simulation model will be to execute performance analysis comparisons in order to gain insight into system design alternatives. The primary mission of the model may preclude the model from performing detailed and specific performance analysis of certain design alternatives. However, a certain level of this type of analysis is expected and may provide insight into the development of other models to specifically test various design alternatives.

Analysis of system performance will involve measuring the effects on the system's GOS, a dependent variable, while varying several of the system's independent, or controllable, variables. Table 2 below lists these system design variables.

Table 2. System Design Parameters

Communication System Parameters	
Independent	**Dependent**
Number of Working Channels	Grade of Service (Probability of Queuing or Blocking)
Call Arrival Rate	
Call Service Time	
Data Message Size	
Bit Error Rate	

2.3.2 Performance and Physical Parameters

The computer simulation model will be developed with the following performance and physical parameters.

1. The simulation software shall be compatible with standard IBM PC compatible computer platforms and shall operate in the Microsoft Windows environment.
2. The simulation software shall be able to be ported to the Microsoft Windows 95 or NT platform will no modeling software changes necessary.
3. The model shall estimate the number of working channels with a 95% confidence level of +/- 1% when designing systems to a 10% GOS, or +/- 0.5% when designing systems to a 1% GOS.
4. The model shall complete a single simulation, otherwise known as a replication, to the confidence levels stated in item 3, in less than 15 minutes.
5. The user will be presented with a means to input traffic profile data into the model and a means to obtain the results.
6. The system shall present the user with a means to create and execute batch runs of multiple replications.
7. The user shall be able to use the model at the completion of a one hour training session.

8. The simulation computing platform shall operate in a standard office environment.

2.3.3 Use Requirements

A runtime version of the simulation model will be available for use by each market focus group and each distant hub location. Due to the standard nature of the hardware/software computing platform, the model will be available for use at anytime. The simulation modeling software shall be able to be fully installed and made operational on any IBM PC compatible computer, within 15 minutes. As stated in Section 2.3.2, the model shall complete a single run in less than 15 minutes. The model will have the capability of batching multiple runs together and executing the batch with no external assistance.

2.3.4 Operational Deployment

The operation deployment of the simulation model is shown in Table 3 below. The software shall be distributed on 3.5 inch diskettes.

Table 3. System Deployment Locations

Location	Number of Runtime Versions
Utility Systems Engineering	2
Federal Systems Engineering	1
North America Engineering	2
International Engineering	2
Atlanta Hub	1
Tampa Hub	1
Kansas City Hub	1
Los Angeles Hub	1
Philadelphia Hub	1

2.3.5 Operational Life Cycle

The initial version of the simulation model is expected to be in operational use for 5 years. Communications technology is rapidly changing, and newer versions of the model will be explored and implemented as necessary.

2.3.6 Effectiveness Factors

The effectiveness factors for the simulation model are specified in Table 4. The modeling software's ability to operate on any IBM PC compatible computer system and its ability to be installed and made operational on any such system greatly enhances many of the following figures of merit.

Table 4. Effectiveness Factors

Figure of Merit	Value	Comments
Operational Availability, A_o	95%	Model may be made available on other PCs, when failures occur.
Mean Time between Maintenance, MTBM	N/A	No maintenance required.
Mean Time between Failure, MTBF	1000 hours	
Mean Time to Repair, MTTR	4 hours	

2.3.7 Environment

The system will be used in an office environment, where standard office grade computing platforms can operate efficiently. The operating temperature of such an environment will be between 60 and 90 degrees Fahrenheit. Humidity will typically be in the range of 50 to 70 percent.

Another aspect is the software operating environment. This environment consists of Microsoft Windows 3.1. The modeling software must be designed to operate in this environment and be upgradeable to run in the MS Windows 95 and the MS Windows NT environments.

2.4 System Maintenance Concept

2.4.1 Supportability Concept

The simulation model will consist of both hardware and software. The hardware will be standard, office grade computer systems. These computer systems will be leased and maintained through the leasing organization. The leasing organization will be responsible for hands-on repair and computer system swap out. The software model will be supported at three levels. The model will be constructed and maintained by cooperative undergraduate students. The co-ops will be responsible for development, modifications, documentation, testing (Type I) and debug. The model's design, including both technical and strategic aspects, will be the responsibility of the system engineers within the Wide Area System Design Team. The modeling software vendor will provide the third level of support. The vendor shall be capable of technical support and training support.

Once the model is ready for operational use, software distribution and logistics support will be controlled by the Software Quality Control department.

2.4.2 Reliability Concept

The system's reliability will be limited to the hardware reliability of the computing platform chosen. Standard procedure shall dictate the use of more than one similar computer at a given site. This computer redundancy will allow the traffic model to be moved to another computer platform should one system fail.

Software reliability is a separate issue and will be discussed in detail in the Test and Evaluation Master Plan in Section 6.0.

2.4.3 Maintainability Concept

Computer hardware maintenance shall be performed by the leasing organization. Failed computer systems shall be repaired or swapped out within 4 hours of the reported failure. The modeling software will require no field maintenance. Any modifications to the modeling software will be directed and performed by the Wide Area Systems Design Team.

2.4.4 Equipment Packaging Concept

Standard office grade computer systems shall provide the operating platform for the simulation model.

2.4.5 Manability Concept

The simulation model shall provide an easy to use Direct Manipulation Interface, or DMI, between the user and the model via the computer keyboard and monitor. The DMI shall provide both input parameter and output result computer displays. All operator input/output (I/O) shall also be made available in hardcopy form. The input parameter sheet shall be designed to allow the user to logically enter the traffic profile information in a sequential, straightforward manner. The output results sheet shall provide the user with a clear and concise summary report of the simulation model's results.

A second level of user interaction shall be provided by the interface. The standard input/output sheets shall allow traffic profiling and analysis to be performed by engineering personnel who may be unfamiliar with the inner workings of the model. A second level of input/output shall be available for the more sophisticated user and will not be accessible by the ordinary user. The second-level input sheet will allow the modeled communication system's design parameters to be varied. This will allow performance

analysis to be performed on differing design alternatives. The second-level output sheet will allow the sophisticated user access to sensitive performance data.

2.5 Section Summary

Engineering activities are directed toward the common goal of delivering a system to meet the desired need by the knowledge of how the system will be utilized and supported. The definition of these operational and maintenance requirements represents the starting point to the engineering process. The definition of the system requirements completes this phase of the design process. The next phase of the process is known as advance system planning.

3.0 Advance System Planning

3.1 System Specification

3.1.1 System Description

The land mobile radio communication system which will be modeled allows voice and data communications to be combined in the system. The voice calls may be queued if a channel is unavailable, but data calls will be blocked and dropped from the system. Either call type may be modeled analytically with the Erlang C or Erlang B equations, respectively, but the combination of queuing and blocking does not lend itself well to an analytical solution. However, a simulation model will be able to combine voice and data calls, with queuing and blocking characteristics, onto a common platform for analysis. The model will also allow many of the communication system's design parameters to be built into the model. These parameters include access times, delay times, response times, and characteristics associated with the selective repeat ARQ (automatic retry request) algorithm. External system design parameters, such as the bit error rate (BER), will also be an input variable to the model.

The model will allow a customer profile to be input, and the simulation will produce performance metrics as its output. As stated above in Table 2, the primary controllable output variable sought is the Grade of Service, or GOS. The GOS can be thought of as either the probability of queuing in a queuing system, or the probability of blocking in a blocking system. Secondary performance metrics, such as end-to-end throughput, percent of data calls successful on the first transmission attempt, and time in queue will also be presented in the results.

A conceptual model of a communication system is depicted in Figure 1.

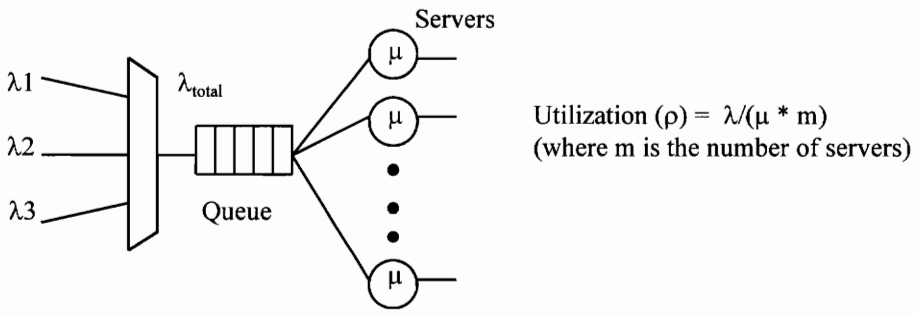


Figure 1. Conceptual Model of a Communication System

The conceptual model shown above represents a statistical multiplexing communication system. The term statistical multiplexing comes from the fact that the servers are assigned based on a demand basis. Voice and data calls arrive into the system with a Poisson distribution. The call interarrival times are, therefore, exponentially distributed about a mean value. The calls arriving into the system are shown with a mean arrival rate, λ. The multiple call arrival streams are combined into a single stream, λ_{total}. As calls arrive into the system, they are assigned to a server for transmission. In the case of voice calls, if all the servers are busy, the call enters the queue and waits. The queue is a First-In-First-Out (FIFO) queue. These call types leave the queue as servers become available. The servers have a service rate associated with them. The service rate, μ, is the number of calls per second the server can handle. The inverse of the service rate is the service time, or in the case of an RF repeater, its transmission time or call duration. The Erlang C *M/M/m* analytical solution assumes that service times are exponentially distributed about a mean call duration. In the case of the actual communication system, this assumption is still valid for voice communications, but is invalid for data communication. A data call has a more deterministic service time. Data call transmission times are based on factors such as message size, the RF channel bandwidth and the bit error rate of the environment.

An important concept that needs to be understood at this point is the utilization of the communication system. The utilization, ρ, is shown below.

$$\rho = \lambda / \mu$$

By observing the relationship between the arrival rate and the service rate, one can easily see the importance of these two parameters. As the arrival rate approaches the service rate, the utilization approaches 100%. If the arrival rate equals or exceeds the service rate, the queue depth will grow to infinity in a queuing system. The utilization factor is sometimes referred to as the system offered load. In order to adequately handle a high offered load, additional servers must be added to the system. By adding *m* servers, the system's utilization now becomes

$$\rho = \lambda / (\mu m)$$

As the above utilization equation shows, *m* servers must be added to a system in order to provide the customer will a utilization which will result in a low probability of queuing or blocking.[3]

The conceptual model in Figure 1 is used to provide a basis and understanding of the design of the simulation model. The simulation model is an analysis means that lies between the general assumptions of the conceptual model and the Erlang solutions, and the detailed intricacies of the actual system. Many simulation models failed because too much detail was built into the model. Attempts at developing complex models often result in a model which is unfinished, incapable of being debugged, incapable of being verified and validated, and incapable of being maintained[1]. Special care will be taken to make simplifying assumptions whenever possible. Tradeoffs between simplicity and detailed design characteristics will be made as warranted.

3.1.2 Functional Tasks and Interfaces

The major functional elements of the simulation tool are shown below in Figure 2.

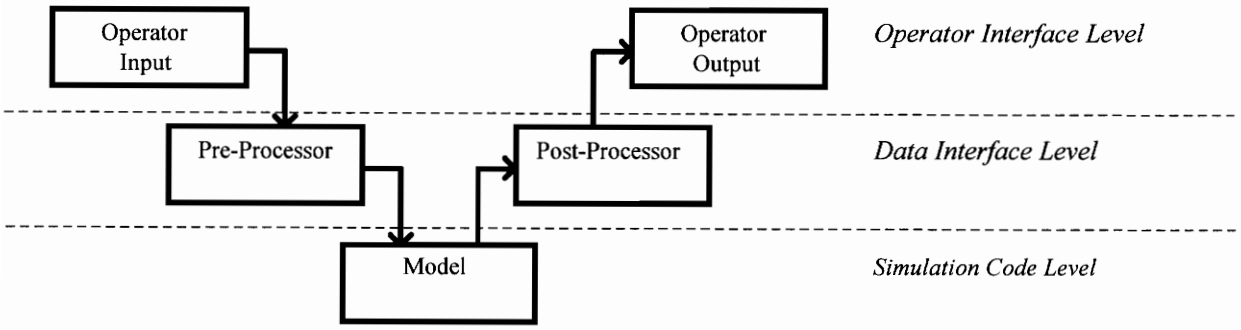


Figure 2. Functional Tasks

As defined in the operational requirements, the system designer will be presented with a means of inputting data, outputting data and controlling the model. The Operator Interface Level shown above will be responsible for this task. An intermediate level task has been defined for a Data Interface Level, which contains the pre- and post- processors. The pre-processor is responsible for accepting the operator supplied information and translating this information into the proper data and format for input into the model. The post-processor will act in much the same fashion, accepting the results from the model and performing the necessary data and file manipulations before transferring the results to the operator's output screen.

The operational requirements for the Data Interface Task can now be expanded. The actions needed to translate input traffic profiles into data the model can accept involve many mathematical calculations. On the output side, the simulation model will present results which must be statistically analyzed over several runs of the model in order to gain confidence in the results. These mathematical needs place additional requirements on the system. The Data Interface Task must be capable of easily accepting numeric data and performing mathematical and statistical calculations on these numbers.

3.1.3 The Model's Input/Output Analysis

An input/output analysis is depicted below in Figure 3. The analysis is a high-level look at the model's input and output parameters and its key processes. This input/output analysis will be used as a basis to further define the I/O as the design progresses.

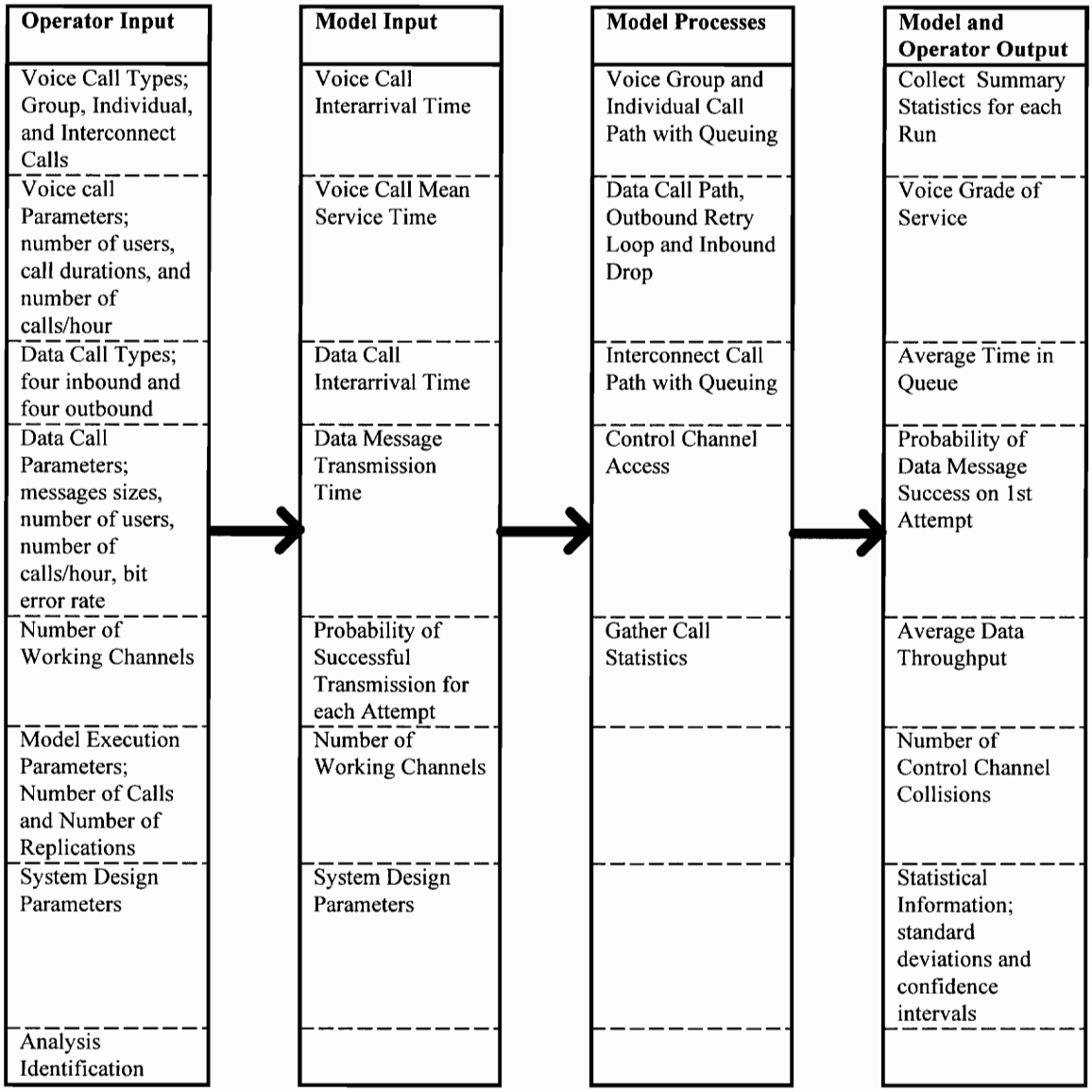


Figure 3. Input/Output Analysis

3.2 Work Breakdown Structure

To better organize and conduct the activities needed to successfully design and develop the simulation model, the Work Breakdown Structure (WBS) shown in Figure 4 was developed.

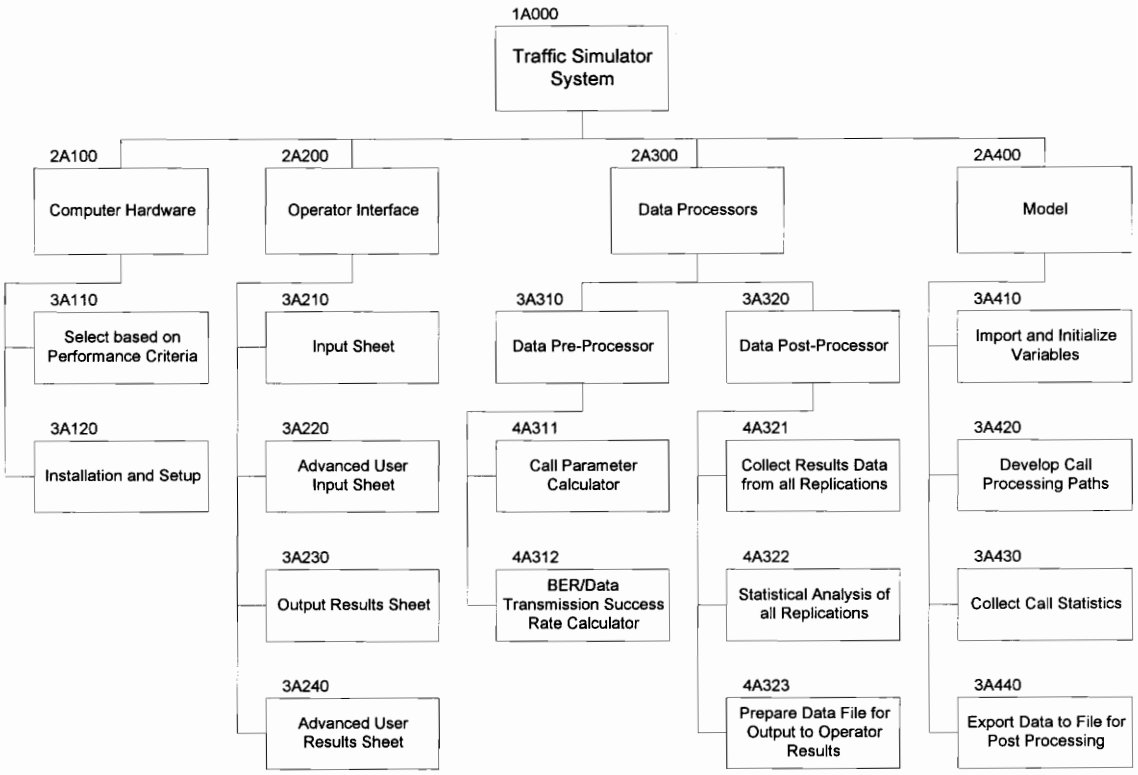


Figure 4. Work Breakdown Structure

3.3 Work Team

A cross-functional work team was assembled to design and implement the traffic simulation model. This work team consisted of members from different market segment groups, the System Parametrics group, and the Wide Area System Design group. This diverse team represented the communication system design function, the communication system test function, and the end users of the model. The work team met informally every week to map out strategy, discuss options and alternatives, review progress, and move the project forward.

3.4 Conceptual Design Review

A conceptual design review was held at this point in the design process. Members of the design review team included persons from Systems Engineering management, System Parametrics, and consulting members of the design work team. Agreement was reached to proceed to the preliminary system design phase.

4.0 Preliminary System Design

The preliminary system design activities included the system functional analysis, a operational task analysis, the allocation of requirements, a trade-off and optimization phase, and a synthesis and definition phase. These activities culminated in a preliminary system design review.

4.1 System Functional Analysis

A system functional analysis was performed to form a design bridge between the system operational requirements and the detailed design. The functional analysis identifies *what* is to be accomplished, not *how* it will be accomplished. The functions of the system are identified, in a top-down design approach, beginning with gross functions at the highest levels of the analysis. Subsequent lower levels of analysis are performed which break the higher level functions down into finer and finer detail. This process aids the systems engineer in defining the requirements which will drive the detailed design activities. After the functional analysis has been performed, one may determine how each objective will be accomplished. Each function identified can be assigned to a hardware device, a software routine, a data table, an operator screen, or a maintenance task. The activity of assigning requirements identified by the functional analysis is known as requirements allocation [2]. Figures 5 through 12 show the functional flow diagrams for the system.

4.1.1 Functional Flow Diagrams

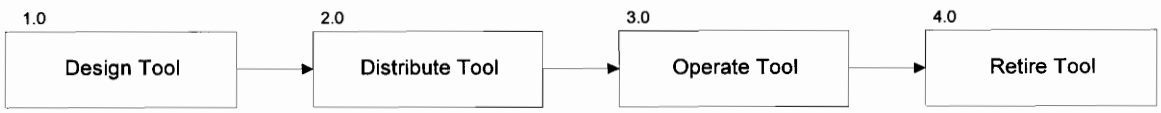


Figure 5. Top-Level Operational Tasks

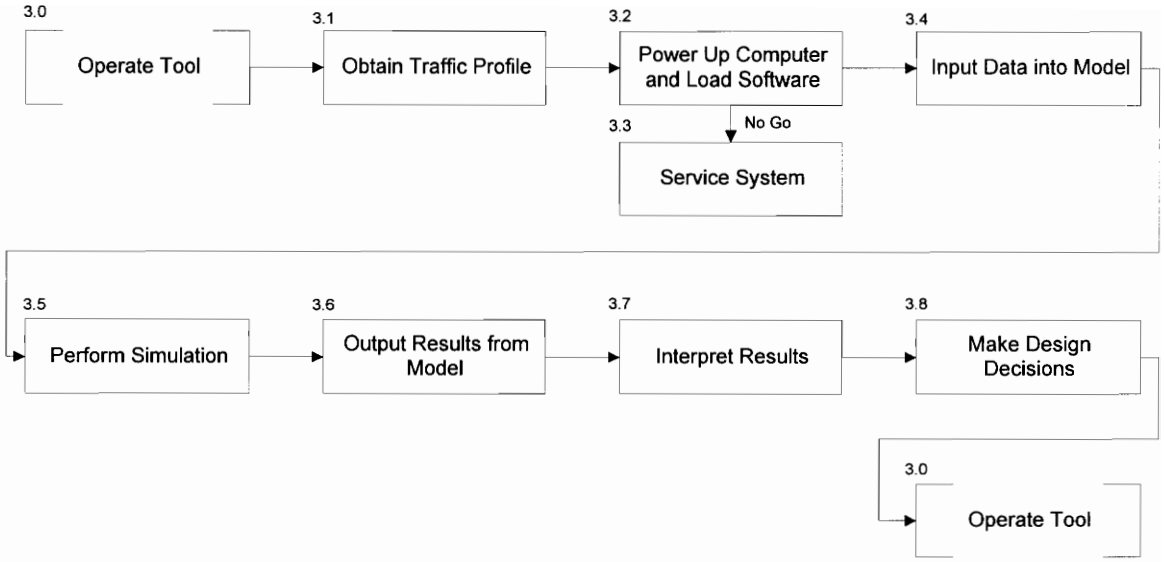


Figure 6. Second-Level Operational Tasks

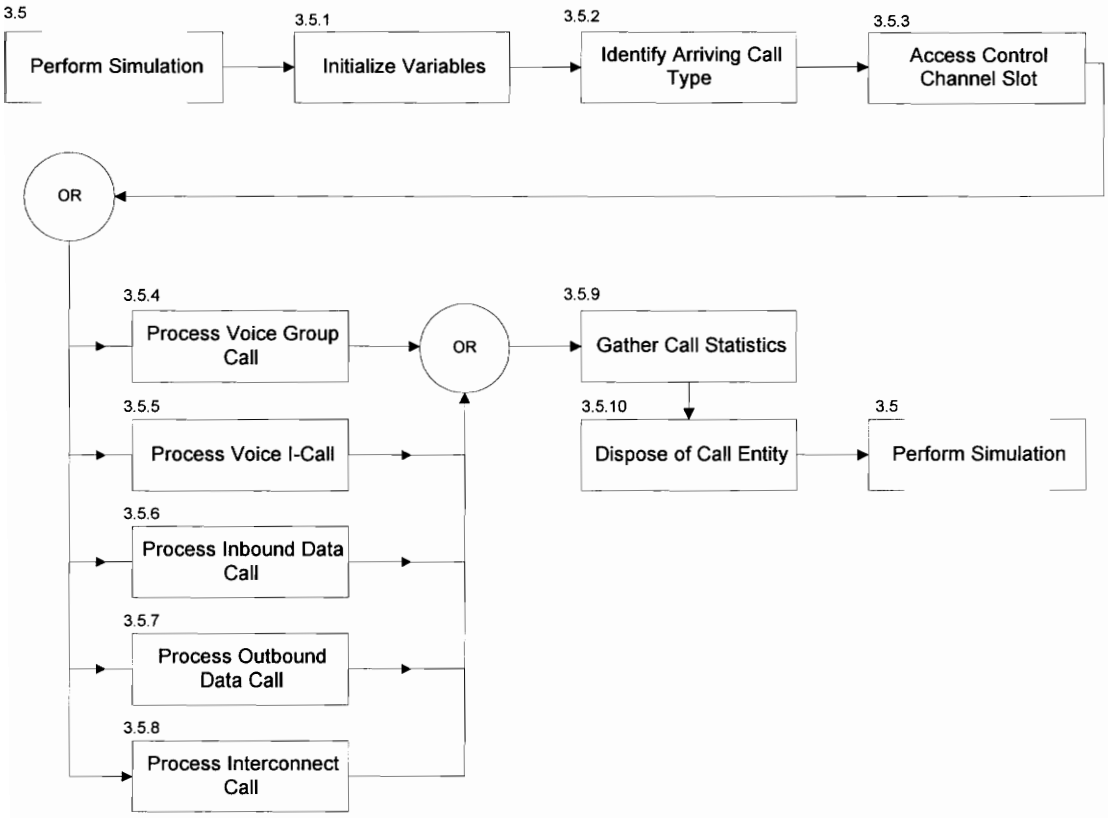


Figure 7. Third-Level Simulation Model Tasks

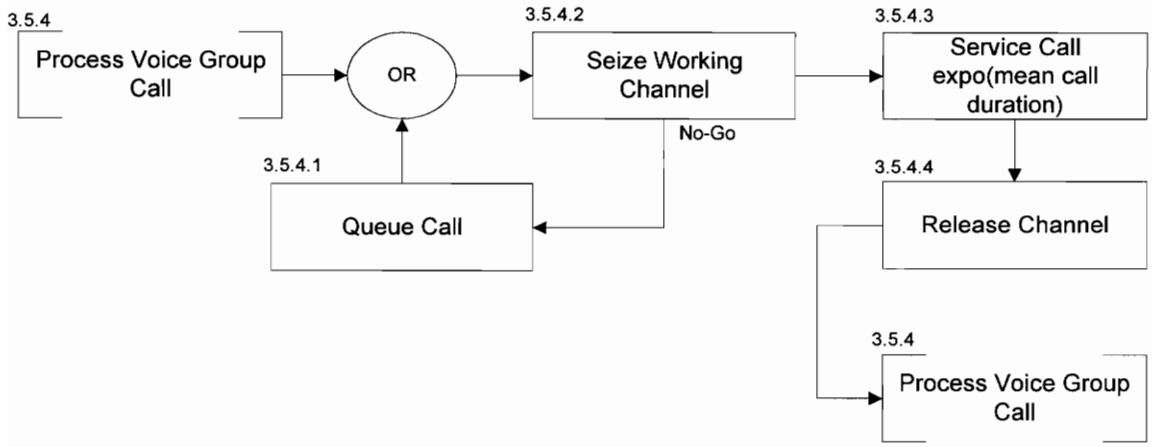


Figure 8. Fourth-Level Group Voice Call Tasks

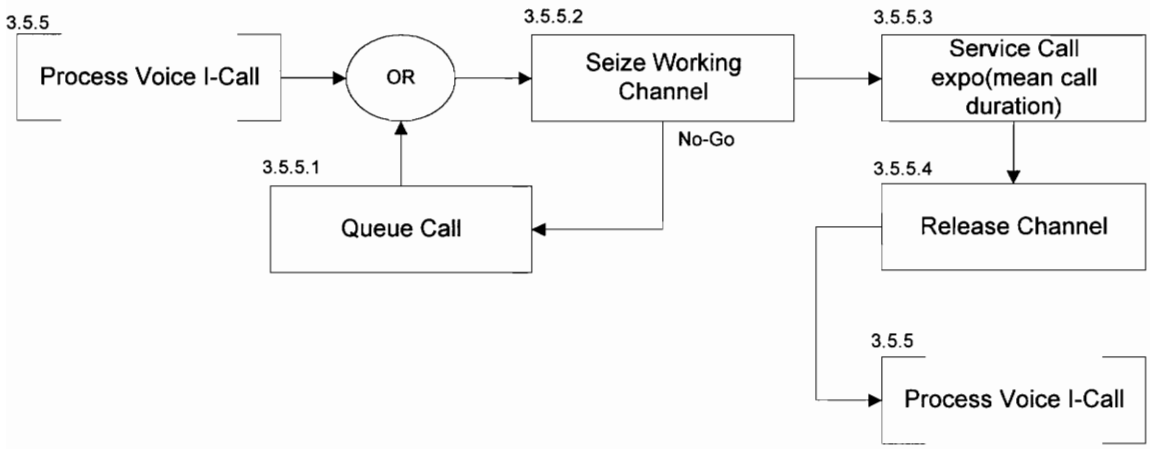


Figure 9. Fourth-Level Voice Individual Call Tasks

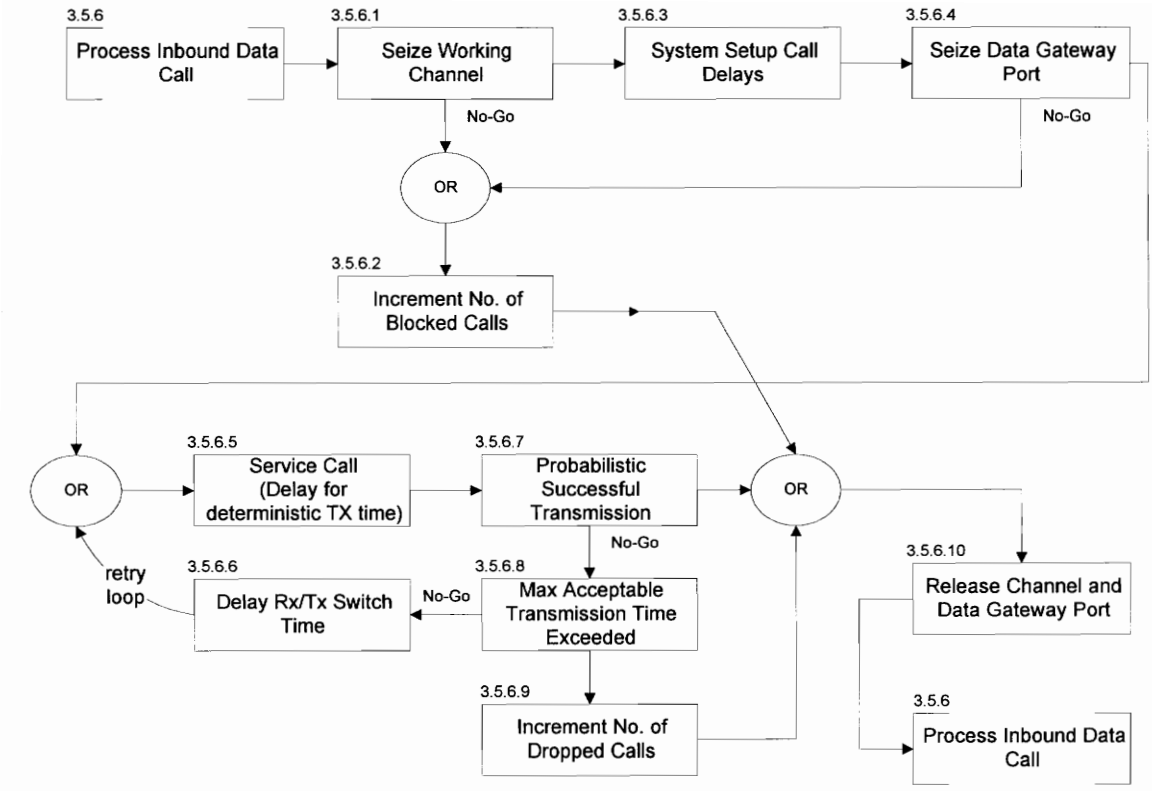


Figure 10. Fourth-Level Inbound Data Call Tasks

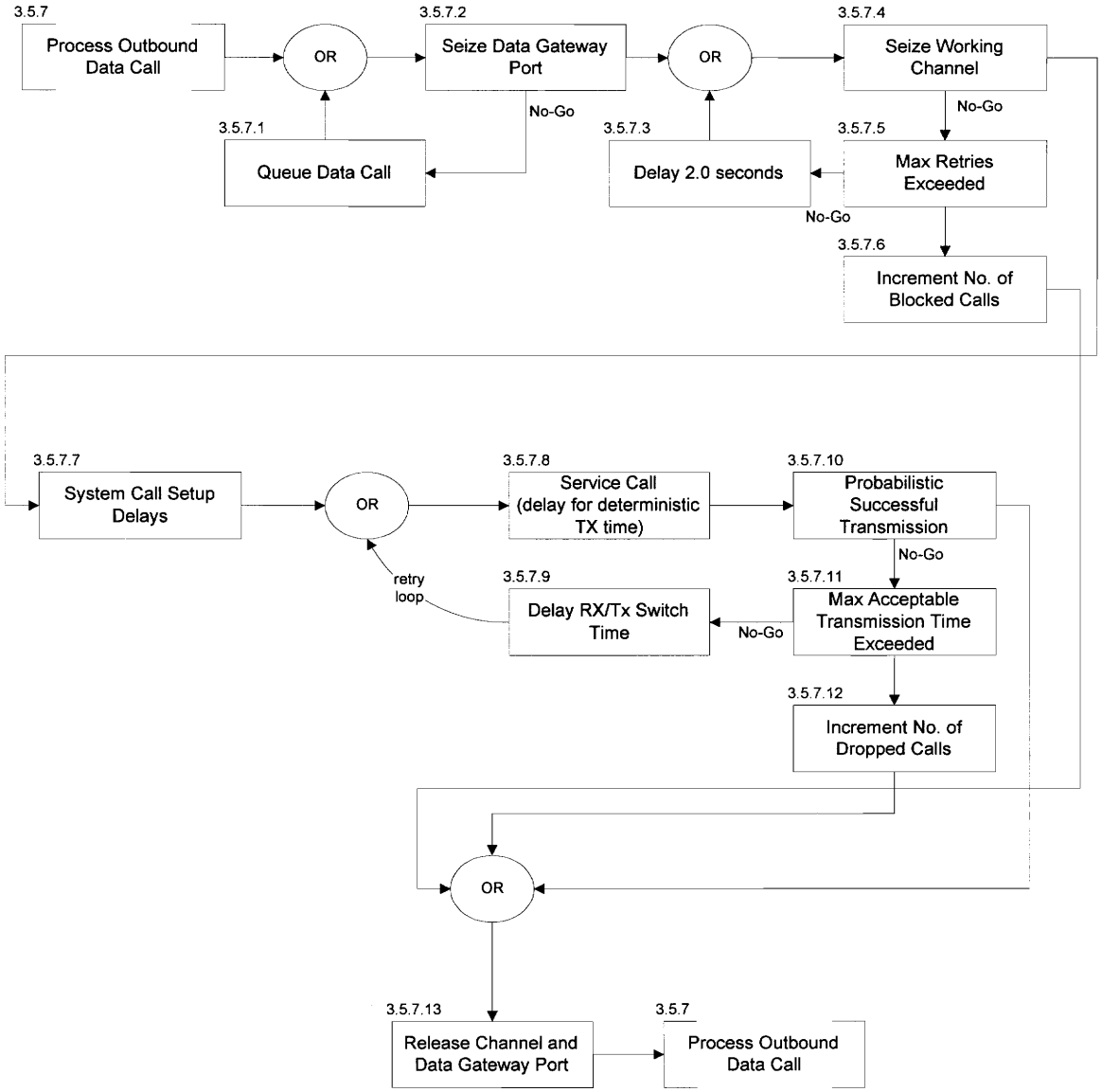


Figure 11. Fourth-Level Outbound Data Call Tasks

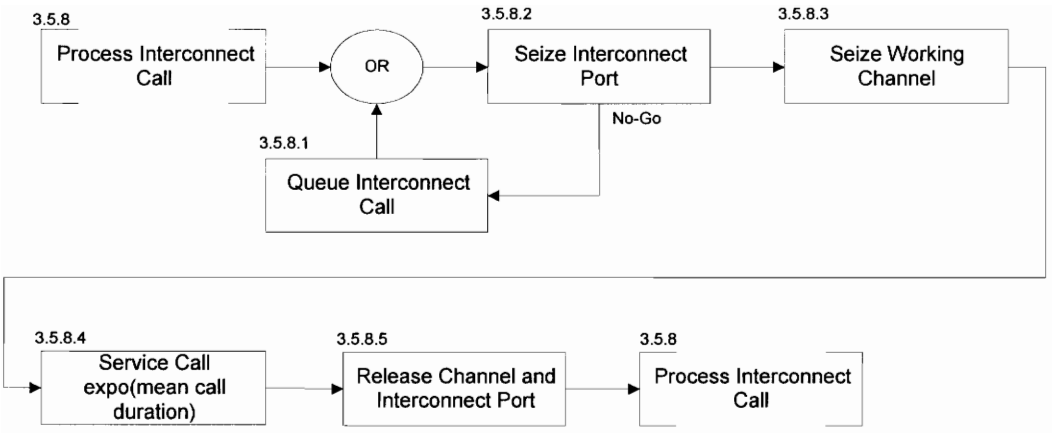


Figure 12. Fourth-Level Interconnect Call Tasks

4.1.2 Operator Task Analysis

An operator task analysis was performed, separate from the operational flow diagrams. This task analysis was done to focus on the operator interface requirements.

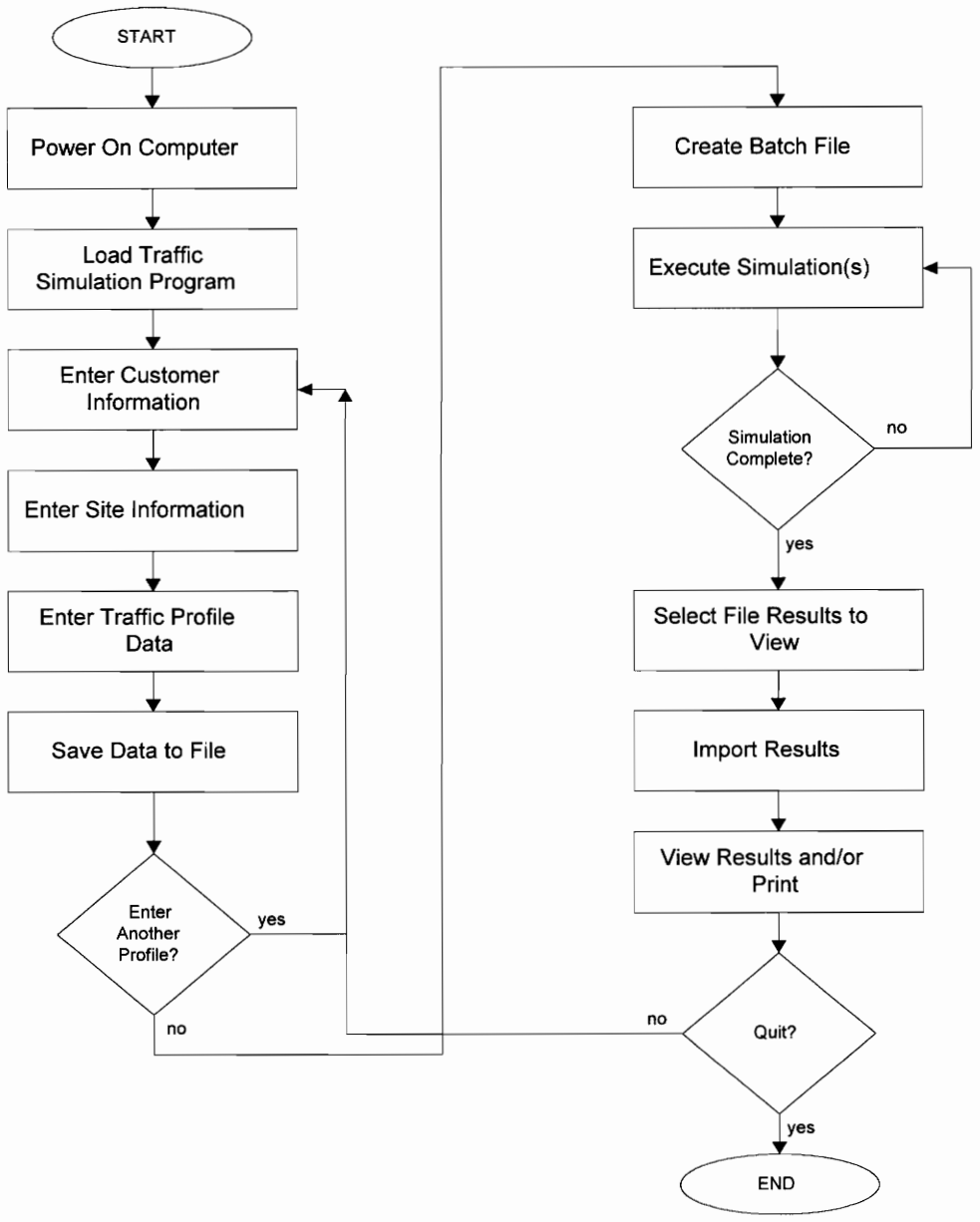


Figure 13. Operator Tasks

4.1.3 System Maintenance Functions

A flow diagram depicting the system maintenance functions of the traffic simulator is shown in Figure 14.

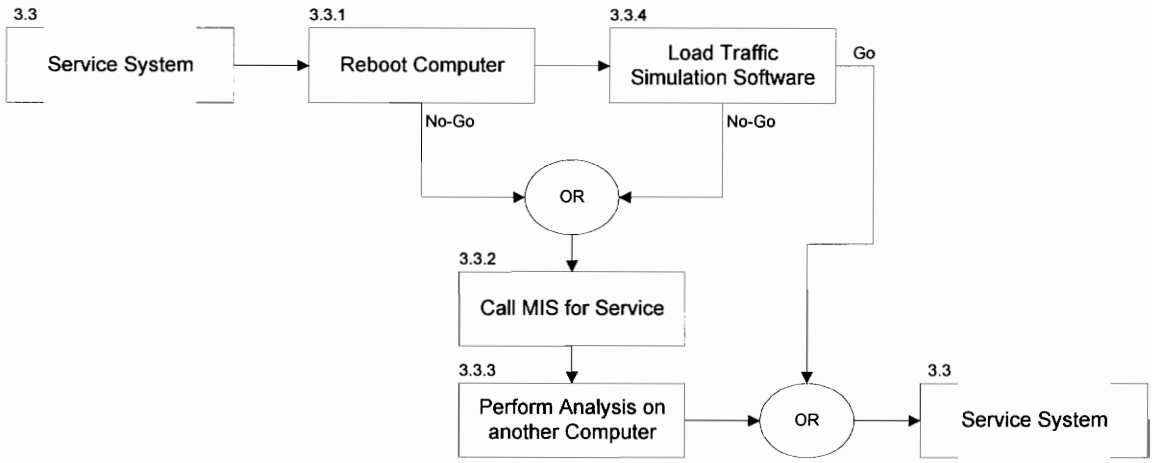


Figure 14. Third-Level System Maintenance Tasks

4.2 *Allocation of Requirements*

At this point in the design, the system requirements are allocated to the proper hardware subsystem and/or software module. The allocation of requirements normally entails defining the requirements for each successive subsystem in the overall system to ensure that the top-level requirements are met [2]. This design step is extremely important when different design groups are responsible for different subsystems. The resource allocation document helps each group to ensure that their portion of the design does not exceed their allotted design budget.

In this simulation modeling project, all the hardware related requirements are contained within a single PC computer system. Since this computer system is purchased as a complete system from a supplier, its allocation of requirements have been rolled up to the top level and are shown below in Table 5. These requirements are supplemented by the minimum hardware requirements as defined by the simulation modeling software vendor.

Table 5. Computer System Requirements

Element	**Requirement**
Cost	Less than $5000
Hard Drive	1 G byte
Memory	16 M byte of RAM
Monitor	SVGA
Processor	486 66 Mhz or greater
Printer	Hardcopy output
Ao	95%
MTBF	1000 hours
MTTR	4 hours

The next step is the allocation of the software requirements, and the determination of new requirements as the design is clarified with finer detail. This step is accomplished by examing the WBS and the functional flow diagrams which have been previously developed. From this information, a data flow diagram (DFD) was generated. A DFD is a tool which allows a software design to be depicted visually, and indicates the direction of data flow between software functional modules. The allocation of the software requirements are included in the discussion in Section 4.4.

4.3 Trade-Off and Optimization

During this stage of the design, many of the design team's discussions centered around the design attributes of the real system. The team struggled with the idea of how much detail should be built into the model. We desired an accurate representation of the system, however, we knew that to built an inordinate amount of detail into the model would jeopardize its completion and success. Building an extremely detailed model would result in enormous complexity, thereby making the model difficult to maintain and modify [1]. After a certain level of accuracy is obtained, efforts to increase the accuracy are considered counterproductive. Therefore, the team's weekly meetings generally centered on the level of detail that should be incorporated into the model, while analyzing the associated risks.

4.4 Synthesis and Definition

The synthesis and definition phase is important to the software developer. The developer must internalize the design in order to fully understand its structure, its logic, and its data components. The developer, through heavy involvement in this phase, begins to synthesize information and requirements into form and function. This stage should be well planned and executed, prior to proceeding to the detailed design phase. The DFD

aids the software developer in this process. A high level DFD for the simulation is depicted in Figure 15. Each entity shown in the DFD is discussed in the following paragraphs. The data components, which were derived through the allocation of the requirements are also presented in the following discussions.

The DFD contains three main data processors, which are depicted by circles. The rectangles depict external interaction interfaces and are the entry and exit points for the operator. Data stores are depicted as entities enclosed within parallel lines. The requirements associated with each of these items is discussed in the following paragraphs. The data tables associated with these entities are also presented.

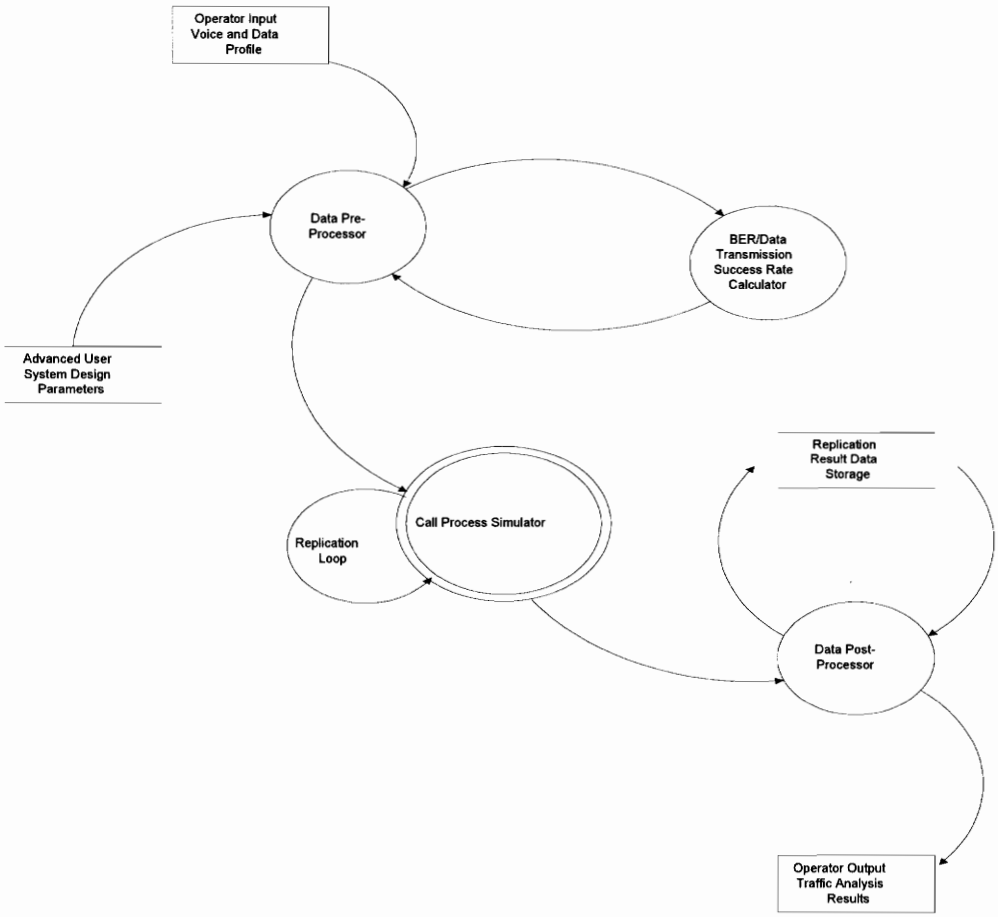


Figure 15. Traffic Simulator Data Flow Diagram

1.0 Operator Input, Voice and Data Profile

This external interface allows the operator to input data into the system. The interface is provided by the computer's monitor and keyboard. The data is sent to the Data Pre-processor for processing. The operator also exercises control over the task of performing a simulation from this interface.

Table 6. Input Data Table

Item	Comments	Units
customer name	80 characters	
notes	128 character descriptive field	
user name	80 characters	
RFP number	proposal identification number	
timestamp	system generated	
voice users	per busy hour	count
data users	per busy hour	count
voice group calls	per user per busy hour	count
voice I-Calls	per user per busy hour	count
voice interconnect calls	per user per busy hour	count
mean call durations	for all voice call types	seconds
data call types I to IV - outbound	per user per busy hour	count
data call types I to IV - inbound	per user per busy hour	count
data message size, types I to IV	range: 1 to 511	bytes
bit error rate (BER)	range: 0 to 5	percent
working channels	range: 1 to 20	count
data gateway ports	range: 4 to 32	
interconnect ports	range: 4 to 32	
simulated calls	calls to be processed by the simulator	count
replications	number of execution iterations	count

2.0 *Advanced User System Design Parameters*

The ability to do performance analysis is a secondary mission objective of the model. This objective places a requirement on the system to give the advanced user the ability to access and modify many system design parameters. These parameters are shown in Table 7.

Table 7. Advanced Input Data Table

Item	Units
control channel slot time	milliseconds
working channel access time	milliseconds
channel assign delay	milliseconds
working channel delay	milliseconds
call setup time	milliseconds
call drop time	milliseconds
channel drop time	milliseconds
ACK time	milliseconds
access slot size	milliseconds
RX to TX switch time	milliseconds
data transmission rate	bits per second
maximum queue delay	seconds
application retries	integer
retry delay	seconds

3.0 *BER/Data Transmission Success Rate Calculator*

A requirement exists to allow the user to input a desired bit error rate (BER) between 0% and 5%. By allowing the BER to be set to discrete values, the system designer is allowed to perform comparative analyzes between best case scenarios at 0% BER and other scenarios at higher BERs. The bit error rate for data transmissions in an RF environment is very high, as compared to wireline

applications. These high bit error rates can be attributed to many causes, some of which are multipath fading, Rayleigh fading, and other forms of RF interference.

The requirement to allow the system designer to perform an analysis with a discrete BER value led to a subsequent requirement for a BER data transmission success rate calculator. This processing entity was developed to calculate the probability of success for a given data message at a given BER. An algorithm was developed to calculate the probability of success while taking factors such as data packet size and the retry attempts into account. The over-the-air Automatic Repeat Request (ARQ) algorithm is also taken into account in these equations. The ARQ scheme used in the real system is a selective repeat ARQ, where only the corrupted packets contained in a multi-packet data burst are retried. This algorithm accounts for packets which are repeated multiple times within a multi-packet data burst to fill the minimum required burst size. Repeating packets within a single burst has the net effect of increasing the probability of success for those repeated packets.

Table 8. Data Transmission Success Rate Data Table

Item	Comments	Units
data types I through IV, probability of success on *i*th attempt	probability of success increases with each attempt	percent
data types I through IV, attempt counter	transmission attempt number	count
data types I through IV, transmission times		milliseconds

4.0 Data Pre-processor

The requirement for a data pre-processor arose from the fact that the data the user inputs is not the same data needed by the traffic simulation model. This input data must be manipulated before it is in the proper form to be presented to the model. The data pre-processor performs this processing function and has the added benefit of keeping with the desire for modular software.

Table 9. Pre-processor Data Table

Item	Comments	Units
interarrival time(s)	for each call type	seconds
voice call service time	mean value	seconds
data call attribute array	consists of call types I to IV, with success rate probabilities for each attempt, and the transmission time	percent and seconds, respectively
voice utilization factor	voice offered load	erlangs
data utilization factor	data offered load	erlangs
system utilization factor	combined voice and data load	erlangs

5.0 Traffic Simulator

The Traffic Simulator data processor is depicted in the data flow diagram as a multi-processing entity. This processing entity is the actual model. The model processes the different call types, each along its own processing path, in a concurrent manner. These calls and their associated processing paths are depicted in the functional flow diagrams of Section 4. The traffic simulator accepts its inputs from the data pre-processor, initializes its variables with this data, and begins executing. At the completion of each replication, the traffic simulator will pause while its result data is transferred to the data post-processor.

6.0 Replication Result Data Store

A traffic simulation analysis is typically made up of many separate replications. The results of each replication are statistically averaged to increase the confidence level of the results. The replication result data store is where the results from each individual replication is stored. The model generates output statistics at the completion of each replication, but is incapable of storing the results for later analysis. This fact led to the requirement for a replication result data store.

7.0 Data Post-Processor

The requirement for a data post processor came from the original requirement of obtaining results of the grade of service with a 95% confidence level. In order to achieve such a confidence level, more than a single replication must be executed. As stated above, the model is incapable of storing its output data for later analysis. Therefore, a data post-processor is needed to collect the output data from the model at the completion of each run and transfer this data to the replication result data store. After the model completes all of its replications, the data post-processor will perform a statistical analysis of the results, computing mean values, standard deviations, and confidence intervals.

8.0 Operator Output, Traffic Analysis Results

The traffic analysis results are passed to the operator's computer monitor, which is depicted as an external interaction interface in the data flow diagram. The data is computed and manipulated into the proper format by the data post-processor.

Table 10. Output Data Table

Item	Comments	Units
grade of service	probability of queuing	percent
arbitrary call delay	probability of a call remaining in queue beyond time *t*	percent
percent of outbound data calls successful on the first attempt		percent
percent of inbound data calls successful on the first attempt		percent
group call duration statistics	minimum, mean, maximum, standard deviation	seconds
individual call duration statistics	minimum, mean, maximum, standard deviation	seconds
interconnect call duration statistics	minimum, mean, maximum, standard deviation	seconds
inbound and outbound data call throughput, types I to IV	minimum, mean, maximum	bits per second
inbound and outbound data call TX times, types I to IV	minimum, mean, maximum	seconds
inbound and outbound data call attempts, types I to IV	average number of call attempts before successful	count
percent of calls blocked at the control channel		percent
total number of calls processed		count
simulation execution time		minutes:seconds

4.5 Preliminary Design Review

A preliminary design review was held at this stage of the design process. All preliminary designs and design activities were referenced to the system requirements to ensure that they were being met. The review team agreed to proceed forward with the detailed design and development phase.

5.0 Detail Design and Development

The detail design and development phase was begun after the formal preliminary design review was successfully completed. The bulk of the workload in this phase was the actual coding of the software needed to implement the model, the pre-processor, the post-processor, and the operator interface. This task required much effort and time from the software developer, but the design team continued to meet on a weekly basis to help resolve any issues or problems. Several actions that occurred early in this phase led to the prototypes of the major system elements. These elements included the model and the operator interface. An additional prototyping effort involved the development of a bit error rate simulator program to verify the correctness of the BER/Data Transmission Success Rate Calculator. These efforts are described in the following paragraphs.

5.1 The Erlang B and the Erlang C Prototypes

An early design activity involved the development and analysis of two prototypes of the model. One prototype would be designed to closely represent the Erlang C *M/M/m* equations and the other prototype would be coded to simulate the Erlang B *M/M/m/m* equations. The Erlang C model was designed as a queuing system with exponential distribution code blocks for a mean interarrival time and a mean service time. The Erlang B prototype was similar to the Erlang C prototype, except it was designed as a non-queuing system. These prototypes were executed with several different traffic loads, and the results were compared to analytical solutions. The results of this comparison showed that the models' predicted probability of queuing or blocking approached the closed form solution of the Erlang equations. By examining the standard error of the mean, it was determined that no significant statistical differences existed between these results. This effort allowed the design team to gain confidence and insight into the modeling software. The team hypothesized that we could develop the queuing paths and the non-queuing paths independently and test and verify their operation independently. The call

processing paths could be combined into a single model without compromising their integrity.

5.2 Operator Interface Prototyping

The preliminary design effort resulted in a solid definition of the operator interface I/O data elements. This definition, along with the operator interface requirements, led to the development of a prototype for the input and output sheets. These prototypes were reviewed by the work team and modifications and improvements were suggested. This prototyping activity lead to the I/O sheets which are shown in Figures 16, 17, and 18.

Note: Some of the terms in the I/O sheets follow product-specific nomenclature.

EDACS Traffic Analysis Simulation Configuration
for ONESITE Model

Model Version	*1.0*

Title:

Notes:

Author:	
Date:	
MBP Number	
Site Name/Num	
Scenario Name/Num	

Site Input Parameters

Voice Call Configuration

Call Type	Active Users	Calls/Hr/User	Duration (seconds)	Calls/Hr	Interarrival Time
Group Calls	0	0	0	0	0
Individual Calls	0	0	0	0	0
Interconnect Calls	0	0	0	0	0

Data Call Configuration

Bit Error Rate	0.75%

511 BYTE MAXIMUM

Data Call Type	Description	Active Users	Message Size(bytes)	Calls/Hr/User	Calls/Hr	Interarrival Time
RH I		0	0	0	0	0
HR I		0	0	0	0	0
RH II		0	0	0	0	0
HR II		0	0	0	0	0
RH III		0	0	0	0	0
HR III		0	0	0	0	0
RH IV		0	0	0	0	0
HR IV		0	0	0	0	0

Max Acceptable Voice GOS	*10%*
Max Prob. of Blockage of data calls	*5%*

Voice System Offered Load	*0.00*
Data System Offered Load	*0.00*
Recomended Number of Voice WC	0
Recomended Number of Data WC	0
Recomended Number of Voice and Data Channels	0
Number of working channels (1-23)	10
Number of EDG Ports	32
Number of lines of Jessica (1-24)	4

Is this a Simulcast System?	
Data Signiling Rate (bps)	9600
Size of Time Slot (ms)	30

Number of Calls/Replication	15000
Recommended SimTime	
Simulation Time (s)	
Number of Replications	15
Disable Internal Delays	Y/N

Figure 16. Operator Input Sheet Prototype

Single Site Post Processor

Version	1.0 BETA		Simulation Time	27030	
Generation Date	3/3/96 12:30		Number of Replications	2	
MBP/Requisition Number	CPC				
Site Name	v&d				
Scenario Name	wc5				
Execution Time	3.43	minutes			
Number of Working Channels		5			
Number of EDG Ports		32			
Number of Jessica Lines		4			
BER		0.75%			

VOICE CALL STATISTICS

	Overall System Results				
	Mean	Standard Dev	Confidence Inteveral		
Voice Grade Of Service	6.55%	0.22%	0.31%		
Voice Arbitrary Call Delay	2.56%	0.25%	0.35%		
	Duration of Various Voice Call Types (Includes queue times)				
Type	Minimum	Mean	Maximum	Standard Dev	95% Confidence
Group Calls	0.51	5.04	44.88	0.02	0.02
Individual Calls	0.54	5.31	40.35	0.06	0.09
Interconnect Calls	0.00	0.00	0.00	0.00	0.00

Control Channel	Mean
CC Collisions/Replication	955.50
Number of CC Assignments/Rep	14917.50
Number of Calls that failed to access the CC	0.00

Calls Entering System in Busy Hour		
	Calls	Calls/Working Channel
Avg. Inbound Data Calls/Hr	511	102
Avg. Outbound Data Calls/Hr	97	19
Avg. Voice Calls/Hr	1378	276
TOTAL CALLS/HR ON SITE	1986	397

Figure 17. Operator Output Sheet Prototype, Section 1

INBOUND DATA CALL STATISTICS

	Value	Standard Dev	95% Confidence
% of Inbound Calls successful on the 1st application att.	93.70%	0.13%	0.18%

Type	Message Size in Bytes	Avg Percent Success
1	16	100.0%
2	70	100.0%
3	160	100.0%
4	0	0.0%

AVERAGES

Type	Inbound Data Calls Times in System	
	Mean	Max
1	1.03	1.76
2	1.51	3.05
3	2.05	3.90
4	0.00	0.00

OUTBOUND DATA CALL STATISTICS

	Value	Standard Dev	95% Confidence
% of Outbound Calls successful on the 1st application att.	99.66%	0.10%	0.15%

Type	Message Size in Bytes	Avg Percent Success
1	501	100.0%
2	0	0.0%
3	0	0.0%
4	0	0.0%

AVERAGES

Type	**Outbound Data Calls Times in System**	
	Mean	Max
1	3.22	9.90
2	0.00	0.00
3	0.00	0.00
4	0.00	0.00

Figure 18. Operator Output Sheet Prototype, Section 2

5.3 BER Equations

As previously described in Section 4.4, a BER/Data Transmission Success Rate Calculator was developed. The purpose of this calculator is to predict the probability of a data message being successfully transmitted over-the-air in a high bit error rate environment. The equations which make up this calculator take into account several important parameters. These parameters include packet size, data message size, data burst size, the number of packets repeated in a data burst, and the bit error rate of the environment. The following explanation clarifies these parameters and how they relate to one another.

A data message is sent from a host computer to the data gateway for transmission over the air. The gateway takes the message and divides the data into smaller packets. A smaller packet has a greater probability of success than the larger data message. Each packet is appended with a cyclic redundancy check (CRC) in order to detect bit errors at the receiving end. The data gateway always transmits a certain number of bytes in a burst of data, regardless of the actual size of the data message. If the data message is larger than the data burst size, more data bursts are required. If the data message is smaller than the burst size, individual data packets are repeated within the same data burst in order to fill the required number of bytes in the burst. Repeating packets in a single data burst increases the probability of success for those packets.

Members of the design team made significant contributions to the development of the equations which describe the above process mathematically. The equations which make up the BER/Data Transmission Success Rate Calculator are shown below.

Let

$N_{bytes/burst}$	Total number of bytes sent in entire burst
N_{mess}	Total message size in bytes ($N_{mess} <= N_{bytes/burst}$)
$N_{bytes/packet}$	Number of bytes per packet
$N_{packets/burst}$	Number of packets per burst
$N_{unique(i)}$	Number of unique packets sent on the *ith* attempt
BER	Bit Error Rate

- Total number of packets sent in a data burst,

  $N_{packets/burst} = N_{bytes/burst} / N_{bytes/packet}$

- The number of unique packets to be sent on the first data burst,

  $N_{unique(1)} = \text{round-up}(N_{mess} / N_{bytes/packet})$

- Probability of packet rejection in *ith* burst,

  $ProbReject_i = [\ 1 - (1 - BER) \wedge (8 * (N_{bytes/packet} + 2)]\ \wedge\ (\ N_{packets/burst} / N_{unique(i)})$

  Accounts for repeated packets in a data burst

  Accounts for 2 byte CRC added to each packet

- Probability of all packets being accepted in *ith* message,

  $ProbAccept_i = (1 - ProbReject_i) \wedge Nunique_i$

- Expected number of packets to be retransmitted on the next retry,

  $Nunique_{i+1} = ProbReject_i * Nunique_i$

An independent verification of the accuracy of these equations was subsequently performed. The test results and validation effort are discussed in Section 7.2.1.

5.4 Section Summary

Major and key points were discussed in this section on the design and development phase. Many other activities occurred during this process. The major focus was the definition and development of the computer software. Periodic design reviews, both

informal and formal, occurred during this phase. As development progressed, test and evaluation efforts began to assume a major role in the overall process. Section 6.0, The Test and Evaluation Plan, describes these activities in detail.

6.0 Test and Evaluation Master Plan

The purpose of the Test and Evaluation Master Plan (TEMP) is to verify that the simulation model meets its intended objectives, without fault. The simulation model must be proven to produce accurate, repeatable results before it can be used with confidence. The TEMP is a systems engineering tool which is used to structure the testing in order to achieve an accurate and fault-free model. The plan also tests to see if the model is meeting its operational, maintenance, and logistical support requirements.

A TEMP is typically divided into four types of tests, type I through type IV [2]. Type I testing is used during the early stages of the detailed design and may involve various methods and models to verify performance and physical design parameters. Type II testing is more formal in nature and usually occurs during the latter part of the detailed design phase. Type II testing may include a combination of the following test types: performance, environmental, structured, maintainability, support equipment compatibility, usability, technical data verification, and software verification. Type III testing is typically performed by user personnel at a field test site. The objective of this type of testing is to verify the operation, support, and maintenance of the system through the use of the system's written procedures. Type IV testing is done during the system's product-use phase to gain information and insight into the system's actual performance characteristics in any area of concern, such as operational performance, supportability, and maintainability.

The Test and Evaluation Master Plan is customized for the traffic simulator to include the verification and validation of the simulation model. Verification and validation testing includes aspects from the type I and type II testing, and is the major focus of this test effort. A simulation model must be correctly implemented and must be a true representation of the real system. These two objectives are proven to have been met

through verification and validation, respectively. After development is complete, two other major steps remain. First, decide how many of the initial observations must be discarded to remove transients from the results. Second, determine when to stop the simulation. These two issues are called transient removal and stopping criterion [1]. Type III testing will be performed at a selected user test site. Type IV testing is expected to take place later in the product-use phase, and will be used to improve the performance, usability and maintainability aspects of the system.

A comparative analysis phase is planned. The purpose of the comparative analysis phase is two-fold. First, different traffic profiles will be used to generate the model's statistical results for benchmarking purposes. These benchmark cases will be used to ensure that future upgrades and modifications to the model do not change its results significantly. The model must be able to regenerate a benchmark's results after a code change to retain its validated status. Second, a comparative analysis will be carried out to compare the analytical solutions to the simulation model's output results. One assumes that the two methodologies will give different answers, but a comparative analysis will show the difference between the results. The Erlang C spreadsheet computes the GOS almost instantaneously for every working channel between 1 and 20. The simulation model is expected to take a maximum of 15 minutes per run per working channel. The speed of the Erlang C spreadsheet gives it the edge when it comes to design efficiency. By comparing the results of the two tools, one can determine if the results differ enough to favor one method over the other. The model is assumed to give a more accurate assessment of the communication system's performance, but this assumption will remain unconfirmed until the model is validated.

6.1 Verification Testing

The purpose of verification testing is to prove that the model has been correctly implemented. The testing is performed in the hopes that program bugs and logic errors

will be detected and corrected. The following subsections describe the tests and methods which will be used to properly verify this traffic simulation model.

6.1.1 Top-Down Modular Design

A simulation model which has been developed using a top-down modular approach will be much easier to implement and debug [1]. The model will be of higher reliability and will also be easier to maintain. Modular construction allows each module to be tested individually, with its own predefined data interfaces. This method is also known as unit testing [4]. As individual modules are tested and verified, they are combined into larger segments and tested as components. This process continues until the model has completed development. A model, or any software project, coded with this process in mind will better lend itself to any software testing methodology. These testing methodologies include items such as flowgraphs, path testing, transaction flow testing, data flow testing, domain testing, and logic testing [4]. The person responsible for the code development is responsible for this level of testing.

This verification step, which will be carried out by a design review team, will be done by inspecting the model and its design documentation to ensure that it was designed and coded in a top down modular approach. If the code appears unstructured, this will serve as a warning and should introduce questions as to the model's reliability.

6.1.2 Antibugging

Antibugging is a technique employed to find program errors [1]. This technique employs constructs including test variables, such as loop and entity counters, to check the model's output. Some level of antibugging should be verified to have been completed by the code developer during the design review. Other antibugging checks can be carried out by examining the model's output and determining if the results are logical and intuitive. These antibugging system checks should include tests such as the ones listed below.

1. The total number of calls sent into the model should equal the total number of calls that exited the system.
2. The model should process call entities at a rate equal to the call per hour arrival rate defined in the operator's input sheet. This assumption will be valid for lightly loaded systems, such as those with less than a 10% GOS.
3. A data call retry loop counter should never exceed four attempts at the data gateway.
4. A data entity timer should never exceed seven seconds inbound, and thirteen seconds outbound.
5. On a lightly loaded system, with the bit error rate set to 0%, every data call should be successful on its first attempt.
6. Outbound data calls should have a higher success rate than inbound calls, due to the outbound data calls' retry mechanism, and the lack thereof for inbound calls.
7. The number of control channel collisions should never exceed the number of attempts. The number of collisions should increase as the system offered load increases.
8. The measured service times of voice calls should closely approximate the call durations input at the operator's spreadsheet.

6.1.3 Structured Walk-Through

One or more structured walk-throughs will be performed. A code walk-through allows the code developer to show and explain the model's code to the design team. The act of verbalizing and stepping through the code segments will aid both the developer and the team to identify concerns, possible bugs, and coding implementation errors. This process will increase the confidence level that the model is correctly implemented [1].

6.1.4 Execution of Simplified Cases

A major portion of the verification process will involve having the model execute simplified cases. The first step is to verify that the model's voice path produces comparable results to the Erlang C spreadsheet. Next, the data path will be compared to the Erlang B spreadsheet. In both cases, all system design parameters will be nullified in an attempt to have the model behave according to the Erlang assumptions. Once each path is independently verified, a higher degree of confidence will exist for the correctness of the combined voice and data paths.

In the voice path, all delay parameters will be disabled. The voice path will then have calls arriving with an exponential distribution, queuing if no servers are available, calls being serviced with exponentially distributed service times, and no outside influences such as system delays. A comparative analysis will be performed to show the differences between the Erlang C spreadsheet and the model. The model's voice path will be deemed as verified if no significant statistical differences between the two results exist. No significant statistical differences exist if the Erlang GOS curve falls within the 95% confidence intervals of the model's GOS curve. The next step in the voice verification process is to enable all the internal system delays, and again compare the model's results and the Erlang C results. The differences between the two results should indicate that a significant difference exists. However, this difference should be expected and explainable, due to the increase in the effective service times because of the added system delays.

Once the voice path is verified, the data path will be exercised in a similar manner. First, all system design delay parameters will be removed from the data path. The data path will be exercised and its results will be compared to the Erlang B analytical solution. In both cases, calls arrive into the system with an exponential distribution and are dropped if the servers are busy. However, a difference still exists which may result in a significant

difference between the two models. The Erlang B *M/M/m/m* analytical solution treats call service times with an exponential distribution. The model treats data call durations in a more deterministic fashion. With all of the system design parameters disabled and the bit error rate set to 0 percent, the call service time is simply a function of the message size and the data rate. The expected outcome should result in significant, but explainable differences in the results. If differences do exist, a subsequent test will be executed where the model's code that calculates the deterministic service time will be replaced with an exponential distribution code block. This test should confirm that the two methodologies are arriving at essentially the same solution. The next step will involve enabling all the system delays in the data path and performing a second comparative analysis. Again, the differences between the two methods should be expected and explainable. Any counter-intuitive results which are obtained will be subject to additional scrutiny to determine if hidden bugs or logic implementation errors exist.

At the conclusion of these tests, one can assume that both the voice path and the data path can be exercised concurrently and the results should be considered verified based on the test results. The system design parameters that were enabled in the latter part of each individual path test are considered valid based on the expert judgment of our team's communication system specialists.

6.1.5 Continuity Tests

The model will be executed with slightly different input parameters and the output will be examined for continuity. For example, the number of working channels will be varied for a given input traffic load. The output should indicate that the probability of queuing and blocking decreases as the number of channels increase. Any discontinuities in a graph of these parameters will indicate an implementation or modeling error.

6.2 Transient Removal and Stopping Criteria

During the testing phase, a secondary objective exists. This objective includes removal of startup transients from the results and determining when to stop the simulation. A simulation model will produce a result which is based on statistical averages. A model is typically run for a large set of calls and these calls produce a result which is a statistical representation of the set. The more calls that are allowed to propagate through the model, the closer the model's results will approach a steady state value. Since the results of interest are the steady state results, they should not include the startup transient state. A model with a certain number of calls to process may be executed multiple times, and the results of each of the independent replications may be statistically combined to produce a result approaching a steady state value. From this information, a simple conclusion can be drawn. The longer a model is run, the more accurate its results will become. Therefore, an optimization problem exists between the required accuracy and the length of time the model is executed. As stated in the system's operational requirements, the model shall produce results for a 10% GOS with a 95% confidence interval of 1.0%. Also, a result of 1% GOS shall be accurate with a 95% confidence interval of 0.5%. Test runs will be executed with varying numbers of calls per replication and with varying numbers of replications. Each unique run, with these varying execution parameters, will produce a mean GOS, with a standard deviation, a 95% confidence interval, and an execution time. The data from these runs will be assimilated to determine the optimum number of calls per replication and the optimum number of replications. Widely differing traffic profiles and system offered loads will produce widely differing results for the number of calls per replication and the number of replications. Therefore, an average customer profile will be utilized during this analysis.

6.3 *Validation Testing*

Validation occurs when the assumptions which were used in developing the model are correctly implemented and result in a model which produces results similar to those of the real system. Three key aspects to model validation need to be addressed [1].

1. Assumptions
2. Input parameter values and distributions
3. Output values and conclusions

Validity testing is performed to compare the aforementioned three key aspects to any combination of the following sources.

1. Expert intuition
2. Real system measurement
3. Theoretical results

Not all forms of validation are possible. At the present time, only a limited real system analysis could be performed and compared to the model's results. Even if all forms of validation were available to the analyst, 100% validation, as well as 100% verification is impossible. However, this test plan will attempt to provide a reasonable validation effort to give a high degree of confidence that the model has been correctly designed and implemented.

6.3.1 Expert Intuition

Expert intuition will be used to a great extent to validate the model. The design team which has been assembled is a knowledgeable group of individuals. System expertise is available from the group which covers areas relevant to the real communication system. These areas of expertise include hardware configuration, software configuration, operating characteristics, the ARQ protocol, RF transmission theory, and data communications theory. The design team will meet on a weekly basis through the design

and development phase to provide input to relevant design issues and to review the current status of the model. The team is expected to participate in the design process throughout the entire design life cycle. Therefore, they will have input into the validity of the assumptions, input parameters and distributions, and output values and conclusions. The results from verification testing will also be subject to the scrutiny of these experts. All results will be presented and validated with this panel of experts. Any results which are not readily explainable will be investigated further and may result in additional tests being performed.

6.3.2 Real System Measurement

Performance comparisons between an actual system and the model will produce the most reliable validity test. However, it is also the most difficult and time consuming method. A voice-only system analysis was performed several years ago on a real system and the results were validated against the Erlang C spreadsheet. This same real system's input and output results will again be used as the standard, but in this instance, the model will be validated against these results.

Two real system measurements will be performed which focus on data calls. The service time of a call is considered critical to the system's performance. Therefore, it is imperative that the service times used by the model closely approximate the actual system. The bit error rate and data transmission times will be validated against laboratory data.

6.4 Usability Analysis

As mentioned earlier, an operator interface (OI) layer was written to provide ease of use when loading the model with input data and examining its results. This OI must be user friendly and intuitive. Once the OI is near completion, it will be presented to the design team for review. The team will check for clarity of purpose, logical layout and flow, and

standard toolbar menu structuring. Verification that all required inputs and outputs are accessible to the operator will also be performed. These inputs and outputs are listed in Tables 6 and 10 in Section 4.4.

A second level of usability will be performed as part of the type III testing. The completed system will be provided for use at a field office. The users will be presented the system documentation, including the user's manual, and support and maintenance documents. The users will be given a one-hour training session on the use and operation of the model. They will be given a traffic profile and asked to perform an analysis using the model. The users will be observed during the operation of the model. Notes will be taken during this process, and problem areas will be identified. After completion of this test, the users will be interviewed and asked to comment on the model's usability. This survey will be used to identify areas for improvement in the operation of the model.

6.5 Comparative Analysis Plan

The purpose of the comparative analysis is to determine the validity of the Erlang C spreadsheet. The spreadsheet is a fast and efficient design tool for predicting the required number of working channels for a given GOS design goal. Three possible outcomes exist for this analysis. First, significant differences will exist between the model and the Erlang C spreadsheet, thereby leading to the conclusion that the model be used for any and all traffic profiles. Second, no significant differences will exist between the two tools' results, thereby suggesting that the Erlang C spreadsheet be the design tool of choice based on its speed and efficiency. Finally, the model and the Erlang C spreadsheet will show no significant statistical differences in some cases, and be significantly different for other cases. This analysis plan will attempt to characterize the conditions under which the Erlang C spreadsheet can be used, and when it becomes necessary to utilize the simulation model to obtain accurate results.

Two case studies will be examined. The cases will both contain voice and data loads, but will differ from light to heavy system offered loads. Each comparative analysis performed will be represented graphically, depicting the Grade of Service versus the number of working channels. The message profile used in this analysis is shown in Table 11. Table 12 shows the number of users and the calculated values of the system utilization for each test case.

Table 11. Combined Voice and Data Message Profile

Call Description	**Msg Size**	**Calls/hr/user**	**Duration (secs)**
VOICE			
Group Call	-	3.0	4.0
Individual Call	-	0.2	2.8
OUTBOUND DATA			
Type I	34	14.1	-
Type II	58	1	-
Type III	78	13.1	-
Type IV	511	26.2	-
INBOUND DATA			
Type I	34	40.3	-
Type II	58	1.0	-
Type III	74	13.1	-

Table 12. System Utilization

			System Utilization (erlangs)		
			Voice Load	Data Load	
	Voice Users	Data Users		0% BER	0.75% BER
Case #1	107	27	0.37	0.50	1.41
Case #2	107	100	0.37	1.84	5.22

7.0 Analysis Package

The results of the Test and Evaluation Master Plan are presented in this section. Each individual test is briefly introduced, the test results shown, and the results discussed. Table 13 below summarizes the individual tests of the TEMP and indicates which tests have been completed and included in this analysis package.

Table 13. Simulation Model Verification and Validation

Test Description	Status
Voice-only comparison	complete
Data-only comparison	complete
Data-only, Erlang B version	complete
Transient and stopping criteria	complete
Comparative analysis, voice and data combined load	complete
Real system measurements	complete

7.1 Model Verification Analysis

The model's verification process involved a wide and diverse range of activities. The individual processes performed included activities such as structured top-down design, software test and debug, structured code walk-throughs, and systems analysis. The results presented in this section focus on the systems analysis aspects of the verification process.

7.1.1 Voice-Only Comparison

In this test, the model's voice path is compared to the Erlang C results. Two comparisons will be shown. First, the model will be executed without any system design delay parameters enabled. This first test will allow the model to behave in a fashion similar to the Erlang C equations. This test is expected to show little difference between the two method's results. The second test will compare the model, with its internal delay

parameters enabled, to the Erlang C analytical solution. A significant difference between the two methods is expected. The voice-only traffic profile is given in Table 14.

Table 14. Voice Profile

Voice-Only Load - 906 Users		
	Calls per User	Mean Call Duration
Group Calls	3.0	4.60
I-Calls	0.2	4.90

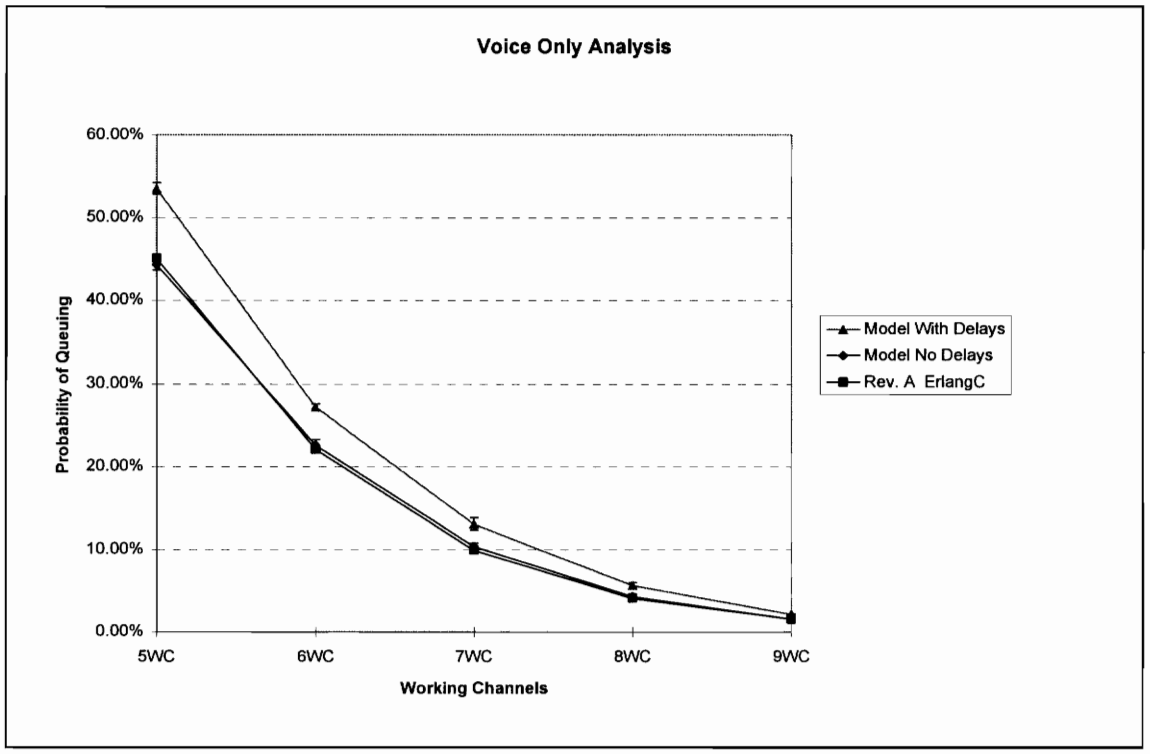


Figure 19. Voice-Only Comparative Analysis

Results

The curves representing the model's results are shown in Figure 14 with vertical error bars. The bars represent the 95% confidence interval of each data point. From the graph shown above, one can conclude that no significant differences exist between the model's results without internal delays and the Erlang C results. This is evident from the fact that the Erlang C data points fall within the 95% confidence interval of the model's data values. However, significant differences do exist when the model's internal design parameters are enabled, but these results are expected. By enabling the system design parameters, delays are introduced to the call processing path. These delays have the net effect of decreasing the mean service rate, μ. This decrease in the service rate increases the utilization factor, which has the net effect of increasing the probability of queuing. From these test results, the voice path is considered verified.

7.1.2 Data-Only Comparison

The data-only comparison tests are meant to verify the data path in the model. The objective of this test is to determine if the results taken from the inbound data path approach the Erlang B analytical solution. The inbound data path is a call blocking path. The outbound data path will retry a call up to three times if a channel is unavailable and, therefore, is excluded from this comparative test. However, the outbound data path was subsequently modified to block calls by zeroing its 'maximum number of retries' loop variable. The results taken from the outbound data path matched the inbound data path results.

The first test had all the system design parameters disabled. This enabled the model to closely approximate the Erlang B analytical solution. Next, the system design parameters were enabled in the model for this test series. Two additional tests were performed. These tests were needed in order to isolate and identify variables which caused the results

between the model without the system delays and the Erlang B solution to disagree. These tests involved replacing the logic responsible for the deterministic service time with a code block which would treat the service times with an exponential distribution, as does the Erlang B *M/M/m/m* solution. The results are shown in Figure 20 and a discussion of these results follows. The message profile used is shown in Table 15. The total system offered data load was created by five data users.

Table 15. Data-Only Message Profile

Message Profile		
Inbound Data-Only	**Message Size**	**Calls/User/Hour**
Type I	15	400
Type II	100	83
Type III	155	250
Type IV	511	300

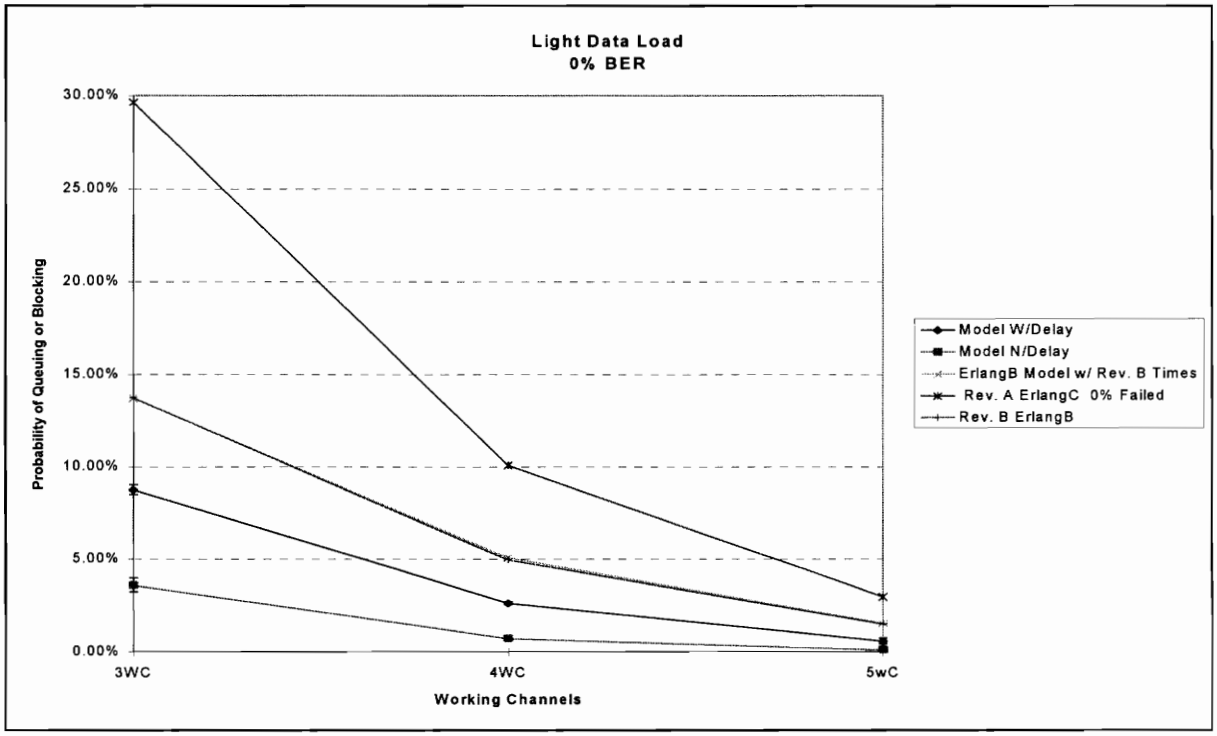


Figure 20. Data-Only Comparative Analysis

Results

The probability of blocking decreases when the system design parameters are disabled in the model. This result is seen by viewing the curve of the model with delays (Model W/Delay) and the curve of the model without delays (Model N/Delay). The decrease in the probability of blocking is the expected result. However, the Erlang B analytical solution seen in the graph has a much higher probability of blocking than either of the two model curves. This significant difference was attributed to the fact that the Erlang B solution treats the call service times with an exponential distribution, whereas the model treats the service times in a deterministic fashion. The model's service times are a function of the message length, the over-the-air baud rate, and the bit error rate of the environment. To validate this assumption, the model was reconfigured to process the call service times with an exponential distribution. The mean call duration was set to the same value used by the Erlang B analytical solution. The resulting curve (Erlang B Model w/ Rev. B Times) is overlaid onto the Erlang B curve (Rev. B Erlang B), showing no significant differences between the two methods. The two curves match so closely that it is difficult to see the two data series as separate curves in the graph.

This test concluded that the decrease in blocking probability of the model's results, as opposed to the Erlang B solution, is due to the manner in which the model treats the service times. The service times are computed with the Data Transmission Rate Calculator. This calculator is verified, validated and discussed in Section 7.2.1. Therefore, the inbound data path is considered verified. The outbound data path is also considered verified. The outbound path results matched the inbound path results with its maximum number of retries set to zero. Also shown in Figure 20 is a curve depicting the results taken from an Erlang C analytical solution (Rev. A ErlangC 0% Failed). This curve is seen to have a higher probability of queuing than all of the other data curves.

7.1.3 Transient Removal and Stopping Criteria

Testing was performed to determine the transient removal and stopping criteria. During the voice-only and data-only analysis, an observation was made indicating that voice calls took longer to approach steady state than did data calls. A test was devised with a voice-only profile to determine the minimum values for the execution parameters which allowed the model to meet the operational requirements. These execution parameters are the number of calls per replication and the number of replications. The requirement stated that the model shall produce results for a 10% GOS with accuracy within 1% with a 95% confidence interval.

A voice system offered load of 11.77% GOS with 7 working channels was selected. This load was input into the model and batch executions were performed with varying calls per replication and varying replications. The results from this analysis are shown in Figure 21. Each GOS data point is plotted with its 95% confidence interval. The model was executed with its system design delays enabled. The resultant GOS is biased with a higher probability of queuing than the Erlang C solution of 11.77%.

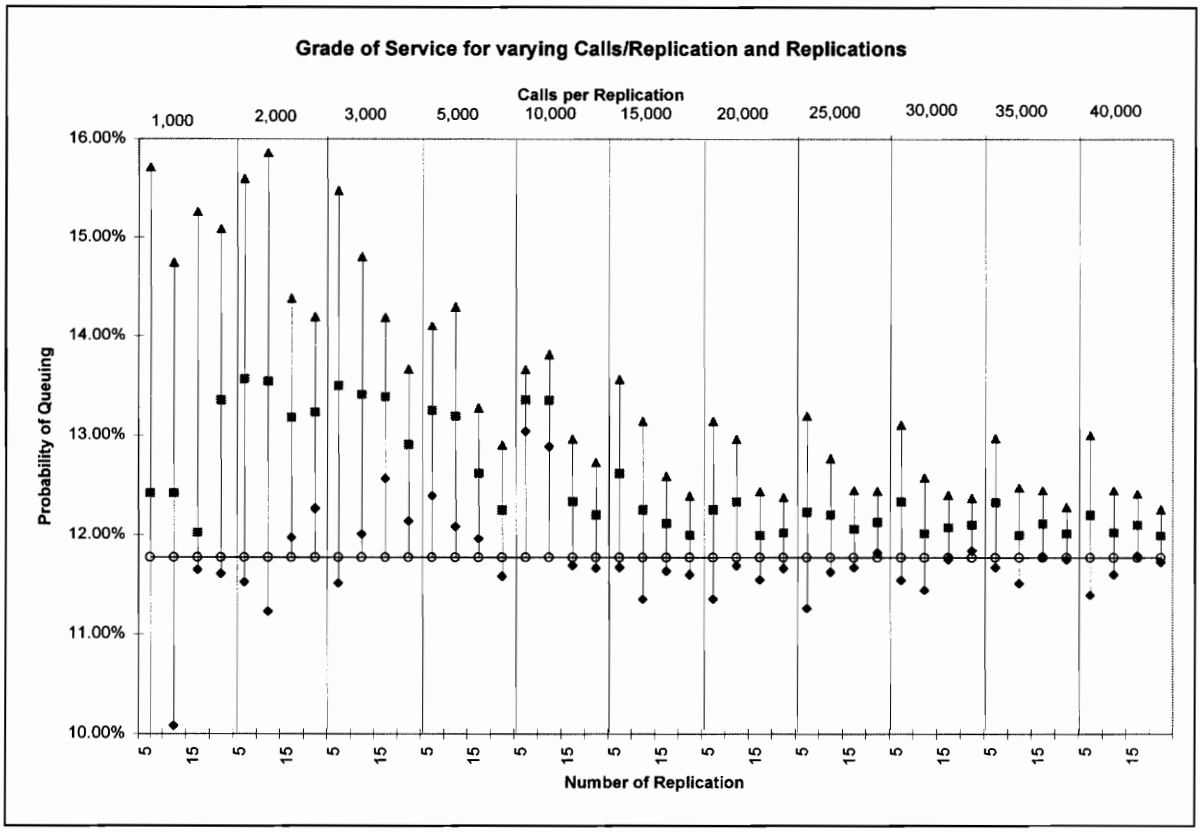


Figure 21. Model GOS for varying Calls/Replication and Replications

The data in the above graph was reproduced in a 3-dimensional graph for better viewing. The y-axis values are the standard deviations about the grade of service mean values. The 3-dimensional graph is shown in Figure 22.

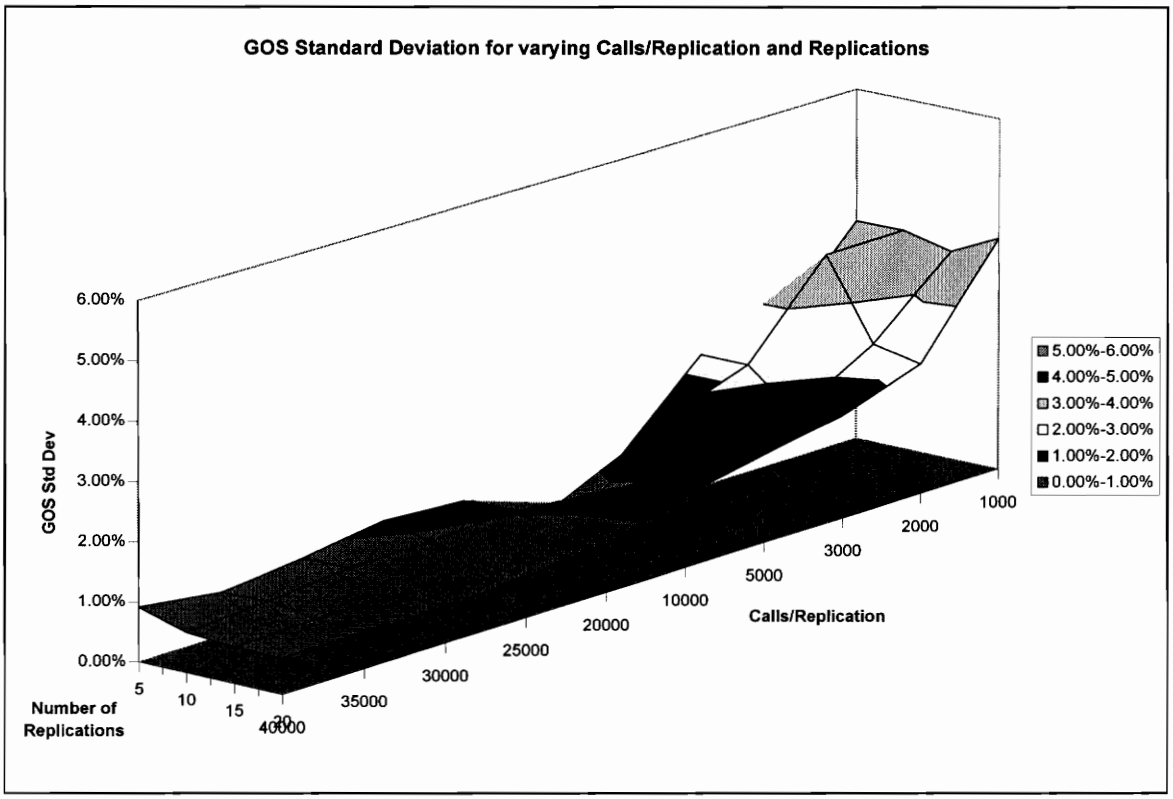


Figure 22. GOS Standard Deviations for varying Calls/Replications and Replications

Results

This traffic profile meets the minimum requirement criteria at 15,000 calls per replication and 15 replications. The execution time of the model was 4.52 minutes. This time meets the 15-minute maximum execution time requirement. Therefore, the model will be configured with the aforementioned execution parameters as default values.

7.2 Model Validation

The efforts to validate the model fall into two broad categories, expert intuition and real system measurements. The analysis presented in this section focuses on the real system measurements which were performed in an attempt to validate the model. Two critical

assumptions are addressed in these tests. These assumptions are discussed in the following two sections and are considered critical because they are used to estimate the service rate parameter, μ. The service rate parameter is critical in determining overall system performance. A sensitivity analysis of the service rate parameter is performed in Section 7.4.1

7.2.1 BER Calculator versus the BER Simulator

The accuracy of the BER/Data Transmission Success Rate Calculator, herein called the BER Calculator, is critical to the accuracy of the overall simulation model. This fact was recognized early in the detail design phase, and provisions were made to compare the results from the BER calculator to a prototype BER simulator. The BER simulator is an executable program which randomly inserts bit errors at a desired rate into a data stream. The BER simulator keeps track of the message sizes transmitted and the average number of transmission attempts per message. The simulator is used in a controlled, lab environment where the possibility of bit errors due to outside influences are negligible. Results and comparison of the BER simulator to the BER calculator are shown in Figure 23.

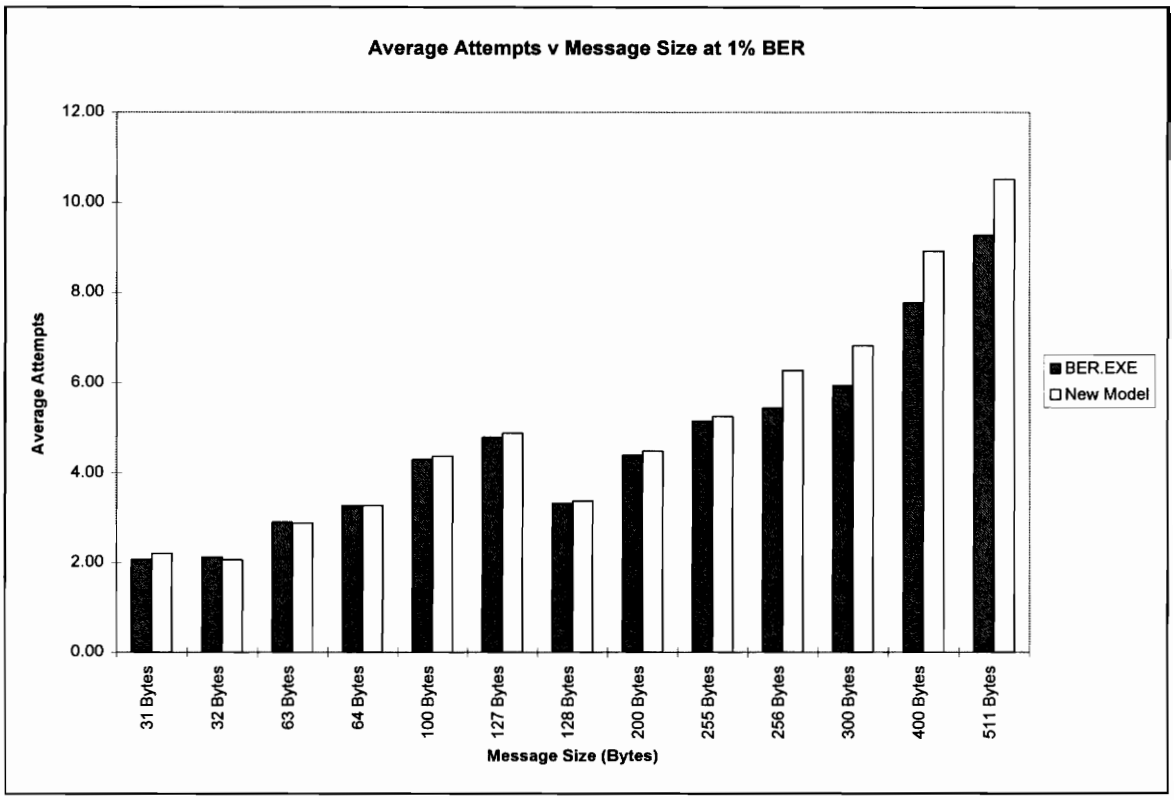


Figure 23. BER Comparative Analysis

Results

From the data shown above, one may conclude that the BER calculator and the BER simulator produce like results for message sizes below 256 bytes. However, for message sizes of 256 bytes and above, a significant difference of up to 8.5% in the results is observed. Message sizes above 255 bytes differ from smaller messages in the fact that the data message is sent over-the-air in two data bursts instead of one. Further investigation showed that the BER simulator executable failed to take this into consideration. Thus, the results show the BER simulator to be overly optimistic in predicting the number of attempts for message sizes requiring two data bursts. From these tests, the BER calculator is considered validated. Figure 23 appears to fail the data

continuity test. However, the packetization scheme changes abruptly at different message sizes and, therefore, discontinuities in this graph are expected.

7.2.2 The Data Transmission Rate Calculator

The data transmission rate calculator estimates the overall transmission time for a given message size. The calculator takes many system design parameters into consideration, such as baud rate, outbound transmission time, return acknowledgment time, and the radio's RX-to-TX switch time. These times, plus the channel access and delay times, represents the amount of time that a channel is removed from the resource pool of available channels. The calculator also computes end-to-end message transmission times. These times include the time a channel is busy, as described above, plus the time required to transmit a data message to and from the radio network to the host-end computers via a wireline interface.

A test was performed in the laboratory where data messages of various lengths were sent over-the-air and the actual transmission times were captured. The model was executed with these same message lengths. The model's output results sheet calculates the statistical mean, minimum, and maximum transmission times of each message size. In this test, the bit error rate was assumed to be 0 percent. Therefore, the resultant times listed below are the minimum values obtained from both methods. The results of these tests are contained in Table 16.

Table 16. Data Transmission Time Comparative Analysis

Data Message	Lab Time (sec)	Model Time (sec)	Percent Difference
Type I	0.97	0.94	3.1%
Type II	1.36	1.34	1.4%
Type III	2.16	2.11	2.4%

Results

The results in Table 16 show a close correlation between the actual transmission times and the statistical times produced by the model. The largest difference between the two time values is 3.1%. These results indicate that the Data Transmission Rate Calculator has been correctly implemented into the model and is considered validated.

7.3 Comparative Analysis and Results

The following analysis involves a case study of two traffic load scenarios. The traffic profiles are identical, with the exception of the number of users. The traffic profile is shown below in Table 17. The first case study represents a moderate voice load combined with a relatively light data load generated by 27 data users. The second case study uses the same voice load and the same message profile as in the first case. However, the number of data users was increased to 100 in order to generate a heavy data load. In each comparative analysis test case, the model's results and the Erlang C spreadsheet results are shown for a 0% BER test case and a 0.75% BER test case. Three model result curves are shown in each graph. A GOS curve is shown for the probability of queuing a voice call and two probability of blocking curves are shown for the inbound and outbound data calls.

Table 17. Combined Voice and Data Comparative Analysis

Call Description	Msg Size (bytes)	Calls/hr/user	Duration (secs)
VOICE - 107 Users			
Group Call	-	3.0	4.0
Individual Call	-	0.2	2.8
OUTBOUND DATA			
Type I	34	14.1	-
Type II	58	1	-
Type III	78	13.1	-
Type IV	511	26.2	-
INBOUND DATA			
Type I	34	40.3	-
Type II	58	1.0	-
Type III	74	13.1	-

7.3.1 Case Study #1

The traffic profile contained in Table 17 was used in this analysis. The number of data users was set to 27, which represents a relatively light data load. The results of the analysis are shown in Figures 24 and 25.

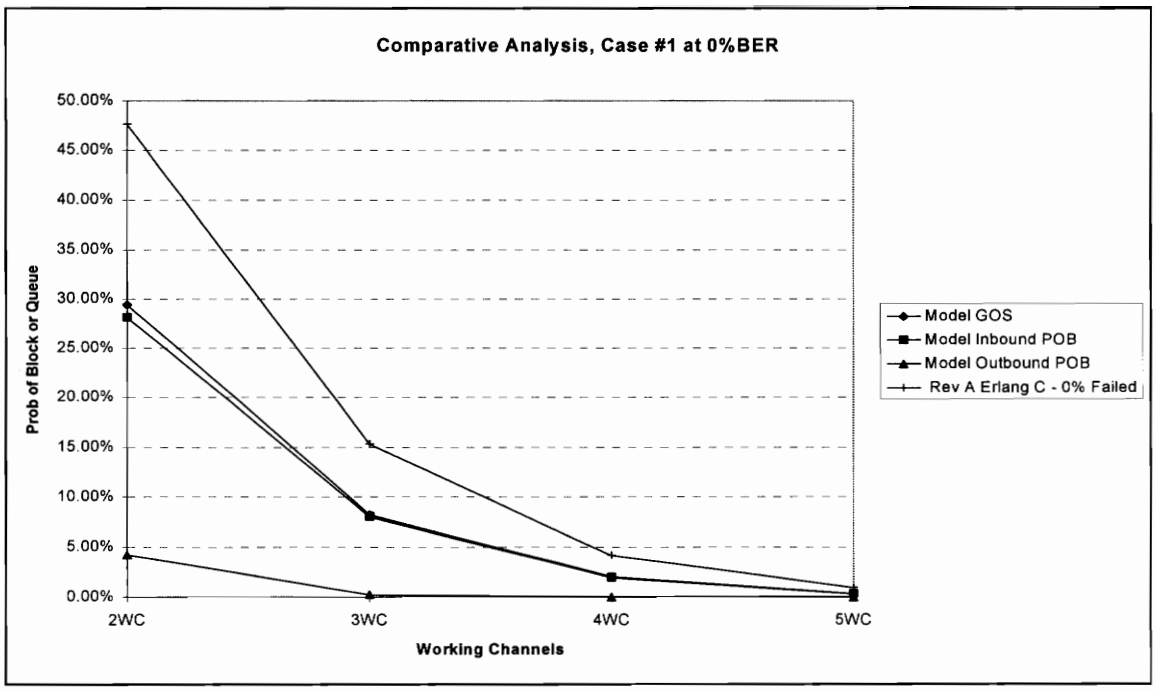


Figure 24. Case Study #1 Comparative Analysis

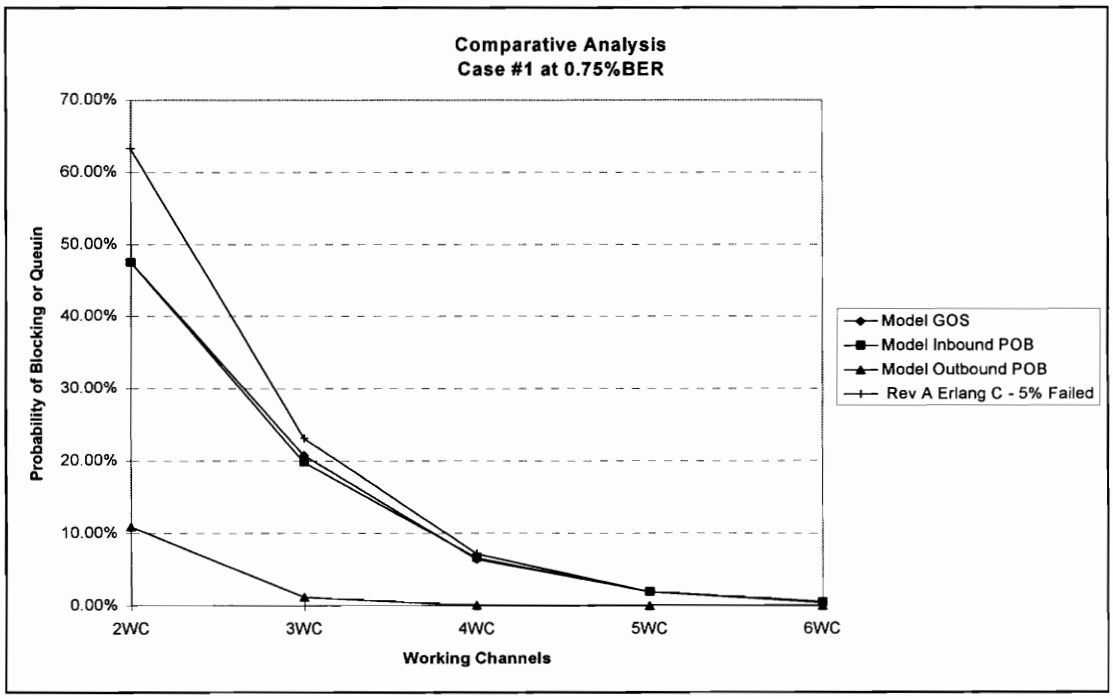


Figure 25. Case Study #1 Comparative Analysis

Results

The 0% BER test case shows the Erlang C solution converging with the model's GOS and inbound probability of blocking curves as the probability approaches 0%. At higher GOS values, the Erlang C curve diverges and exhibits a higher GOS than the model for a given number of working channels. At 0.75% BER, the model and the Erlang results converge at a higher GOS. With a design goal of 15% GOS or less, either solution would provide the same answer in the form of the number of required working channels.

7.3.2 Case Study #2

The traffic profile for case #2 is identical to the profile used in case #1, except the number of data users is increased to 100. This increase in users was done to increase the data load while maintaining the same voice load. The results are graphed in Figures 26 and 27.

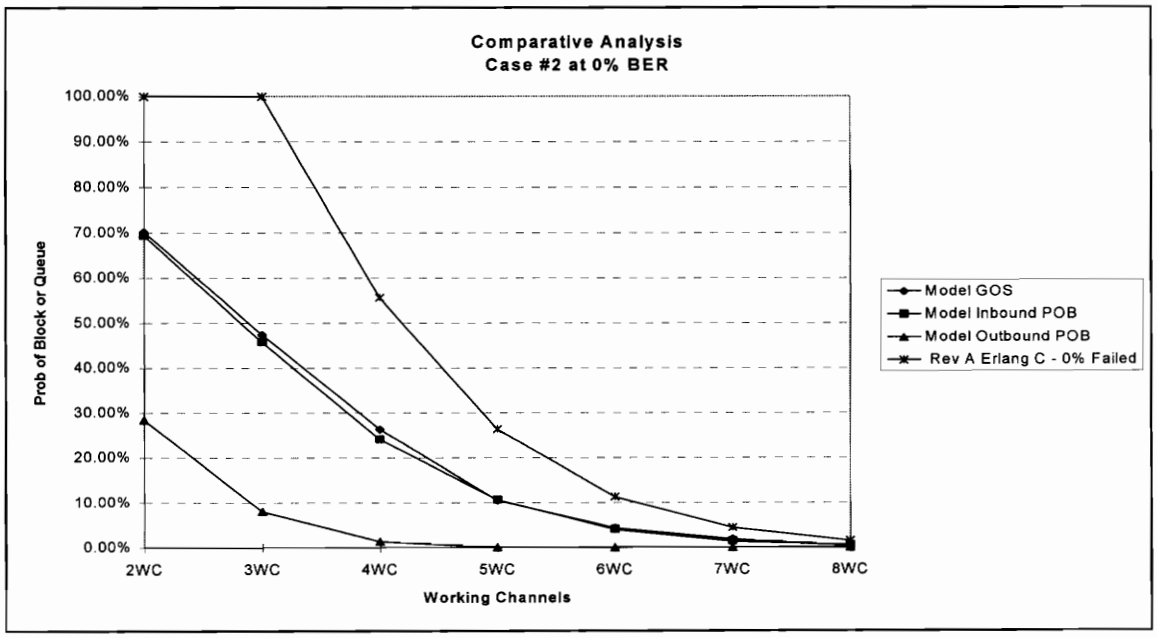


Figure 26. Case Study #2 Comparative Analysis

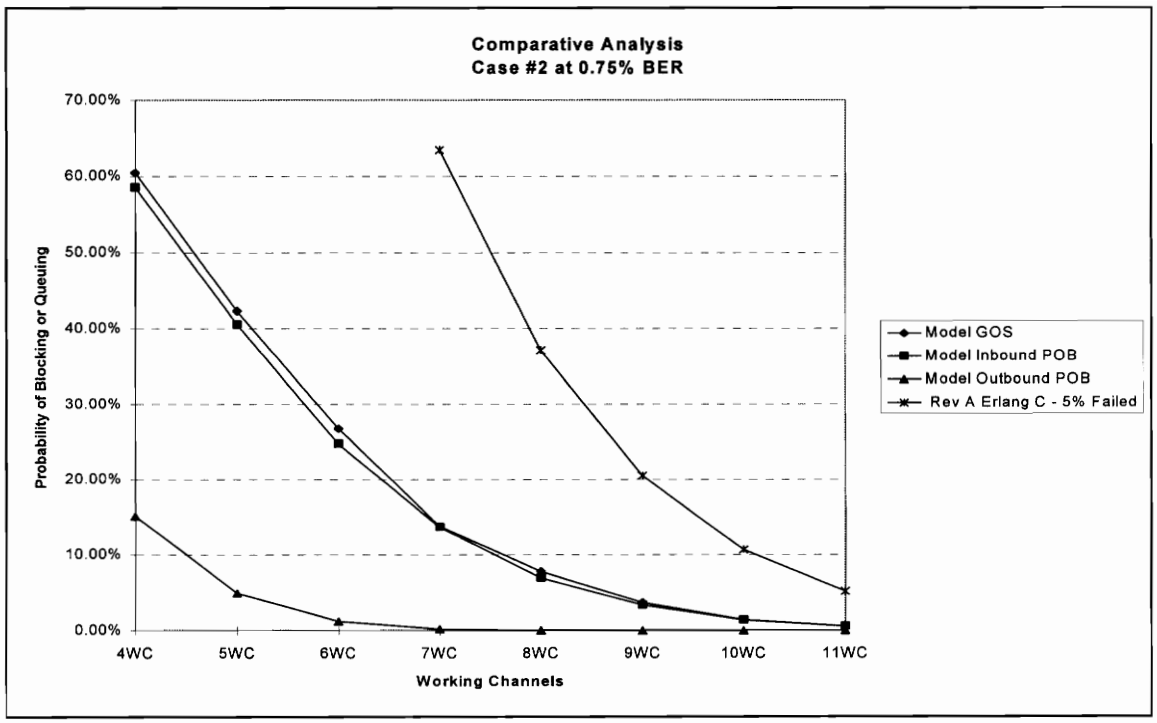


Figure 27. Case Study #2 Comparative Analysis

Results

In both the 0% BER and the 0.75% BER test cases, the Erlang C solution gives a higher probability of queuing than the model predicts. At a 10% GOS, the 0% BER test case shows the Erlang C solution requiring 1 additional working channel than the model requires. At 0.75% BER, the Erlang solution requires 3 additional channels. At 1% GOS, the Erlang C solution and the model's solution agree in case study #1 and for the 0% BER test in case study #2. However, in the 0.75% BER test in case #2, the Erlang C solution projects a much higher probability of blocking than the model predicts.

Figure 28 compares the results taken from both case studies. The curves represent the number of working channels needed, versus the data system offered load, to achieve a 10% GOS design goal. The curves diverge as the data load increases and decreases. From these findings, the author suggests using the model for any traffic profile that

contains data calls when designing for a 10% GOS. Further characterization needs to be performed before verify this assumption.

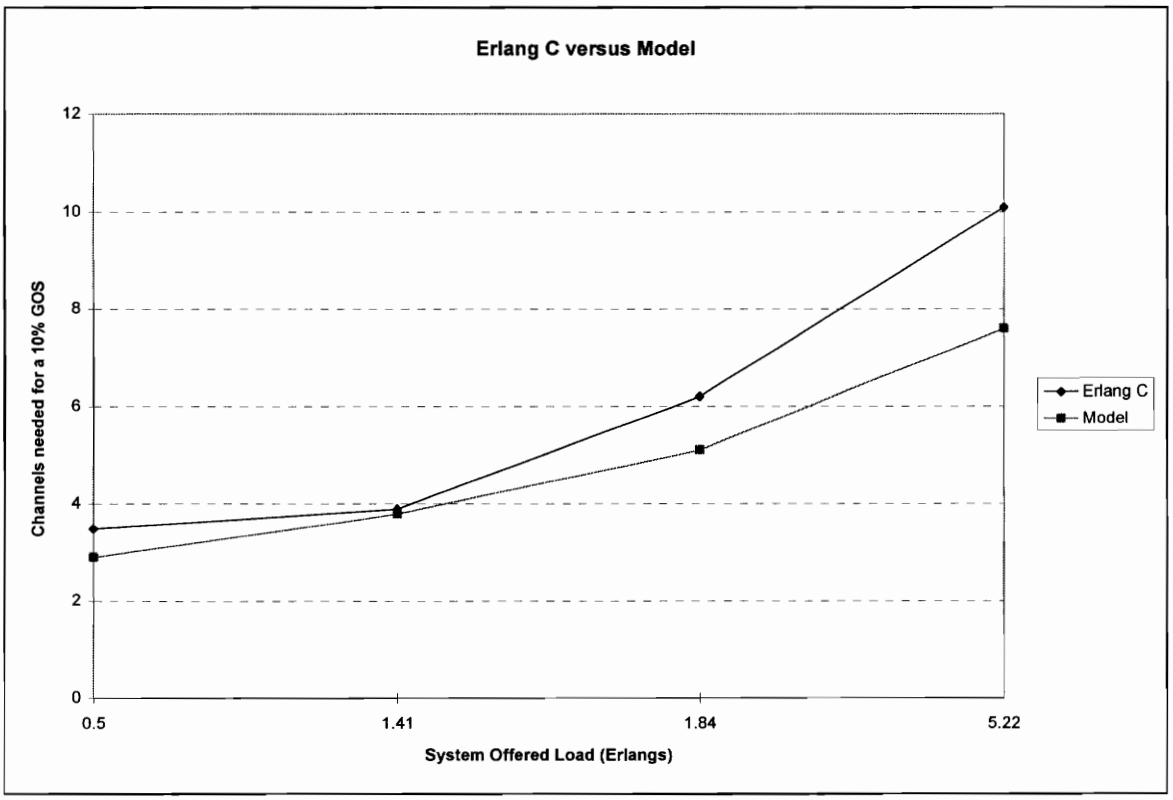


Figure 28. Erlang C and Model Comparison

These results indicate that the Erlang C spreadsheet and the model do not agree in all cases. The results tend to agree in a system dominated by voice communications. In such a scenario, the closer the design GOS is to 0%, the Erlang C solution may be used with confidence. However, as the data load becomes a significant portion of the overall data load, the Erlang C solution can no longer be expected to produce the same results as the model.

7.4 Performance and Sensitivity Analysis

One of the primary uses of simulation models is to aid the design engineer in the selection of the best design alternative. The designer may use a model to vary one or more design-independent parameters to determine the net effect on one or more design-dependent parameters. The author highly recommends this approach in order to optimize the design of the real communication system. However, the primary objective of this model is to perform traffic loading analysis. Therefore, testing designed primarily for performance comparisons of different design alternatives is not included in this report. However, by examining the results taken to verify and validate the model, several avenues for further exploration emerge. Some of the results suggest that modifying certain parameters and transmission techniques may yield performance improvements.

7.4.1 The Effects of the Bit Error Rate

The results of the data-only analysis indicate that the bit error rate is critical to the performance of the system. The results of the heavy data load test scenario with a 0% BER and a 0.75% BER are depicted in Figure 29.

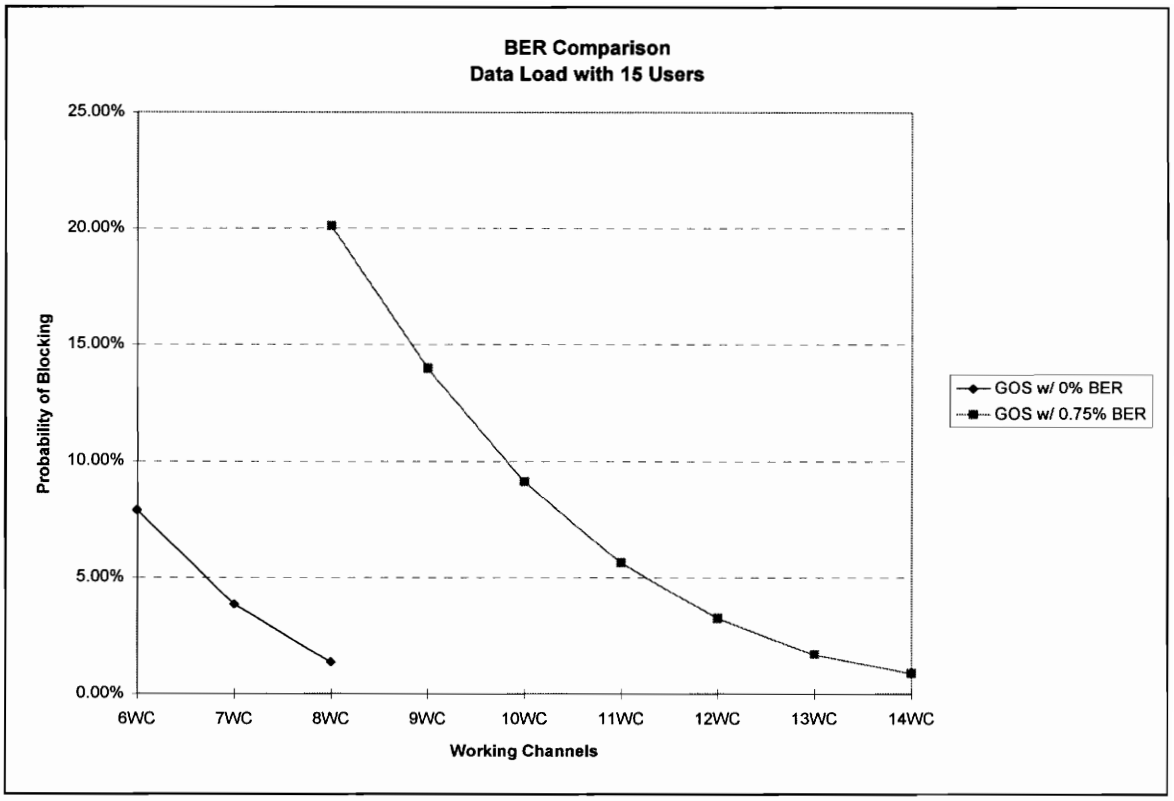


Figure 29. Effects of BER on Working Channels

The GOS curves shown in the above graph indicate a large difference in data transmission performance.

Further investigation led to the development of the following table. Table 18 shows the contributing factors to the utilization factor. Recall that the utilization factor (ρ) is computed as the call arrival rate (λ) times the mean call duration ($1/\mu$).

Table 18. Contributing Factors to the Utilization Factor

	Data Users	5	10	15
	Calls/sec	1.435	2.869	4.304
BER	Mean Call Duration			
0%	0.645	0.926	1.85	2.78
0.75%	2.168	3.11	6.22	9.33

The utilization factor increases proportionally with both parameters. An increase in either the number of data users or the BER causes a corresponding increase in the utilization factor. The number of data users in a given system is considered fixed and cannot be lowered to decrease the utilization factor. However, the call durations associated with the higher BER may be improved upon. The 0.75% BER mean call duration is 236% higher than the 0% BER mean call duration.

Forward error correction may be able to enhance system performance by minimizing the number of data message retries, thereby decreasing the mean call duration. As an example, assume that parity check codes can be added to the data packet transmissions to allow a near zero data message retry scenario. Also assume that adding parity check codes will double the data message length. The model was used to aid in this sensitivity analysis. Each data message in this test message profile was doubled in length. The resultant mean call duration is 1.011 seconds. The forward error correction scheme's mean call duration is 57% higher than the best case, as opposed to the call duration being 236% higher than the best case without any forward error correction. Therefore, forward error correction appears to be a technique which may be able to significantly improve data performance.

7.4.2 The Effects of the Data Message Retries

During the data path testing, the outbound data path consistently performed better than the inbound data path in terms of blocking probability. The reason is due to the fact that an outbound data message will be retried up to three times after failing to seize a working channel before it is discarded. An inbound data message is discarded immediately after finding no channels available. The graph shown in Figure 30 shows an inbound and an outbound probability of blocking curve. The data message profiles and the call arrival rates are identical for both paths. The inbound data load requires three fewer working channels than the outbound data load. This result indicates that performance advantages may be gained by incorporating retries into the inbound data path. The author recommends that further analysis be pursued in this area. A similar concept to incorporating a retry mechanism is data queuing. The model should be modified to utilize data queuing and performance comparisons should be executed. These comparisons could be used to determine if incorporating retries or data queuing leads to the best overall performance improvement.

The improvements made to the blocking probability would need to be made while minimizing any increase to the message throughput times.

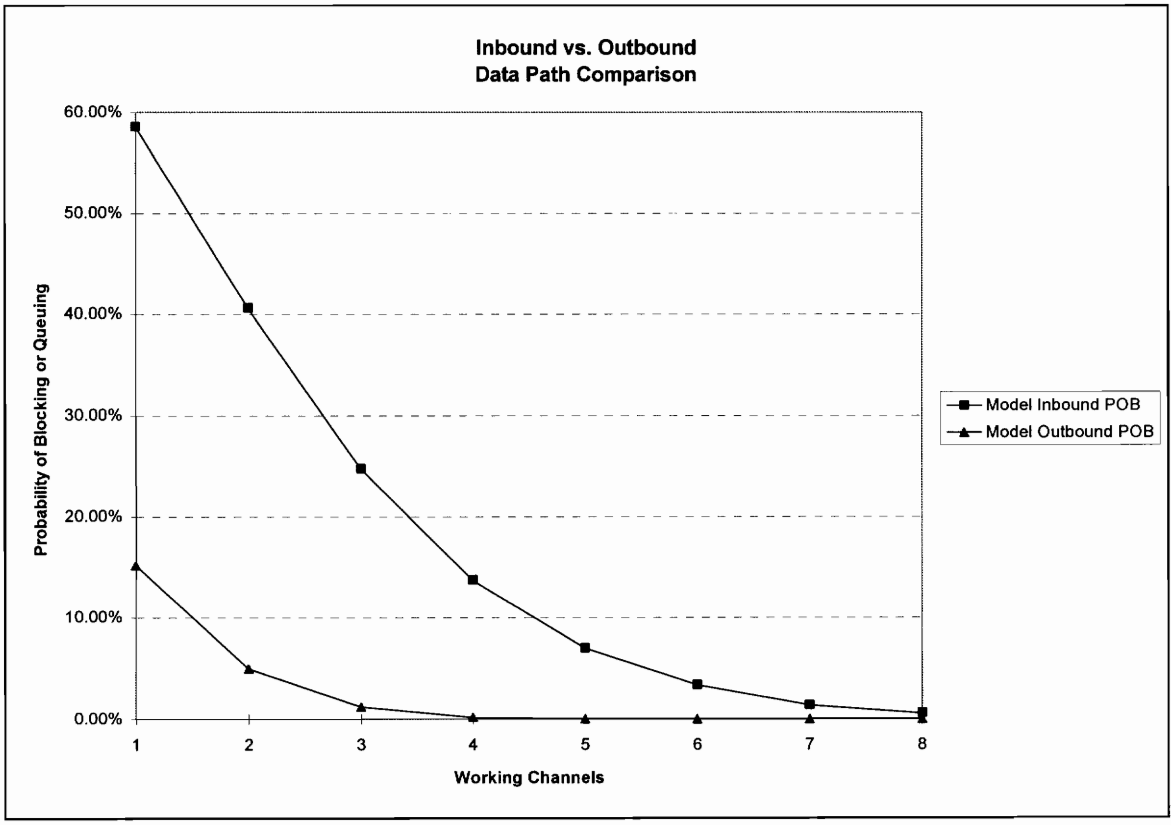


Figure 30. Performance Comparison of Inbound and Outbound Data Paths

7.5 Section Summary

The analysis package presented in this section shows the results of the verification and validation effort, the comparative analysis, and the performance and sensitivity analysis. These tests are discussed in the TEMP in Section 6.0.

8.0 System Utilization and Life Cycle Support

The traffic analysis simulation model will begin its roll out for field use as soon as the user documentation and training materials are completed. The model will be placed into a beta test mode where a controlled number of users will begin to use and evaluate the model. After a 30-day period, modifications to the model will be made based on the users' feedback. After the beta test period, the system will be rolled out to the remaining deployment areas.

Continued system support will be the responsibility of the Wide Area System Design Team. Team members will be responsible for model modifications, support, and training. The team will also be responsible for the design and development of new models to accurately predict the performance of new and improved communication systems. The present model's expected life is five years. The field of telecommunications is changing at a rapid pace and new models representing the new technology will replace the current version.

9.0 Conclusion

This paper focused on the design and development of a simulation model to aid system designers of land-mobile radio communication systems. The model is used to predict the number of working channels required to adequately handle a customer's traffic load. The design, development, test, and evaluation of the modeling system followed a top-down systematic approach. The employment of systems engineering methodologies to structure this process led to the successful implementation of the model. A rigorous verification and validation process has ensured that the model is correctly implemented and that the assumptions upon which the model is based are true and correct.

Preliminary indications from the analysis phase suggest that the model be used for traffic analysis whenever data calls are contained in the traffic profile. The Erlang C spreadsheet should be used to analyze voice-only traffic loads.

The model has provided valuable insight into determining the system parameters which are critical to the performance of the real communication system. The author recommends using the model to further study design alternatives to increase overall performance of the communication system.

10.0 References

1. Jain, Raj, "The Art of Computer Systems Performance Analysis", John Wiley & Sons, Inc., New York, 1991

2. Blanchard, Benjamin S.and Fabrycky, Wolter J., "Systems Engineering and Analysis", Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1990

3. Bertsekas, Dimitri, and Gallager, Robert, "Data Networks", Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1992

4. Beizer, Boris, "Software Testing Techniques: Analysis, Testing, and Verification", Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1990