Pittsburg State University
Pittsburg State University Digital Commons

Problems, 1950-1991 College of Education

5-1-1974

An Approach to the Problem of Computer Simulations in Business

Judith Miller
Kansas State College of Pittsburg

Follow this and additional works at: https://digitalcommons.pittstate.edu/problems

 Part of the Business Commons

Recommended Citation

Miller, Judith, "An Approach to the Problem of Computer Simulations in Business" (1974). *Problems, 1950-1991*. 21.
https://digitalcommons.pittstate.edu/problems/21

This Graduate Research is brought to you for free and open access by the College of Education at Pittsburg State University Digital Commons. It has been accepted for inclusion in Problems, 1950-1991 by an authorized administrator of Pittsburg State University Digital Commons. For more information, please contact digitalcommons@pittstate.edu.

AN APPROACH TO THE PROBLEM OF COMPUTER SIMULATIONS IN BUSINESS

A Problem Submitted to the Graduate Division in Partial Fulfillment of the Requirements for the Degree of Master of Science

By

Judith Miller

LIBRARY
MAY 15 '74

KANSAS STATE COLLEGE OF PITTSBURG
Pittsburg, Kansas
May, 1974

LIBRARY

ACKNOWLEDGEMENTS

Sincere appreciation is expressed to Dr. Darrell Wiener, Problem Advisor, who made this research problem worthwhile and educational.

Sincere appreciation and many thanks are due my mother, Virginia, and my brother, Doug, for their patience, encouragement, and faith that made this endeavor possible.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

CHAPTER I

NATURE OF THE PROBLEM

Introduction

Computer terminals in business are becoming increasingly popular. By calling prestored programs from the computer memory and feeding necessary data from a terminal device in their offices, modern executives have been able to make better use of quantitative techniques in the decision-making process.

The desk-like terminals provide executives direct access to information files of their companies and to computerized analysis of current operations. Gupta pointed out that the broad applications of terminal processing has allowed the executive greater control without stretching his span of management. If students in business courses are pre-exposed to this type of computer processing, it will undoubtedly add enrichment to their future careers in the world of business.[1]

Computer simulations are currently being used in business, industry, and government. Business executives

---

[1]Roger Gupta, Electronic Information Processing (New York: The Macmillan Company, 1971), pp. 243-44.

have used models of systems to evaluate decisions regarding price, product line, advertising, and marketing. These are but a few of the potential uses of computer simulations for business.

The use of computer simulations for the college classroom has been limited even though they are easy to implement. College use of the computer has been confined to problem-solving for research and doing administrative work.

The learning and instructional advantages of using computer simulations in the college business classroom are many. According to Loschetter, some of the most important advantages are: (1) it can eliminate the artificial classroom environment and put the business student in an imitation of the business, environmental, and social conditions of the real world, and (2) it gives the business student an insight into the importance and usefulness of the computer as a tool in the decision-making process.[2]

Statement of the Problem

The problem of the study was to determine the feasibility of using computer terminals for computer simulations and the BASIC language program to build the

---

[2]Richard Loschetter, "The Computer In The Business Classroom," Collegiate News and Views, XXV (Winter, 1971), 19.

models of systems in the design process of computer simulation.

Purposes of Study

The purposes of the study were to (1) assemble information that describes the fundamental concepts and development of computer simulation and models of systems, (2) assemble information that describes computer terminals and the BASIC programming language, (3) write a computer terminal program that would perform a computer simulation to solve a problem for a system, and (4) show one application of computer terminals for the business student to use as an instructional aid in the classroom.

Hypothesis of the Study

1. A computer terminal program can be written to perform a computer simulation to solve a simple business system problem.
2. The application of system simulation and the computer terminal will be a useful aid to the business student in his class work.
3. Students in business can understand and need to learn to use computer terminals.
4. Business students will become more familiar with data processing during their educational years in college.

5. Business students will be able to relate better the uses and services of data processing to their future careers in the field of business.

Need for the Study

A new computer became available at Kansas State College of Pittsburg in the spring of 1974. The computer has terminals located at different points on campus for use by students in their college classes or any part of their education. The expressed purpose and hope of the faculty and the Data Processing personnel were that the terminals would be of benefit and would be used frequently to serve the needs of all students on campus. Thus, it was important that many uses for the terminals be found and defined, and aids be made available to help the students use the computer terminals.

College students have been observed to be shy and unaware of the uses of data processing in their education. Yet, it is important they have contact with data processing during these years so that they can and will use it later in their careers.

Loschetter, in his article, "The Computer In The Business Classroom," made the following comment:

> With the introduction of the computer on college campuses, a new and exciting instructional tool became available to educators. To date

> outside of the data processing and computer science areas, little use has been made of this device as an instructional aid.[3]

In an article, "The Collegiate Business School Today," McGuire gave some interesting views of the past trends in business schools and five important ways he felt business education should move in the future. McGuire summarized these future steps as: (1) recapture a real world concern; (2) emphasize administration; (3) use more quantification and computerization methods; (4) have more studies in leadership, organization, motivation, and creativity; and (5) have more emphasis on management. He saw a new approach in business schools emerging that stressed economics, computers, behavioral sciences, and mathematics. He felt, though, that students were not using or did not know how to apply the tools in these areas to business, and that there was a definite gap that needed to be filled. In the past the schools had moved away from the business world and on to the campus. A real need existed for colleges to change this trend.[4]

The use of terminals connected to a central computer was a further extension of the computer. Problem-solving in mathematics, statistics, and other sciences can be fun

---

[3]Ibid., pp. 19-22.

[4]Joseph W. McGuire, "The Collegiate Business School Today," Collegiate News and Views, XXV (Spring, 1972), 1-5.

and educational when completed on the computer terminals. Students can utilize their time more effectively with the use of the computer. The researcher has seen and continues to see many ways that computers can be of value to students in the field of business. It is evident, though, that students must be given an initial introduction to computer terminals in their classes and also be given some aids that would give them direction in its possible application and use. This study showed the application of a computer simulation for solving a business problem that served to introduce business students to the value of computer terminal processing in their education and future careers.

Delimitations of the Study

The delimitations of this study were developed from two distinct categories. They were those involving the computer terminal and those that were related to models of systems for computer simulations.

The first delimitation was that the problem was written for specific computer terminal equipment, IBM's System 370 Interactive Terminal Facility. The simulation program was written in the BASIC computer terminal language for use on this specific type of computer equipment. Some programming modifications would need to be made if this simulation program were to be used at other data processing installations if different equipment were used.

Another delimitation was that this problem did not include any instruction for the operation of the computer terminal equipment. The instructions were not difficult and could be furnished by the Data Processing Center personnel.

A delimitation that stems from computer simulations and models of systems was that only one business system model was simulated using the BASIC language program. Other computer simulation programs need to be written to consider the value of this application for building models with the BASIC computer language and for computer terminals.

The problem and system were elementary in illustrating the principles of building a probabilistic model for computer simulation. The possibility of building deterministic and probabilistic models for complex business systems and computer simulation of difficult problems was not evaluated.

A final delimitation was that there was no way at this time for the researcher to determine if computer simulation and the use of computer terminals added value to the education of the business students or enriched their future careers in the business world.

Limitations of the Study

A very important limitation of this study was the motivation of the business student to use the computer

terminals as an aid in classroom work. The business student would have to devote additional time to acquire a knowledge in writing BASIC computer terminal programs to take advantage of this application of the computer terminal in his classroom work. The business student would also have to understand the concepts of simulation and building models of systems.

Another limitation was that the simulation program was not tested on the computer terminal equipment. The IBM computer terminal equipment for the KSCP Data Processing Center was not available at the time of the completion of this study.

A final limitation was that the researcher was aware that there existed special computer languages to build models of systems for simulations and the BASIC computer language and the computer terminal may limit the extent of the use of this type of application.

Definitions of Terms

The definition of terms was important in this study because the reader should have a working knowledge of these terms to effectively understand the paper. They were not intended, though, to be as technically defined as those that would be needed by persons working in the fields of data processing or simulation. The terms were also listed

in two categories, one for data processing and one for simulation, for easier reference.

The following data processing terms were defined:

Alphanumeric: Alphanumeric characters are those characters that consist of alphabetic letters, numeric digits, and special characters such as $, %, and *.

Coding: The process of writing the programming statements for a specific problem is called coding.

Data: Information to be processed by the computer is referred to as data.

Data Processing: The term, data processing, is used to describe operations performed by automatic equipment.

Diagnostic: An error message typed on the terminal indicating a specific problem with a program or data is called a diagnostic.

Documentation: The process of documentation means identifying and organizing all of the relevant information concerning a specific program.[5]

Field: The term, field, refers to a group of related characters of data or information.

File: A group of related records of data or information is known as a file.

Fixed-point number: Fixed-point numbers are those numbers in which the decimal has an exact position.

Flowchart: A flowchart is a pictorial diagram that shows the logical flow of data or operations necessary to solve a particular problem.

Input device: A device used to transmit information to the computer is known as an input device.

Instruction: A group of combined characters and symbols recognized by the computer as an order to perform a specific operation is an instruction.[6]

---

[5]C. Joseph Sass, Basic Programming for Business (Boston: Allyn and Bacon, Inc., 1972), p. 288.

[6]Ibid., p. 289.

Integer: An integer is a whole number that does not contain any fractional part.

Loop: A group of statements executed a number of times within a program is known as a loop.

Matrix: An ordered array or two-dimensional table is referred to as a matrix.

Output device: The device that accepts information transmitted from the computer is called the output device.

Program: A logical sequence of steps and statements to be performed by the computer in order to solve a specified problem is referred to as a program.

Storage: Storage is the part of the computer where program information is retained, normally called the memory unit.

Terminal: An input/output device enabling the user to interact directly with the computer system is called a terminal.[7]

Test data: Test data is that data especially created to facilitate easy check-out of a program to determine if errors exist.[8]

Variable: The name assigned to a location within the computer, whose value may change during the execution of a program, is a variable.

The following simulation terms were defined:

Analysis of system: The analysis of system means to analyze the behavior of the system by observation of the inputs and outputs and by explanation of the transformation of the inputs to outputs of the system.

Attributes: Attributes refer to the characteristics that define and describe an entity.

Closed system: Closed systems are those systems where all entity attribute values are defined and changed by events within the system.

Continuous simulation: A simulation in which time moves as in the real world is known as a continuous simulation.

---

[7]Ibid., p. 292.

[8]Ibid.

Design of system: The process of putting the components or elements of a system together is called the design of system.

Deterministic model: A deterministic model is one which does not have any uncertainty, future states can be predicted with reasonable accuracy.

Discrete simulation: A simulation in which time changes only periodically is known as a discrete simulation.

Entity: An entity is defined as a physical or abstract object that causes an activity in a system.

Environment: Objects outside the control of the defined system of a simulation are referred to as the environment.

Event: The occurrence of an activity in a system is known as an event.

Model: A model is an abstraction, simplification, or idealization of a system in the real world.

Open system: Open systems are those systems where entity attribute values are defined or changed by events within the system and the environment.

Probabilistic model: A probabilistic model is a model which has uncertainty; future states are determined by probability.

Probability: The chance that something will or will not happen is called probability.

Probability distribution: The probability of the frequency of an event occurring over a certain number of times is known as a probability distribution.

Random number: A random number means the appearance of this one number does not dictate the appearance of the next number.

Simulation: A representation of a part of the real world is referred to as a simulation.

State: A state is defined as a point in time during a simulation in which the value of the system can be evaluated.

System: A set of objects together with relationships between the objects and between characteristics of the objects is called a system.

System variable: An entity attribute value that is changed through time by the system is known as a system variable.

CHAPTER II

REVIEW OF RELATED LITERATURE

The literature reviewed for this research showed the extent to which computers and computer terminals were being used in education and the extent to which simulation was becoming a part of the curriculum. References were found that illustrated the use of computers and the use of simulation in classroom, but articles on the use of computer terminals for simulation applications were not found.

Computers in Education

The manner and rapidity with which computers were changing the education processes necessitated the discussion of this area first as it related to this problem. The applications possible for education today via computers affected many of the thought processes and ways of doing things. Computer technology has brought about many changes in the field of education within the past decade. This time span was short when compared to other changes of equal importance that have taken as long as thirty to fifty years to make as much of an impression. The computer was viewed by many to be as important as the

printed textbook. The entry of computers into the education process has reorganized the essential teaching and learning techniques and has also made important and very relevant changes in the administrative work of education, according to Gupta.[1]

Education in the area of computer processing has become an important subject area in all institutions of higher learning. The subject of computer programming was offered and in some instances, may be required, in many disciplines. Students in business, engineering, mathematics, and science find it an essential element in obtaining solutions to mathematical or statistical problems in all areas of research. Gupta pointed out that some educators considered computer science a new discipline and rated it as important as the long-established disciplines of mathematics, the physical sciences, and the social sciences.[2]

The use of the computer has enabled the student to explore even greater frontiers and to learn by discovery and research. According to Gupta, the use of the computer has freed the student from the tedious and frustrating task of performing complex calculations. It has increased the number of applications the student can make in his learning processes.[3]

---

[1]Roger Gupta, Electronic Information Processing (New York: The Macmillan Company, 1971), p. 22.

[2]Ibid., p. 23.

[3]Ibid., p. 28.

Gupta concluded that in the future computers will play a very dominant and beneficial role in education and advanced research. The impact of the computer on the future generations of students and, consequently, on society was bound to be enormous.[4]

Studies of Classroom Uses of Computers

A study conducted at Mayfair City College in Chicago involved the application of computer processing in business mathematics courses. Sutherland pointed out that after learning the logical steps for solving a problem in business mathematics, a considerable amount of time was consumed in performing mathematical calculations. A more desirous approach would have the students devote this time instead to exploring and analyzing more problems while letting the computer do the repetitive mathematical calculations.[5]

The study at Mayfair City College was concerned with the question of whether the student could afford the time in his course work to become knowledgeable about computers. The most valuable contribution from the study was that a student did not need to have a great deal of knowledge

---

[4]Ibid., p. 30.

[5]Angela Sutherland, "More Problem Solving in Business Math With the Computer," Journal of Business Education, XLVI (March, 1971), 262.

about the computer to be able to use it as a tool. According to Sutherland, it was possible, in two forty-minute sessions, to provide students with sufficient background in the use of the computer to enable them to solve the problems in business mathematics. The student did not become a programmer, but learned to use only a few computer instructions needed to perform mathematical calculations and to store and to retrieve the material. The students used these instructions with prepackaged routines for handling input and output.[6]

Students in the Mayfair City College study became intrigued with the possibility of using the computer for more sophisticated problem solving and voluntarily pursued deeper applications of problem solving in business mathematics.[7]

Dascher in his article, "EDP in the Elementary Accounting Course," felt that because computer processing in the business community was increasing, it should be a part of the business curriculum through the use of the computer in elementary accounting courses. There were many accounting applications that could be assigned to computer processing, and the accountant did not have to be a sophisticated programmer to be a sophisticated user of the

---

[6]Ibid.

[7]Ibid.

of the computer. The computer provided the service and the support for information needs of accounting reports.[8]

Computer Supplemented Instruction, a design developed by Dascher was used to bring accounting course work and computers together. This type of instruction was a method which provided student usage of the computer without the technical involvement of programming. Students used the relatively simple data manipulation of the accounting entries of debits and credits by using a plus-minus concept of accounting and input was limited to a small set of control and data cards. According to Dascher, the fact that the computer gave immediate results to students so that they could see and correct errors, lessons were more quickly learned. The use of this type of instruction also provided some basic computer exposure to the business student.[9]

In an article by Perritt entitled, "Innovations in an Elementary Accounting Program," he discussed some of the new innovations made at Northern Illinois University in the accounting program. The business department had been faced in recent years with enrollment increases in the accounting classes; hence, new methods of instruction and/or teaching aids had to be developed. One of the solutions to this dilemma was the use of the computer. The

---

[8]Paul E. Dascher, "EDP in the Elementary Accounting Course," Collegiate News and Views, XXVI (Winter, 1972-73), 11-12.

[9]Ibid., pp. 12-15.

innovations had been fruitful and each semester more classes as well as applications in accounting on the computer had been added.[10]

Laughlin in an article entitled, "Computer Related Instruction Developments in Economics," told about a project to develop materials which integrated the use of the computer in economics course offerings in macro and micro analysis. Programs had been written to build economic models to demonstrate different theories of economics, such as the simple interaction of the multiplier and accelerator, and the relationship of savings, investment, income, employment, price level, demand for money, and interest rate. The author of this article concluded that it had added value to the economic courses for both the student and the instructor, and smaller computer systems did not limit the possible applications for economic instruction.[11]

Educational Use of Simulations

In the article, "The Computer in the Business Classroom," Loschetter illustrated the use of the computer in the business classroom for computer simulation exercises. The computer simulation exercises used allowed students

---

[10]Roscoe D. Perritt, "Innovations in an Elementary Accounting Program," Collegiate News and Views, XXVI (Fall, 1972), 13-15.

[11]William M. Laughlin, Jr., "Computer Related Instruction Developments in Economics," Collegiate News and Views XXV (Winter, 1971), 3-6.

LIBRARY

to make predictions about the future based on certain restrictions and then the students were evaluated on their decision-making processes. The students were placed in future business situations and learned the basic essentials that go into making the important and right decisions.[12]

The use of computer simulations in the classroom in the Loschetter study was the responsibility of the instructor. He secured prewritten computer simulation exercises. Loschetter summarized how the simulation exercises were implemented in the following steps: (1) the class was divided into teams, (2) the exercise and required input documents were distributed, (3) the cards were key punched for input by the students, (4) the exercise was run on the computer, and (5) the teams were given an opportunity to re-evaluate their decisions.[13]

According to Loschetter, computer simulation exercises were available from computer manufacturers, industrial firms, and some universities. Any college that had leased time on a computer could take advantage of these economically through the use of batch processing of the computer programs and the data.[14]

Loschetter felt that the computer simulation exercise as a computer-assisted instructional technique displayed a

---

[12]Richard Loschetter, "The Computer in the Business Classroom," Collegiate News and Views, XXV (Winter, 1971), 19.

[13]Ibid., p. 21.

[14]Ibid., p. 20.

LIBRARY LIBRARY

new dimension in learning. Students learned subject matter and basic computer concepts in an exciting and novel environment. The study concluded that the simulation exercises on the college campus had given a new and worthwhile tool to educators.[15]

Summary of the Chapter

Many articles were found on the importance of computers in education and on many levels of education. The articles showed an increasing amount of use of computers in business courses at the college and university level.

The computer was an important addition to a variety of instructional techniques in the business classroom. The business instructor using the computer as a tool did not need to be an expert in data processing. Through the use of the computer, interaction and communication among students, teachers, and data processing personnel increased and made such use very worthwhile. The ability of the computer to respond quickly to input brought eager interest among the users. The action involved in operating a key punch and other input devices stimulated the interest of those students who learn best by physical action. Most had viewed their experiences with the computer in the classroom as successful.

---

[15]Ibid., p. 21.

CHAPTER III

METHOD OF PROCEDURE

The method of procedure was best explained by defining and discussing the three major divisions or areas used as resources for this problem. Each of the areas contributed valuable knowledge and essential parts for the documentary-descriptive method used for the problem.

Library research was conducted at Porter Library, Kansas State College of Pittsburg, and the Computer Center Library at the Kansas State College of Pittsburg. The second area of resource materials came from interviews with IBM computer manufacturer representatives and KSCP Data Processing staff members. Prior education and prior work experience of the researcher made up the third division of resource material for the problem.

Porter Library at KSCP was used as a source of information for books that described the background and the concepts of simulation, computer simulations, systems, and the processes for building models of systems. This material led to needed material on analysis of systems and design of systems.

The KSCP Computer Center Library had manuals, periodicals, and books that furnished documents on computers,

computer terminals, and programming. These references gave background, definitions, types, and uses of data processing in all fields of education as well as business, industry, and government.

Interviews were conducted with IBM representatives about the IBM System/370 Computer and its time-sharing system referred to as the Interactive Terminal Facility. Resource material was also gathered by writing to IBM for manuals describing the computer terminals and the BASIC programming language.

Interviews were conducted with the members of KSCP Data Processing staff on the new computer, IBM System/370, and the new computer terminals. In addition, the expected uses of these terminals for the college were discussed. As the final step, the program application written for this problem was checked by the KSCP Data Processing Center head programmer for accuracy.

The educational background and the experience as a computer programmer at KSCP contributed information on computers, computer terminal processing, and the BASIC programming language to the study. Visits were made to the University of Kansas Medical Center to see and to use their computer terminals. The computer terminals were a part of an IBM time-sharing system and the BASIC programming language was available for users. The personnel at the University of Kansas Medical Center explained the way in

which they use the computer terminals and the BASIC language programs written by the users. A trip was made to the Southwest Missouri State College campus, Springfield, to view and to work on their computer terminals and to talk to the staff of the computer center about the use of terminals by the students. The time-sharing system used at Southwest Missouri State College was also an IBM product using the BASIC programming language.

Knowledge of the researcher, acquired through classes on simulation was a valuable reference for the problem. The researcher studied the special simulation computer programming languages that exist to build models of systems and for computer simulations. The study of these languages was a necessity in understanding computer simulation and finally in using the BASIC language in the process of building models for a computer simulation.

Special simulation languages were computer programming languages designed especially for simulation and nothing else. Many systems had several basic characteristics in common and a wide variety of simulations therefore had programming tasks in common. Simulations were simplified with the preprogrammed routines of these languages. Programmed routines performed tasks such as time keeping and statistical analysis, and provided certain tools such as random number generators and report generators. When complex

systems were involved in a simulation, special simulation languages became vital.

The GPSS language (General Purpose Systems Simulator) was studied and was found best suited for simulations of systems based on queuing concepts. The SIMSCRIPT language was also studied and was important for systems with discrete time keeping tasks. SIMSCRIPT was similar to the FORTRAN language used mostly for mathematical applications. Books describing computer simulation and building models of systems with the FORTRAN language were also reviewed.

When using the special simulation languages, analysis of system was still required for computer simulation. The language aided in the design of system with a general structure for the model of the system and routines for the computer simulation algorithm steps.

The final step in the method of procedure was the selection of a business system problem for a computer simulation. The two processes, analysis of system, and design of system, for simulation were applied to the business system. The BASIC programming language was used for the design of the model of the system and the simulation.

After the resource material was collected, the background and the definition of both simulation and computer simulation were reviewed and described in Chapter IV. The definitions of simulation varied in the resource material and these definitions were explained. In simulation, it was

important to explain the meanings of system and model, for all computer simulation applications require a system and a model. The two processes, analysis of system and design of system, were given as further information related to a description of computer simulation.

In Chapter V, the research completed for the time-sharing computer systems and the BASIC programming language was discussed. The definition of a computer program and the steps followed in writing computer programs were included in the chapter. Important and fundamental concepts of the BASIC programming language with examples of BASIC programming statements served to explain this resource material. A brief review of the history of computer terminals, time-sharing computer systems, and the BASIC programming language introduced that chapter.

This was followed by Chapter VI in which a description of the result of the problem and the study of the computer terminal and the BASIC programming language and simulation were presented. The application of a business system simulation for a computer terminal was explained. The steps used in the analysis of the system and the design of the system were given. The BASIC language program for the model of the system and the flowchart of the design of the system were shown in that chapter.

The summary, conclusions, and recommendations of the research study were presented in Chapter VII.

CHAPTER IV

COMPUTER SIMULATION

In the most general sense, simulation means the representation of reality. Verbal descriptions and schematic or diagrammatic representations of some part of the real world constitute simulation forms. These forms of simulation were not new. McMillan and Gonzalez pointed out that simulations using mathematical expressions and equations that closely approximate the real world were another form of simulation that was a more recent development. When the term "computer simulation" was used, it specified a special kind of mathematical simulation that was not intended to be solved analytically, but rather to be simulated on an electronic computer.[1]

There were many problems in the social sciences, physical sciences, engineering, and business fields that could be stated in a mathematical form, but were too difficult to be solved by analytic methods and so must be solved by computer simulation, according to Wyman. More and more computer simulation was being used to study such problems

---

[1]Claude McMillan and Richard F. Gonzalez, Systems Analysis, A Computer Approach to Decision Models (Homewood, Illinois: Richard D. Irwin, Inc., 1968), p. 23.

with the cost of computation decreasing and becoming more economical.[2]

Computer simulation provided an effective means of testing and evaluating alternatives for proposed systems without affecting the real system. In the IBM: GPSS IV Manual, it was pointed out that several hours, days, weeks, or even years of operation of a system could be simulated in a matter of minutes on a computer. Simulation was not a precise likeness of the system, but rather a symbolic representation of the system. The simulation could not provide optimal solutions, but it did yield very useful insights into the behavior of very complex systems. It provided measurements which would be impossible to obtain in any other way.[3]

Business simulation as defined by McMillan and Gonzalez meant setting up for a digital computer, mathematical expressions and equations that describe a business system based on the description and assumptions about the business operation. The simulation then provided a method to test different management policies and market situations to determine their effect on the company success. Simulation

---

[2]Forrest Paul Wyman, Simulation Modeling: A Guide to Using SIMSCRIPT (New York: John Wiley and Sons, Inc., 1970), p. 1.

[3]IBM: GSPP IV Manual (New York: International Business Machines Corporation Programming Publications, 1971), p. 1.

allowed experiments to be conducted for a system without affecting the real system.[4]

Systems of Simulation

The most important concept in the simulation process was the "System." The system was defined for every form of research and learning, and it also had a special meaning in simulation. In simulation a system was referred to as a set of objects together with the relationships between the objects and between the characteristics of the objects. The objects were called entities and the characteristics of the objects were called attributes in simulation.

The kinds of entities in business systems were limitless. They could be physical objects such as machines, raw materials, finished products, clerks, and machine operators or they could be abstract objects such as profit goals, sales quotas, production standards, or costs. The objects or entities of the systems performed activities or changed their behavior at points in time. The activity or behavioral change of an entity of the system was referred to as an event.

Entities were described by defining their attributes. The attributes were characteristics or properties of the entities. For example, if machine was an entity of the

---

[4]McMillan and Gonzalez, Systems Analysis, p. 26.

system, the attributes could be price, weight, speed, cost of operation, or production rate. Another entity example for a system, inventory item, might have attributes such as quantity on hand, cost of storing units, or number of units used last year. There were also an endless number of attributes that could be used to describe an entity. McMillan and Gonzalez in the book, Systems Analysis, A Computer Approach to Decision Models, indicated that both entities and attributes must be limited to those most important for a specific simulation.[5]

Entities and attributes of a system were tied together by their relationships. These relationships tied the system together and if it were not for these relationships, the term, system, would be meaningless. The relationships that could be defined were endless and only the ones important for the specific simulation were defined in the analysis. According to McMillan and Gonzalez, they should be limited to those that had an important effect on the way the system behaved when it underwent change. An example of a relationship used in a business system simulation would be sales volume and advertising expenditure.[6]

After a system was defined by its entities, attributes, and relationships, it was necessary to identify the environment of the system. McMillan and Gonzalez stated that the

---

[5]Ibid.

[6]Ibid., p. 2.

environment was defined as the set of all objects that, when there was a behavioral change in the objects, it affected the defined system. Behavioral changes of entities in the defined system could in turn affect the environment entities.[7]

To separate the objects of a system from the objects of its environment was difficult. McMillan and Gonzalez suggested that one way to distinguish between them was to consider whether or not the activity or behavioral change of an object was influenced or subject to management control. Customers, banks, unions, venders, competitors, or the economy were examples of objects beyond management control and could be objects of the environment of a system.[8]

Systems could be divided into subsystems, and the entities of the total system could be members of several of the subsystems. McMillan and Gonzalez pointed out that the subsystems could be studied separately or as a whole of the total system. The study of subsystems involved a study of the microscopic behavior of a system. The study of a total system was a study of the macroscopic behavior of a system.[9]

Systems were classified into closed and open systems. An open system meant there was an exchange of activities

---

[7]Ibid.

[8]Ibid.

[9]Ibid., p. 3.

or information with the environment of the system. A closed system would have no exchange of such activities or information.

A system was described by its "state." In the simulation process, the entity attributes of a system took on different values at different points in time. At any one point in time in the simulation process, the state of the system was the current value of the entity attributes of the system. In describing the system state, a list was made of the values of the attributes of the entities to determine the specific system state.

The state of the system was either stable or unstable. If the values of the attributes of the entities remained constant or within defined limits in each system state evaluation, the system was considered to remain stable, according to McMillan and Gonzalez. In contrast, an unstable system would be one that would have different entity attribute values in each system state.[10]

The system was in equilibrium when there was an absence of external activity with the system. McMillan and Gonzalez pointed out that when a system was exposed to external activity and its state was changed only temporarily and equilibrium was obtained again, it was classified as stable.

---

[10]Ibid., p. 5.

If the system state was never returned to equilibrium the system was referred to as unstable.[11]

The term, information feedback system, was also used in simulation. According to McMillan and Gonzalez, it referred to systems in which a portion of the output of the system was used to influence future states of the system. The feedback into the system controlled future decisions made in the system.[12]

System entity attributes that took on different values for different states of the system were called system variables or state variables. The value of these variables were generated by the system and depended on what happened earlier in the system. System entity attributes that remained constant were called parameters. The values did not change during a simulation of the system. McMillan and Gonzalez indicated that initial values must be given for both system variables and parameters, but the system variables would take on different values depending on the relationships evaluated in the simulation of the system. An example of this differentiation could be explained with the entity machine. The attribute production rate for the entity machine would be a system variable and the attribute cost of the machine would be a parameter.[13]

---

[11]Ibid., p. 4.

[12]Ibid., p. 6.

[13]Ibid., pp. 31-32.

Random variables were also characteristic for many business systems and this meant that there was uncertainty in the input values to the system. The random variables caused many successive changes in the system. Another characteristic for the systems was the fact that the relationships of the entities and attributes of the system were not well behaved. It was difficult to trace down an exact or general relationship for different states of the system.

The typical systems that were processed on the computer were very complex. McMillan and Gonzalez pointed out that this complexity meant that the systems had a large number of entities, attributes, variables, parameters, and many relationships and events to which the systems were responsive.[14]

Models of Simulation

Simulation methodology involved experimentations with models of systems. There were many kinds of models for systems. They could be constructed from physical objects or they could be representations in pictorial form. Flow diagrams and organizational charts were examples of the pictorial form. A mathematical model was another way of representing a system. A mathematical model consisted of a

---

[14]Ibid., p. 26.

set of equations whose solution explained or predicted changes in a system.

A form of mathematical model was used in computer simulations. McMillan and Gonzalez explained that these mathematical models were a result of analyzing and describing a system of the real world. The abstract nature of mathematical models made them subject to manipulation and could give precise information. Computer models were simply defined as mathematical models expressed or written according to a particular set of rules so that the model could be processed by the computer.[15]

Mathematical models for simulation were subdivided into deterministic models and probabilistic models. McMillan and Gonzalez, in their book, Systems Analysis, A Computer Approach to Decision Models, pointed out that if there was an optimum design and the design was within defined and unvarying limits, the model was deterministic. Deterministic models were devoid of uncertainty, and changes of the system state could be perfectly predicted.[16]

More typically, though, models were probabilistic, because most systems were characterized by having entity attributes that took values which were the result of an interaction of factors which were not well understood, according to McMillan and Gonzalez. The entity attribute

---

[15]Ibid., p. 12.

[16]Ibid., p. 13.

values were determined by successive trials of stochastic processes. A stochastic process was defined as a repetition of experiments, the results of which were determined by chance. If the entity attribute values of a system were determined by stochastic process or probabilistically, the knowledge of the system was less than perfect. When uncertainty existed and future states of the system were anticipated, subject to the probable occurrence of a sequence of events usually in the environment, the model was called a probabilistic model. With uncertainty more the rule than the exception in the real world systems, most models would be probabilistic.[17]

Models of systems for computer simulation were moved through time, so that the dynamic behavior of the systems could be studied. In the simulation process, time began at zero when the parameters and system variables of the model of the system had initial values. Next, various events were generated which caused changes to take place and which resulted in new values for the system. Each event advanced the model through time.

Building a Model of the System

To accomplish the goals of simulation for any system, the elements of the system were isolated and the logical rules governing their interaction were formulated. McMillan

---

[17]Ibid.

and Gonzalez stated that this gave a description of the system called a model of the system. The model was limited to those aspects of the system which were pertinent to the types of answers sought in the simulation.[18]

Two processes were necessary in the simulation of a system: (1) analysis of system, and (2) design of system. The technique for analysis of system had these basic steps. First, the system had to be understood thoroughly and to do this, the analyst began with observations of the system and its events. Each part of the total system was examined because each part interacted and had an effect on every other part. In the book, Simulation Modeling: A Guide to Using SIMSCRIPT, Wyman emphasized that the simulation model builder discovered which of these interactions were significant and which were not necessary for the model of the system. After this observation and familiarization with the system came the second step, the formulation of an hypothesis, which was a possible explanation of the behavior of the system. This hypothesis gave the goals and boundaries of the model of the system. Third, the analyst separated the elements of interest of the system into exogenous and endogenous event variables. Exogenous event variables were those which were beyond the scope or bounds of the study; their originating causes were unknown and

---

[18]Ibid., p. 1.

their behavior could only be determined by a statistical distribution. Endogenous event variables were those for which cause and effect were known and which could be controlled in the model of the system. Fourth, the analyst defined the entities and their attributes. The choice of entities for the model of the system was subjective and depended on the goals of the system. Fifth, the relationships between the entities were discovered and defined. These relationships provided the structure of the model of the system and were as close to reality as possible. The resulting analysis gave an abstraction and simplification of the system and the events.[19]

Having observed the system, having specified the parts of the problem of the system, having formulated a hypothesis, and having defined the entities and attributes with their relationships, the next step was to turn to the second process, design of system. In the design of the model of the system, McMillan and Gonzalez pointed out that the model builder had alternate ways of putting the model of the system together. The model builder sought to design a model of the system that optimized the behavior of some measure of the system. Design of system for simulation was primarily concerned with developing an algorithm that solved the model of the system based on the behavior of the system.

---

[19]Wyman, Simulation Modeling, p. 2.

The model builder also specified the data required to solve the algorithm, referred to as input. Finally, the model builder defined the information or solution of the model. This was referred to as the output of the model of the system.[20]

These basic steps for a design of system for simulation are shown in Figure 1, Design of System.

Figure 1
Design of System.

Any information required by the algorithm was referred to as data. Some models required a large amount of data or input while others required very little or in some cases, no data at all.

To construct the algorithm, a sequence of steps was devised for the computer. These steps processed the data in such a way as to solve the model and produce the output or solution. In the book, Systems Analysis, A Computer Approach to Decision Models, McMillan and Gonzalez stated

---

[20]McMillan and Gonzales, Systems Analysis, p. 9.

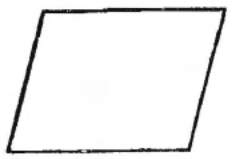
that the exact definition of an algorithm was a set of arithmetical equations and logical operations in an ordered sequence, to achieve the goals of the model of the system. The logical operations allowed decisions that altered the sequence of steps in the algorithm. The output gave all the required information concerning the solution.[21]

In designing the model for the computer, the model builder specified the steps of the design in a graphic form before writing the computer program. This was called a documentation of the model of the system. This documentation was very helpful in the design of system process and in coding the model into the symbolic language for the computer to process.

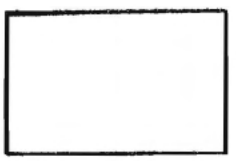
To document or represent the model of the system in graphic form, symbols were used. Each step of the design was specified by a symbol and these symbols were shown in the order of the sequence that they were used to solve the model. The form of each symbol suggested the process needed to solve the design of system algorithm. The symbols were connected together by arrow line segments that represented the sequence of the design of the model of the system. Shown in Figure 2, Design Symbols, are the basic symbols used and an explanation of when they were used in the design.

---

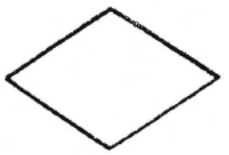
[21]Ibid., pp. 18-19.



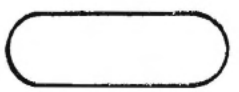
This symbol had two different uses in the design of the model of the system. It represented the data required or the solution produced in the model.



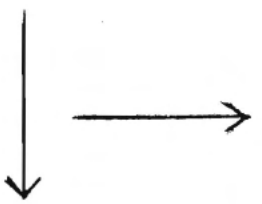
An arithmetic operation or equation of the design of the model of the system was represented by this symbol.



A logic operation in the design of the model of the system was represented by this symbol.



This symbol represented the beginning or the ending of the design of the model of the system.



The arrow segments connected the symbols and showed the sequence of the steps of the design of the model of the system.

Figure 2
Design Symbols.

The scientific method was applied in building models of systems. According to McMillan and Gonzalez, this meant that the model accounted for all known facts, and secondly, the model enabled predictions of the system to be made which could be tested by any independent observer.[22]

The final step in simulation was to evaluate the model. The testing or validating of a model was done by making more observations and measurements of the system, making further experiments with the model of the system, and comparing the results of the computer simulation to the observations.

Summary of the Chapter

Simulation meant a representation of reality. There were many different forms of simulation or ways of representing the real world, such as verbal descriptions and pictorial designs. Another way was with mathematical equations. If a mathematical equation was extremely difficult to solve analytically, it was solved on a computer and this was known as a computer simulation.

Simulation provided a method for testing and evaluating alternatives for proposed systems or a method of studying the behavior of systems, without affecting the real system. The simulation of a system was not a precise

---

[22]Ibid., p. 9.

representation, but was a symbolic representation. The simulation did not result in an optimal solution for a problem, but did give valuable insight into a problem. The simulation of a system provided the best way of discovering the behavior of a system. Hours, days, weeks, or years of the operation of a system were simulated in a matter of minutes on the computer.

Without a system there would be no application of simulation. A system was defined as a set of objects with definable characteristics that were tied together by relationships. These relationships tied the system together. The objects of a system were called entities and the characteristics used to describe the objects were referred to as attributes. An activity of an object of the system was known as an event.

Systems were classified many different ways; such as, open and closed systems, and natural and man-made systems. Also, systems were divided into subsystems. Systems were described at any one point in time by evaluating the characteristics of the objects. The result of this evaluation was called the state of the system.

Models of systems were descriptions of systems. Models were not precise analogs of systems, but were a likeness of the system. Computer simulations required the two processes of analysis of system and design of system. The analysis of a system began with observation of the system and the

identification of the parts of the problem of the system. The analysis furnished a hypothesis of the behavior of the system, a definition of the entities, attributes, and relationships of the system, and the events of the system.

The design of system was the building of a model of the system that was processed on the computer. The primary purpose of the design was the construction of an algorithm to solve a problem of a system, or to study the behavior of a system.

Computer simulations were used to solve problems in the field of business as well as social sciences, physical sciences, and engineering. Simulations in the field of business were used primarily to evaluate different alternatives of management policies that could be applied to business operations.

CHAPTER V

COMPUTER TERMINALS AND BASIC

Development of BASIC Language and Time-Sharing Computer Systems

The BASIC programming language was created with the appearance of "time-sharing computer systems." Time-sharing was the utilization of one central computer by many different users. Each user had a certain fixed amount of time on the computer and they shared this time on a rotating basis. The user had access to the computer by means of a typewriter-like device called a remote terminal. The terminal was connected to the computer by telephone lines and the user utilized the terminal as an input-output device to communicate with the central computer.

In BASIC Programming for Business, Sass stated that one of the first time-sharing systems developed was at Massachusetts Institute of Technology in 1961. In the next few years, time-sharing systems were developed by the Digital Equipment Corporation and the Rand Corporation. A majority of the major computer manufacturers had time-sharing systems.[1]

---

[1]C. Joseph Sass, BASIC Programming for Business (Boston: Allyn and Bacon, Inc., 1972), p. 4.

There were many advantages of time-sharing computer systems that made it one of the fastest growing segments in the computer field. Most important of these was that it had provided computer access for so many people. Other advantages were lower costs brought about by many users sharing the cost of one central computer, the ease of operation with the teletypewriter terminal, and the conversational mode that permitted the user to communicate with the computer in a question and answer response.

Sass pointed out that the capability of bringing the computer to the user had brought many more users to the data processing field. Many of the new users were not computer programmers, yet they needed the capabilities of the computer to assist in some activity in their occupation. They needed a language that was easy to learn, not technical, yet one that provided the maximum benefit of the computer. The BASIC programming language was developed by John G. Kemeny and Thomas E. Kurtz for use on the computer terminals of the time-sharing system at Dartmouth College in 1964. BASIC stands for BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE. The language code was similar to English and the programs required very few instructions.[2]

---

[2]Ibid., p. 6.

Programs and Programming Defined

Before learning about the BASIC programming language, an understanding of the general concepts of a program and programming was necessary. A program was a set of directions performed in logical sequence on the computer to solve a particular problem. Programs were written in a programming language used to communicate with the computer. The languages differed according to the type of computer and the type of problem. The program must be written according to the rules of the programming language and for the specific type of computer, since the computer could do only what it was instructed to do.

There were several steps involved in program writing. The first was defining the problem. The definition involved analyzing what data would be needed for input, deciding upon procedures to follow in order to obtain the solution, and the data required for the output. Second, a method was created to solve the problem. The method was called the algorithm of the program. It consisted of a set of steps that used the data in such a way as to lead to the solution of the problem. The logical flow of directions to solve the problem on the computer was the third step. A flowchart was used for this step. A flowchart was a pictorial representation that showed the algorithm. The fourth step was writing the program. This was called coding the program and it must follow the rules of the

programming language used. The flowchart created in the third step was a guide for the coding of the program. Fifth, the running or execution of the program on the computer was performed. This was a test to see if the program led to the solution of the problem.

Of the five steps used in program writing, establishing the flow of directions to solve the problem shown by a flowchart was the most important step, according to Sass. The flowchart helped to define the problem by illustrating each separate part required to solve the problem. The flowchart showed the source of input, the logic used for the solution of the problem, and the output. The flowchart was also a guide in writing the program statements for the computer and it became a documentation of information for the problem.[3]

The pictorial representation of the steps used special symbols. Each symbol had a special meaning and the symbols were connected with directional arrows. The arrows gave the sequence of the procedures to be performed by the computer that would lead to the solution. Inside each symbol there was a written explanation that described the procedure step represented.

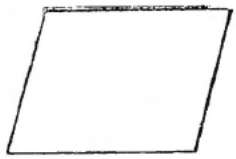
Shown in Figure 3, Flowchart Symbols, are a few of the most used symbols in flowcharting with an explanation of what the symbol represented or meant.

---

[3]Ibid., p. 12.

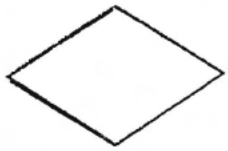
The Terminal Symbol represented the start or end of the flowchart.

The Process Symbol represented the defined operations that caused changes in value, form, or location of information.

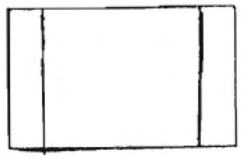


The Input/Output Symbol represented information available for processing or the recording of processed information.

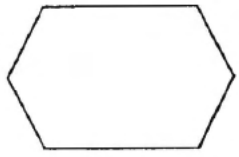
The Offpage Connector Symbol represented entry to or exit from a page.



The Decision Symbol represented a decision that determined which of a number of alternative paths to follow.



The Predefined Symbol represented a named operation or program steps specified in a subroutine.



The Preparation Symbol represented instruction modification that changed a program or initiated a routine.

The Flowline Symbol linked the symbols together and showed the sequence of the flowchart steps.

Figure 3
Flowchart Symbols.

BASIC Program Writing

After understanding the general concepts of programming and programs, the reader was prepared to study concepts of the BASIC language. The general concepts which follow were used to serve only to introduce the reader to the BASIC language. These concepts were based on IBM's Time-Sharing system and their version of the BASIC language.

In the ITF: BASIC Terminal User's Guide, it was stated that a BASIC program consisted of a number of entries that comprise the steps necessary to solve a problem. Each entry had a line number and was called a BASIC statement. The line number was an integer between 0 and 99999. The line number specified the order in which the entries of BASIC statements were processed.[4]

The BASIC statements were organized according to specific rules and were divided into two types: executable and nonexecutable according to the ITF: BASIC Terminal User's Guide. Executable statements specified a program action and nonexecutable statements provided information or data for program action. An example of an executable statement was INPUT R4, R3, R2, R1. An example of a nonexecutable statement would be DATA 67,32,90. BASIC entries were retained in the computer in the numeric

---

[4]ITF: BASIC Terminal User's Guide (New York: International Business Machines Corporation Programming Publications, 1970), p. 69.

sequence of the line numbers, but they did not have to be entered in a numerical sequence. Executable and nonexecutable statements could also be intermixed. The BASIC statement after the line number was composed of constants, variables, expressions, and/or labels or messages combined with BASIC command key words.[5] Examples of BASIC statements were:

```
700 LET S3=B7 * 678.2 + G2
710 READ R1, Y8, R2, Y7
720 IF A$=B# THEN 620
730 PRINT ' THE TOTAL ENROLLMENT IS ', T4
```

Constants were either numeric, character, or internal values. Constants had a defined value that did not change throughout the execution of a program.

Numeric constants were real numbers, either whole or fractional. For example, the numbers 105, -21, 18.945 were all considered numeric constants. Numeric constants could be represented as above or in exponential form. The exponential form was usually used for very large or small numbers. To express the number 59.2×10^7 in BASIC, it was written 59.2E7. This was evaluated as 59.2 raised to the seventh power of base ten. In summary then, any group or string of characters that was a decimal number was called a numeric constant.

---

[5] Ibid.

Character constants were strings of characters made up of alpha-numeric characters. These constants were always enclosed by a pair of single or double quotation marks. The computer stored all character constants as eighteen characters, filling them with blanks if the length was less than eighteen. Examples of character constants were 'STORE A56', 'PROGRAMMING', and 'INTEREST 5%'.

Internal constants were strings of characters that represented a value that had been predefined by the BASIC language. The three internal constants defined were for the values of e,PI, and the square root of two; each had a name, as was specified by the ITF: BASIC Terminal User's Guide. The programmer used internal constants by referring to them by their name.[6]

Variables were also used in BASIC entries. There were four types of variables: simple numeric, simple character, array numeric, and array character variables. A variable represented a string of characters whose value was assigned and could change during the running of a program. This meant the data could take on different values at different times during the execution of the program and for each different execution of the program.

A simple numeric variable was named by a single alphabetic character or an alphabetic character followed by a

---

[6]Ibid., p. 73.

digit. Numeric variables stored only numbers. Examples of simple numeric variable names were H8, W, and K6.

Character variables were named by an alphabetic character followed by a dollar sign. These variables contained strings of characters made up of alphabetic letters and/or numeric digits. Examples of simple character variable names were T$, U$, and N$.

Array numeric variables and array character variables were sets of character strings. Array numeric variables contained numbers only and they were identified by naming them with a single alphabetic character. Array character variables contained alphabetic letters and numeric digits and were identified by naming them with an alphabetic character followed by a dollar sign. Each character string in the array character variable set was eighteen characters in length.

All numeric variables initially had the value of zero. All character variables were initially set to a value of blank by the BASIC system.

Expressions contained constants, simple variables, array variables and/or functions which were combined with operators. Expressions represented a value or values and were divided into three different types: scalar, array, and relational. Scalar expressions represented a single value. Array expressions resulted in a set of values, and relational expressions represented a true or false value.

There were five mathematical operations possible in BASIC expressions: exponentiation, multiplication, division, addition, and subtraction. The mathematical operation was coded by using a symbol for the operation. The ITF: BASIC Terminal User's Guide gave the following list of symbols and the mathematical operation they represented.

1. + addition
2. - subtraction
3. * multiplication
4. / division
5. ** exponentiation

Expressions could also contain parentheses and unary operators. The unary operators were the plus and minus signs for the numeric values.[7]

The expressions in BASIC were evaluated according to rules of precedence. The highest precedence was the parenthesis. If there were parentheses within an expression in the BASIC statement, this part of the expression was evaluated first. The second highest precedence was exponentiation. Multiplication and division shared the third level of precedence and subtraction and addition made up the fourth or lowest level of precedence. For those levels of precedence that had two mathematical operations possible, the computer evaluated the expression working from left to right.

---

[7]Ibid., p. 74.

BASIC Language Commands

The following is an explanation of BASIC programming concepts of the most commonly used BASIC commands. They are briefly explained and further reference to the ITF: BASIC Terminal User's Guide should be made. The BASIC commands were explained here in order that the Simulation Programs discussed later in this research problem in which the BASIC language was used could be better understood.

The BASIC commands were grouped together according to the similarity of their use. The following types of BASIC commands were discussed: Input-Assignment Commands. Output Commands, Transfer Commands, Array Commands, Subroutine Commands, Function Commands, File Commands, and System Commands. Several miscellaneous commands that were used for almost every type of program were also included.

Input-Assignment Commands. There were three methods available for entering data into a BASIC program. They consisted of the LET command, the INPUT command, and the READ command which was used in conjunction with a DATA statement.

The LET command entered data into a program by assigning values to variables. The assignment of the value to the variables could be done by assigning numeric or character constants and variables to the variables or by assigning values to the variables with mathematical expressions or functions. The general format was LET (variable name) =

(character or numeric constants and variables or mathematical expressions or functions). According to Sass in BASIC Programming for Business, this BASIC command included an equal sign and this could be interpreted as assigning the evaluation on the right side of the equal sign to the variable on the left.[8] Examples of the use of the LET command were:

```
LET B = 5 + N * XJ
LET B$= 'TOTAL'
```

The LET command was also the principle way to perform computations in the BASIC programs since it evaluated mathematical expressions and functions on the right side of the equal sign and assigned the result to the variables on the left side of the equal sign.

There was one other use of the LET statement. This was called the Multiple LET statement. A single mathematical expression, function, numeric, or character constant or variable could be assigned to more than one variable at a time. In the following BASIC Multiple LET statement, the result of the expression 56 ** 3 + 67 would be assigned to variables E6, T, and R2.

```
LET E6, T, R2 = 56 ** 3 + 67
```

---

[8]Sass, BASIC Programming, p. 21.

The INPUT command was the second method for entering data into a BASIC program. It assigned specific values to variables during the execution of a program. Because the values were assigned during the execution, each different execution of the same program allowed the user to vary the values assigned to the same variable. When the program was run and the INPUT command was encountered, the computer printed a question mark on the teletypewriter. The user of the program at this time supplied the value or values that were assigned to the variables in the INPUT command.

The INPUT command was usually used when only limited amounts of data were entered into the program or the data varied for different program executions. The variable names in the INPUT command must correspond to the types of values entered by the user, or the computer would terminate the execution. The general format of the INPUT command was INPUT (variable or variables separated by commas). The PRINT command was often used with the INPUT command to print a message which identified the variables for the data the user was to supply and type in at the terminal. An example of the INPUT command was:

```
INPUT   A, R$
?       785.3, 'LANGUAGE'
```

In the previous example, the computer typed the question mark and the user typed the values 785.3 and LANGUAGE following

the question mark, for the variables A and R$ named in the INPUT command.

The READ command and the DATA statement was the third method for entering data into a BASIC program. The READ command specified the variables which were assigned the values supplied in the DATA statement. The READ command was used for entering large amounts of data into a program.

According to ITF: BASIC Terminal User's Guide, the first value in the DATA statement was assigned to the first variable in the READ command; the second value was assigned to the second variable, and so forth. Each variable in the READ command was assigned the successive values in the DATA statement; there must be at least one DATA statement when using the READ command in a program. The variable name types named in the READ command had to correspond to the type of values given in the DATA statement, but the variable types could be intermixed. The DATA statement did not have to precede or follow the READ command; it could be placed anywhere before the END command of a program. The programmer could change the DATA statement before any execution of the program when different values were needed for the variables. The general format for the READ command and the DATA statement was READ (variable or variables separated by commas) and DATA (numeric or character values, separated by commas).[9]

---

[9] ITF: BASIC Terminal User's Guide, p. 92.

An example of these commands was:

```
READ  C, N$
DATA  59.321, 'PROGRAM'
```

The RESTORE command was used with the READ command and DATA statement if it was necessary to use the same DATA statement for more than one READ command. The RESTORE command resets the reading of the DATA statement so that the subsequent READ would start assigning values at the beginning of the list of values in the DATA statement.

Output Commands. The method for output of processed information on the terminal used the PRINT command. There were several important concepts that must be understood about the printing that occurs at the terminal.

The format of the print line was to some extent controlled by the BASIC system. The programmer controlled the density of the line, but the format of the values printed was standard. Each PRINT command caused the computer to begin a new line on the terminal.

Horizontally, the print line was divided into what was called full print zones, each one having eighteen print positions. The comma between the variable names or constants in the PRINT command caused the computer terminal teletype to move across the page to the next full print zone. To explain this concept, consider the PRINT command, PRINT 'TOTALS', A, D. The computer teletype printed TOTALS

starting at print position one on the line, skipped to position nineteen to print the value for the variable A, and finally would move to print position thirty-seven to print the value of the variable D.

The programmer controlled and changed the number of print zones on a line by the use of semicolons between the variable names and constants. In ITF: BASIC Terminal User's Guide, it was stated that the use of semicolons was called using packed print zones. With the use of packed print zones for numeric data, the length of the printed field was determined by the size of the value. For example, if the length of the numeric value was two to four characters, the packed zone was six characters in length, and the number 321 appeared as X321XX, where X represented a blank. If the length of the numeric value was five to seven digits, the packed print zone was nine characters. For example, the number 321445 appeared as X321445XX. If the data item was a character variable or constant, the size of the packed print zone was equal to the length of the character variable or constant. The character variable that contained the word "item," had a print field of four characters.[10]

The PRINT command contained numeric or character variables or constants, expressions, and format items. In BASIC Programming for Business, Sass gave several important

---

[10]Ibid., p. 29.

variations when using the PRINT command. The first one was to print values of variables or results of some calculations. The general format of this PRINT command was PRINT (variable or variables separated by commas). A second very helpful use was the PRINT command to print a character constant to label or identify variables that were the output of the program. Character constants were also used to print messages to the user of the program. The general format for this variation of the PRINT command was PRINT (a message or label enclosed by quotation marks). A third possibility was using the PRINT command to print a blank line. This made the output of the program more readable and improved its appearance. The format for this variation was the PRINT command with nothing following it. The fourth variation was printing both character constants and numeric or character variables on the same line. The general format for this variation was PRINT (label or message and variables separated by commas).[11] The following were examples of these PRINT command variations:

```
PRINT V, M
PRINT 'THIS MESSAGE IS FOR THE PROGRAMMER'
PRINT
PRINT 'THE INVENTORY LEVEL IS', A5
```

The PRINT USING command was a BASIC command similar to the PRINT command. It was used though, for controlling

---

[11]Sass, BASIC Programming, pp. 26-29.

the format of numeric variables to be printed in a program. The PRINT USING command specified the variables to be printed and the statement number of the IMAGE statement to be used to print these variables. The IMAGE statement gave the format of the print line. A colon identified the start of the IMAGE statement. All the alphabetic characters in an IMAGE statement were printed as they appeared in the IMAGE statement and the "#s" were replaced by the values of the numeric variables given in the PRINT USING command, according to the ITF: BASIC Terminal User's Guide.[12]

The general format of the PRINT USING command was PRINT USING (image statement number), numeric variable or variables separated by commas). The general format of the IMAGE statement was IMAGE: (character constants and format specifications). The character constant was any alphabetic character except the "#" (number character). An example of the PRINT USING command and IMAGE statement was:

```
305 PRINT USING 670, M7, G
670 IMAGE:  THE FINAL TOTAL   $ ###.## FOR ITEM ##
```

The PUT command caused values to be placed in or written on a specified file. This was another way to obtain processed information in BASIC, not on the terminal teletype, but on a storage device such as a magnetic disk pack or tape. The general format was PUT file name, (constants

---

[12]ITF: BASIC Terminal User's Guide, p. 30.

or variables separated by commas). An example of using it was:

```
PUT 'DGF', C7, 2134.90, X8.
```

The file name in the preceding example was 'DGF' and the values written on the file were the values of the numeric variables C7 and X8 and the numeric constant 2134.90.

Transfer Commands. The transfer commands or statements gave the programmer two important programming capabilities, altering the sequence of the execution of a program and building program loops. Altering the sequence of execution meant that the order of the program statements were not processed in the original sequential order in which they were written or entered.

A program loop was a series of program statements executed a variable number of times, but only coded once in the program. With the use of program loops, it was not necessary to rewrite the same statements over again every time they were needed. Loops were very important in programming because they saved time in writing statements and shortened the length of the programs.

There were several BASIC commands that were used to alter the sequence of execution of a program and to build program loops. They were considered transfer commands because they transferred control. The following transfer commands were discussed: the GOTO command, the IF-THEN command, the FOR command, and the NEXT statement.

The GOTO command had two forms. The first form caused control to be unconditionally transferred to a specified statement. This was called the SIMPLE GOTO statement. The GOTO command was followed by a line number that indicated the next program statement to be executed which altered the original sequential order of processing. The general format was GOTO (statement number). An example of the GOTO command was:

GOTO 240

The second form of the GOTO statement caused control to be transferred to one of a set of statement numbers. Which statement number the control was transferred to depended on the value of an expression specified in the GOTO command. In ITF: BASIC Terminal User's Guide, this was called the COMPUTED GOTO statement and was known as a conditional transfer of control. The expression in the command was any valid BASIC expression. The line numbers were those to which the transfer of control would go depending on the value of the expression. The value of the expression was truncated to a whole number. When the GOTO command was executed, the computer transferred control to the first line number specified, if the value of the expression was a one; or control passed to the second line number if the value was two, or to the third line number if the value was a three, and so on. If the value of the expression was less than one

or greater than N (N being the number of line numbers listed in the GOTO statement), the next executable statement following the GOTO command was processed. The sequence of the program was not altered if this occurred. The general format of the GOTO command was: GOTO (statement numbers separated by commas) ON (arithmetic expression).[13] An example of the COMPUTED GOTO command was:

GOTO 45, 32, 67, 17 ON 88 + A/T - 6

The IF THEN command provided a method for the programmer to alter the sequence that the program statements were executed. It was a conditional transfer of control. The transfer of control depended upon a relationship tested. The relationship involved a comparison between two expressions. The expressions were of various complexities. Expressions contained character or numeric constants or variables or it was an arithmetic expression. The IF THEN statement also included a logical operator to indicate the type of relationship or the type of comparison to be made. The ITF: BASIC Terminal User's Guide gave the following as the types of comparisons that could be made: less than, greater than, equal to, not equal to, less than or equal to, and greater than or equal to. If the relationship between the two expressions was false, the next statement in the sequence of the program was executed. If the relationship tested was true, transfer

---

[13]Ibid., p. 82.

of control occurred and went to the statement number following the THEN in the IF THEN statement. The general format was IF (character or numeric constant or variable or expression) relational operator (character or numeric constant or variable or expression) THEN (statement number).[14] Examples of the IF THEN command were:

IF N4 > X + 56*33 THEN 590

IF M7 ≥ Y3 THEN 370

IF G$ = J$ THEN 780

IF E - 56.97 < Y THEN 128

The FOR command and the NEXT statement were used for the programming technique of looping, according to Sass. A program loop caused a specified group of program statements to be executed a certain number of times. The FOR command identified the beginning of the loop and caused the repeated execution of the statements that followed it up to the matching NEXT statement. The NEXT statement identified the end of the loop and was known as the physical end. Control was passed to the statement that sequentially followed the NEXT statement when the loop statements had been executed the specified number of times.[15]

The FOR statement specified the number of times the loop or program statements would be repeated and also

---

[14]Ibid., p. 28.

[15]Sass, BASIC Programming, p. 97.

specified the numeric variable used as a counter. Each time the loop was executed, there was an increment of one added to the counter variable. When the value of the variable or counter exceeded a numeric constant or variable used to specify the number of times to repeat the execution, the loop ended.

The constants or variables in the FOR command that gave the starting and ending values for the counter variable were called the range specifications. There was an increment of one for the counter variable unless it was explicitly coded otherwise with the use of the STEP option. In the following example: FOR I5 = 1 TO 50 STEP 2, an increment of two would be added to the counter variable I5 every time the loop was executed. The general format of the FOR command was FOR (numeric variable) = (numeric constant or variable or expression) TO (numeric constant or variable or expression) STEP (numeric constant or variable or expression). The general format of the NEXT statement was NEXT (numeric variable). The numeric variable used as a counter in the FOR command (on the left side of the equal sign) was the same numeric variable used in the NEXT statement. An example of these commands was:

```
50 FOR X9 = 1 TO G5 STEP 3
51
52
53 NEXT X9
```

Subroutine Commands. In the ITF: BASIC Terminal User's Guide, it was indicated that subroutines were written for programs using certain BASIC statements. Subroutines were program segments that represented programming steps used frequently or needed more than one time. A subroutine was composed of a group of program statements separate from the main program and executed by altering the sequence of the main program and skipping to the subroutine. The GOSUB command and the RETURN command provided what was called the linkage to and back from the subroutine.[16]

The GOSUB command was used to transfer control or skip to the subroutine. The transfer of control went to a specified statement number that was the first statement number of the subroutine to be executed. The general format was GOSUB (statement number).

The RETURN statement was used with the GOSUB statement. It caused program control to be transferred back to or returned to the next logically executable statement following the GOSUB command. The general format was RETURN (character string). The character string was a comment or message which was optional. It did not affect the execution of a program and only appeared when the program was listed. An example of the GOSUB command and the RETURN command was:

---

[16]ITF: BASIC Terminal User's Guide, pp. 51-52.

```
500  GOSUB 710
510
520
 .
 .
710
720
730
740
750  RETURN END OF SUBROUTINE FOR RANDOM NUMBERS
```

The BASIC statements numbered 710 through 750 were the subroutine statements. The statement number 710 following the GOSUB command caused the transfer of control to the subroutine. When the subroutine statements had been executed, the RETURN command caused a transfer of control back to the BASIC statement numbered 510. Some programs required subroutines that skipped to other subroutines. In this case more than one GOSUB command was allowed before the RETURN.

Function Command. A function was a named arithmetic expression which computed a single numeric value from an argument. The argument of a function was either an arithmetic variable or an expression. It represented a numerical value on which the arithmetic operation specified in the definition of the function was performed. Two types of functions existed in the BASIC language: the Intrinsic functions and the User functions.

The Intrinsic functions were functions supplied by the BASIC language. There were twenty-four of these functions

defined in the ITF: BASIC Terminal User's Guide. The Intrinsic functions consisted, for example, of functions that computed the sine of x radians named SIN, cosine of x radians named COS, square root of x named SQR and logarithm of x named LOG.[17]

The second type of function, the User function, was written by the programmer. A User function was named and defined by the BASIC DEF statement. The name was a single alphabetic character preceded by the letters FN. Examples of function names were FNC or FNR. An example of the DEF statement to define and name a function was:

```
DEF  FNC(X) = X ** 3 + 45.89 * X
```

The numeric variable X enclosed in parentheses after the function name FNC was called the dummy variable. The dummy variable was a simple numeric variable. The function performed the defined calculation on the argument value substituted for this dummy variable. After a function was defined, it could be used anywhere in the program. This gave the programmer the advantage of performing the same calculation on many different values without duplicate coding.

Array Commands. An ARRAY was a table of data and all data within an array were of the same type. In BASIC, there were two kinds of arrays, numeric and character. Numeric

---

[17]Ibid., p. 107.

arrays contained or included only numeric values and character arrays contained or included only character values.

If an individual data item or member of the array was in a program statement, the BASIC system automatically set up the array at this time with ten members. This was called implicit declaration of an array. Explicit definition of an array was the array defined with the use of the DIM statement, the number of data items in the array being specified in the statement. This was necessary only if the array was to contain more than ten members. An example of an explicitly defined array was DIM T (25). This set up storage space for an array named T and twenty-five data items for that array. The number, 25, in parentheses was called the bounds of the array.

To refer to an individual data item in any array, the array name was given followed by the location of the data item in parentheses. For example, A(1) would refer to the first data item and A(3) would refer to the third data item or member of the array T. The number in parentheses giving the location was called the subscript. It was important to remember the difference between a subscript and the bounds of an array. The subscript was used to refer to a member of the array. It was any valid arithmetic expression such as a numeric constant or numeric variable. The bounds of an array defined the total number of data items or members of

an array. The bounds of the array was used only in conjunction with a DIM statement and was a positive integer.

Numeric arrays were either one or two dimensional arrays; character arrays were only one dimensional arrays. An example of a two dimensional array was T(5,2). This had ten members just like the one dimensional array defined as S(10). The difference was that the members of T were divided into two dimensions so that the programmer thought of five small arrays, each with two members. The following examples were how each member of the arrays, T and S, were referred to in a program.

T(1,1), T(2,1), T(3,1), T(4,1), T(5,1)
T(1,2), T(2,2), T(3,2), T(4,2), T(5,2)
S(1), S(2), S(3), S(4), S(5), S(6), S(7), S(8), S(9), S(10)

If the bounds of the array was two dimensional, two subscripts were used to refer to a member.

Numeric arrays were named by a single alphabetic character. The BASIC system initially set all data items in a numeric array to zero, according to the ITF: BASIC Terminal User's Guide. Character arrays were named by a single alphabetic character followed by a $ sign. The BASIC system initially set all character array data items to blanks and each member was automatically set as eighteen characters in length. Character arrays were not used in MAT array statements.[18]

[18]Ibid., pp. 32-34.

Arrays were given values through the use of the LET statement, the READ statement, the INPUT statement, or the MAT statements. MAT statements could only be used, though, to supply values for numeric arrays. When supplying input values for arrays with the READ or INPUT statements, every array member assigned a value had to appear in the statement. An example of the INPUT statement to assign values to the array defined DIM T(5) was coded as INPUT T(1), T(2), T(3), T(4), T(5).

Another way of supplying values to an array was to use a program loop, the FOR NEXT statements, and the READ statement. It would be coded as:

```
FOR N = 1 to 5
READ T(N)
NEXT N
DATA 89, 54, 3, 21, 77
```

The subscript, N, for the array, T, in the READ statement was used to specify every member of the array so that the location of every member was used during the execution of the program loop. An example of program statements that supplied values for a two dimensional array named L was coded as:

```
DIM L(5,7)
FOR J = 1 TO 5
FOR N = 1 TO 7
INPUT L(J,N)
NEXT N
NEXT J
```

The BASIC language provided for array matrix operations with a group of program statements called MAT statements. These MAT statements performed matrix assignment operations and matrix mathematical operations. Only numeric arrays that had been previously defined either explicitly or implicitly were used in the MAT statements. In several of the MAT statements, it was possible to define new bounds for the array; this was called redimensioning an array. All MAT statements used the word, MAT, and the array(s) name. Individual array member names were not used in MAT statements. The BASIC system provided the following matrix assignment statements for the input and output of arrays.

The MAT GET statement read a file of numeric data into the array named in the statement. An example of this statement was: MAT GET 'IF', X(30). The MAT INPUT statement assigned values from the terminal during execution to members of a numeric array. An example of this statement coded was: MAT INPUT Y. The MAT READ statement read numeric data from the DATA statement into a numeric array named in the MAT READ statement. It was coded as: MAT READ Z(20).

The MAT PRINT statement printed each member of an array at the terminal. The array was printed row by row; the first row of each array began at the start of a new line. An example of the statement coded was: MAT PRINT Z. The MAT PRINT USING statement printed at the terminal an array

in the format of the associated IMAGE statement. An example of this statement coded was: MAT PRINT USING 50, X. The MAT PUT statement created or added the values of a numeric array to an output file. It was coded as: MAT PUT 'VFL', X.

The following were MAT statements that performed other matrix assignment operations and mathematical operations for arrays. An example of each coded follows the explanation of the statement. The MAT Simple Assignment statement assigned the members of one array to another array. MAT X = Y. The MAT Addition or Subtraction Assignment statement assigned either the sum or the difference of the members of two arrays to the members of a third array. MAT X = Y(+ or -) Z. The MAT CON function assignment statement assigned the value, one, to all members of an array and also provided the option of redimensioning the array. MAT X = CON (N,N).

The MAT IDN function assignment statement changed an array to its identity matrix. MAT X = IDN (N,N). New bounds could be specified. The MAT Inversion assignment statement assigned the mathematical matrix inverse of one array to another array. MAT X = INV (Y). The MAT Multiplication Assignment statement provided the mathematical matrix multiplication of two arrays. The product was assigned to a third array. MAT X = Y * Z. The MAT Transpose assignment statement placed the matrix transpose of one array into another array. MAT X = TRN (Y). The MAT ZER function assignment statement assigned the value, zero, to

all members of an array and it also provided the option of specifying new bounds for the array. MAT X = ZER (N,N).

File Commands. A file was a group of related data that was treated as a unit. For example, data collected about the amount of inventory sold over a period of time might form a series of related data which could be called a file. The BASIC language system provided a method of storing this data so that it could be used in the program currently being processed or in any later program. These files were retained in what was called the BASIC System Library. A library was a part of a computer storage unit. Files were initially created, using the PUT statement. The GET statement was then used after the file was created when a program needed to use the data in the file.

In the ITF: BASIC Terminal User's Guide, it was pointed out that whenever a file was created, it must be named. This was accomplished by including the file name enclosed in single or double quotes in the PUT statement. For example, the BASIC statement, PUT 'AFL', X, Y, Z, would create an output file, name it AFL, and place the values X, Y, and Z in it. File names could be any length, but certain restrictions existed for the first three characters of the name; such as, it could not contain a period, a comma, or a semicolon. These restrictions existed because the BASIC system recognized file names by the first three characters only. Longer names were truncated and shorter file names

were filled with blanks. It was important then that the first three characters be unique and to avoid duplication, it was best not to exceed three characters.[19]

The files created were automatically retained in the BASIC system library storage area of the computer. Once a file had been created, it could be used as input to the same program or it could be used as input to some other program by the use of the GET statement.

When using the file after its creation, it was essential to know how many items it contained or that the file had what was called an "end of file indicator" as the last data item. When using the file later, tests could be made for the last data item or the "end of file indicator."

Once files had been created, they had to be activated or opened before they could be used. This was done by the BASIC system automatically the first time the file name appeared in a PUT statement or in a GET statement, and the file was deactivated or closed by the BASIC system after the execution of the program. If a file was created in a program and then was needed in the same program later, it had to be closed with a BASIC program statement called the CLOSE statement before it could be reopened. The RESET statement was another BASIC statement used in conjunction with files that was important. The RESET statement provided

---

[19]Ibid., pp. 45-46.

the capability of repositioning the file so that the first data item was available again.

The following summary of the File Commands, PUT, GET, CLOSE, and RESET that were used with a file was given in the ITF: BASIC Terminal User's Guide. The GET statement performed the function of reading an input file and assigning to the variables named in the GET statement the values of the data items in the file. The general format was GET (file name, variable, variable). An example of the statement coded was GET 'AFL', A, B, C, D. The PUT statement performed the function of placing in a file the values listed in the PUT statement. The general format was PUT (file name, variable, variable). The statement would be coded as PUT 'AFL', W, X, Y, Z. The RESET statement performed the repositioning of a file to the beginning data item in that file. The general format was RESET (file name). An example of the statement coded was RESET 'AFL'. The CLOSE statement provided for deactivating either input or output files. The general format was CLOSE (file name) and an example of using it was CLOSE 'AFL'.[20]

Miscellaneous Commands. Every program written in BASIC was concluded with an END statement. The END statement indicated the logical end of a program and was the last line of the program with the highest line number. The END

---

[20]Ibid., pp. 46-49.

statement terminated the computer execution of a program. The general format was END.

The STOP command terminated the running or execution of a program. The only difference between the STOP command and the END command was that more than one STOP statement was allowed in a program and it could be placed anywhere within the program. The STOP statement was used most with subroutines. The general format of the STOP command was STOP (character constant). The character constant did not affect the execution and only appeared when the program was listed. An example of the STOP statement with a character constant was:

STOP THIS IS END OF THE RANDOM NUMBER ROUTINE.

The REM statement provided a method for the programmer to use comments in a program. These explanatory remarks gave directions for using the program. They also identified the different sections or specified other very needed information. This was known as documenting a program and it was a very valuable aid to both the programmer and future users of the program. The REM statement was not part of the execution or running of a program; it only appeared when the program was listed at the terminal. The general format was REM (character constant). Examples of the REM statement were:

```
REM THIS IS A PROGRAM TO BUILD A MODEL
REM THIS ROUTINE COMPUTES THE INTEREST ON A BOND
```

The PAUSE statement halted program execution and a message was printed at the terminal giving the line number of the PAUSE command. The user of the program would resume execution by pressing the carrier return key. A character constant or message was optional and did not affect the execution of the program. The character constant was used only when the program was listed. The general format was PAUSE (character constant). An example of the PAUSE statement was:

```
PAUSE 'THE OPERATOR SHOULD ENTER VALUES AT THIS POINT'
```

Systems Commands. The BASIC programming language had a group of commands called System commands. These commands were used with the programming commands. They were not part of the computer program to solve the problem; they were BASIC language commands to use in running or executing the programs and in testing them.

The Systems commands were orders issued to the computer that did not cause the computer to perform any logical or mathematical operations. They did not have any problem-solving capability; instead they controlled the function of the computer. The Systems commands are briefly defined and discussed here and the manuals from IBM provided the information necessary to use the Systems commands. There

were several terms that were defined to clarify the Systems commands.

The word, mode, meant a method of operation of the computer. There were two possible modes in which the system operated: control mode and edit mode. The edit mode had a test submode for debugging purposes. Library was an area of storage where programs, text collections, and data files were saved. Text collection was an ordered sequence of logical lines created in the edit mode with the text option. Text collections were lines of data or documentation or parts of programs.

The ITF: BASIC Terminal User's Guide gave the following information about System commands. The AT command established a breakpoint which would interrupt program execution when that breakpoint was reached. It permitted the user to intervene immediately before execution of the specified command. The DELETE command deleted all or part of a program or collection of text in the edit mode. In the control mode, it deleted an entire program, text collection, or a file from the private library. The EDIT command placed the system into the edit mode. In the edit mode the user then created or modified programs or text collections. The END command ended the current mode that the computer was operating in. The GO command was used in the test submode to start or resume execution of a program.[21]

---

[21]Ibid., pp. 111-16.

In the ITF: BASIC Terminal User's Guide, it was indicated that each System command had specific uses for the different operating modes of the command. The LIST command, in the edit mode, caused the system to display all or part of a program or a text collection at the terminal. In the test submode, values of specified variables were displayed at the terminal. The LISTCAT command provided a listing of the names of every program, text collection, and file that the user had in his private library along with its type: BASIC, text, or file. The LOGOFF command ended a terminal session. The LOGON command initiated a terminal session. It was the first command used. The MERGE command caused part or all of one program or text collection to be inserted into another program or text collection. The NOTRACE command turned off one or more traces established by the TRACE command. It turned off the tracing mechanism for variables, branch points, files, and intrinsic functions.[22]

The OFF command turned off one or more breakpoints established by the AT command. It was used in the test submode. The RENAME command renamed a library member. It gave a new name to a program, text collection, or a file. The RENUM command caused the lines of a text collection or statements of a program to be renumbered. The RUN command caused a program to be executed or to be tested as it was

---

[22]Ibid., pp. 116-18.

executed. The RUN command executed the current program in short or long form arithmetic, according to the ITF: BASIC Terminal User's Guide. The SAVE command caused the current program or text collection to be saved in the private library of the user. The TRACE command established traces for variables, branch points, files, and intrinsic functions, so the user was aware of program changes when they occurred. This command monitored program execution by keeping track of these changes.[23]

Summary of the Chapter

The use of computers in our society has increased rapidly. Smaller and less expensive computers were available and the fundamental knowledge of computers was required in education, industry, business, and government. New and different problems were solved by applying the capabilities of computer systems. A very significant and important extension of the computer was accomplished with the development of commercial time-sharing systems and the development of the BASIC programming language.

Time-sharing provided access to many users. One central computer was shared by many people and each had a set time to use the computer on a rotating basis. The user communicated with the computer by means of a remote terminal, a

---

[23]Ibid., pp. 119-21.

teletypewriter device.

The BASIC program language was developed on a time-sharing system located at Dartmouth College during the period from 1964-66 by two professors. Many versions of the language existed for different time-sharing systems. The BASIC version described in this chapter was for the IBM Time-Sharing System.

BASIC means Beginner's All-Purpose Symbolic Instruction Code. The language was developed for the users who did not want to be computer programmers. The BASIC program language was easy to understand and learn, yet sophisticated applications had been programmed in the BASIC language.

A computer program was a specific set of directions performed in a logical sequence that identified to the computer how to solve a problem. There were five steps for writing programs: (1) problem definition, (2) method of solution called the algorithm, (3) logical flow of algorithm steps shown with a flowchart, (4) coding the program, and (5) execution, testing, and documentation of the program. After the five steps were reviewed, the BASIC program language was studied.

A BASIC program was composed of a number of entries. Each entry consisted of a line number and a BASIC statement. The line number specified the order in which program entries were processed. The BASIC statements were made up of constants, variables, expressions, and messages combined

with the BASIC command key words.

The programmer had to understand the rules for the constants, variables, expressions, and BASIC command key words in order to use the BASIC language. He then proceeded to write the program. The BASIC command key words were easy to use in writing the program because they had inherent meaning from the English language. The name BASIC was all it implied for beginners, yet it was all-purpose in the types of applications possible for everyone.

CHAPTER VI

APPLICATION OF COMPUTER SIMULATION

This chapter demonstrated and illustrated the use of computer terminals and the BASIC programming language for computer simulations. The steps of a computer simulation explained in Chapter IV were used and the two processes of computer simulation, analysis of the system and design of the system, were followed. The flowchart of the design of the model of the system and the computer program written in the BASIC language for use on the IBM System/370 computer terminal for the simulation of the system were included.

The first process for computer simulation was the analysis of the system. It involved making the observations of the system to see how the system interacted or behaved, isolating the elements of the system, and formulating the logical rules governing the interaction. The following were the observations of the system made by the analyst.

1. Continental Rent-A-Car was a branch office of Ritter Rental Car Agency.
2. The manager of each branch of this agency decided his own policy on the size inventory of cars to keep in stock available for rental.

3. The home office of the Ritter Rental Car Agency levied a penalty charge of five dollars per day for each car not rented and a penalty charge of two dollars per day for each car demanded, but not available.
4. The home office of the Ritter Rental Car Agency gave a credit of thirteen dollars per day for each car that was rented by the branch office.
5. In the past, Continental Rent-A-Car had experienced a daily demand of cars from zero to five, and the manager had records showing the daily demand during the past year.
6. During the past year, Continental Rent-A-Car had zero cars demanded 5 percent of the time, one car demanded 10 percent of the time, two cars demanded 25 percent of the time, three cars demanded 30 percent of the time, four cars demanded 20 percent of the time, and five cars demanded 10 percent of the time.
7. The manager of Continental Rent-A-Car was interested in knowing the inventory level of cars that would maximize his profit, based on his previous car demand experience.

The hypothesis of this system, or how the system behaved, and a statement of the simulation problem and goal followed next in the analysis of the system. The problem was to

optimize the behavior of the system by the measure of inventory level of the system to achieve the goal. The goal of the system was to maximize the profit of the business operation. This problem was called a system optimization study. This system was classed as an open system and a man-made system. The model of the system was a probabilistic model because uncertainty existed in the system. The events, which were activities and behavioral changes of the objects of the system, were based on a discrete probability distribution. The goal of the system, maximum profit, was not a continuous one, but changed as certain specifications or events within the system occurred. When the event, car daily demand, was less than or equal to car stock level, the states of the profit of the system changed; when the event, car daily demand, was greater than car stock level, this changed the profit of the system to different states.

The analysis also included a separation of the events of interests of the system into exogenous and endogenous events. The exogenous events could not be explained; they were determined by a statistical probability distribution. They were outside the control of the system. The endogenous events were within the control of the system. The exogenous and endogenous events that occurred affected the way the system behaved and caused the system to assume different states.

The events of car daily demand were exogenous events. The following could occur: the daily demand for cars could be zero, one, two, three, four, or five. The endogenous events were the six possible levels of car stock: zero, one, two, three, four, or five.

Next in the analysis, the elements of interest of the system were isolated. The objects of the system and the objects of the environment of the system were identified. The objects were called entities. The characteristics or properties of the objects were also defined. These characteristics were referred to as attributes of the entities. The manager of Continental Rent-A-Car was an entity of the system. This entity was characterized with the following attributes: (1) expected profit, and (2) car stock. Car was an entity of the system and possessed the following attributes: car units stocked, car units demanded, and car units rented.

Profit was the third entity of the system and possessed the following three attributes: rental dollar, overstock dollar, and overdemand dollar. Rental dollar was the credit given the branch office by the home office for every car rented. Overstock dollar was the penalty charge levied by the home office for each car that was not rented per day and overdemand dollar was the penalty charge levied for each car not available, but demanded per day. These three attributes

were called parameters because they were fixed values and did not change because of the events in the system. The value for the rental dollar was thirteen, the overstock dollar value was fixed at five, and the overdemand dollar was set at two.

The fourth entity was identified as customer who had the attribute, car demand. This entity was part of the environment of the system because the demand for a car by a customer was not controlled by Continental Rent-A-Car.

The definition of the relationships of a system provided the structure of a system and was the final step in the analysis. There was a relationship in this system between the attributes; car stock and car demand, and the entity, profit. If the attribute, car stock, was zero or greater, and greater than or equal to the attribute, car demand; it changed the state of the entity, profit, of the system. If the attribute, car stock, was less than the attribute, car demand; the entity, profit, of the system state was changed.

The second process for a computer simulation was the design of the model of the system. It involved stating the inputs, the algorithm, and the output of the model. The input provided the necessary information to solve the model of the system. The algorithm gave the method to solve the simulation problem; the logic and steps used in the method were shown in the pictorial form of a flowchart. The output

was the information that stated the solution of the model of the system. The algorithm was the most important part of the design and was used in coding the program for the simulation on the computer.

The necessary inputs for this model were values for the events of car stock level of zero, one, two, three, four, and five. The events of car daily demand were also inputs into the model. These event values were based on a discrete probability distribution. The event values are shown in Figure 4, Car Demand Event Values.

Possible Event	Probability
0	0.05
1	0.10
2	0.25
3	0.30
4	0.20
5	0.10

Figure 4
Car Demand Event Values.

The output of the model of the system was the simulation solution of the expected maximum profit and the car stock necessary to obtain this profit.

The algorithm was explained below verbally and with mathematical equations. For each possible value, zero through five, of stock level event, combine a different value, zero through five, of daily demand event and compute

the expected profit. If the value for the event of stock level was greater than or equal to the value for the event of daily demand, compute a value for profit equal to the rental dollar (thirteen) times the value of car units demanded. Take from this value, the value of the over-stock dollar (five) times the value of car units stocked in excess of the value of car units demanded.

If the value for the event of stock level was less than the value for the event of daily demand, compute a value for profit equal to the rental dollar (thirteen) times the value of car units stocked. Take from this value, the value of the overdemand dollar (two) times the value of car units demanded in excess of the value of car units stocked.

The next step in the algorithm was the calculation of the expected profit for every possible car stock level event. After all expected profits were figured, the maximum expected profit was determined.

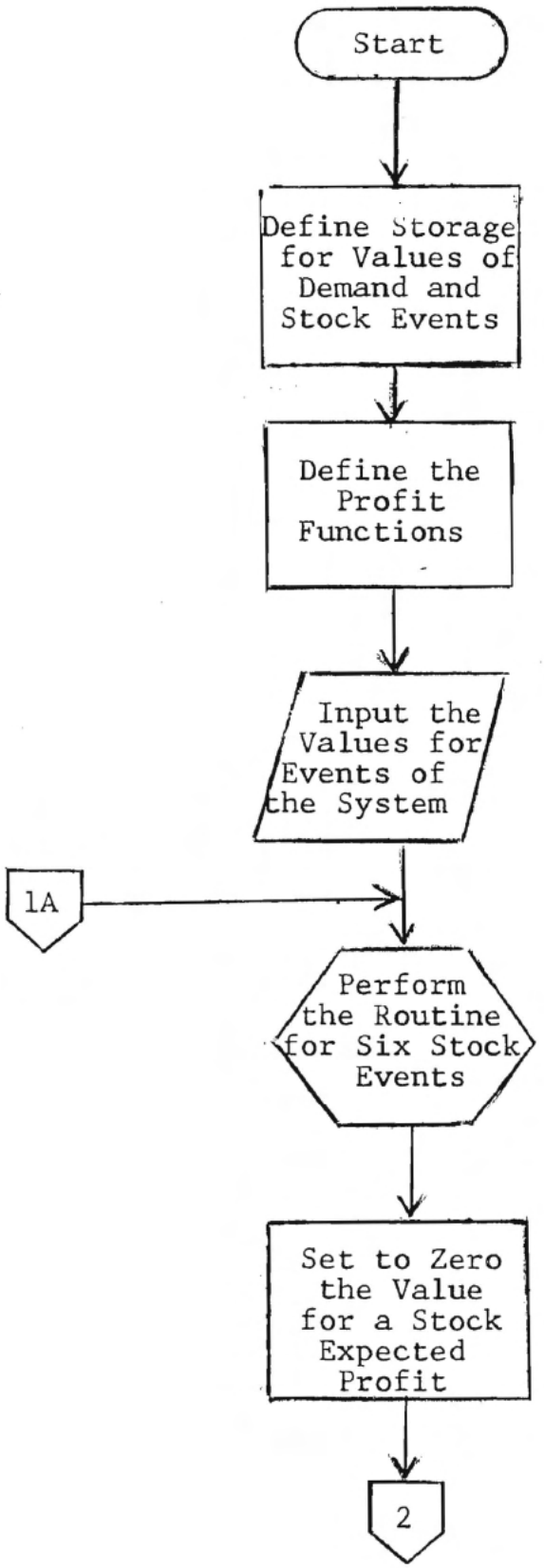
"S" was the symbol for stock level event value, "D" the symbol for daily demand event value, and "P" the symbol for profit. The mathematical equation for the profit when $S \geq D$ is:

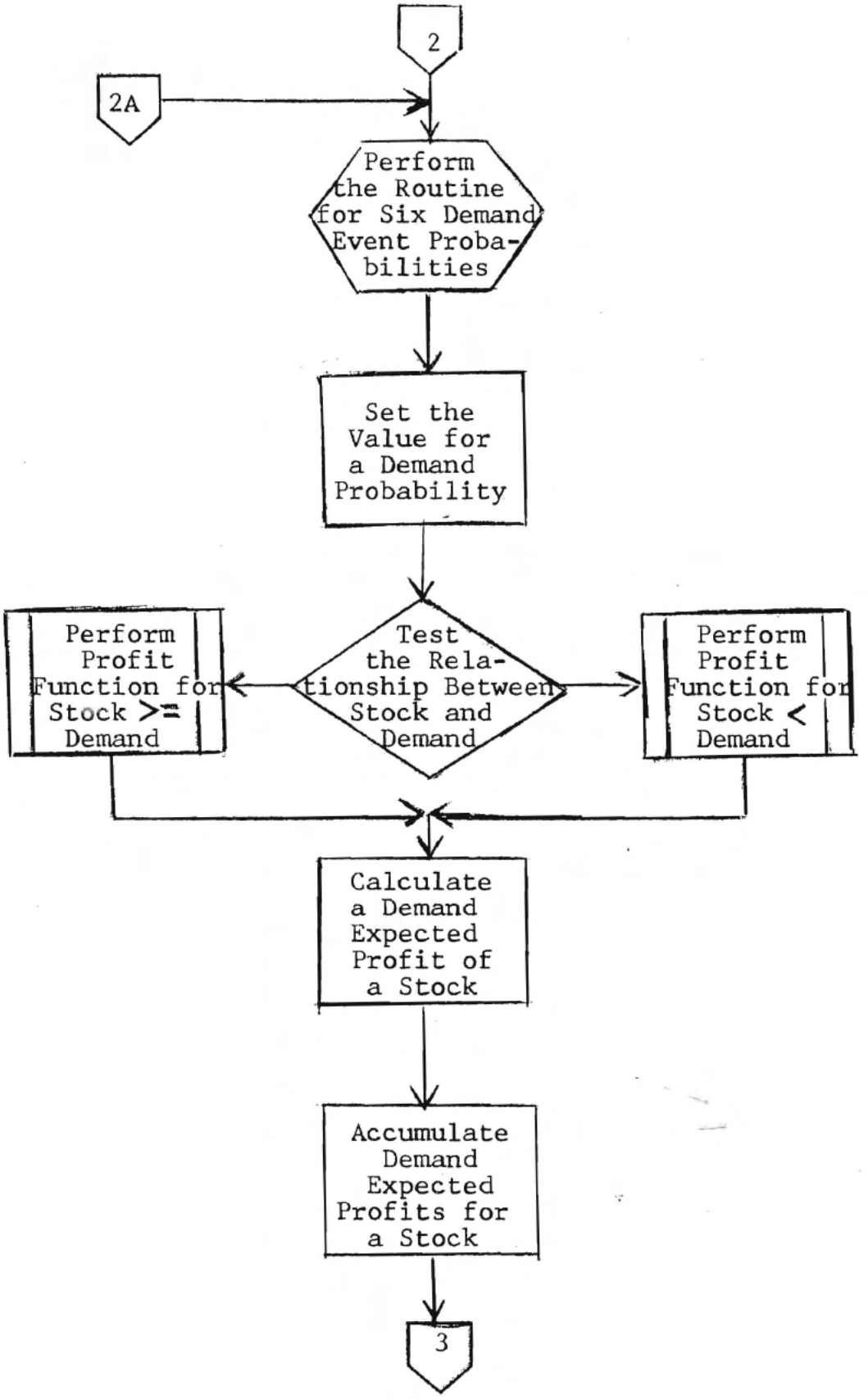
$$P = (D \times 13.00) - ((S - D) \times 5.00)$$

The mathematical equation for the profit when $S < D$ is:

$$P = (S \times 13.00) - ((D - S) \times 2.00)$$

The flowchart used in the design of the model of the system is shown in Figure 5, Flowchart of Model, pages 92-94.


Start
Define Storage for Values of Demand and Stock Events
Define the Profit Functions
Input the Values for Events of the System
1A
Perform the Routine for Six Stock Events
Set to Zero the Value for a Stock Expected Profit
2


2
2A
Perform the Routine for Six Demand Event Probabilities
Set the Value for a Demand Probability
Test the Relationship Between Stock and Demand
Perform Profit Function for Stock >= Demand
Perform Profit Function for Stock < Demand
Calculate a Demand Expected Profit of a Stock
Accumulate Demand Expected Profits for a Stock
3

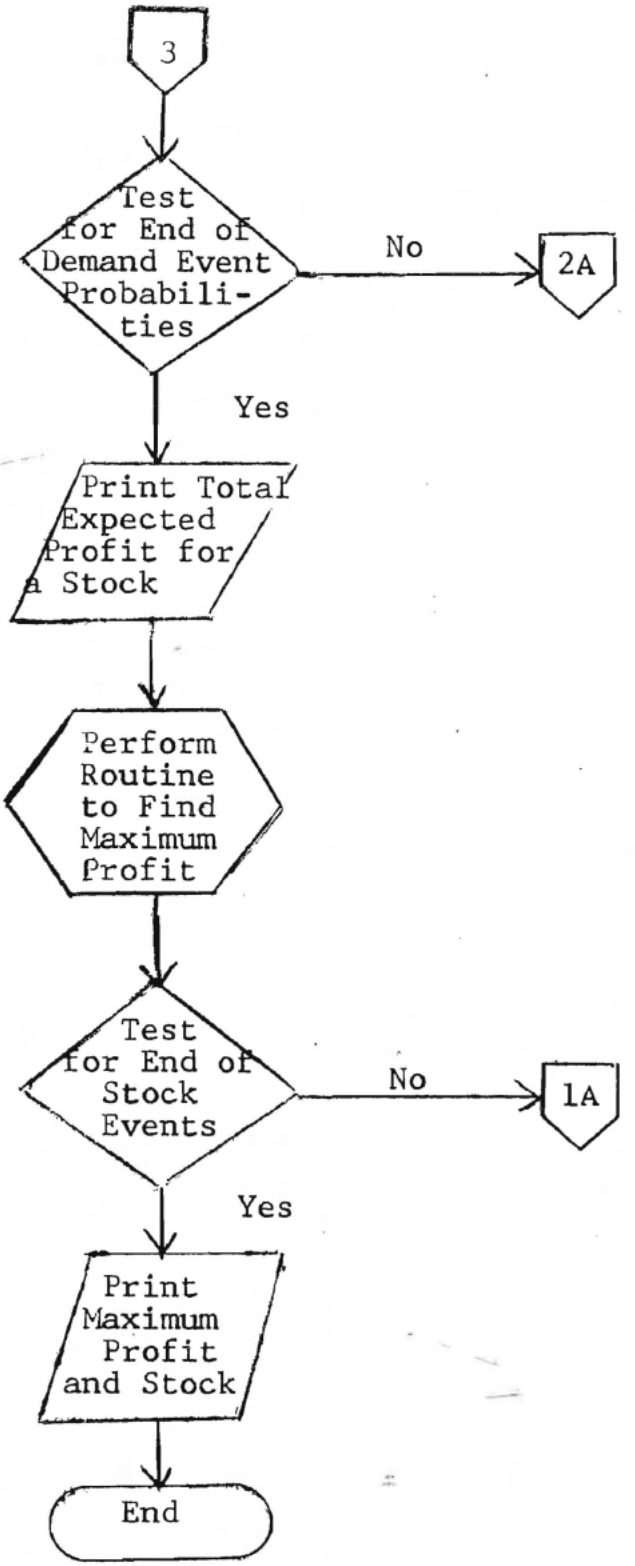


Figure 5
Flowchart of Model.

The last step in the design was coding the computer program for the model of the system. The computer program was written in the BASIC language explained in Chapter V. It showed examples of using various BASIC commands and statements discussed in that chapter. The program also demonstrated the use of functions and the programming techniques of "looping" and subroutines. The REM command was used more often than necessary, but it was done as a guide for the reader in understanding the program and for documentation purposes. Figure 6, Computer Simulation Program, illustrates the program.

```
 10   REM SIMULATION PROBLEM TO FIND MAXIMUM PROFIT
 20   REM FOR CONTINENTAL RENT A CAR
 30   REM DEFINE STORAGE AREA FOR SAVING THE PROBABILITY
 40   REM DISTRIBUTION FOR CAR DAILY DEMAND
 50   DIM P(10)
 60   REM PROFIT FUNCTIONS
 70   DEF FNA(X) = ((13.00 * X) - (5.00 * Y))
 80   DEF FNB(X) = ((13.00 * X) - (2.00 * Y))
 90   REM READ THE VALUES FOR PROBABILITIES OF DEMAND EVENTS
100   READ P(1), P(2), P(3), P(4), P(5), P(6)
110   DATA 0.05, 0.10, 0.25, 0.30, 0.20, 0.10
120   REM C IS VARIABLE USED TO STORE COMPUTED EXPECTED PROFIT
130   REM FOR EVERY CAR DEMAND OF A CAR STOCK
140   LET C = 0.0
```

```
150  REM THIS SETS A LOOP FOR THE SIX STOCK LEVEL EVENTS
160  FOR L = 1 TO 6  STEP 1
170  REM INITIALIZE EXPECTED PROFIT FOR EACH CAR STOCK
180  LET E = 0.0
190  REM SET UNITS OF CAR STOCK TO RANGE FROM ZERO TO FIVE
200  LET S = L - 1
210  REM THIS SETS A LOOP FOR SIX DEMAND EVENT PROBABILITIES
220  FOR T = 1 TO 6
230  REM SET UNITS OF CAR DEMAND TO RANGE ZERO TO FIVE
240  LET D = T - 1
250  REM STORE PROBABILITY FOR DAILY DEMAND IN VARIABLE P2
260  LET P2 = P(T)
270  REM TEST RELATIONSHIP BETWEEN EVENTS OF STOCK LEVEL AND
280  REM DAILY DEMAND
290  IF S<D THEN 370
300  REM COMPUTE PROFIT FOR RELATIONSHIP OF CAR STOCK GREATER
310  REM THAN OR EQUAL TO CAR DEMAND
320  LET Y = S - D
330  LET C5 = FNA(D)
340  GOTO 410
350  REM COMPUTE PROFIT FOR RELATIONSHIP OF CAR STOCK LESS THAN
360  REM CAR DEMAND
370  LET Y = D - S
380  LET C5 = FNB(S)
390  REM COMPUTE EXPECTED PROFIT USING DEMAND PROBABILITY
400  REM VALUE FOR A DAILY DEMAND
```

```
410  LET C = C5* P2
420  REM ACCUMULATE DAILY DEMAND EXPECTED PROFITS
430  REM FOR A CAR STOCK
440  LET E = E + C
450  REM END OF LOOP FOR DEMAND EVENT PROBABILITIES
460  NEXT T
470  REM PRINT THE TOTAL EXPECTED PROFIT FOR A CAR STOCK
480  : CAR STOCK ## EXPECTED PROFIT $###.##
490  PRINT USING 480, S, E
500  REM ALTER SEQUENCE OF EXECUTION FOR MAXIMUM PROFIT ROUTINE
510  GOSUB 590
520  REM END OF LOOP FOR STOCK LEVEL EVENTS
530  NEXT L
540  REM OUTPUT SIMULATION SOLUTION
550  PRINT 'MAXIMUM', 'EXPECTED PROFIT', R
560  PRINT 'CAR STOCK', Q
570  STOP
580  REM BEGINNING OF ROUTINE TO FIND MAXIMUM EXPECTED PROFIT
590  IF L = 1  GOTO  640
600  REM TEST RELATIONSHIP BETWEEN PREVIOUS AND CURRENT
610  REM EXPECTED PROFIT FOR MAXIMUM VALUE
620  IF E<R  THEN  660
630  REM STORE MAXIMUM EXPECTED PROFIT AND CAR STOCK
640  LET R = E
650  LET Q = S
660  RETURN
670  END
```

Figure 6
Computer Simulation Program.

The application of a computer simulation for a computer terminal and the use of the BASIC programming language to build a model of a system was illustrated in this chapter. The application for the computer simulation began with the selection of the business system, Continental-Rent-A-Car, that had a problem to be solved in the operation of the business. The analyst then followed the processes of analysis of the system and design of the system for a computer simulation to find a solution for the problem.

The problem for this simulation was to optimize the behavior of the size of the inventory level of cars to achieve the goal of the system, maximum profit. Uncertainty in car daily demand events of the system was characterized by a discrete probability distribution. The result of the design of the system was thus a probabilistic model. Entities (objects), attributes (characteristics), relationships, exogenous and endogenous events of the system, and the environment of the system were isolated and defined. Manager, customer, profit, and car were all entities in this system and performed events or changed behavior so that the state of the system was changed.

The design of the model of the system was shown with a flowchart and the computer simulation program was coded as final steps in the computer simulation exercise.

CHAPTER VII

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary

The use of computer terminals in the business world was growing in importance. Computers and computer terminals had given the business executive easy and quick access to all types of information in the operation of business and had provided analysis techniques for business operations never before possible. As businesses grew in size and complexity, analysis techniques for making the best decisions were needed. Simulation has helped fulfill this need.

Simulations of systems were representations of systems in the real world. These representations took the form of models. With these models proposed alternatives for solutions of problems were studied and evaluated without affecting the real system. These simulations then aided the business executives in the decision-making process.

The use of the methodology of simulation in the college business classroom had been scarce, though it would have been beneficial for business students to be exposed to this type of systems analysis technique for the business problem situations they studied. Simulation also would have

prepared the student for his future career in the business environment. The uses of data processing, computers, and computer terminals had not received enough emphasis in the college education of the business student.

This research was conducted to determine if computer terminals and the BASIC computer language could be successfully used to build models of systems used for practical business applications of simulation.

The purpose of this research study was directed toward assembling information about simulation, the BASIC computer language, and the computer terminals. From this information an application of simulation for use by the college business student was developed.

The documentary-descriptive method of research was used for this research. A study was conducted of the fundamental thoughts of simulation and the process of computer simulation. It was necessary to understand and to learn the concepts of systems and models and how they were applied in simulation. The BASIC computer language and computer terminal processing was researched. Finally, a business system with a problem was selected and the analysis and the design processes for simulation were used to build a model of the system for the computer to solve the business operation problem.

Conclusions

There existed a growing interest and importance in the systems concept, in the study of the business environment, and in the use of computers and computer terminals in the business world. The systems concept and the technique of systems analysis were more valuable and extensive with the aid and capabilities available via computers and programming.

The systems concept forced the business systems analyst to think of a whole rather than a part. This was essential in the study of the complex system of businesses today. The systems concept made the business systems analyst consider and investigate the interaction of all parts of a business system and its environment. It lessened the chances of overlooking any pertinent influences that affect the behavior of systems and that affect the problem situations associated with business system events. The systems concept was important for making the right decision.

The use of time-sharing computer systems and computer terminals in the business environment was popular as well as important. The decision-making process in the operation of businesses had been improved with the analysis techniques available through the computer. Time-sharing was the shared use of one computer by many different users. The users had access to the facilities of the computer via a computer terminal. To communicate, the user had to know a software language of the computer.

The BASIC computer language was developed specifically for the computer terminals and for time-sharing computer systems. The BASIC computer language was also created for people not involved in data processing. It was oriented towards those who only wanted the capabilities of the computer to assist in an activity of their occupation, education, or research. The BASIC computer language code and programming techniques were designed to be easy to understand and to learn by anyone outside the field of computer programming. The computer terminals and the use of BASIC computer language provided the capabilities necessary for the activities and the applications of systems analysis and simulation.

Systems analysis was important in the business world of today. Simulation was used in the work of the systems analyst and computer simulation required the two processes of analysis of system and design of system. Simulation was a symbolic representation of a part of reality; a simulation of a system was a parallelism of a system in the real world.

A system was a set of objects; the objects were defined by certain characteristics. These objects were tied together with definable relationships and these relationships bind a system together. A system object was called an entity in simulation. The entity characteristics were called attributes and the attributes had either different values or remained the same, when changes occurred in the system state. A

system state was a point in time when the values of the system were determined or evaluated.

The symbolic representation of a system was called a model. Models were formed of verbal descriptions, pictorial designs, and mathematical equations. When a model was constructed from a mathematical equation which was solved on a computer, this was known as a computer model.

The purpose of systems analysis and of computer simulation was to study how systems behave and to evaluate proposed solutions for problems in systems.

Writing a computer program was not difficult if the fundamental steps were followed. The technique of writing computer programs made the programmer think about the whole as well as its parts just as it was essential in the technique of systems analysis. The programming steps were: (1) definition of the problem by specifying data for input, deciding upon procedures to obtain a solution, and specifying the data for output; (2) creating a method for the solution called an algorithm; (3) developing a logical flow of steps for the algorithm shown with the flowchart; (4) writing the program in a computer language called coding; and (5) running or executing the program.

The BASIC computer program language provided the essential concepts and instructions for the operation of the computer so the second process of computer simulation, design of system, was accomplished. Design of system was

the process of building the model of the system for the computer.

A BASIC computer program was a number of entries that were steps for solving a problem, and in a computer simulation, the entries were steps for solving the problem of a system or steps for studying the behavior of a system. The BASIC computer entry contained constants, variables, and/or expressions. Each of these had a purpose in the definition of a model of a system. Constants were parts of memory or values that did not change; they were used for the entity parameters of a system. Variables were the values or definable parts of memory whose value changed, and they were used for the entity attributes or system variables. The expressions were used to simulate the relationships between the entities of the system because expressions contained the constants (parameters of entities), variables (attributes of entities), and logical and/or mathematical operators that bound the system together.

The BASIC computer language commands were instructions to the computer to perform certain computer operations. The computer operations performed the events in the system that led to a solution of a problem of a system or gave insight into the behavior of a system. The BASIC input-assignment commands were used to define the values of entity attribute parameters and entity attribute variables and were also used to change the value of the system variables. The

BASIC output commands were used to give the solutions of the computer simulation of a system. The transfer commands were used to make decisions for the system and caused only certain events to occur. The function command was necessary for defining events and evaluating changes of the state of the system. The subroutine commands performed the same event several times that caused a change in the state of the system within a simulation. The system type of commands made possible the testing of the model of the system or validating it and also started and stopped the events of the computer simulation.

Recommendations

The following recommendations were made based on observations of the researcher, the related literature reviewed for this problem and the research conducted for this study.

Only one application of computer simulation was presented; therefore, this research problem needs further investigation. Other simple business systems situations should be selected and analysis and design of system applied. Models should be built for these with the BASIC language program and solved via the use of the computer terminals. It was further recommended that more complex computer simulations of systems using the BASIC language and computer terminals be tried. These should involve the analysis and design of systems based on uncertainty with continuous probability distributions and

a system that was based on the concept of queuing.

It was recommended that care should be exercised that the first applications of simulation used by business students not involve complex systems or systems with difficult probability concepts. The applications should begin with systems that have certainty and then progress to systems with uncertainty.

An application for a simulation of a business system should be selected, analyzed, and designed using the BASIC computer language. The model should then be executed on the computer terminal. The same simulation application should then be analyzed and designed with a special computer simulation language such as SIMSCRIPT, GASP, DYNAMO, or GPSS and the model executed on the computer. The two computer simulations should then be compared for ease, understanding, and efficiency.

College courses in simulation and computers should be offered. The curriculum should include: (1) an introductory course in systems analysis for business students at the undergraduate level and this course should include the fundamentals and concepts of computer simulations, (2) an introductory course in computers and computer terminal processing offered to the undergraduate business students, (3) an introductory course, including both systems analysis and computer processing should be considered and offered to

the students at the undergraduate level, and (4) a course in computer simulations should be offered at the graduate level and this course should include the study of the special software simulation computer languages such as SIMSCRIPT, GPSS, DYNAMO, and GASP. All undergraduate business students should be required to take the introductory course in systems analysis.

BIBLIOGRAPHY

BIBLIOGRAPHY

Dascher, Paul E. "EDP in the Elementary Accounting Course." Collegiate News and Views, XXVI (Winter, 1972-73), 11-12.

Gupta, Roger. Electronic Information Processing. New York: The Macmillan Company, 1971.

IBM: GPSS IV Manual. New York: International Business Machines Corporation Programming Publications, 1971.

ITF: BASIC Terminal User's Guide. New York: International Business Machines Corporation Programming Publications, 1970.

Laughlin, William M., Jr. "Computer Related Instruction-Developments in Economics." Collegiate News and Views, XXV (Winter, 1971), 3-6.

Loschetter, Richard. "The Computer in the Business Classroom." Collegiate News and Views, XXV (Winter, 1971), 19.

McGuire, Joseph W. "The Collegiate Business School Today." Collegiate News and Views, XXV (Spring, 1972), 1-5.

McMillan, Claude, and Gonzalez, Richard F. Systems Analysis, A Computer Approach to Decision Models. Homewood, Ill.: Richard D. Irwin, Inc., 1968.

Perritt, Roscoe D. "Innovations in an Elementary Accounting Program." Collegiate News and Views, XXVI (Fall, 1972), 13-15.

Sass, C. Joseph. BASIC Programming for Business. Boston: Allyn and Bacon, Inc., 1972.

Sutherland, Angela. "More Problem Solving in Business Math With the Computer." Journal of Business Education, XLVI (March, 1971), 262.

Wyman, Forrest Paul. Simulation Modeling: A Guide to Using SIMSCRIPT. New York: John Wiley and Sons, Inc., 1970.