MAC TR-129

USING INTERACTIVE GRAPHICS IN SIMULATING THE HOSPITAL EMERGENCY ROOM

Richard W. Weissberg

This research was supported by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 2095 which was monitored by ONR Contract No. N00014-70-A-0362-0006

Supervision for this research was supported by the National Science Foundation, "Innovative Resource Planning in Urban Public Safety Systems," Grant No. GI-38004

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

FOREWORD

This document, adapted from a Master's thesis in Electrical Engineering, is being jointly published by two research activities at M.I.T., Project MAC and the "Innovative Resource Planning" Project.

Project MAC is an M.I.T. interdepartmental laboratory for computer research and development involving faculty and students from the Departments of Electrical Engineering and Mathematics and the Sloan School of Management. The name MAC is an acronym derived from several titles: man and computers, machine-aided cognition, and multiple-access computers. The broad goal of machine-aided cognition implies the development of new ways in which on-line use of computers can aid people in their creative work, whether it be research, engineering design, management, or education.

The research project, "Innovative Resource Planning in Urban Public Safety Systems," is a multidisciplinary activity involving faculty and students from the M.I.T. Schools of Engineering, Architecture and Urban Planning, and Management. The administrative home for the project is the M.I.T. Operations Research Center. The research focuses on three areas: 1) *evaluation criteria*, 2) *analytical tools*, and 3) *impacts upon traditional methods, standards, roles, and operating procedures*. The work reported in this document is associated primarily with category 2, in which a set of analytical and simulation models are developed that should be useful as planning, research, and management tools for urban public safety systems in many cities.

Supervision, computer time, and other expenditures associated with the work reported herein were supported through the above two research programs by the Advanced Research Projects Agency, Department of Defense (under Office of Naval Research Contract Number N00014-70-A-0362-0006) and the National Science Foundation (Grant GI-38004).

Richard C. Larson
Joseph C.R. Licklider

ABSTRACT

The hospital emergency room is a complex system having many interrelated factors contributing to its operation. The emergency room administrator has limited control over certain of these factors: numbers of beds, nurses, doctors, x-ray units, for example. Other factors such as patient arrival rates and demands made upon available resources are largely uncontrollable. One of the main problems facing the emergency room manager is to find a reasonable balance among the many factors over which one has control in the face of a range of values of the factors over which little control is possible.

A computer program has been designed which uses computer graphics and interaction with the user to create a flexible modeling environment for analysis of hospital emergency rooms. In projects involving analysis of public systems, it is especially important that close communication be maintained between the public administrator and the analyst. Tools of the type which concern the present research can make a significant contribution towards this end.

The emergency room was chosen as the basis for the research for two reasons: First, the author had been a member of a team which performed an analysis of the Cambridge Hospital emergency room in Cambridge, Massachusetts, and therefore was somewhat familiar with the emergency room system and factors relevant to its analysis. Second, the emergency room is a system which in many hospitals is rapidly approaching a crisis: like the medical care system as a whole, the emergency room is experiencing profound changes in the demands being made of it. Patient arrival rates are increasing at an exponential rate. For many, the emergency room has become the primary source of medical care. Thus the very role of the emergency room is becoming unclear. The rapid changes in volume and nature of demand being experienced by the emergency room suggest that time invested in analysis and planning of the system would be well spent.

The program, the Tool for Interactive Graphical Emergency Room Simulation (which, for ease of discussion, is referred to as TIGERS) is a simulation-based modeling environment which has been implemented on the PDP-10 computer of the Programming Technology Division of Project MAC at M.I.T. This first effort, although general in scope, is based upon the emergency room at Cambridge Hospital. A preliminary model based upon this emergency room has been implemented. Valuable feedback has been obtained from Dr. Peter Mogielnicki there, and it is expected that other doctors in the Boston area may soon try out the system as well. The main thrust of the research is being concentrated not on designing a highly accurate model of a particular emergency room, but rather on development of a tool which can be used for such a purpose.

The actual implementation of the simulation within TIGERS involves the design of a model and the translation of the model into data bases and events. The task is made somewhat easier in that TIGERS provides all major data bases and several utility subroutines. The graphics updating is automatic, and routines are provided which make

trivial the creation of light buttons for changing any relevant parameters.

The hardware upon which the present system is implemented is currently too expensive for practical application in most situations, but graphics technology is developing rapidly and is fast entering the realm of practicability for smaller installations. Both to the analyst and to the public administrator, the medium represents a potentially useful means of making simulation models more intuitive and easier to understand.

Acknowledgements

I would like to express my thanks to all of the people who aided and encouraged this work:

to Professor Richard C. Larson who supervised this thesis, for his continuous encouragement and wise advice throughout the period of research. He is responsible for inspiring my interest in urban service systems, and for this I am truly grateful.

to Dr. R. Peter Mogielnicki, who made it possible for me to spend time in the Cambridge Hospital emergency room, who provided valuable feedback from where it was most needed and who helped me to keep my feet on the ground.

to Professor J.C.R. Licklider, for his continued support and encouragement, and for creating an environment where research of the type reported here is possible.

to my co-workers in the Programming Technology Division who evolved the language MUDDLE and taught me how to use it. I would especially like to thank Ed Black who spent many long hours making his MUDDLE graphics compiler compatible with TIGERS, the program which is the subject of this thesis.

to Larry Markel, Steve Purks, and Jim Shields, my co-workers in the May 1972 study of the Cambridge Hospital emergency room. Material incorporated from that study was very valuable to the present work.

Richard Weissberg

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I: INTRODUCTION TO THE MEDICAL CARE PROBLEM

1.0 The Crisis in the Medical Care Sector

The health care industry has become the second largest industry in the United States. In 1970, Americans spent 67.2 billion dollars, 6.9 per cent of the Gross National Product, on their health. And the industry is fast growing: between 1966 and 1970, health expenditures increased at an average annual rate of 12.3 percent, compared to a growth rate of 6.3 percent for the economy as a whole. The Department of Health, Education, and Welfare has projected a figure of 105 billion dollars for fiscal year 1974 [1], and expenditures in the health enterprises in the United States are projected to reach between 156 and 189 billion dollars by the end of the current decade. Thus our society will be spending between 8 and 9.8 percent of the Gross National Product for health. If present trends continue, the health establishment may well be the nation's largest industry, in terms of manpower and expenditures, by 1980. [2]

Unfortunately, even our present vast expenditure of over $350 per capita annually does not ensure a high level of medical care in the United States. Rate of infant mortality is lower in twelve other industrial countries. Men in seventeen other countries live longer than Americans do, and women live longer in ten. There is a growing consensus in the United States that the medical care sector, while continually costing more, is not performing its functions well for all those who could benefit by them; a nearly ubiquitous view that there is a health care "crisis." Between 1960 and 1971, costs of hospital care climbed more than 50 percent, compared to a 31 percent increase in the consumer price index. A day in a hospital cost $35 in 1960 and $75 in 1971. [3] A typical company sponsored health insurance plan, including full hospitalization and surgical benefits for a family, cost $550 in 1971, more than double the amount of just five years before.

The symptoms of the crisis are all too clear -- skyrocketing expenses and inadequate care for a significant sector of the population -- but the cure is not so easily seen. Berki and Heston articulate the nature of the problem: "The multiplicity of proposals for the cure of this 'crisis' in terms of delivery, organization, financing, and control is evidence that dissatisfaction with the performance of the medical care sector has reached the status of a politicized social problem. As in other complex diseases, while there is agreement that something is drastically wrong, there is no consensus on either the diagnosis or the therapy." [4]

The fact that we spend such vast amounts on health care, and the fact that health care expenditures are rising at a rate far out of keeping with the rest of the economy, lead one to believe that we are not allocating our resources, i.e. our health dollars, as wisely as we might. Dr. David D. Rutstein expressed this belief in his book, *The Coming Revolution in Medicine*: "The hallmark of our present haphazard system of medical care is lack of efficient systematic allocation of resources to meet the public need. Implicit is a lack of planning which really reflects the pitiful inadequacy of research on the provision of medical care. Indeed our blind faith in the infallibility of the physician has made it seem in the past as if research on medical care were unnecessary." [5] Essentially what we have done has been to leave the allocation and distribution of resources almost exclusively to the marketplace. Apparently the marketplace has failed us in this case, as we find ourselves caught in a burgeoning spiral of ever increasing expense that buys little significant improvement to the system.

Our lack of planning is catching up to us. Even though our medical care system has perhaps never been run nearly as efficiently as it might have been, only in the last decade has the situation started to get out of hand, generating the so-called crisis that is now all too manifest. The problem was first recognized as early as 1933. In that year the classic Lee-Jones study, undertaken under the auspices of the Committee on the Cost of Medical Care (CCMC) pointed out that "The problem [the provision of good medical care] will not solve itself through the operation of undirected economic forces." [6] People, however, were not yet ready to listen.

Thirty-five years after the publication of the Lee-Jones study, the report of the National Advisory Commission on Health Manpower was issued. This commission was established because the problem-turned-crisis could no longer be overlooked. Costs were rising out of all reasonable proportion compared to the rest of the economy; there was insufficient manpower to meet the requirements of the system structured as it was. The commisson reported essentially the same problems as reported thirty-five years earlier. The report very clearly proclaimed a health crisis, pointing out:

> *The crisis, however, is not simply one of numbers.* It is true that substantially increased numbers of health manpower will be needed over time. But if additional personnel are employed in the present manner and within the present patterns and systems of care, they will not avert, or even perhaps alleviate, the crisis. *Unless we improve the system* through which health care is provided, care will continue to become less satisfactory, even though there are massive increases in cost and in numbers of health personnel. [7]

We are faced not with a shortage, but with a problem of misallocation.

1.1 Alleviating the Problem

It is not surprising that the concept of efficiency is the mainstay of medical reformers. More efficient operation would presumably provide more services for more people, limit the cost to third parties, and similarly limit the skyrocketing costs to patients, insurance companies, and the government. The government, which pays two fifths of the nation's medical bills, should be especially interested in improving efficiency, since it would retard inflation as well as costs. Doctors too would benefit from more efficient operation. "By increasing his productivity, he can see more patients, respond to growing patient criticism of the availability of doctors, increase his earnings, and help more people." [8]

Optimizing such an incredibly complex system as our whole medical care system is a problem of huge magnitude. Indeed, optimizing it is probably impossible, but even making significant improvement is an awesome task. Where do we begin? The problem really exists at several levels, and the task needs to be addressed at each one.

At the outermost or most "macro" level, we are faced with broad political and social questions. Who should pay for medical care? How should the overall system be structured? Can everyone have a family doctor, and if not, where does he go for his primary medical care? What about medical centers? How should they be organized? By neighborhood? By region? Should doctors be paid on a fee basis, or are there other structures of payment which might reduce unnecessary surgery and be more conducive to other "more efficient" treatment?

In the middle levels are questions which, although not as political or social, still concern broad issues that will affect the very nature of the medical care system. For example: should all hospitals be general, equipped and staffed to handle a wide range of needs; or should a network of specialized or semi-specialized centers be set up? How do we evaluate the utility of various possibilities?

Finally, at the most "micro" levels, we face such questions as how to run the facilities themselves. How do we staff the intensive care unit? How many doctors do we need? Where can paramedical personnel be used, and where is a doctor essential? Where can we use machines? How many beds do we need such that the probability of all being full is below a given level?

Thus we see that the decisions involved in planning a more efficient health care delivery system span a wide range of positions in the decision making hierarchy -- from the highest levels of government to administrators of individual medical centers. It is these people who are called upon to make the decisions which will hopefully bring our medical care system to an improved level of operational efficiency. As we stated earlier, their task is not easy.

1.2 Operations Research

A body of knowledge and analytical methods known as "operations research" is evolving, which is directly applicable to certain aspects of the medical care problem. At each level of the hierarchy, a basic concern of decision makers is the allocation of resources so as either to maximize the accomplishment of the stated objectives with given resources or to minimize the resource costs of achieving the given objectives. Operations Research (hereafter sometimes abbreviated O.R.) is concerned with realizing such allocation procedures. Dr. Rutstein discusses a bit of the history of O.R., and points out how O.R. is beginning to be applied in the health care sector:

> Historically, operations research was first used in the deployment of troops and *materiel* in World War II. The success of operations research in military logistics led to the extension of its use to industrial production problems in war industry, then to industrial problems in general, to business management, and to economic planning. Now it is being applied to public health and medical care. In a policy statement on operations research prepared for the World Health Organization with Professors Marcel-Paul Schutzenberger and Murray Eden, we pointed out that O.R. is useful in the kinds of major decisions that frequently face the medical administrator. They are as follows:
>
> 1. How does one assess the relative needs for, and the values of, alternative values that must draw upon limited resources in funds, material, and trained manpower?
>
> 2. How shall available resources be best allocated and applied once a decision on priorities has been made? [9]

1.3 Our Misuse of Available Resources

Although it is only a beginning, several studies have been made or are now underway investigating the problems of the medical care sector, at least at the broadest levels, i.e. addressing the important political and social questions. This is indeed encouraging -- but this type of planning is necessarily relatively long range planning. At the level of hospital administration -- the level where the crisis is being most acutely experienced -- very little has been done, although the field is ripe for applications of O.R. It is well known that many of today's hospitals have a *surplus* of hospital beds while their clinics are becoming more congested than ever. Such statistics suggest that we would do well to invest more time in questioning our present patterns of hospital management. The skyrocketing costs of hospital care, along with the ever increasing difficulty of getting an appointment at a hospital clinic, is evidence that hospitals, rather than searching for more viable systems, have been trying to cure themselves only by pouring more money into the already existing one. The situation is unfortunate because it is really in the hospital where the crisis must be confronted first; it is there that first aid can be administered to the ailing medical care system.

There are really two aspects of the problem faced by the O.R. researcher in the medical care field, or for that matter in any field of public administration. First is the relatively straightforward problem of devising relevant analytical methods and techniques. Second is the more subtle problem of getting them implemented.

Administrators are making increasing use of these analytic tools, but not nearly as much as might be expected. The answer to the question of why not is complex. First there is a problem of hesitancy to leave a system that works, even poorly, for a system which is untried, or at least unknown to the administrator. People are uncom fortable in unfamiliar territory. There is security in "the way we've been doing it," and we are not inclined to set off down dark paths that we do not know and therefore cannot really trust. It is a known aspect of human nature that administrators are often reluctant to implement, or even investigate, any changes to an existing system until action is forced by a complete failure of the system. Another common obstacle to acceptance of more rigorous analytical methods is misrepresentation: It is not uncommon for the decision maker to be exposed to analytical techniques, but led to believe, either by overzealous analysts or simply by wishful thinking, that these new methods are being presented as a panacea. Then he either immediately sees or later discovers that they are not, and he loses faith in them altogether. This communications gap between the analyst and the administrator is destructive -- the administrator presented with a "solution," only to be disillusioned, is naturally going to lose any faith he might have had in the analytical techniques. Operations research does not offer an instant solution, but it does offer a powerful set of quantitative tools.

Another reason that O.R. is not being employed to best advantage in the medical care sector is that hospital administrators are often physicians, who, not surprisingly, rarely have strong backgrounds in management. There is, of course, no reason why they should --

physicians are specialists. The unfortunate aspect of the system is that they are forced to be big business administrators as well.

1.4 Motivation for Present Research

The research of this report was undertaken as a step towards bridging the gap between the administrator and the analyst. The goal of the research was to use the medium of computer graphics to develop an analytical tool which might be used by a hospital administrator, and which would improve communication and encourage co-operation between the administrator and the analyst.

One of the reasons for the communications gap is that quantitative analytic techniques are often highly technical, and although the administrator understands the statement of the problem he is often forced to view the technique itself as a black box which is open only to the analyst. There is a need for tools which lend quantitative insight into complex problems while avoiding the necessity of reams of computer printout or formidable looking sets of equations. Computer graphics, which is only now beginning to be widely used, offers a possible answer to this need.

This research is intended as a first step in the development of such tools.

References

1. U.S. Dept. of Health, Education, and Welfare, *A Study of National Health Insurance Proposals Introduced in the 91st Congress*, A Supplementary Report to the Congress, July, 1971, p. 84.

2. Projections of national health expenditures, 1975 and 1980. From Research and Statistics Note 18-1970, Office of Research and Statistics, Social Security Administration, U.S. Dept. of Health, Education, and Welfare. Washington, D.C., October 30, 1970.

3. *Time*, vol. 97, June 7, 1971, pp. 86-88.

4. S. E. Berki and A. W. Heston, Introduction, *The Annals of the American Academy of Political and Social Science*, vol. 399, January, 1972, p. iv. This volume, which is entitled *The Nation's Health: Some Issues*, provides an excellent discussion of the issues facing the health care sector in the United States.

5. D. D. Rutstein, *The Coming Revolution in Medicine*, Cambridge, Massachusetts, M.I.T. Press, 1967, p. 49.

6. R. I. Lee and L. W. Jones, *The Fundamentals of Good Medical Care: An Outline of the Fundamentals of Good Medical Care and Estimate of the Service Required to Supply the Medical Needs of the United States*, Hamden, Conn., Archon Books, 1962, p. 127.

7. *Report of the National Advisory Commision on Health Manpower*, vol. 1, November, 1967 (Washington, U.S. Government Printing Office), p. 2.

8. D. Mechanic, "Human Problems and the Organization of Health Care," *The Annals of the American Academy of Political and Social Science*, vol. 399, January, 1972, p. 3.

9. D. D. Rutstein, op. cit., pp. 50-51.

CHAPTER II: THE HOSPITAL EMERGENCY ROOM

2.0 Introduction

The hospital is a large, complex system comprising numerous smaller subsystems. It was decided that in light of the goals of this research project, more would be accomplished by concentrating the research effort, at least at first, on a representative subsystem of the hospital.

It is worthwhile, therefore, to examine one of these subsystems in greater depth: a facility found in diverse forms and sizes but common to most hospitals is the emergency room (hereafter sometimes abbreviated "E.R."). In a sense a microcosm of the medical care system as a whole, the emergency room is experiencing profound changes in the demands being made of it. The number of patients crowding into emergency rooms has tripled in the past fourteen years, from 18 million in 1958 to 44.1 million in 1968 to an estimated 60 million in 1972. [1] The very role of the emergency room is becoming unclear. A study of patients attending one emergency ward found that it appeared to be the *primary source of medical care* for at least one fourth of its patients. [2]

Dr. Reinald Leidelmeyer, a co-founder of the American College of Emergency Physicians, cites two reasons for the avalanche of emergency room patients.

"In the first place, the American people have become enormously mobile. It is a rare occurrence indeed that a newcomer to an area takes the time to find himself a physician before he really needs one.

"Secondly the family physician -- who had been the focal point of initial medical care through the ages -- became more and more scarce."

Thus, points out Dr. Leidelmeyer, the patient chooses to wait a few hours in the emergency ward rather than a few weeks for an appointment with his doctor. [3]

The rapid changes in volume and nature of demand being experienced by the emergency room suggest that time invested in analysis and planning of the system would be well spent. The emergency room is a complex system, and managing it effectively is not simple: the system has many variables (e.g. number of doctors, number of beds, number of nurses, patient arrival rate, etc.); numerous complex trade-offs (e.g. an efficient E.R. must strike a balance between numbers of beds, doctors, and nurses); and is highly stochastic in nature (it can be empty at one moment and overloaded five minutes later). Yet with all this complexity, most emergency rooms are planned, organized, and staffed without the aid of available analytic techniques. Development of quantitative tools might aid the hospital manager in several ways; for example:

1. improvement of staffing patterns
2. evaluating proposed changes in facilites or personnel
3. designing new facilities
4. estimating future demands on the system, and the system's ability to handle them.

More specifically such quantitative tools might be used as aids in:

1. reducing patient waiting time
2. improving doctor utilization
3. reducing unneeded time that the patient spends in the treatment facility.

2.1 An Interactive Graphical Simulation

The primary research of this report involves the design and implementation of an interactive program for simulation of a hospital emergency room. The emergency room presents an excellent focus for our study, because it is relatively small, yet quite complex, and it can probably benefit significantly from the use of analytical techniques. Recall from Section 1.4 that the purpose of this research effort is to use the medium of computer graphics to develop an analytical tool which might be used by a hospital administrator, and which could improve communication and encourage co-operation between the administrator and the analyst. Work has been undertaken towards this end by the author at the Programming Technology Division of M.I.T's Project MAC. A program has been written employing the PDP-10 computer, the Evans and Sutherland display processor, and associated supporting software developed by the Programming Technology Division. The program is the Tool for Interactive Graphical Emergency Room Simulation, which, for simplicity and ease of discussion, we call TIGERS.

2.2 Cambridge Hospital

It was decided that TIGERS, at least at first, would be based on a specific emergency room. Although generality is unquestionably desirable, the idea of graphical simulation is best investigated first by applying it to a relatively specific case. Once initial investigations are complete, work can begin on making the program more general.

The TIGERS system is based on the emergency room at the Cambridge Hospital in Cambridge, Massachusetts. It is an emergency room in the medium size range, with about one hundred patients per day arriving for treatment. In the next sections we shall attempt to describe the Cambridge Hospital Emergency Room, and give the reader a feeling for the facilities, the staffing patterns, and the types of patients who arrive there for treatment.

2.2.1 Role in the Community

The purpose of the Cambridge Hospital Emergency Room (hereafter sometimes abbreviated CHER) is to provide prompt, high quality medical care on a twenty-four hour basis to all who arrive there. Although the term "emergency room" usually implies a place which handles only very serious medical problems, the emergency room at Cambridge Hospital could be classified as a type of ambulatory clinic. The great majority of people seeking service at the emergency room do not need immediate medical attention. In fact, only about five percent of the arriving patients at CHER are what one would normally call "emergencies," and only about one percent are pre-emptive emergencies requiring the entire E.R. staff. (This situation is not unique to Cambridge Hospital.)

Especially for the poorer sections of Cambridge, the CHER takes the place of the old style family physician. This is not unusual for hospitals in middle or lower class areas. In the clinic at the Cambridge Hospital, and at other clinics in the area as well, it is often weeks before one can get an appointment. It is not surprising that the faster paced emergency room is often used instead.

Figures 2-1, 2-2 and 2-3 present a picture of the type of patient entering the CHER. Figure 2-1, the diagnosis of emergency room patients, shows that most patients have minor complaints such as superficial sprains, while the true emergencies, such as cardiac arrest or major fracture, make up only a few percent of the patients. Figures 2-2 and 2-3 show that few patients (less than ten per cent) are brought to the CHER by ambulance (and many of these are simply elderly patients transported from nursing homes), while most are able to come to the hospital by themselves. Finally, less than ten per cent of the patients arriving at CHER have problems serious enough to result in their being admitted to the hospital, and many of these ten per cent are not true emergencies.

2.2.2 Physical Plant

A diagram of the CHER is given in Figure 2-4. A typical patient walks into the emergency room entrance (1) and immediately registers with the receptionist. He fills out the registration form, which becomes his record of emergency room treatment. Then he sits in the waiting room (3) until called by a nurse, who brings the patient to one of the five beds in the general treatment room (6). (He may be assigned to a chair if the complaint is minor.) In the general treatment room the

Superficial injury -- strain, sprain, contusion	19%
Upper respiratory infection -- viral syndrome	12%
Lacerations, bites, and punctures	13%
Symptoms and ill-defined conditions	.9%
Skin disorders	.6%
Alcoholism Psychiatric disorders	3-4%
Circulatory disorders Minor fracture Gonorrhea Dx not specified Lower Respiratory Infection	2-3%
Gastritis - gastroenteritis Urinary tract infection Healthy person Otitis media Allergic reaction Major fracture Pregnancy and abortion	1-2%

Figure 2.1 Common Diagnostic Categories

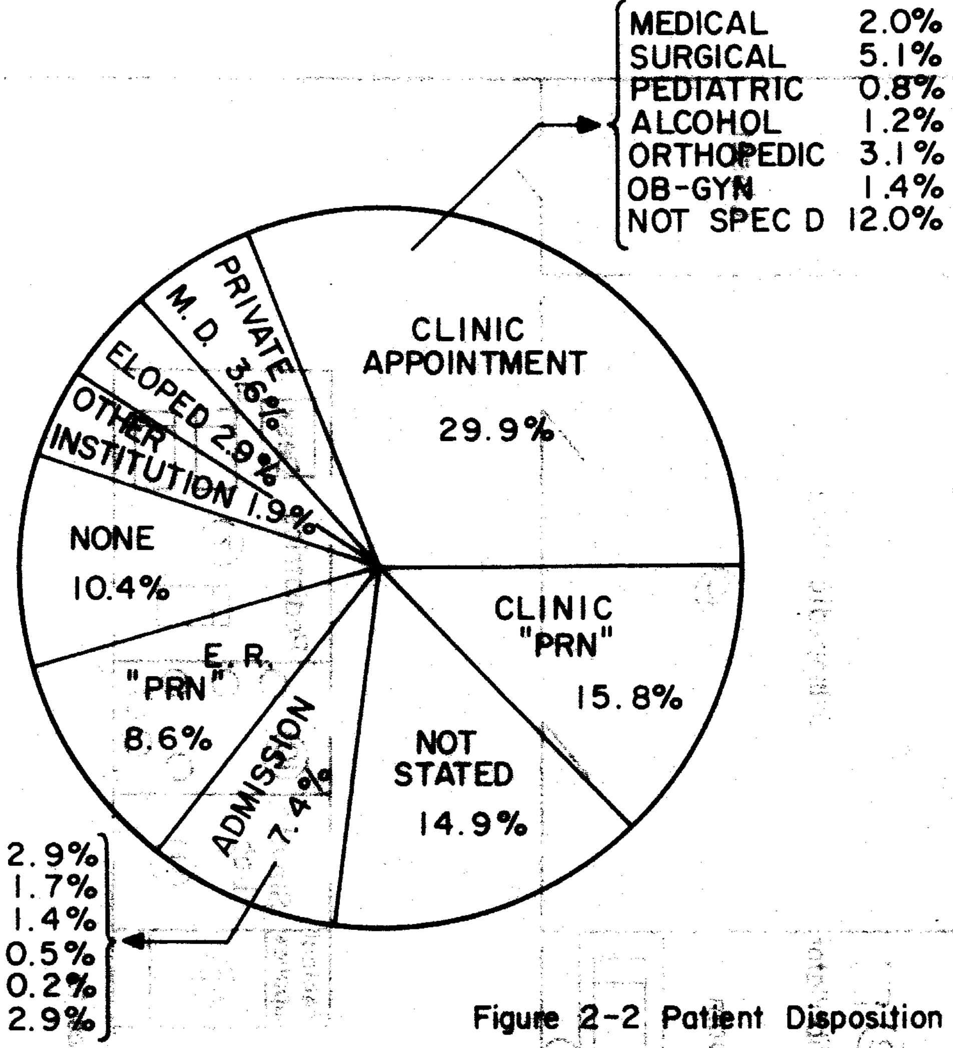
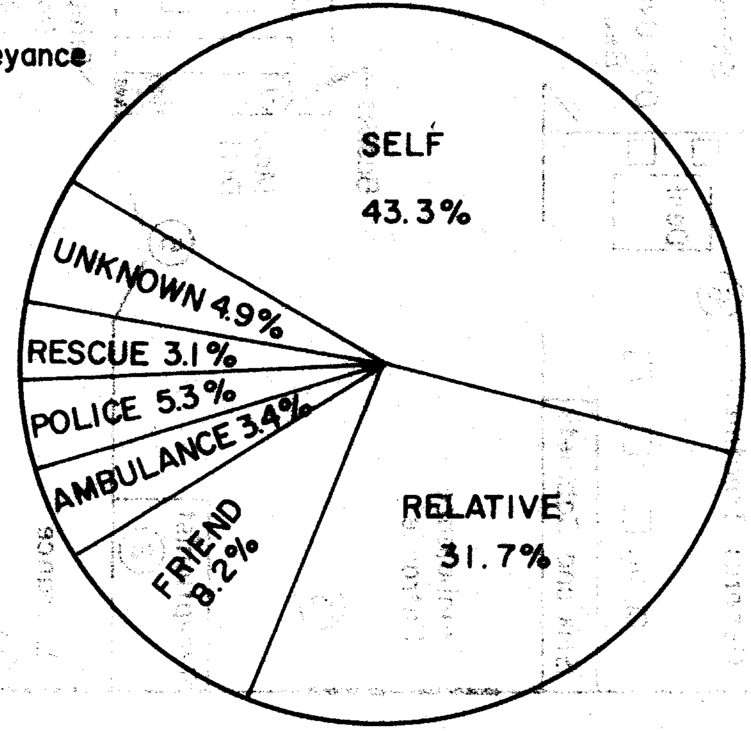


Figure 2-2 Patient Disposition

Figure 2-3
Patient Conveyance



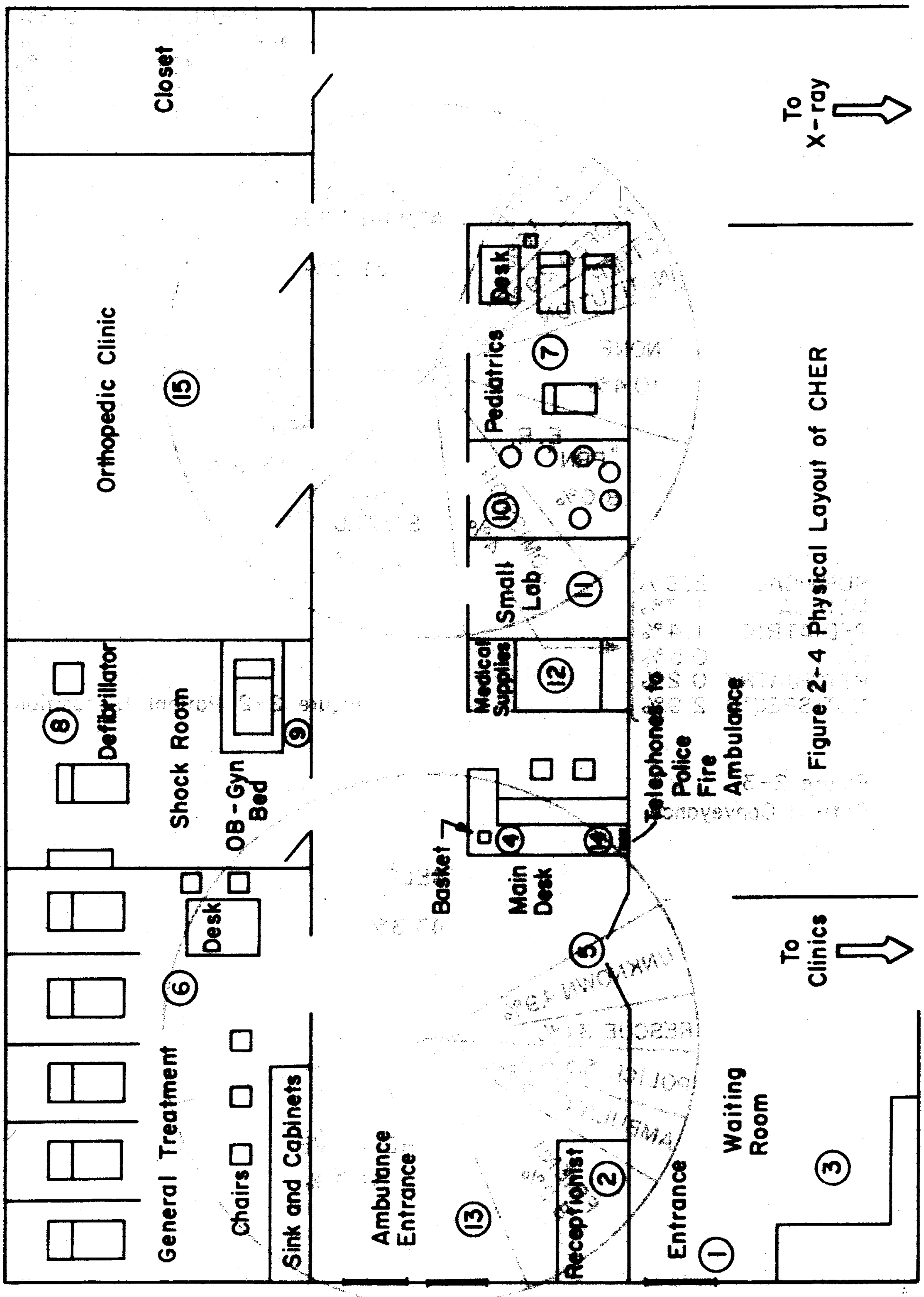


Figure 2-4 Physical Layout of CHER

patient is seen by a doctor (when one becomes available), who determines the required course of diagnostics and treatment. Very often the patient requires rather lengthy "non-doctor treatment" such as laboratory tests, x-rays, treatment by a nurse, etc. When the patient has been treated, he is released; and he either is admitted to the hospital proper or exits through the emergency room entrance (which is also the exit) (1).

A patient who is brought to CHER by an ambulance or other emergency vehicle (usually on a stretcher) enters the emergency room through the ambulance entrance (13). In this manner he can enter CHER without passing through the waiting room or filling out a form. The E.R. staff is generally informed of the impending arrival of such a patient about five minutes before the vehicle actually gets to the door, which gives them time to prepare for dealing with a "real" emergency. This is made possible by three direct telephone lines (14) to the police, fire department, and ambulance services.

Immediately adjacent to the main treament room is the shock room (8), which is used for very serious cases such as cardiac arrests. It is the shock room that plays the role that most people normally associate with that of the emergency room. In this room are all necessary supplies for treating a cardiac arrest or other very serious emergency; every effort is made to keep this room clear for such occurrences. The only other use of this room is for obstetric-gynecological (OB-GYN) cases; the bed for pelvic examinations is in a corner of the room, away from the other equipment.

Any patient under fourteen years old is considered a pediatric case and is taken to the pediatrics room (7) instead of the main medical treatment room. The pediatrics room contains three beds and a desk and is used exclusively for persons under fourteen.

Other rooms in the emergency room are a small staff lounge (10), a small laboratory (11) where doctors can perform simple tests such as urinalysis, a medical supplies closet (12), and the orthopedic room (15). This last room is actually the hospital's orthopedic clinic and is not a part of the emergency room proper. When not being used as a clinic, however, it is used for E.R. psychiatric patients, both as a waiting room and as an interviewing room. Also, it is used for patient interviews with such people as social workers and staff members of the alcoholic detoxification program.

The general medical room (6) contains five beds (each made private by surrounding curtains), a number of chairs for patients who don't need a bed, a sink, various medical supplies, and a single desk for the doctors' use. The records of the patients being treated are kept at this desk. The paper work (i.e. writing up patient records) is done mostly at this desk; therefore a telephone and the laboratory and x-ray requisition forms are also located here.

Filling in the diagnosis and treatment on a patient's emergency room treatment record can also be done at the main desk (4). Medical reference books are there, as well as a regular telephone (in addition to the three special lines). The main desk is used largely for research, telephoning, and consulting with the staff doctor assigned to assist in the emergency room.

2.2.3 Staffing

The CHER staff consists of a receptionist, two or three nurses, one or two interns, a staff doctor; and, at various times, medical students, nurses' aides, orderlies, and paramedical students. The staffing pattern is shown in Figure 3-2, along with the average number of arrivals to CHER, by hour of the day.

The staffing pattern is designed to deal with the time-dependent arrival rate of patients to the emergency room. The high demand hours are 9 a.m. to 11 p.m.; from 10 a.m. to 7 p.m., during which time the average patient arrival rate is about five per hour, there are always at least two interns and a staff doctor. Also during the day there are three nurses on duty. Theoretically the interns provide the primary medical care, while the role of the staff doctor is that of a consulting physician. (When the room begins to become overcrowded, however, the staff doctor also provides primary care.) In practice, however, some of the staff doctors, while still acting as consultants when needed, engage in primary patient treatment all the time. In such cases there are effectively three full-time doctors in the emergency room.

During the evening the arrival rate begins to taper off, until at night the rate is greatly reduced, on the order of one patient per hour. At night only one intern and one nurse are on duty. Generally from 7 p.m. to 1 a.m., a "moonlighter" is also employed; he is a staff doctor or a resident whose role is analogous to that of the staff doctor during the day. From 1 a.m. to 7 a.m., however, the only doctor on duty is the one intern on the "night shift."

It is the CHER nurse who first sees the entering patient. She assigns the patient a bed and measures his vital signs; and then assists the doctors in working with the patients, watching them, obtaining specimens, filling out lab and x-ray requisitions, and various other duties.

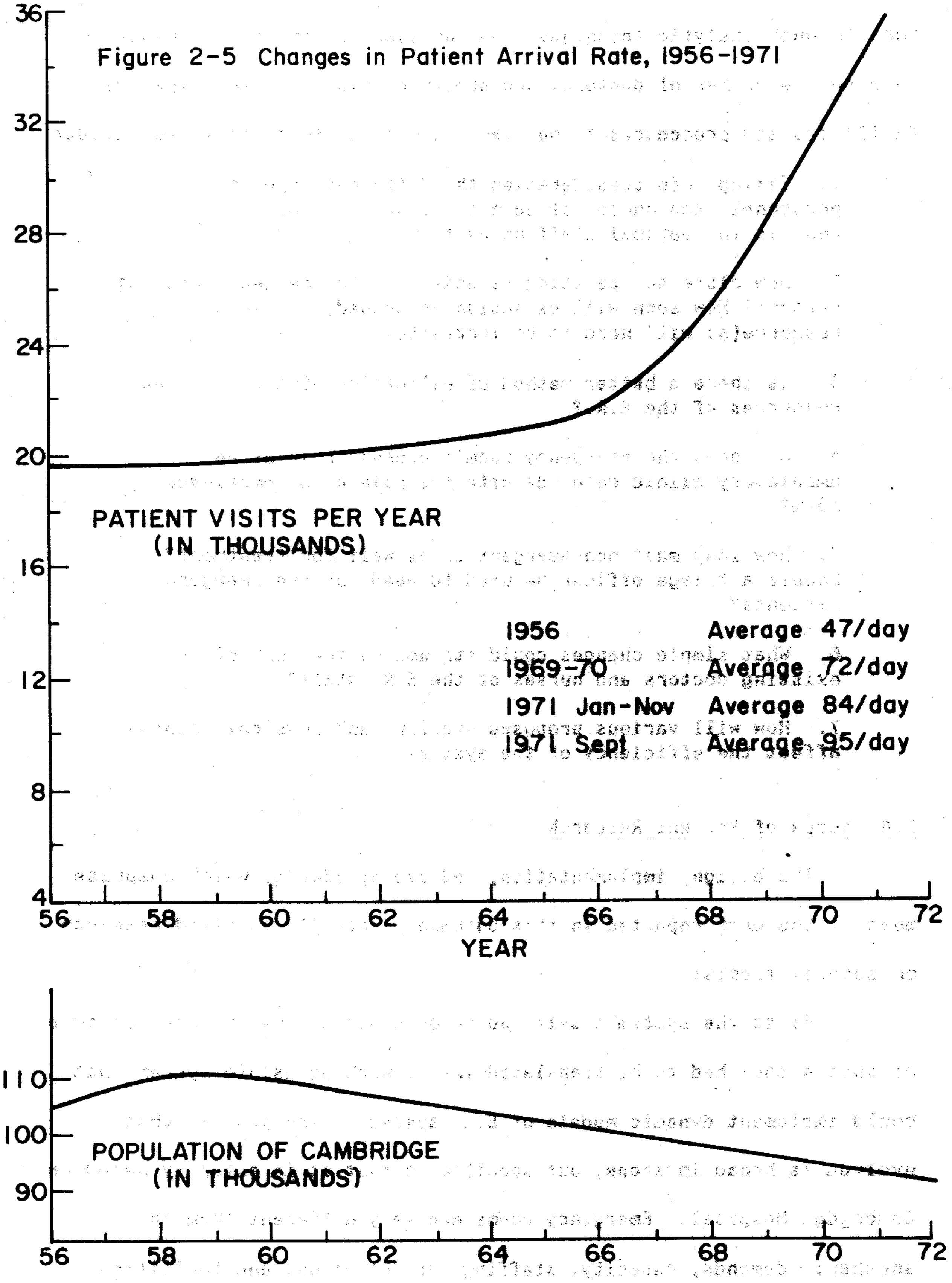
Since the Cambridge Hospital is a teaching hospital, it has an arrangement with the local medical schools whereby medical students are assigned to CHER as assistants. CHER is able to request the number of such students that are to be assigned at a given time. Recently, this has been limited to one or two students. The calibre of these students varies greatly; some function almost as another intern, while others tend to get in the way and hinder service.

2.3 Need for Research

To fulfill its goal of providing fast, efficient, twenty-four hour medical care, the CHER has to solve a number of problems. The first is how to deal with the increased demand. Currently the CHER sees about 100 patients per day. The increasing demand on CHER in the past sixteen years is shown in Figure 2-5. The number of patients has risen from 19,000 per year in 1956 to 30,000 per year in 1970, while the population of Cambridge has remained fairly constant. Such a rapid rate of growth necessitates either a physical expansion of the present system or a reorganization in such a way that the present quality of service will not be diminished.

Constraints on the hospital's methods of meeting this growth are both physical and monetary. Thus simply increasing the size of the E.R. staff may not effectively increase its capacity. It is probable

Figure 2-5 Changes in Patient Arrival Rate, 1956-1971



that through analytic techniques, we can quantify the relationship between the number of doctors, the number of beds, and the types of facilities and procedures to be used. Questions to be answered include

1. Taking into consideration the different types of personnel, the number of each type, and the various shifts, what is the optimal staffing pattern?

2. How close to operating capacity is the emergency room at present? How soon will expansion be needed, and what resource(s) will need to be increased?

3. Is there a better method of allocation of the existing resources of the E.R.?

4. How does the emergency room's present role as an ambulatory clinic coincide with its role as an *emergency* room?

5. How long must non-emergent cases wait for treatment? Should a triage officer be used to weed out non-emergent patients?

6. What simple changes could streamline the work of the existing doctors and nurses of the E.R. staff?

7. How will various proposed staffing and physical changes affect the efficiency of the system?

2.4 Scope of Present Research

The design, implementation, and use of TIGERS, which comprise most of the work reported in this document, actually entailed research on several fronts:

First the system itself had to be created; the theoretical idea of such a tool had to be translated into a working usable system that could implement dynamic models of E.R. systems. The program that evolved is broad in scope, but specific in that it is actually based on Cambridge Hospital. Emergency rooms are very different from one another: demands, capacity, staffing, organization, and facilities --

all can vary with surprisingly great latitude. The techniques incorporated by the program, however, are general; and TIGERS does illustrate the potential of a more general system.

Secondly, in order to use the user-oriented system that was designed, a test model had to be designed and then implemented as a simulation.

Finally, a third aspect of the research concerned human engineering. It was desired that TIGERS be usable by persons not necessarily trained in operations research and/or computer science. In fact it was desired that TIGERS be a self contained package usable by all intelligent (but not especially technical) users, especially medical personnel and hospital administrators. Usually such models are designed for use only by highly trained technical specialists. It is felt that there is significant value in having such a tool be available to the non-technical user who knows the problems faced much more intimately than does the outside consultant. In order to insure that TIGERS was made as useful as possible to those for whom it mattered most, the hospital personnel, periodic communication was maintained with people at Cambridge Hospital, especially Dr. Peter Mogielnicki. Through this communication, valuable feedback was obtained towards making the system as useful and usable as possible.

This report discusses each of these three aspects of research. In Chapter III, the idea of modeling is introduced; and the trade-offs and pitfalls that must be considered in designing a model of the E.R. are discussed. The May 1972 study by Markel et al. is introduced and discussed in depth. Chapter IV discusses TIGERS, both as an interactive tool and as a modeling environment. Chapter V discusses the test model that was actually implemented under the TIGERS system. Finally, Chapter VI summarizes what TIGERS is and what it is not, and attempts to put the program in a realistic perspective. Also Chapter VI speculates about the future and points out areas where future research might be fruitful. A complete subroutine by subroutine description of the TIGERS system and a sample of source code from the program are included as appendices.

References

1. *U.S. News and World Report*, vol. 72, June 26, 1972, pp. 78-81.

2. E. R. Weinerman et al., "Yale Studies in Ambulatory Medical Care, V: Determinants of Use of Hospital Emergency Services," *American Journal of Public Health*, vol. 56, July, 1966, p. 1044.

3. *U.S. News and World Report*, vol. 72, June 26, 1972, pp. 78-81.

CHAPTER III: MODELING THE EMERGENCY ROOM

3.0 Modeling -- Introduction

In order to apply quantitative methods to a real, physical system, one must create an abstraction, a *model*, that describes the system in terms of aspects of the system relevant to the study. Using the model, the analyst can perform experiments which would be difficult and/or expensive to carry out on the real system. A well designed model can predict how the system will react to changes in its environment. A model can be thought of as a "black box" which accepts descriptions of the conditions of interest, and outputs information describing how the system would behave given this set of conditions. Often, setting up these conditions in the real world is impracticable; but the model, when it serves its purpose, provides an abstract representation of the real world which the analyst can manipulate more easily than he can reality.

Generally, the analyst defines a set of hypothetical conditions by assigning values to a set of relevant input parameters, and the model describes the system's response by assigning values to relevant output variables. For example, we might wish to know how much longer an emergency room patient would be expected to wait in a waiting room if the demand for service were to double. A model of the system might accept as input the demand for service expressed in requests per hour, and generate as output the expected waiting time in minutes. The model does not, of course, duplicate all aspects of the system; it incorporates only specific aspects in which the analyst is interested. The model just mentioned, for example, would probably shed little insight on the question of how demand would change if the service facility were moved to another location.

Often an apparently complex system can be modeled well using a model with a well defined analytic solution. A number of classical models have evolved which are applicable to whole classes of real world situations. An excellent example is the so called "M/M/1 single server queuing model." This model assumes a service system involving a single server whose service time is characterized by an exponential probability distribution function (pdf). [1] The interarrival time of customers arriving requesting service is also assumed to be characterized by an exponential pdf. If the server is free, the customer receives immediate service. When the server is busy, arriving customers enter a queue, and customers are served in a first come first served manner as the server becomes available.

The solution of such a model yields a set of equations which express such useful values as the expected waiting time in the queue and expected number of people in the queue in terms of the known or postulated parameters, average arrival rate and average service time. Often a situation does not exactly fit the model but is close enough that significant insight can still be obtained from its use. For example, a situation might fit the above queueing model with the exception that its service time pdf deviated somewhat from the exponential. The predicted waiting time in queue, however, might well still be close enough to the true value to be of use to the analyst. It is, of course, extremely gratifying to the analyst when the system under study "fits" one of these models with well defined analytic solutions.

Some real-life situations are not so co-operative and do not lend themselves to such models. Frequently, however, models of such systems are solvable using computer simulations. The simulation technique, which involves dynamically simulating "events" of the real world on a digital computer, is generally more expensive, but it can often be used to great advantage with systems that are too complex to be solved analytically.

3.1 Modeling the Emergency Room

Of first priority in any analytical process is to clarify which questions about the system are of interest. Often these questions can take the form of "How does x vary with y?" or "How does x vary with y and z and w and ... ?" Formulating these questions involves isolating two sets of parameters; a set of outputs (x's) and a set of inputs (y's, z's, etc.).

In this section we outline our approach to modeling the emergency room. Basic issues are discussed, and an attempt is made to communicate the mode of thinking necessitated by the modeling process.

3.1.1 Isolating Relevant Parameters

The quality of treatment in the Cambridge Hospital Emergency Room -- and in many of the nation's emergency rooms -- is high. Problems in these systems arise primarily when patients cannot avail themselves of this treatment because the system is overloaded. Thus as a first step in our analysis of the emergency room, it is decided that the quality of medical care per se administered to each patient is not at issue in this analysis. Therefore it addresses itself *not to the problem of inadequate medical technology in the E.R., but rather to the fact that requests for service from the E.R. are exceeding its capacity to handle then effectively.* It is assumed, therefore, that the system runs smoothly and delivers adequate service as long as the system is operating below "capacity." It is only when demand for service increases that queues begin to develop and the system bogs down as it becomes "swamped." [2]

Once it is decided that in the present research medical care per se is not at issue, the system can be viewed as a "service system," in which patients arrive requesting "service" and receive it as facilities become available. The efficiency of such a system can be quantified through such metrics as delays experienced by the patient. Thus the following parameters can be considered relevant indices of the operational effectiveness of an emergency room system:

1) number of patients in waiting room queue

2) time patient is kept waiting before service begins

3) time a patient actually spends in service

4) fraction of patient service time not usefully spent.

3.1.2 Obstacles Inhibiting Development of an Analytic Solution

The emergency room is clearly a service system -- patients arrive requesting service, are served as servers become available, and are released. But unfortunately it does not fall conveniently into the classic queueing model template. The definition of the key parameter, service, is obscured by the complexities of the system. In a system such as, say, a police patrol force, it is relatively easy to define a clear cut server and define a distinct service time. The server is the patrol unit, and the service time is the number of minutes that elapse between the time a unit is dispatched on a call and the time it resumes patrol or is dispatched to answer another request for service.

In the E.R., an obvious definition of service time is time spent in treatment. But what defines treatment? What is the server? In the police patrol example, the server is clearly the patrol unit. In the E.R. example one is at first tempted to define the E.R. bed as the server. But this is clearly inadequate; no service is being rendered when a patient is lying in a bed waiting for a doctor to become available. The analyst is then tempted to decide that the doctor is the server, but this is equally unsatisfactory. A patient using the x-ray facility is unquestionably receiving treatment, but if the doctor were defined as the server, the patient at x-ray would be classified as receiving no service.

Thus the analyst of the hospital emergency room is quickly confronted with the problem that there are several types of treatment in the E.R., all of which constitute "service." Service can take the forms of treatment by trained personnel and/or use of physical facilities. (Personnel does not only mean doctors, however; just as important are the nurses and paramedical persons.)

Not to be overlooked is the problem of how to deal with "blocked time." A patient is said to be "blocked" when he is waiting for a resource (e.g. doctor, nurse, lab report), but not receiving any other type of service. Large amounts of blocked time indicate that more of that resource is needed in order to be operationally compatible with the rest of the system. For example, a large number of beds would be worthless if, even though every patient requesting service were immediately assigned a bed, he spent most of his time in a "doctor blocked" state waiting for a doctor. In such a system, a wasteful imbalance would be present: if the demand were large, more doctors would be needed; or if the demand were not so great, then expense has been wasted on extra beds. In any classical serving system model, it is assumed that using the resource that is the server (or part of it) constitutes service. In the emergency room, however, it is possible to have a patient who is really receiving no service except in that he is consuming one of the resources.

Another classic example of blocked time is the patient who has been treated and is essentially ready to be released from the E.R., but who, for one reason or another, is still occupying a bed. Such a situation might occur, for example, in the case of an aged person who cannot leave unaccompanied, but whose companion has not yet appeared to take responsbility for him. Another instance of this "exit blocked" state often occurs for E.R. patients who are finished with all E.R.treat ment and are to be admitted to the hospital. Admission to the hospital involves a considerable amount of administrative overhead, and it is not uncommon for the patient to spend considerable periods of time waiting for this processing.

As one gets deeper into the analysis of the emergency room, such basic questions as when a patient is being served and how much service he is receiving become significant. In the standard service system models, the only variation in service received by various customers is in length of time of service. Patients in the emergency room receive not only different lengths of service but also highly varying types of service. A patient in a blocked state waiting for a doctor is only receiving service in that he has a bed. At the other extreme is a cardiac arrest patient who may be receiving service from numerous facilities (defibrillator, monitoring equipment, etc.) and personnel (continuous service from several doctors and nurses). Between the two extremes are a range of situations difficult to classify.

One example is patient observation. Sometimes it is necessary for a patient simply to wait in a bed for an "observation period." A patient with a drug overdose, for example, will stay for a few hours to make sure that there are no more complications. During this time, a doctor or nurse will see him a few times, but for the most part he just waits until the doctors are satisfied that he is well enough to be released. A person under observation uses up a bed, but how much personnel time does he use? What service is he receiving? Such questions have no well defined answers.

Similar questions can be asked about such situations as those in which laboratory analyses are ordered for the E.R. patient. The patient awaiting a lab report may be receiving no treatment except in that he is occupying a bed, and yet somewhere in the hospital his blood sample is being analyzed. How do we classify this type of treatment? Where does it fit into a model? Again, such questions have no well defined answers.

A significant part of any analysis is to gather data from the system being analyzed upon which to base parameter values of the model. A value for average service time, for example, needs to be determined. In the E.R. this can pose a serious problem. It has already been noted that patients receive varying degrees and types of service. A patient might require ten minutes of nurse time, five minutes of x-ray time, thirty seconds of doctor time, and thirty minutes of bed time. Another might need twenty minutes of nurse time, twenty minutes of doctor time, ten minutes of lab time, and two hours of bed time. This variation poses only a minor problem to the data gatherer. The serious problem is that the services are not delivered in easy to measure packages. Returning once more to the police patrol example, one notes that data gathering is a rather straightforward process. Service can be measured in patrol unit minutes. There is no analogous unit of service for the emergency room. Furthermore, whoever is using a server (patrol unit) has the entire server dedicated to him. Service begins when the unit is dispatched to answer the call and ends when the unit resumes patrol or begins service on another call. Service in the E.R. is generally not at all dedicated. The emergency room is a constantly changing scene with doctors and nurses continually moving from patient to patient, thirty seconds here, two minutes there. Measuring service received by each patient becomes a formidable problem.

The problem is made even more complicated by service being rendered by non-E.R. personnel and facilities. The E.R. is not a closed system. It is common practice to have a doctor called from the floor in order to consult with one of the E.R. doctors. The most frequent occurrences of this type occur when a surgeon or an orthopedic specialist or a psychiatrist is needed. Also, in the case of patients to be admitted, a doctor from the floor is called to initiate and supervise preliminary medical care (care administered before the patient is actually officially admitted by an admitting officer). These "outside doctors" use the E.R. facilities, including one of the beds for the patient, for what are sometimes large periods of time. Several non-E.R. facilities may be involved in the care of an E.R. patient. The most common of these are the x-ray facilities and the laboratory (used for cultures, blood analysis, etc.). Specimens obtained in the E.R. are taken to the third floor laboratory by a nurse or nurse's aide. (Note: Most patients walk to x-ray, but if one of the beds of the general treatment room is needed for transporting the patient to x-ray, its space is left empty. Another bed is not moved in.)

The reader may note that underlying this entire discussion of the complexities of the emergency room has been the idea of *balance*. Smooth and efficient operation of the emergency room is absolutely dependent upon a proper balance existing among the available resources. There is no single server in the emergency room. Beds are useless without personnel, and personnel are useless without a place to work. Carrying this idea one level further, doctors can do little unless there are also enough nurses in the system, and nurses, in turn, are only useful if there are enough doctors. The marginal utility of an additional unit of any resource is highly dependent upon the existing quantities of other resources.

3.2 The May 1972 Study

In May, 1972, one of the first efforts in emergency room analysis was undertaken by Markel, Purks, Shields, and Weissberg. [3] Their purpose was to gain insight into the interrelationships between number of beds, staffing patterns, and arrival rate of patients; and how these parameters affected the quality of service as measured by the indices defined in Section 3.1. The study was one of the first attempts at quantifying these relationships.

In some ways, the TIGERS model is an extension of the work of this May 1972 study. In this section, we therefore discuss this study in considerable detail. In the last section we made an attempt to communicate the type of thinking necessary for modeling the E.R. This section, in addition to describing the work of the May 1972 study, is also an extension of that attempt.

3.2.1 Simplifying Assumptions

It was decided early in the analysis that the emergency room could be viewed as a multi-server "service system" as described in Section 3.1. Once it was decided which parameters were relevant to the analysis (end of Section 3.1) it was necessary to make a number of simplifying assumptions. This is necessary for any modeling process; factors which do not significantly affect the parameters of prime concern can only add overhead.

It was decided not to include pre-emptive emergencies in the model, because the vast majority of the CHER's patients are non-emergent, and it is this type of patient that is eventually going to strain the capacity of the system. Furthermore, although an emergency such as cardiac arrest requiring the entire emergency room staff for an hour or more seriously disrupts the system, it happens so rarely that the facilities of the shock room are thought by the hospital to be adequate to fill present needs. (The bed with the defibrillator, etc. is at this time used only for pre-emptive emergencies, not for non-emergent patients.)

Another assumption involved peripheral facilities: the use of the psychiatric, pediatric, gynecological, and alcoholic detoxification facilities and staff usually does not involve members of the E.R. staff; and the facilities used by these departments are independent, for the most part, of the general medical treatment room. For this reason the mathematical model did not include these facets of the E.R.

The physical capacity of CHER is about seven patients in the general treatment room. There are five beds in this room, and new patients are seldom called in unless there is a bed free. However, there are also a few chairs; and many patients, not in need of a bed or who come back from x-ray and find no beds available, sit in chairs. From the physical size and layout of the room and from their own observations, the team determined that seven was a reasonable estimate of the maximum number of adult patients in the system at one time. (The nurses almost never let more than seven in the room at one time. If there are many people at x-ray, then even if beds are free, no more people are called in; thus the room is prevented from becoming "flooded" when those at x-ray return.)

Because the rate of arrivals at night was low, it was decided that it would be inefficient to increase the night staffing above the present one intern policy. If too many people to be handled by one person arrive, the intern can call elsewhere within the hospital for help. Thus, the Markel et al. model concentrated on a daytime situation with an arrival rate of five people per hour and a staff of two interns and one staff doctor on duty.

3.2.2 The Two Stage Model

The main object to be modeled was the general treatment room. Typically a patient is called in by a nurse, who assigns him a bed and takes his vital signs (pulse, temperature, etc.) before he is examined by a doctor. While a patient is in the emergency room, he is seen many times by a doctor, usually for periods of a minute or two, rather than for one uninterrupted period. In addition to the emergency room doctor, the patient also receives a number of non-emergency room doctor services, such as laboratory tests, x-rays, consulting physicians, nursing, and resting and waiting. These "non-doctor" services overlap each other and overlap the doctor time also (e.g. a nurse often helps a doctor treat a patient). To be absolutely true to life, doctor time should be modeled as the sum of a random number of random variables interspersed with a random number of random-length non-doctor services.

The approach that the team took towards fitting the emergency room to a model with an analytic solution was to divide the time a patient spends in the emergency room into two categories: "doctor" and "non-doctor" time. That is, a patient in the emergency room receives two services, those of a doctor and those of all other facilities and personnel. An inherent important assumption, indeed a key assumption of their model, is that all non-doctor time and all doctor time are aggregated into two uninterrupted service periods.

All patients are assumed to avail themselves of non-doctor services first. This includes nursing, being seen by non-emergency-room doctors on a consulting basis, getting a blood test or x-ray, or being under observation. Next, in the model all patients are seen by a doctor for an uninterrupted period of time. When non-doctor functions overlap doctor functions (e.g. being seen by both a doctor and a nurse) the time is counted as being "doctor time."

Because there are at most three regular doctors in the emergency room at one time, and one of these is the staff doctor whose role theoretically is as a consultant for the interns, the number of people being treated at any one time is usually less than seven. While there are seven "beds," only three of them can be seen by a doctor at once. The others, if they have finished their non-doctor service (no other medical treatment being administered), are considered to be in a "blocked" state, i.e. waiting for a doctor.

Since the team decided to concentrate their analysis only on the physical capacity and the number of doctors of the CHER, their model does not explicitly account for nurses, medical students, paramedical students, orderlies, etc. They are, however, incorporated into the non-doctor time of a patient.

3.2.3 Model Structure

The team's model of the emergency room is shown in Figure 3-1. Seven patients can be treated at one time, and each patient receives two basic types of service, "doctor" and "non-doctor." While seven patients can be in the system at any time, only three or fewer (assuming there are three doctors on duty) will be seeing a doctor at any one time. The remaining patients will either be receiving care other than from an emergency room doctor or will be blocked, waiting for a doctor to see them. Thus the model is a two stage multi-server system, with no queue allowed between the two stages. If x doctors are busy with patients, then 7-x beds are available for other patients, both for non-doctor services and for waiting to see a doctor. A queue is allowed to form in front of the first stage of the model; this queue corresponds to the waiting room.

Service times are assumed to be exponential. The distribution of doctor and non-doctor times of a 100% sample, taken for eleven hours of observation on each of two days, is shown in Figure 3-3. The distributions support the exponential service time assumptions. The mean doctor time is the parameter $1/\alpha$, and the mean non-doctor time is the parameter $1/\beta$. For the first model of the present system, it was assumed that two interns and a staff doctor were on duty in the emergency room and that the staff doctor followed his theoretical role, acting as a consultant to the interns. It was also assumed, however, that when the system was full (seven patients), the staff doctor would deliver primary care. Because the staff doctor was not as available for help or consultation while he was a primary server, it was assumed that

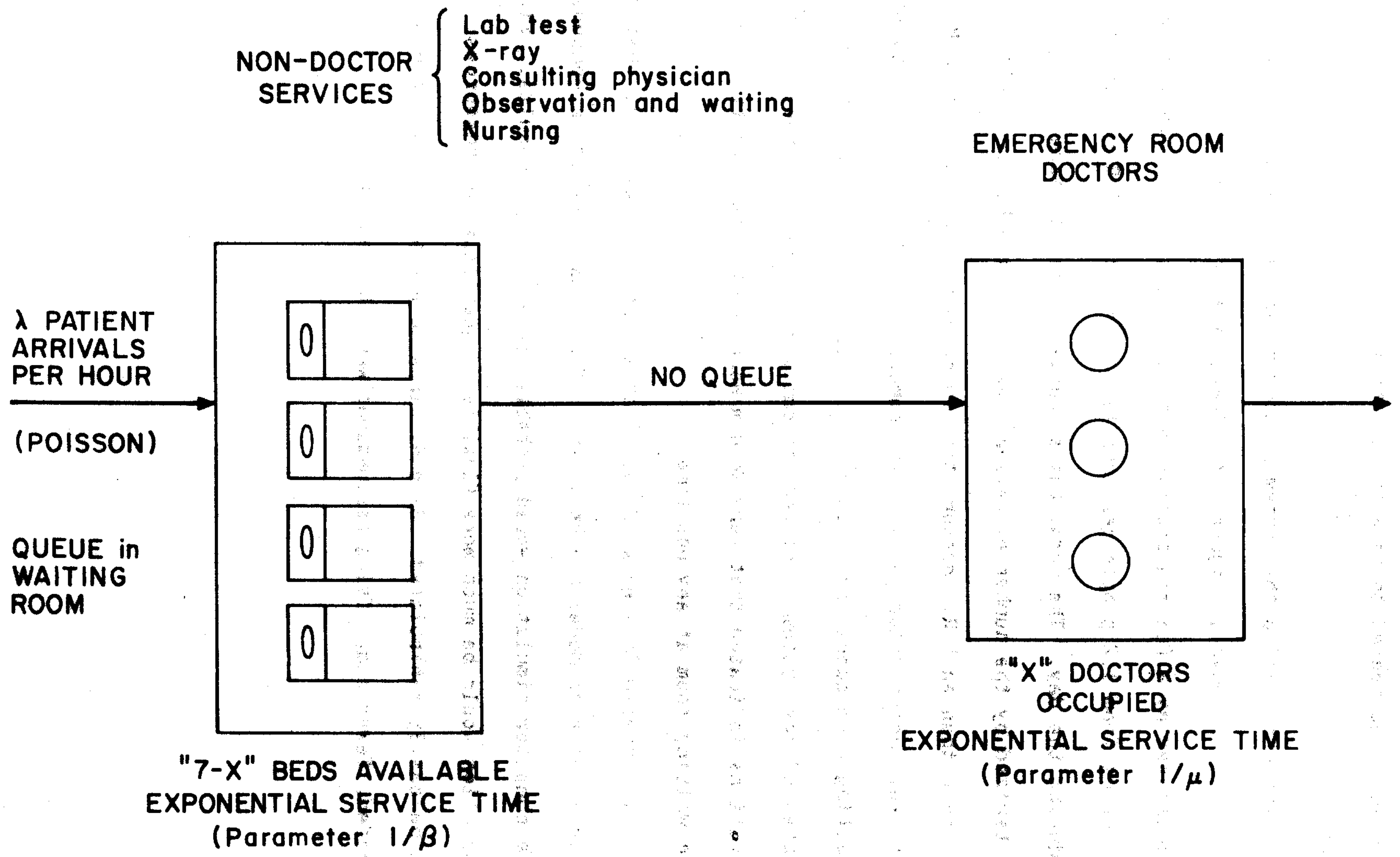


Figure 3-1 Two Stage Model Used by Markel, et al.

the average time needed to care for a patient increased;   was used instead of

The arrivals to the system by hour of the day for a one week 100% sample are shown in Figure 3-2. The arrival rate can approximately be represented by a Poisson process with an adult arrival rate lambda which varies from 1.2 people per hour late at night to 4.8 people per hour during the day. The model has seventy-six states, each characterized by the number of patients being treated by facilities or staff other than an E.R. doctor, the number being treated by an E.R. doctor, the number "blocked," and the number in the queue. Length of the waiting room queue was limited to five in order to keep the number of states of the system model from becoming unreasonably large. Also, observations indicated that very seldom were there more than five people in the waiting room at any one time.

The preliminary work of Markel et al. provided evidence that the two stage queueing model can be useful. There was, however, a consensus that a computer simulation would probably yield stronger results. First a simulation would be much more flexible -- quantities of resources could be easily manipulated. Second, since no analytic solution would be needed, fewer simplifying assumptions would be necessary.

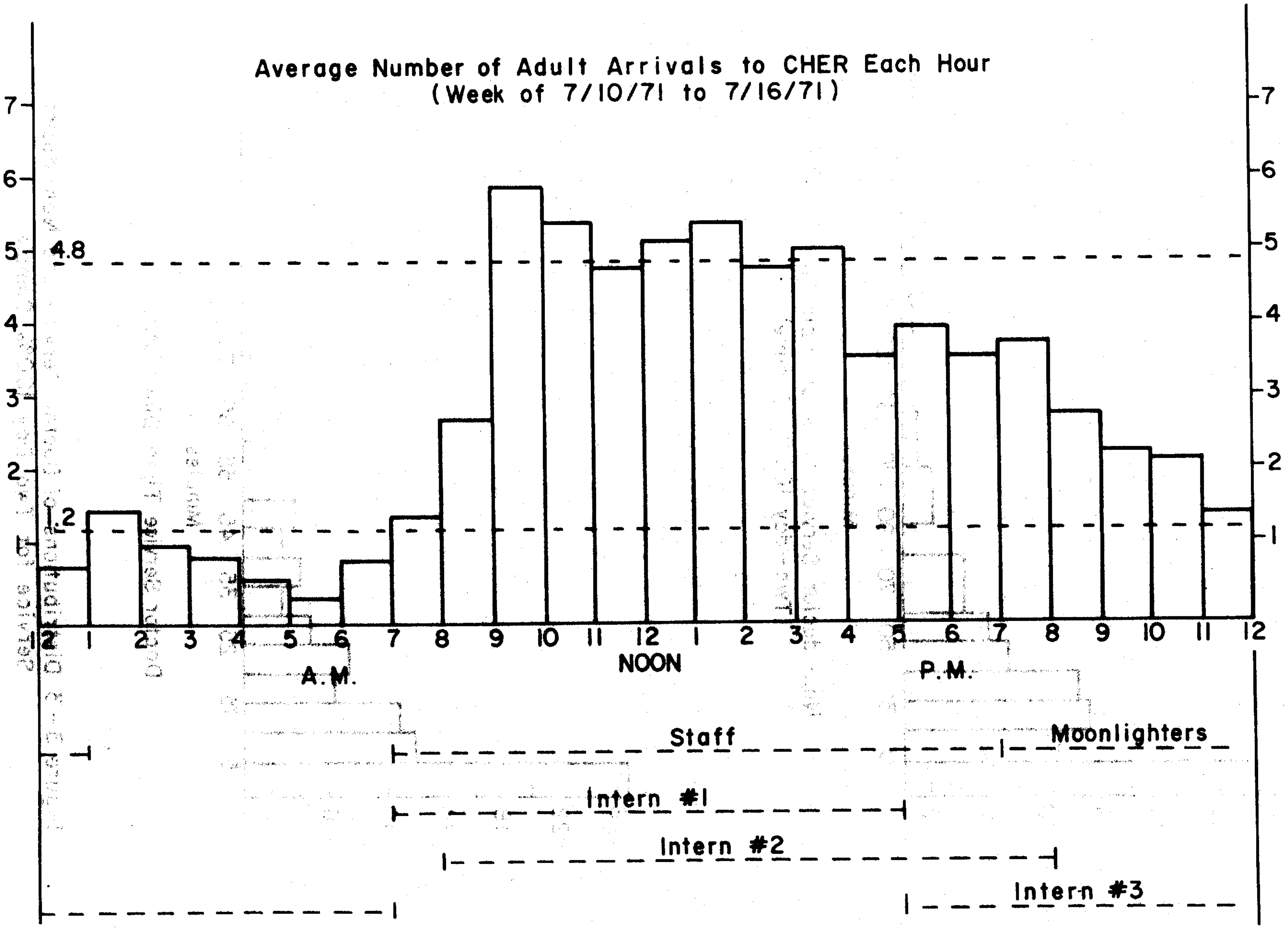

Average Number of Adult Arrivals to CHER Each Hour
(Week of 7/10/71 to 7/16/71)
7
6
5
4
3
2
1
4.8
1.2
12
1
2
3
4
5
6
7
8
9
10
11
12
1
2
3
4
5
6
7
8
9
10
11
12
A.M.
NOON
P.M.
Staff
Moonlighters
Intern #1
Intern #2
Intern #3

Figure 3-2

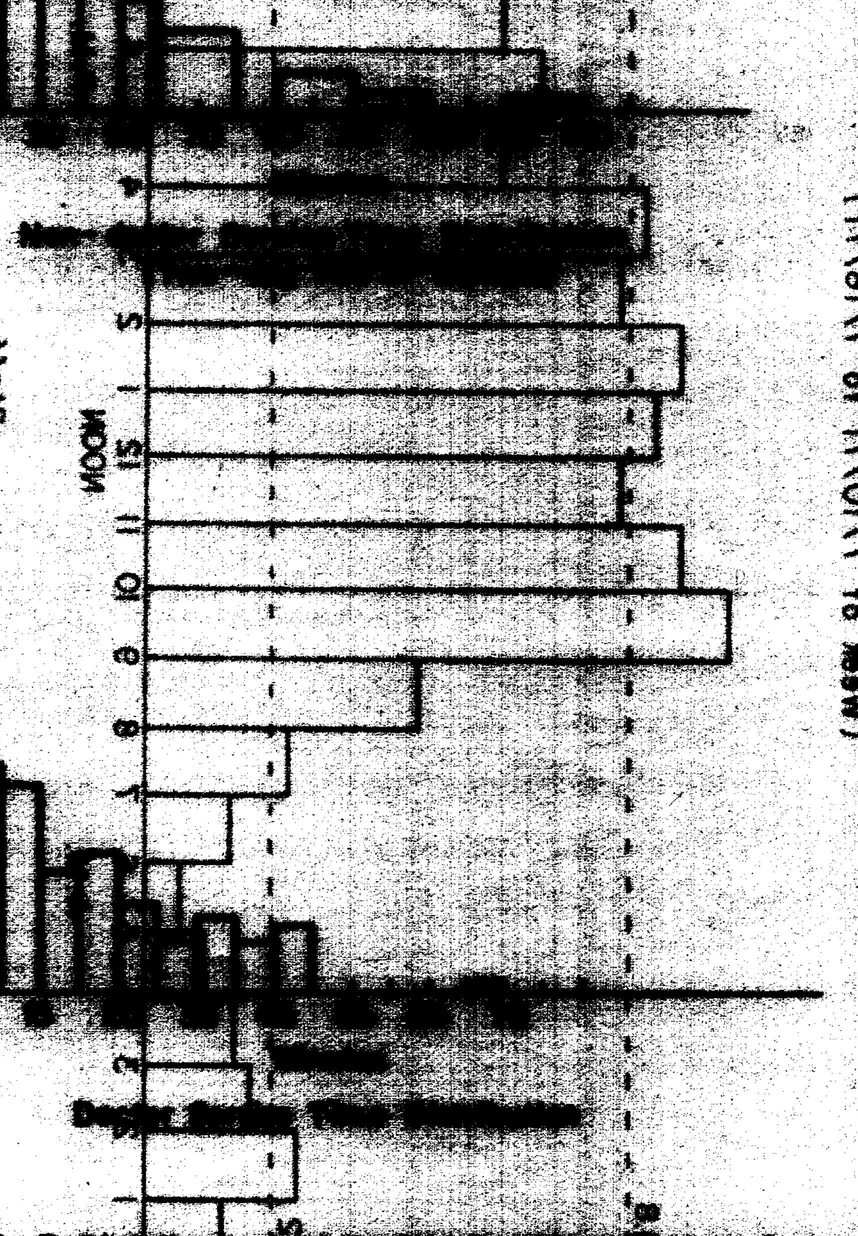


Figure 3-3 [illegible]

References

1. An excellent introduction to probability theory can be found in Alvin W. Drake, *Fundamentals of Applied Probability Theory*, New York, McGraw-Hill, 1967.

2. Also important in the delivery of emergency medical care is the delay between the onset of the emergency and the reporting of the emergency to the formal emergency medical system. Research is in progress investigating the nature and causes of this delay, but it does not fall within the scope of the system modeled in this report.

3. L. Markel, S. Purks, J. Shields, and R. Weissberg, "A Study of the Cambridge Hospital Emergency Room," May, 1972, unpublished.

CHAPTER IV: TIGERS

4.0 Introduction

As mentioned in Chapter I, the present research was motivated by the goal of creating an analytical tool which might be used by the hospital administrator, and which could aid in bridging the communications gap that exists between the administrator and the analyst. The design and implementation of the program TIGERS represents the greater part of the work of this research project. A direct outgrowth of the Markel et al. study, the TIGERS system was developed with two main goals in mind: 1) to design and implement a simulation-based modeling environment which would necessitate fewer simplifying assumptions than the model developed by Markel, Purks, Shields, and Weissberg; and thus make possible a more detailed and policy-relevant model; 2) to develop a medium of communication for presenting the quantitative results of the analysis in an intuitive manner requiring little training in systems analysis to comprehend. Pursuant to the goals outlined in Chapter I, the primary research effort was concentrated not on designing a highly accurate model of a particular emergency room, but rather on development of a tool which could be used for such a purpose.

Any analysis of a public system is almost necessarily a co-operative effort shared by the analyst and the public administrator, if the project is actually to result in constructive change of the system. The analyst has effective mathematical tools and skills, but it is the administrator who has intimate familiarity with the problem at hand and who can keep the analysis on a track where it can be of other than academic value.

In this chapter, the Tool for Interactive Graphical Emergency Room Simulation is discussed in detail, as the environment is described both from the point of view of the hospital administrator and that of the analyst. We first define the type of simulation upon which TIGERS is based. We then introduce the idea of graphical interaction and the modeling philosophy which makes the TIGERS model somewhat different from other simulation models. Next the user viewport and interaction with the model are introduced, and finally the idea of TIGERS as a modeling environment is dicussed.

4.1 Interactive Graphical Simulation

The TIGERS environment is based upon a so-called *event paced* simulation, in which the model has associated with it certain data bases, and the simulation executes a sequence of events which may change these data bases and/or schedule future events. For example, the model might incorporate the event "send patient to x-ray." This event changes two data bases: it decrements the count of number of people in the main treatment room, and increments the count of the number of people at x-ray. It may append the patient to a queue of patients waiting for the x-ray; or if no one is currently using the x-ray and no queue exists, the "send patient to x-ray" event will determine how long the patient will be at x-ray, and generate another event, "return from x-ray to the main treatment room."

In a typical computer simulation environment, the use of a simulation program involves three steps: 1) A set of parameter values is determined. 2) The simulation is run for a preset length of time or number of events, or until some preset condition is met. 3) The printed output of the values of variables deemed important is examined. These three steps may be repeated a number of times. All information communicated to the user from the program is transmitted through the "printout."

The philosophy of the TIGERS program differs from that of most other simulation programs. A simulation under TIGERS is an interactive graphical process, i.e. a process which maintains continuous two-way communication between the user and the program. The simulation communicates with the user, not through computer printout, but through a dynamic graphical display on a CRT scope. Through this medium, TIGERS keeps the user continually informed of the present state of the simulation. As he is monitoring the progress of the simulation, the user has the ability at any time to start or stop the simulation or make parameter changes or request information. Its interactive graphical nature facilitates use by the doctor or public administrator as well as by the trained analyst. More accurately, one might say that such a system is *accessible* by both, and therefore can hopefully serve as a focus towards creating a more effective team. In the next sections we shall discuss in greater detail the ramifications of these graphical and interactive aspects of the program.

4.1.2 [illegible] In a typical computer simulation environment [illegible] values is [illegible]

[illegible] shall discuss in greater detail the ramifications of the [illegible] interactive aspects of the program.

4.1.2 User Interaction with Simulation Programs

The key feature introduced by a high degree of interaction is great flexibility, since flexibility (as we shall discuss below) is largely a function of degree of interaction. Flexibility is desirable because it encourages the extraction of a large quantity of information from the program in a relatively short time. Degree of interaction can be broken down into three categories.

The "lowest" category is that of a so-called "batch programming" system. In order to execute one run of a simulation, the user decides upon a set of parameters, types them up (usually on punched cards), and submits his deck (typed program) in an input bin. Hours or days later the user can pick up his deck and printed output from his output bin. If the user wants to try out another idea, he must go through the same process again.

The next category of interaction is that of standard time-sharing. It is significantly better in that the user communicates with the program not through punched cards, but through an on-line teletypwriter. The user submits his job from his console, and the computer types the relevant results of the simulation on the user's console in minutes. This is, of course, a significant improvement over batch processing.

A third category of interaction is interactive not only in that the user can change parameters before running each version of the simulation, but also in that he can interact with the simulation at any time during its execution. Such interaction was strived for in designing the TIGERS program. The graphical trace allows the user to follow the simulation as it runs, and as long as it looks interesting, he can let it continue to run. If he sees an interesting state, he can temporarily stop execution and investigate the state in depth. If he wishes to change a parameter at any time during the simulation, he is free to do so. Thus simulation in a TIGERS-like environment becomes a process which the user can continuously observe and in which he can participate. Such a program is not bound to the three steps described earlier which characterize most other simulations; the graphical trace creates a situation such that the program can be run for as long, but only as long, as the user deems it useful.

4.2 Interaction With the Simulation

Implementing an E.R. model under TIGERS necessitates interaction with the program at more than one level. There is no rigorous hierarchy, but if forced to differentiate, one might isolate two primary facets of the use of TIGERS. On one level, the program is an interactive graphical simulation with which the user can experiment with a given model through use of the program's two operating modes. On another level TIGERS is a modeling environment where subroutines are written and data bases are structurally modified, or sometimes even newly created.

In this section we discuss in detail the use of the interactive program to experiment with a given model. We will refer to the experimenter as the *user* of the program. We describe below the man-machine interface and the use of TIGERS at the "user level."

4.2.1 Physical Facilities

The program TIGERS communicates with the user primarily through the graphical display on the CRT scope. The user, in turn, can give commands to TIGERS through the hand-held stylus (pen) and tablet. (There is also two-way communication through the teletypewriter, but it is generally unnecessary at the user level.) The facility is shown in Figure 4-1. The user sits in front of the CRT with the tablet horizontal directly in front of him. The pen and tablet allow the user to "point" to objects displayed on the CRT. The tablet is a surface that is sensitive to the position of the stylus. The surface of the tablet maps into the face of the scope such that touching a point on the tablet with the pen is like touching the corresponding point on the scope. In this discussion when we refer to "touching" an object on the display, what is actually meant is touching the corresponding point on the tablet. (In actual use, the user quickly forgets that he is really touching the tablet. Because the pen position is always displayed on the scope, the user feels as if he were touching the screen itself.) While the program is running, the user can at any time point to or touch displayed objects.

Certain objects called *display buttons* are sensitive to being touched by the pen, and it is through the display buttons that the user can give commands to and otherwise communicate with TIGERS. In TIGERS, all light buttons take the form of a label and an associated sensitized area indicated by a square. For example, the button to command initialization of the data bases before an execution of the simulation looks like this:

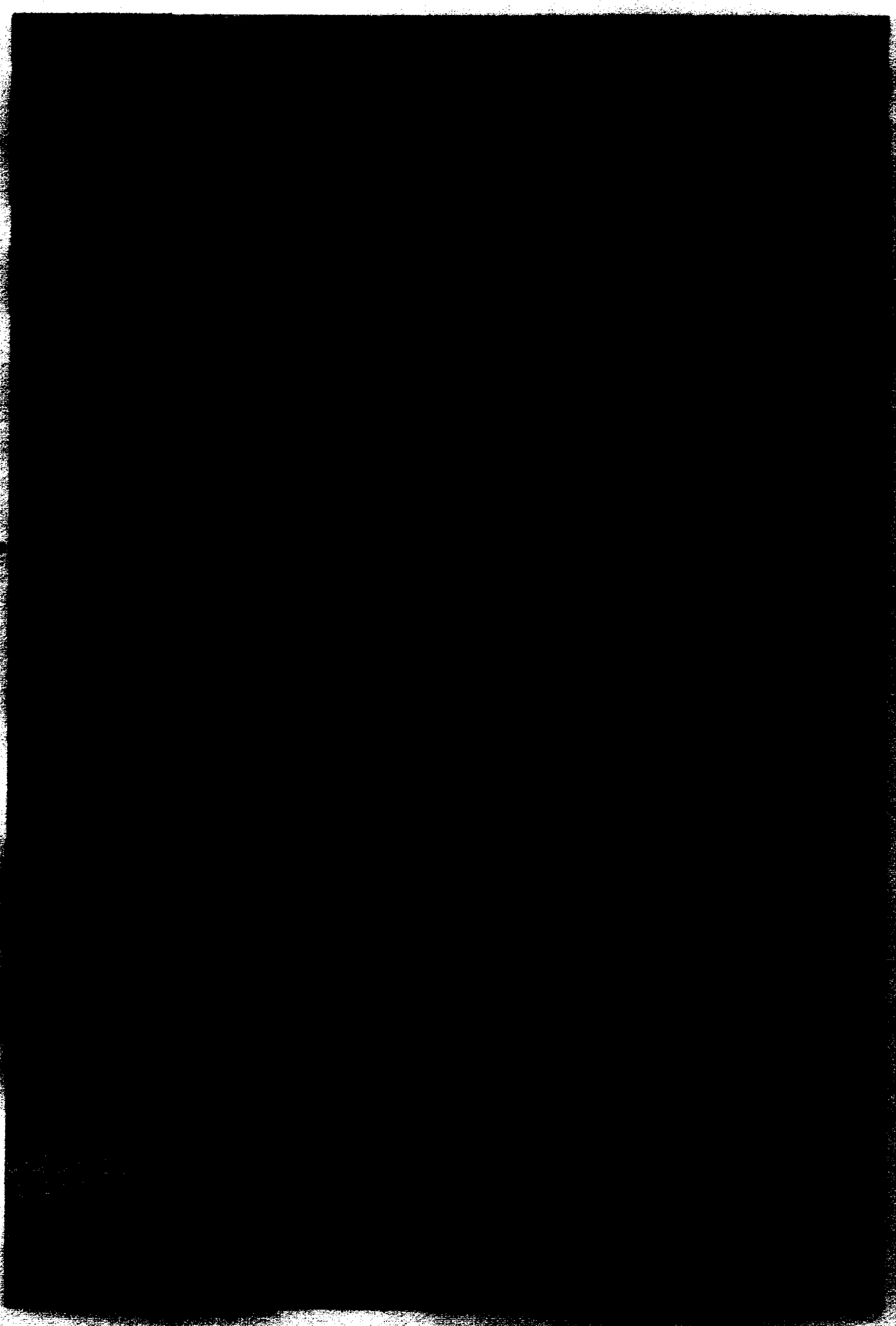


Figure 4-1 TIGERS User Seated in front of CRT and Tablet

☐ INITIALIZE

Whenever the square is "hit" (touched) by the pen, the states of all the resources are reinitialized. (A more detailed description can be found at the end of Section 4.2.2.)

TIGERS operates in two possible "modes," Modify and Simulate. Simulate is the mode in which actual simulations take place, and Modify is the mode in which the user can change the values of various parameters. When TIGERS is first started, the program is in Simulate Mode in a "stopped" state; i.e. the simulation is stopped, but ready to be started. TIGERS is inactive but listening, waiting for a command (i.e. for a button to be hit). The CRT in an initialized Simulate mode appears as in Figure 4-2. This is the stage upon which the scenario of the simulation takes place.

<u>4.2.2 Simulate Mode</u>

TIGERS' stage is composed of thirteen separate sectors: Main, Blocked, Waiting, Lab, X-ray, Shock, E.R. Staff, On Call, Lounge, Nurses, and three unlabeled sections. (Blocked, Shock, and Lounge are not used in the current implementation.) Each sector represents a different subsystem of the emergency room. The mobile elements of the system are the stick figures with various shaped heads (see Figure 4-7) which represent the patients and the emergency room personnel. For lack of better identifiers, nurses have triangular heads; doctors, hexagonal; consultants, octagonal; and patients, square heads. Numbers and types of persons in each sector are indicated by the presence of the

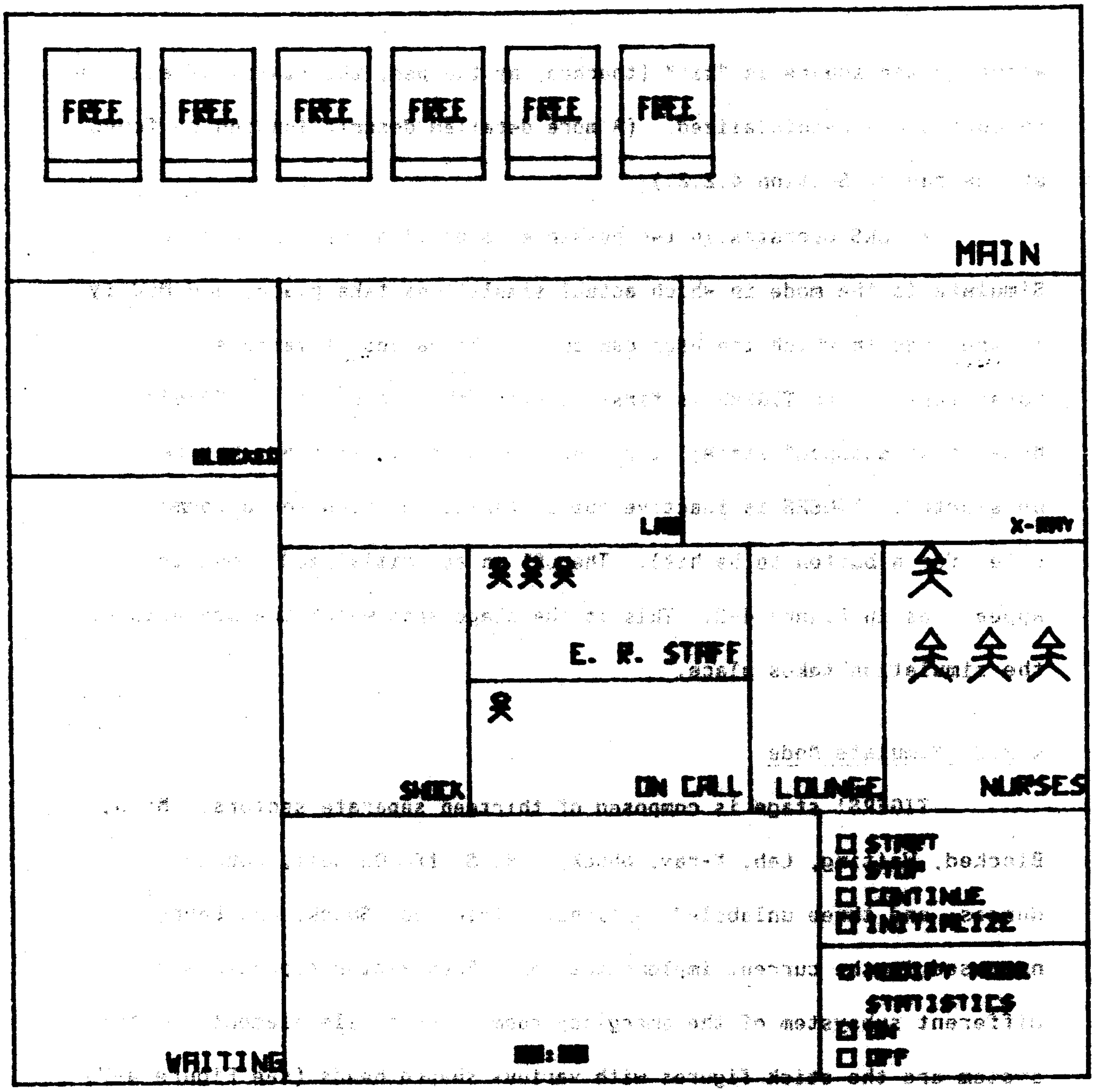


Figure 4-2 The CRT in Simulate Mode Just After Initialization

appropriate number of the appropriate symbol (stick figure) in the sector. The clock at the bottom of the screen indicates the elapsed simulated time since the last START of the simulation.

Three of the sectors are indicators of available emergency room personnel:

Nurses -- emergency room nurses

E.R. Staff -- doctors (including interns) assigned full time to the emergency room

On Call -- doctors who, although not stationed in the E.R., can be called into the E.R. when it becomes crowded.

These three sectors indicate number of staff who are on duty and not busy, i.e. ready to treat patients. Thus when the program is initialized, these sectors indicate the total number on duty, since at time zero none of the staff members are busy. Thus the parameters of the model of Figure 4-2 are initialized at four nurses, three doctors, and one doctor on call.

When the simulation is initialized, no patients are indicated in any of the sectors, because patient arrivals do not begin, of course, until the simulation has started.

The sector marked "Waiting" represents the waiting room. As the simulation generates arrivals, they are assigned to a bed if possible. If this cannot be done (e.g. if no bed is free, or all nurses are busy), the patient joins a queue in the waiting room.

The area labeled MAIN at the top of the screen represents the main treatment room. The bed-like objects represent beds. Associated with each bed are four "fields" (Figure 4-3). Field 1 is the patient location field (Figure 4-4). It indicates that either the bed is not in use, the patient is in the bed, or the patient is at x-ray. Field 2 is a patient state indicator field (Figure 4-5). It may indicate the following states:

> OBSERVED -- under observation. Often a patient will stay in an E.R. bed, sometimes for several hours, simply to ascertain that he is ready to be released. Patients with drug overdoses, for example, are often watched for a while before they leave the E.R.
>
> LAB WAIT -- lab blocked, waiting for lab test result. Often, a patient's treatment is delayed until his test results are back from the lab.
>
> EXIT WAIT -- exit blocked, finished with treatment, but exit delayed. Delay might be caused by administrative red tape, or sometimes simply by lack of transportation.

These states are critical because patients in these states account for much of the use of a significant E.R. resource -- bed space.

Field 3 is a "server indicator" field (Figure 4-6). It may indicate either that a patient is being served by one of the E.R. personnel or a consultant from the floor, or that the patient is in a "blocked state" -- receiving no treatment and waiting for such service.

Field 4 specifies the "type" of patient occupying the bed (Figure 4-6). Patients may be classified by type of treatment necessary; and by monitoring the types of patients in the main treatment room, the user can ascertain which types make the greatest resource demands of the system.

In a typical "scene" there might be a couple of patients in beds being seen by doctors, a queue of patients waiting for the x-ray, and several available E.R. personnel. A change of state (e.g. a patient moving from bed to x-ray) is displayed as it occurs. The user is able, for example, to see a patient sent to x-ray, see queues build, or notice

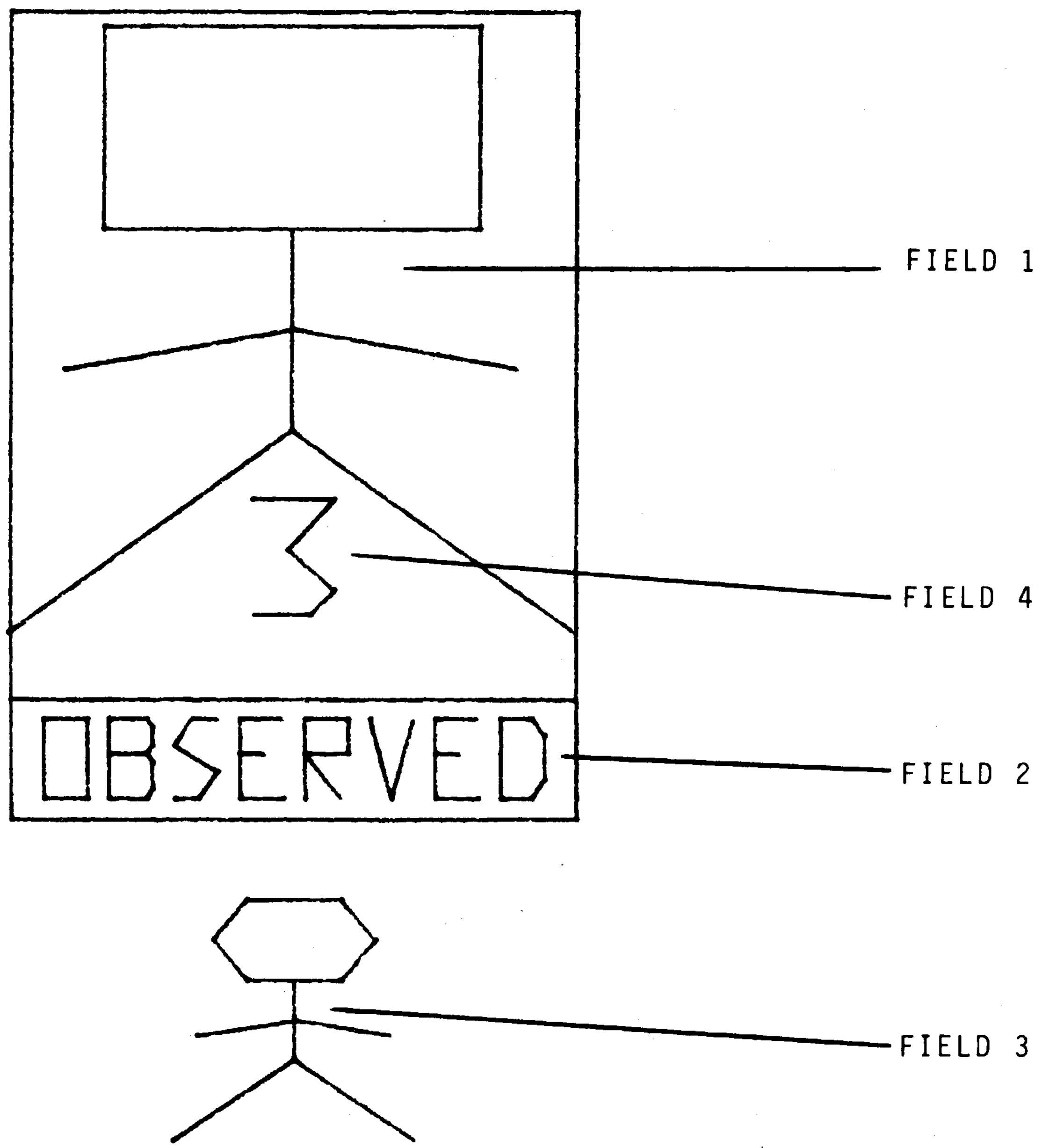


Figure 4-3 Four Fields of TIGERS Bed

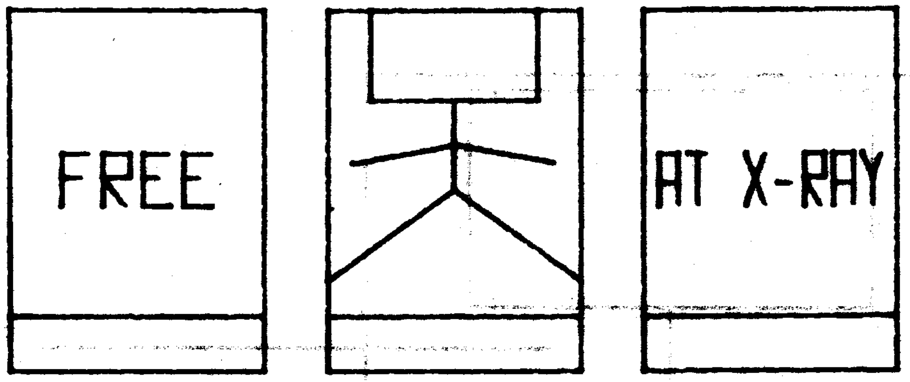


Figure 4-4 Bed Field 1: Three Possible States

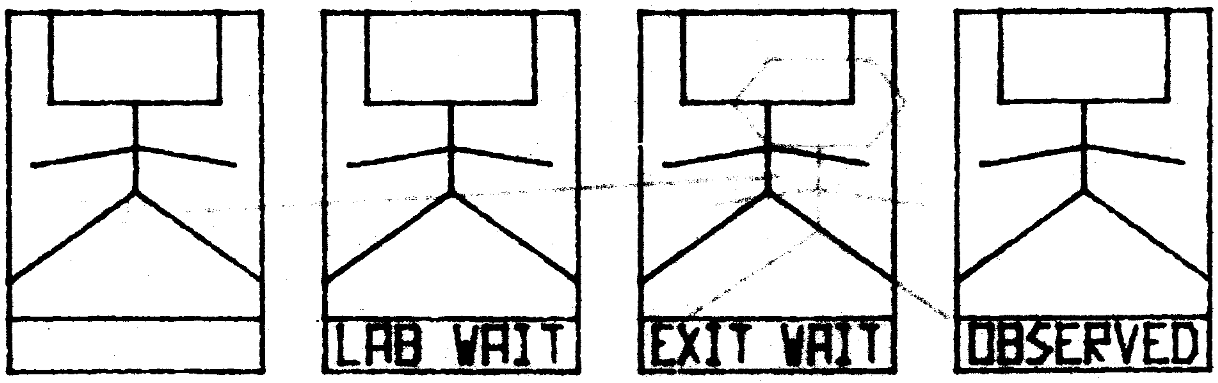


Figure 4-5 Bed Field 2: Four Possible States

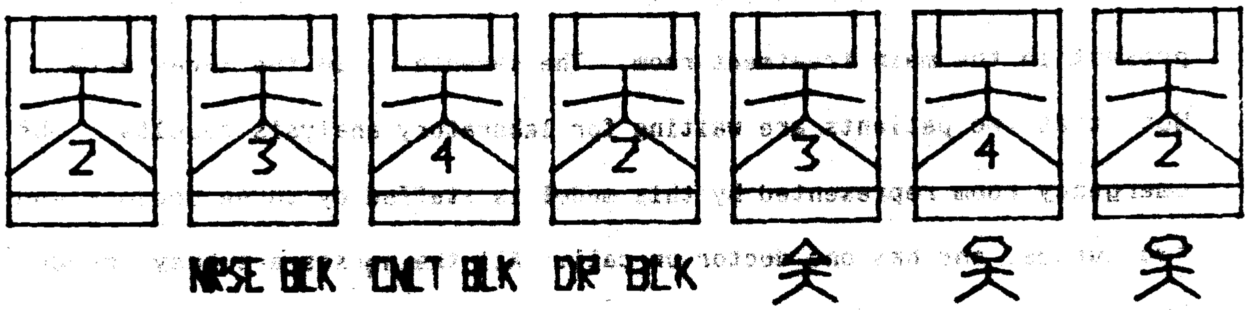


Figure 4-6 Bed Field 3 and Bed Field 4:
Seven and Three Possible States, Respectively

a patient in a blocked state waiting to see a doctor.

Figure 4-7 is a typical scene which might appear on the CRT during a TIGERS simulation. In the main treatment room we note seven beds. There are seven patients in this scene, six of which are actually present in the main treatment room. The seventh is in the x-ray area. Note that two patients are waiting for laboratory analysis results. The emergency room represented by this model is staffed by three doctors and two nurses, and has one doctor on call. All the nurses are busy (hence the patient in the nurse blocked state). One of the doctors is busy and the doctor on call is available. One patient is under observation.

An important aspect of the display is that it is intended to lend intuitive "feel" for what is happening as well as convey quantitative information. It is for this reason that queues of patients or staff are represented not as numerals, but as actual collections of stick figures. A crowd gathered in the waiting room conveys a much more intuitive idea of the property "crowded" than does the numeral "11." Similarly a blank space in the x-ray area conveys the ideas "empty" and "not crowded."

It might be argued that the numeral "0" communicates as well as the blank space, and that the numeral "11" does as well as eleven little figures. The author contends that this argument might be valid if one were only interested in the state of one data base, say, the waiting room. But when one is interested in the state of the whole system, the stick figures communicate much more effectively. Using numerals instead of queues of figures adds one level of abstraction. If one were only watching one queue, this abstraction might be insignificant; but when

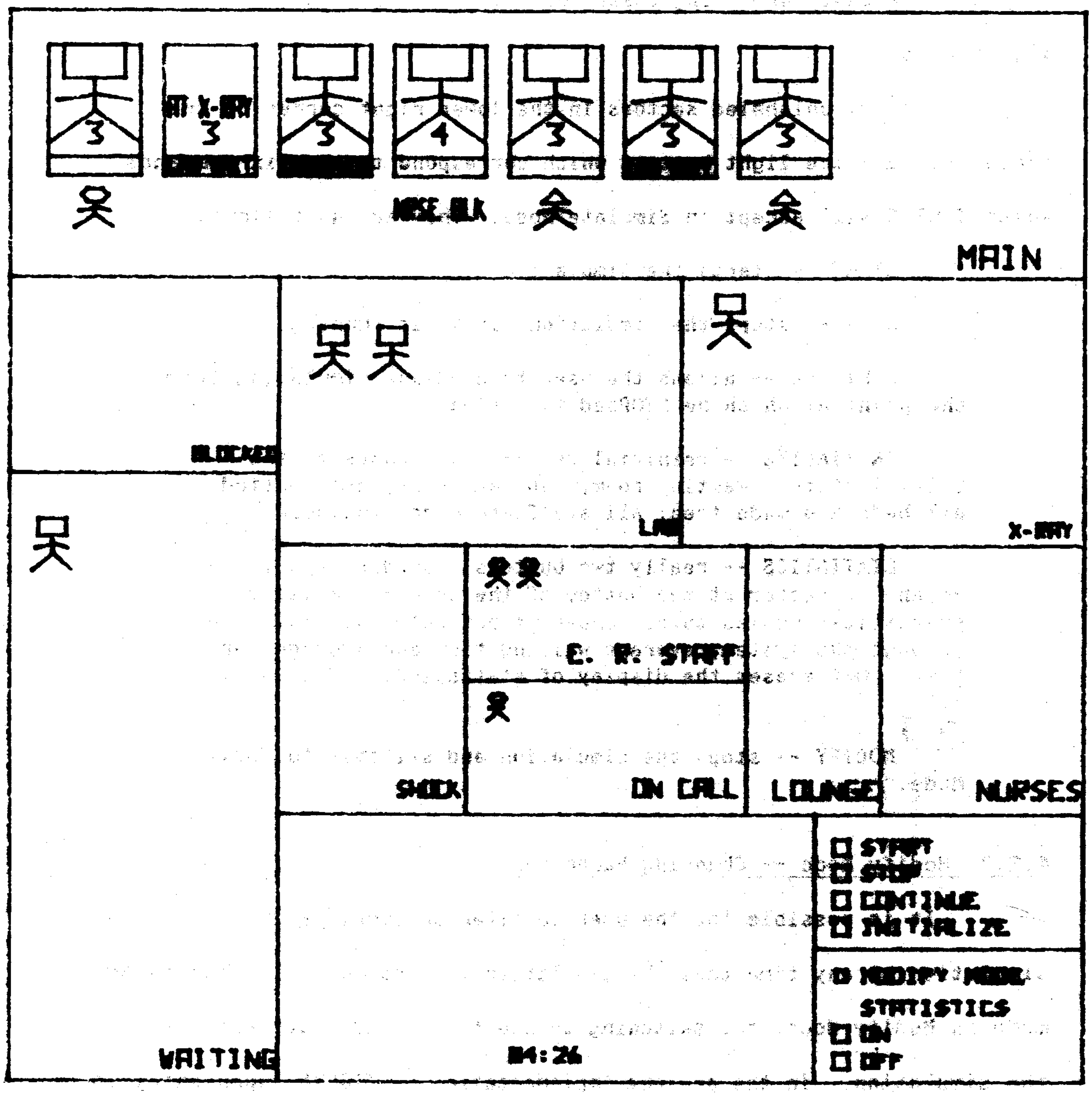


Figure 4-7 Typical Scene During a TIGERS Simulation

several data bases are being observed, the extra level does become significant.

The two unlabeled sectors in the lower right corner of the screen contain the light buttons which correspond to the six commands which TIGERS will accept in Simulate Mode. They are as follows:

START -- starts the simulation.

STOP -- stops the simulation, if it is running.

CONTINUE -- allows the user to continue simulating from the point at which he STOPped execution.

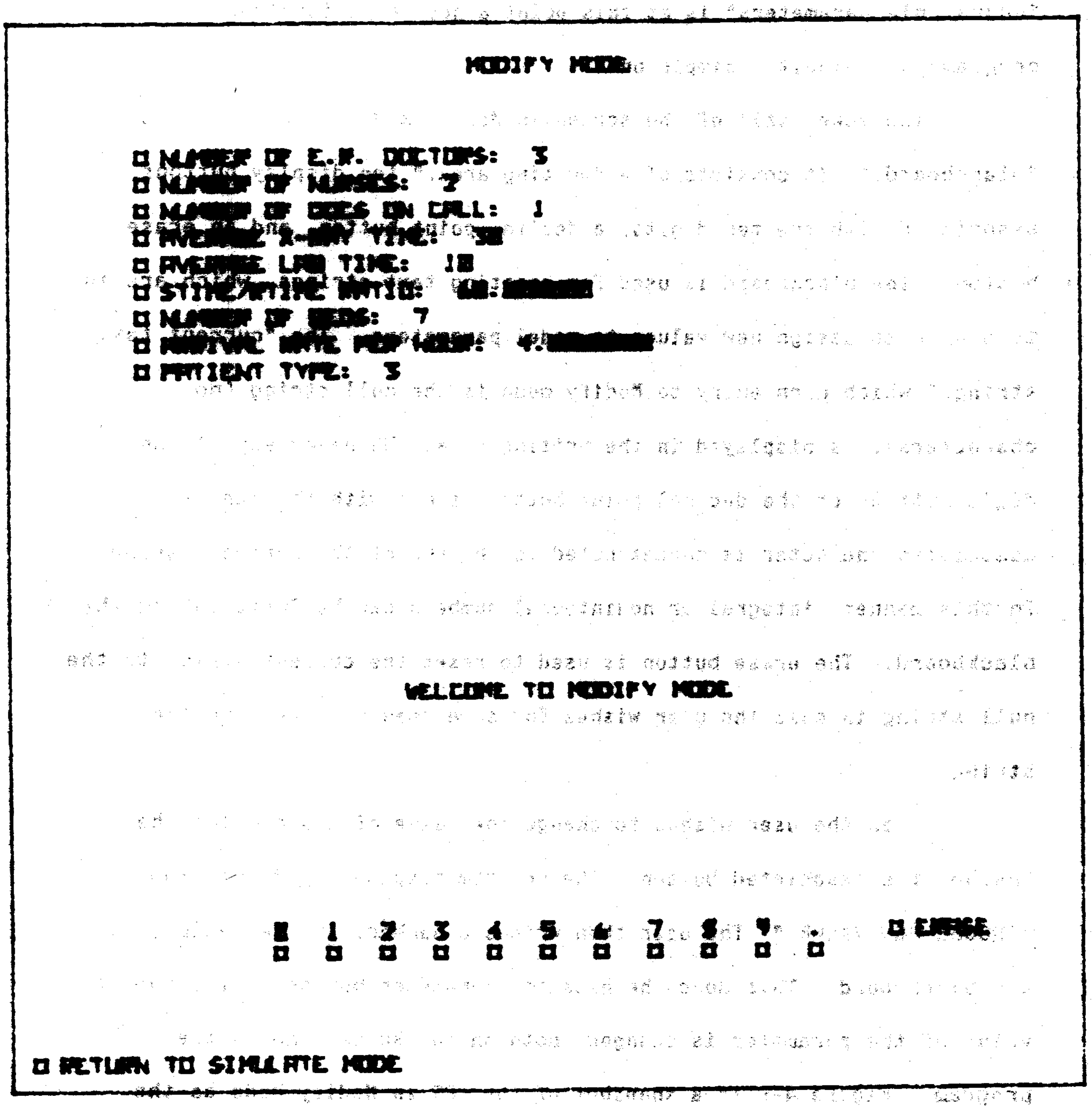
INITIALIZE -- reinitializes all data bases to the initial state. Waiting room, lab, and x-ray are emptied; all beds are made free; all staff are made available.

STATISTICS -- really two buttons. ON displays in the unlabeled sector at the bottom of the screen cumulative statistics such as total number of patients that have gone through the system, average waiting time and average service time. OFF erases the display of statistics. (See Section 5.5.)

MODIFY -- stops the simulation and switches to Modify Mode.

4.2.3 Modify Mode -- Changing Parameters

It is possible for the user to alter critical parameters of the simulation at any time that the simulation is stopped. All changes are made in Modify Mode, and switching to Modify Mode automatically stops the simulation. In the present implementation of TIGERS, upon entry to Modify mode the display appears as in Figure 4-8. The eight parameters in the upper left are the simulation parameters which the user can modify. It is a trivial operation to add or subtract from this set of parameters, but in the present implementation, there is no way for the untrained user to do this automatically. Altering the set of



MODIFY MODE

□ NUMBER OF E.R. DOCTORS: 3
□ NUMBER OF NURSES: 2
□ NUMBER OF DOCS ON CALL: 1
□ AVERAGE X-RAY TIME: 30
□ AVERAGE LAB TIME: 10
□ STIME/ATIME RATIO: 60:[illegible]
□ NUMBER OF BEDS: 7
□ ARRIVAL RATE PER HOUR: 4.[illegible]
□ PATIENT TYPE: 3

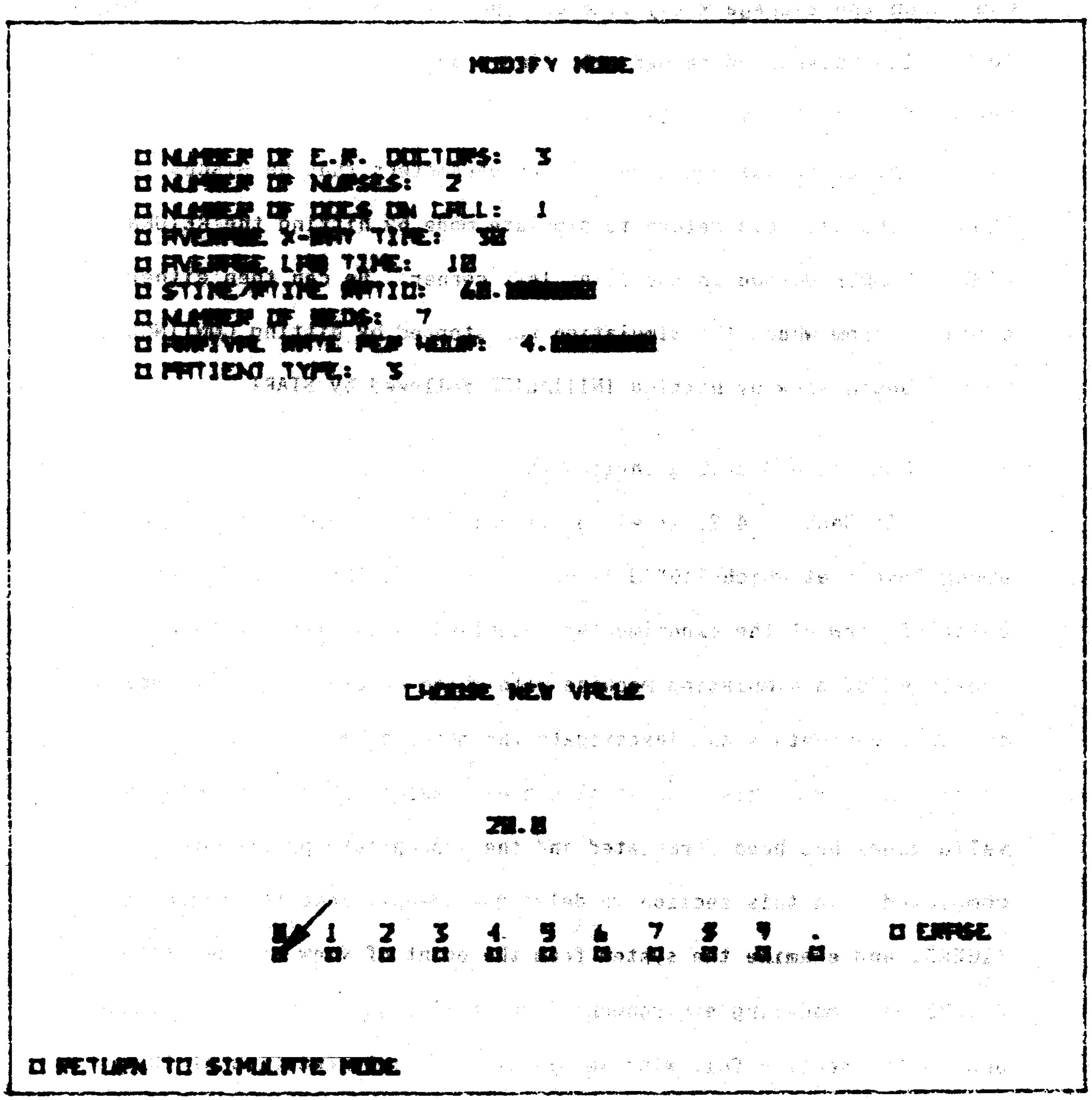
WELCOME TO MODIFY MODE

Figure 4-8 TIGERS Display Just After Switching to Modify Mode

"adjustable parameters" is at this point a job which involves programming, albeit a simple one.

The lower half of the screen in Modify mode is known as the "blackboard." It consists of a "writing area," ten display buttons associated with the ten digits, a decimal point button, and an erase button. The blackboard is used for creating text strings, which are in turn used to assign new values to model parameters. The "current text string," which upon entry to Modify mode is the null string (no characters), is displayed in the writing area. Whenever any of the digit buttons or the decimal point button is hit with the pen, the associated character is concatenated to the end of the current string. In this manner, integral or nonintegral numbers can be "written" on the blackboard. The erase button is used to reset the current string to the null string in case the user wishes for some reason to rewrite the string.

When the user wishes to change the value of a parameter, he touches the associated button. The program responds by displaying "CHOOSE NEW VALUE." The user then writes a number, the new value, on the blackboard. This done, he hits the parameter button again, and the value of the parameter is changed, both on the screen and in the program. Figure 4-9 is a snapshot of the CRT in Modify Mode as the blackboard is being used. The user is modifying the x-ray time parameter for Type 3 patients. He has already hit the parameter button and is now building the text string representing the new value. The pen is still on the "zero" button which appended the last "0" to the text string. The user's next step will be to hit the parameter button again,



MODIFY MODE

- NUMBER OF E.R. DOCTORS: 3
- NUMBER OF NURSES: 2
- NUMBER OF DOCS ON CALL: 1
- AVERAGE X-RAY TIME: 30
- AVERAGE LAB TIME: 10
- STIME/ATIME RATIO: 40.[illegible]
- NUMBER OF BEDS: 7
- ARRIVAL RATE PER HOUR: 4.[illegible]
- PATIENT TYPE: 3

CHOOSE NEW VALUE

20.0

0 1 2 3 4 5 6 7 8 9 . ERASE

RETURN TO SIMULATE MODE

Figure 4-9 Using the Blackboard in Modify Mode

whereupon the average x-ray time of Type 3 patients will be changed to 20.0. The interested reader will find a more detailed description of Modify Mode in Section A.13.

After he has modified all the parameters that he wishes to change, the user can return to simulate mode by hitting the RETURN TO SIMULATE MODE button in the lower left corner. He can then either continue from where the simulation was stopped by hitting CONTINUE, or he can begin anew by hitting INITIALIZE followed by START.

4.3 TIGERS as a Modeling Environment

In Section 4.2, it was mentioned that one might differentiate among levels at which TIGERS is used. We found that TIGERS, from the point of view of the experimenter described in the last section, is something of a simulating machine with which he can set the values of critical parameters and investigate the model's behavior under these conditions. But this user level can only exist after a (hopefully) valid model has been formulated and the appropriate programming completed. In this section we delve more deeply into the logic of TIGERS, and examine the system from the point of view of one using TIGERS as a modeling environment. We first discuss model formulation, and in the section following we discuss the implementation of the formulated model.

4.3.1 Creating a Model: World Lines

The task of formulating the emergency room model is formidable. The problems faced by the Markel et al. group (see Section 3.1.2) also face the analyst using TIGERS. However, the simulation does allow the analyst considerably more flexibility than that team had working only with an analytical model. The Markel et al. group was forced to aggregate all emergency room service into the two classes "doctor" and "non-doctor" services, whereas simulation can take into account seperately such peripheral services as x-ray, lab, outside consultant and nurse. Nevertheless, there is no avoiding the difficulty of the task of creating the model.

Developing a model in TIGERS usually means writing a scenario describing how given data bases are to be manipulated. A simple formalism has been developed for describing the model in terms compatible with the TIGERS environment: a *world line* is a possible course of events that a patient may experience from the time he arrives at the E.R. until the time he exits. The term is used to refer both to the actual sequence of events and to the schematic diagram describing it.

Consider the following possible sequence of events which an accident victim might experience.

1. Patient arrives at E.R.
2. Waits in queue.
3. Called into main treatment room by nurse.
4. Seen by nurse.
5. Seen by doctor.
6. Has blood sample taken and sent to lab.
7. Sent to x-ray.
8. Lab test results returned.
9. Seen again by doctor.
10. Seen gain by nurse.
11. E.R. treatment terminated.
12. Admitted to hospital.

This sequence of events can be described by the world line shown in Figure 4-10. Another patient with say, a sore throat, might have a much less eventful E.R. visit described by the following set of events:

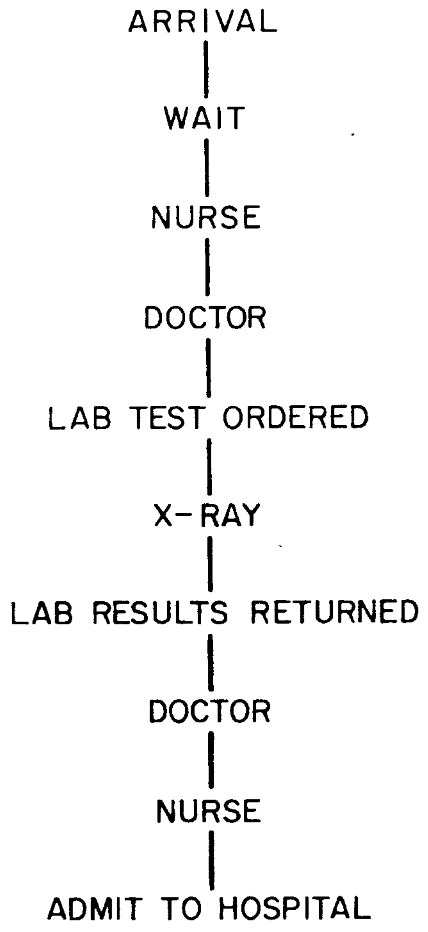


Figure 4-10 Example World-line (long)

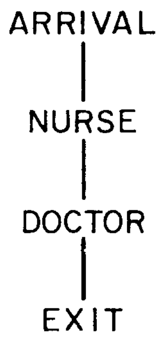


Figure 4-11 Example World-line (short)

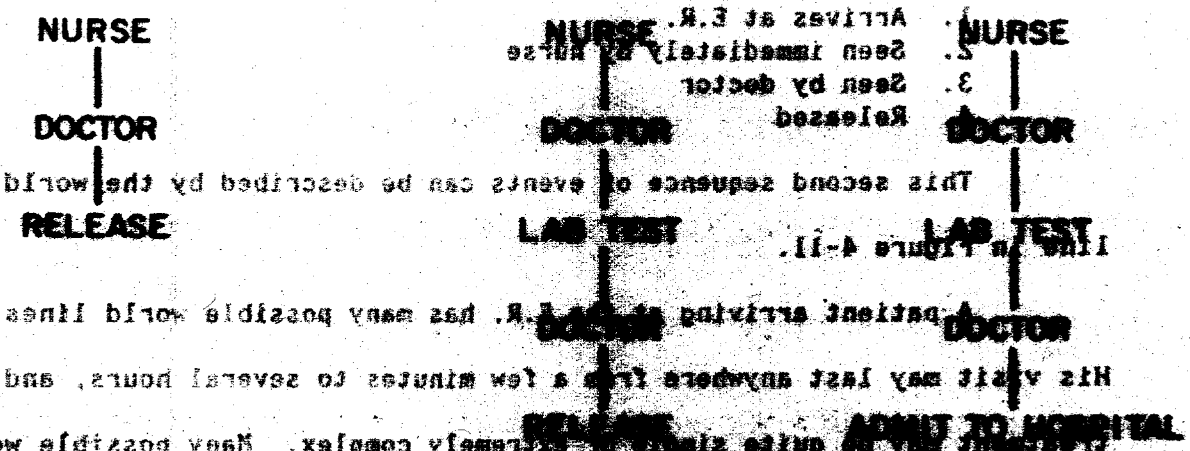
1. Arrives at E.R.
2. Seen immediately by nurse
3. Seen by doctor
4. Released

This second sequence of events can be described by the world line in Figure 4-11.

A patient arriving at the E.R. has many possible world lines. His visit may last anywhere from a few minutes to several hours, and his treatment may be quite simple or extremely complex. Many possible world lines can be combined in a *world line tree*. The world line tree is a construct which can be used to indicate several possible sequences of events which a patient may experience. Consider for example the three world lines in Figure 4-12. A patient in a hypothetical (and highly simplified) emergency room might always experience one of these three sequences of events. In Figure 4-13, we have a world line tree which incorporates all three world lines of Figure 4-12.

Whereas the world line indicates one definite path of a patient through "event space," the world line tree indicates a set of possible paths. Thus if one is told that a patient has an associated world line tree similar to that of Figure 4-13, he knows that the patient's path through event space will be one of the three world lines of Figure 4-12.

Each node of a world line tree with a unique line branching out from it indicates a point in space-time where only one thing, one event, can happen next. This event is indicated by the single branch. A node with more than one outward branch, however, indicates a point in space-time where several possibilities exist. In order to indicate the likelihood of the various possible events, a probability is generally associated with each branch of the tree such that the sum of the



[illegible]

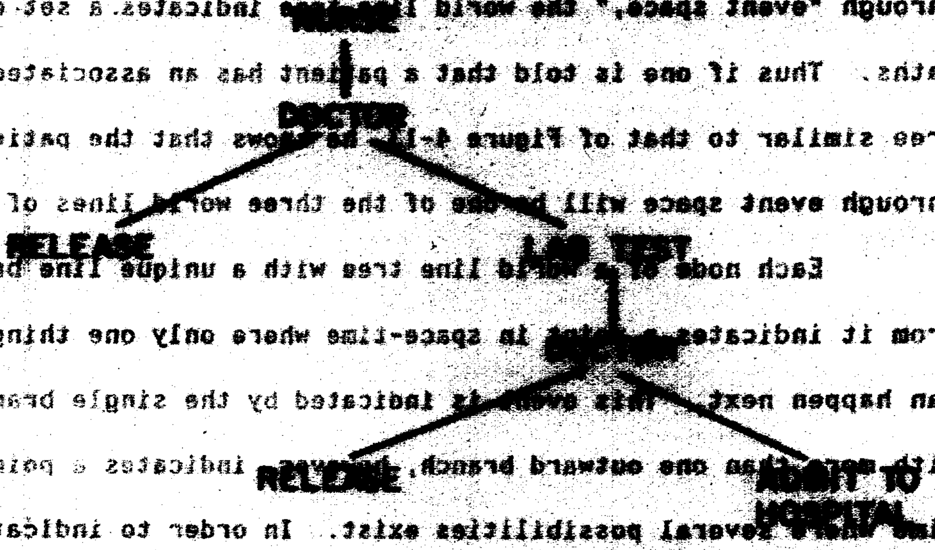


Figure 4-13 World-Line Tree
Incorporating Three World-Lines of Figure 4-12

probabilities associated with the outward branches of any node is one. It is also desirable to have information about how long the patient remains in the state represented by a node. Therefore in addition to associating probabilities with each branch, a conditional probability distribution function (conditional on which branch is chosen) is generally associated with each node.

It can now be noted that the world line tree provides a mechanism to fulfill the need expressed in Section 4.2. In that section it was stated that developing a model in TIGERS means writing a scenario for the way in which the given data bases are to be manipulated. We note that this can be done in three steps:

1. Formulate a world line tree which incorporates all the possible world lines that might be associated with a patient arriving to the system.

2. Assign a probability to each branch of the tree, such that the sum of the probabilities of the branches below each node (except the terminal nodes) is unity.

3. Assign to each node n probability distribution functions (where n is the number of branches emanating from the node) that give information about how long a patient will spend at that node, if he reaches it.

In real life, the world line tree associated with a patient arriving at the E.R. has an extremely large number of branches and nodes. However, the probabilities associated with most of the branches are near zero. The TIGERS program differentiates among patient types by assigning different sets of nodal pdfs and branch probabilities to each type of patient. (The programmer can effectively assign different trees to each type by setting certain probability variables to zero.) The job of the analyst in designing a model in TIGERS is to formulate trees which ignore the largely irrelevant branches and incorporate the relevant ones. Once again we are faced with the trade-off of accuracy versus programming overhead. Clearly if the model incorporates details that have little effect on parameters of interest, computation-time is wasted. On the other hand, if not enough detail is included, the model may not provide useful results.

Thus although he does have considerably more flexibility than he would have if he were using analytical methods similar to those of Markel et al., the analyst-user of TIGERS still cannot avoid the need to make careful simplifying assumptions in generating a model.

The analyst's task is indeed far from simple. "Pruning" the tree without invalidating the model's results is difficult enough. But in addition, once the analyst has decided on the "shape" of the world line tree, it remains to assign values to the node-pdf's and branch probabilities. The only way to do this is to take careful data in the actual emergency room and calculate appropriate values from these data.

4.3.2 Implementation

We have said that TIGERS can be described as an environment in which the analyst can implement an emergency room model. Implementing a model implies writing a simulation. Even within TIGERS, the analyst must still write the simulation; the program will not do this for him (at least not yet). However, TIGERS does make the job significantly easier; it provides a set of subroutines, protocols and data bases which the analyst would otherwise have to design himself. For a large class of models, the analyst does not have to write new routines or create new data bases himself. Rather the process of implementing a model usually involves modifying the data-bases and routines already supplied in the system. The main features which TIGERS offers the analyst are

1. A "simulation frame" consisting of a protocol for defining events, an events scheduler which keeps track of events and executes them at the appropriate times, and routines for adding events to the "schedule."

2. A set of data bases and routines for managing them.

3. A "basic set" of common events such as "send patient to x-ray," "schedule lab report," "generate new arrival," "release patient from system," and several others.

4. A set of graphics routines. Almost all of the graphics is handled automatically by TIGERS. TIGERS supplies functions such that any graphics handling the analyst needs to do is trivial.

The interested reader should consult Appendix A, in which these resources are discussed in depth.

CHAPTER V:
AN IMPLEMENTED MODEL

5.0 Introduction

Chapter IV introduced the TIGERS environment for modeling the emergency room and described interaction with the program. Since it was desirable that the environment be demonstrated and tested by at least token use, a preliminary model was designed and implemented on the system. That model is the subject of this chapter.

In designing a model, one must continually keep in mind what it is he wishes to accomplish with it. Key questions include 1) In what aspects of the system under consideration is the experimenter interested? and 2) What parameters influence these aspects? Specifically, which aspects of a patient's visit to the emergency room are being investigated with the model, and which factors associated with his visit will have some effect on these items of interest.

5.1 Design of the Model

Section 3.1.1 discussed which aspects of the emergency room were of interest in the current research. We established that for our purposes the emergency room can be viewed as a server system, and that the following parameters are relevant indices of the operational effectiveness of the this system:

1) number of patients in waiting room queue

2) time patient is kept waiting before service begins

3) time a patient actually spends in service

4) fraction of patient service time not usefully spent

Essentially, the task decided upon was to combine the available emergency room resources in such a way that the patient time spent in the system is kept to a minimum without sacrificing quality of medical care, thereby minimizing patient blocked time.

As has been discussed at length in Chapter III, the definitions of blocked time and service time are not easily arrived at. We noted that a fine line exists between the two. For purposes of the current model, it was decided that any state in which the patient is using no emergency room resource except a bed shall constitute a blocked state. Otherwise the patient is said to be receiving service. Thus time spent waiting for a laboratory report or under observation is not considered blocked time. The reasoning behind this decision is that a patient should be considered blocked only if he is simply taking up space. Therefore when a patient is in a blocked state, it is an indication that some resource is lacking at that moment, or that something is blocking smooth operation of the system. Thus for the current implementation, the four blocked states are waiting for doctor, waiting for nurse, waiting for consultant, and waiting to be released from the system; that is, doctor blocked, nurse blocked, consult blocked, and exit blocked.

In Chapter III it was mentioned that one of the advantages of a simulation model over an analytical model a la Markel et al., is that the simulation is capable of taking into consideration considerably more detail than the analytical model. However, unnecessary detail can only impede the modeling effort. It is desirable therefore in building the simulation model to take advantage of the capability to explicitly manipulate details, but not waste time on non-productive computation. Ideally, one wishes to consider explicitly in his model all such factors that do have significant effect, while at the same time keeping the model as simple as possible. Too much detail is a common pitfall in the design of simulation models.

As a first step in the building of the model, a number of questions were formulated which explicitly focus on important factors contributing to patient time spent in the system and E.R. resources used. These questions served as a basis for the model design. The list of questions follows:

Is the patient's problem especially non-serious? This question is important because, as was mentioned in Chapter I, a large number of patients requesting treatment at the emergency room have in fact very minor problems. (See Figures 2-1, 2-2, 2-3.) Such a patient should be dealt with explicitly in the simulation model because he demands rel atively little of the emergency room's resources, and his stay is generally short.

Does the patient need x-rays and/or laboratory analysis? These are two important hospital resources, the use of which always makes a patient's stay in the emergency room significantly longer. Furthermore if these resources are needed, the patient frequently uses more of the valuable resource of E.R. personnel time. Delays caused by queues at x-ray or backlogs in the lab also cause a drain on bed space and often constitute blocked time. Thus a bottleneck in the x-ray or lab can eventually manifest itself in longer service times and longer waiting room queues.

Is an outside consultant required? Sometimes the emergency room doctor finds that it is necessary to call in a specialist (e.g. a surgeon) from elsewhere in the hospital. Even more than x-ray or lab use, calling in an outside physician is a sign that the patient being treated has a more complicated problem than most and will therefore make relatively large demands on E.R. resources. Furthermore delays incurred while waiting for the outside physician to arrive constitute blocked time.

Does the patient require observation? Such patients are of concern because they often occupy emergency room beds for hours.

Is the patient admitted to the hospital proper? This question is important because often much time is spent in administrative overhead in admitting a patient to the hospital. Such time can be considered blocked time.

Once this set of questions was formulated, the next step was to devise a simulation algorithm to implement various possible world lines of patients. As stated earlier, the objective was *to make such an algorithm as simple as possible without sacrificing the ability to answer explicitly the above questions.* The aggregation technique used by Markel et al. was again employed in the design of this algorithm, although fewer simplifying assumptions were made.

The world line tree upon which this algorithm is based appears in Figure 5-1. Briefly the algorithm can be described as follows: A patient is seen by the doctor one, two, or three times, depending upon the complexity of the treatment needed. On visit one it is first decided whether the patient has a problem which can be especially easily and quickly treated. If so, the patient leaves the system quickly and does not see the doctor again. Otherwise a second visit is scheduled. It is also during visit one that it is decided whether the patient will use the x-ray or lab facilities.

Doctor visit two does not occur until after the lab report (if it was ordered) is returned and the patient has returned from x-ray (if he went). During visit two it is decided whether an outside consultant physician is to be called.

The third doctor visit occurs only if a consultant was called. The patient may see the doctor a third time when the consultant has left. Note that *the more complicated a patient's case, the larger the demand on the E.R. facilities and personnel.*

At this point the reader interested in further technical details should read Sections A.7.2 and A.8 which describe the model in depth. Figure A-5, which is especially useful, is a flowchart which describes the algorithm used to implement most of this three-visit model. An important aspect of the algorithm is that all decisions based on chance are binary; the entire algorithm is based on yes-or-no questions with probabilities of yes (or no) set by the user. It is felt that by making all such decisions binary, the user can manipulate the model with more

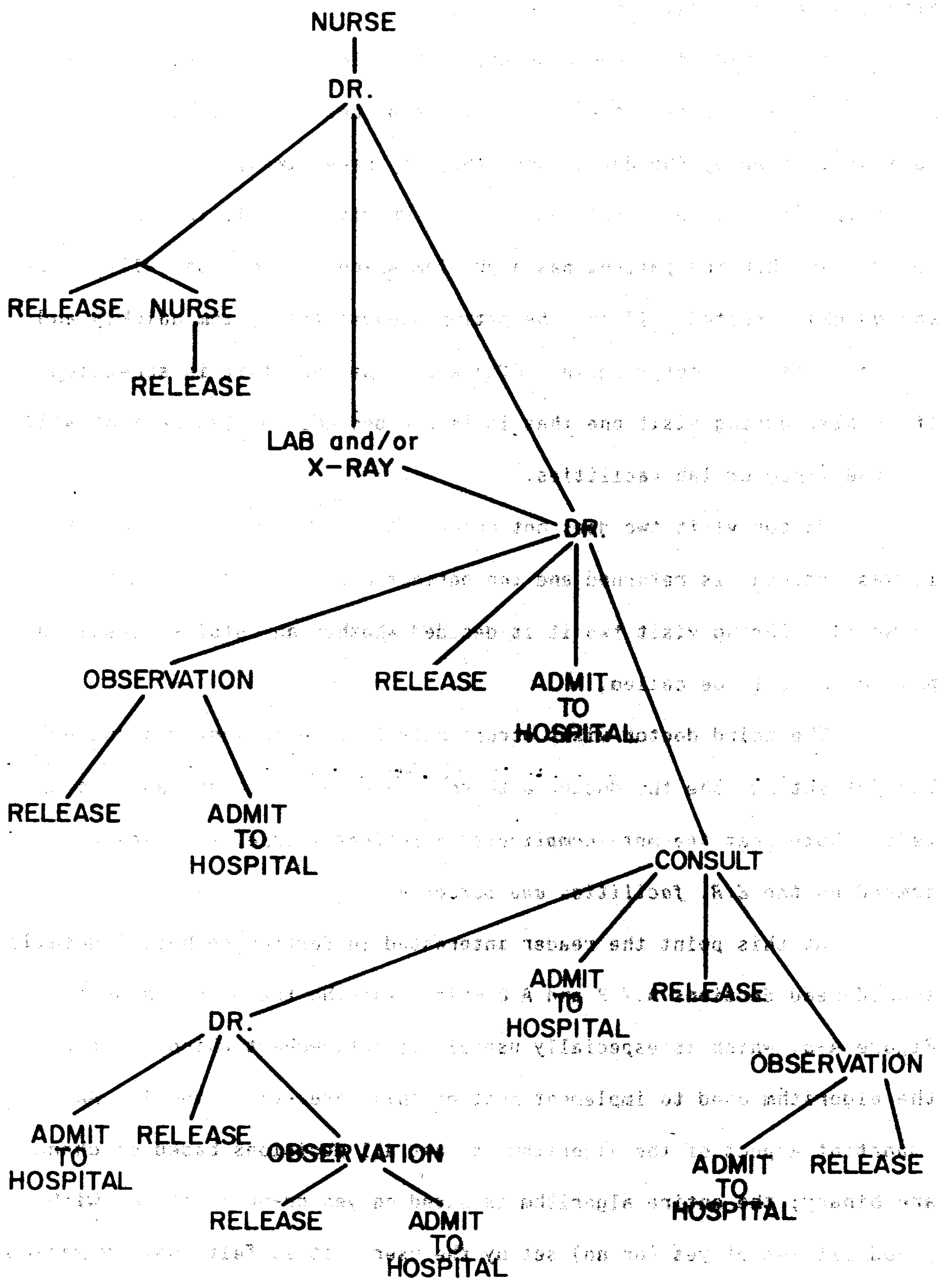


Figure 5-1 World line Tree of Current Model

control than he might otherwise.

A goal of the model design was that major factors influencing length of patient stay and amount of resources needed to be explicitly variable in the model. These are the variables in which the experimenter will be interested. In Section A.8 are itemized all parameters which are set by the experimenter in the current implemented model. We note that each of the critical questions mentioned above is explicitly considered in the model and can be varied by the experimenter.

5.2 Defining Patient Types

The world line tree discussed in the previous section represents all possible world lines for patients under this test-model. The values assigned to each node and branch, however, may differ with each type of patient, as noted in Section 4.3.1. For the sample model, it was decided that three types of patient would be implemented.

In choosing to implement three types of patient in this model, the experimenter divides the population of arriving patients into three classes. *The criteria upon which this division is based depend upon the information the user wishes to extract from the model*. By setting certain world line probabilities to unity and others to zero, the user can effectively assign a single type to all patients with certain world lines or classes of world lines. In this manner, the user can isolate any particular classes of patients that might hold special interest.

For example, suppose the user were an emergency room planner interested in drug-overdose patients who spend a great amount of time in the emergency room bed under observation. These patients, once initial treatment is over, often require little additional treatment. They do, however, often occupy a bed for hours, thus making a significant demand upon the E.R. resources. The planner might be looking for ways to reduce the demand on an overloaded emergency room, and considering setting up a special observation area for such drug-overdose patients. In setting up his TIGERS model, this user assigns all patients of this class to one type, say type 2: First he sets the arrival rate for this type to the appropriate number based on data gathered at the hospital. Then he assigns the value one to the probability parameter which describes the probability of of a patient's undergoing observation. Finally he assigns parameter values for the other types, based on the fact that patients of these types and patients of type 2 are mutually exclusive.

When the experimenter finally runs the simulation, he is able to explicitly observe how much of the overcrowding is being caused by the type 2 patients. Also, by setting the arrival rate to zero, he can observe the effects of removing this type entirely from the emergency room (presumably to the special observation area). He might notice that the removal of the type 2 patients had little effect, or he might notice that the introduction of the special observation area has solved the overcrowding problem of the main treatment room.

5.3 Assigning Parameter Values

Before running a simulation under TIGERS, the experimenter must assign values to all the variable parameters of the model for each patient type. Figures 5-2 and 5-3 are the sheets which the experimenter must fill out in order to supply parameter values for the currently im

Value		Variable
X,.3,.5	1	XR? -- Is an x-ray necessary?
X,.3,.6	2	LAB? -- Is a laboratory analysis called for ?
1, 0, 0	6	EXIT1? -- Does the patient exit immediately after the first doctor visit ?
.5, X, X	3	NUREX? -- If the patient does leave immediately, does he see a nurse first ?
X,.2,.7	4	ADMIT? -- Is the patient to be admitted to the hospital proper ?
X, 0, 1	7	CNSLT ? -- Does the patient see a consult ?
X, X,.X	5	OUT-AFTER-CONSULT ? -- If the patient does see a consult, does he exit immediately after seeing a consult ?
X,.2,.2	8	OBS ? -- Does the patient undergo a period of observation before he leaves the system ?

Figure 5-2 Currently Variable Branching Probabilities
(Typed numbers refer to position in Probsch vector.)

Value	Variable
X,45,45	1 ADMITT -- time spent in administrative "red tape" waiting to be admitted to the hospital
X,25,25	2 ADMIT2 -- time spent in administrative "red tape" waiting to be admitted to the hospital, after having undergone a period of observation
X, X,25	3 CNSDRT -- time spent with a consult
X, X,10	4 CNSLTT -- time spent waiting for a consult
X, X,20	5 DRCNT -- time spent on second doctor visit if consult is scheduled
S, X, X	6 DRIMEX -- time spent with doctor before immediate exit
X, 10,10	7 DRNXLT -- time spent with doctor on first doctor visit if lab report but no x-ray is scheduled
X, 1, 1	8 DRNXNLT -- time spent with doctor on first doctor visit if no lab report and no x-ray are scheduled
X, 15, X	9 DRT2 -- time spent with doctor on second doctor visit if no consult is scheduled
X, X, 10	10 DRT3 -- time spent with doctor on third doctor visit
X, 10,10	11 DRXT -- time spent with doctor on second doctor visit if patient is sent to x-ray
X, 15,15	12 EXITT -- time spent in exit blocked state before leaving emergency room
X,20,20	13 EXIT2 -- time spent in exit blocked state before leaving emergency room, if patient has undergone period of observation
99,30,30	14 LABT -- time before laboratory analysis results are returned
12,12,12	15 NURST -- time spent with nurse
X, 60,60	16 OBSRVT -- time spent under observation
99,12,12	17 XRAYT -- time spent at xray

Figure 5-3 Currently Variable Means of Random Variables

plemented model. The former comprises the probability variables; and the latter, the means of the various probability distribution functions. (Certain of these variables are currently attached to display buttons. The others can all be varied through the teletypewriter. As has been discussed in Chapter IV, it is not difficult to attach frequently changed variables to display buttons.) The ordered triplets which have been filled in the blanks of Figures 5-2 and 5-3 represent values which have been assigned by the author for types two, three, and four, respectively. (Due to an idiosyncracy of the programming, the three types are labeled two, three, and four rather than one, two, and three.) The values assigned are not based on rigorous and detailed data. But they are based upon the author's experience in the Cambridge Hospital emergency room and do represent reasonable values.

For this model, the types have been defined as follows: Type 2 represents the patient with a trivial problem. In the three-doctor-visit model, such cases typically see a nurse only once, see a doctor only once, and are in the E.R. for a relatively short time. Although a significant percentage of the patients arriving at the typical emergency room are in fact of this type, they represent a relatively small drain on the system because they use so few resources and their stays are short. Type 3 represents the patient who has a somewhat more serious problem, and therefore makes significantly greater demands on the E.R. resources. In the three-doctor-visit model, the Type 3 patient sees the doctor twice, and sometimes makes use of various other E.R. resources. The Type 3 patient represents the greatest load on the E.R., since, although the demand per patient is not as great as that of the Type 4 patient, the Type 3 arrival rate is generally much higher. The Type 4 patient represents the most serious cases. Type 4 patients always need to have a consultant called in from the hospital proper, and generally demand more of the available resources than any other type of patient. As already noted, however, they generally occur relatively infrequently.

5.4 Using the Model

The purpose of this section is to communicate an idea of what it is like to sit in front of the CRT and tablet and use the TIGERS program. The author requested Dr. Peter Mogielnicki of the Cambridge Hospital to act as a "token user" and to experiment with the program as he wished. We describe here a part of that session. It would be desirable to have a motion picture to accompany the discussion, but we shall do the next best thing and refer to the figures at the end of this section.

It is important that we note that the displayed statistics, as they are now implemented, can be a bit misleading. First, the displayed statistics are based on patients *no longer* in the system. Any patients visible anywhere on the screen are still in the system and therefore do not affect the cumulative statistics. Secondly patients in the system a long time ago have as much weight in the average as patients just released. It may prove desirable to weight the more recent patients heavier in calculations of averages (so that effects of more recent changes are more easily discerned). Bearing these facts in mind, we continue with our description.

Initially the parameters of the model were set as in Figures 5-2 and 5-3. The program was initialized to five beds, two nurses, three emergency room doctors and one doctor on call. At this point, the screen appeared as in Figure 5-4. The simulation was then started.

This hypothetical emergency room appeared at first to be functioning smoothly, but this was an illusion attributible to the fact that the system begins operation empty -- with all beds free and all facilities available. As soon as the facility had had a chance to fill up, a queue began to form in the waiting room and grew steadily larger. It became clear that for some reason this emergency room was not able to perform adequately in the face of the demands being made of it. Thus after four hours and forty-seven minutes the simulation was stopped, with the "stage" appearing as in Figure 5-5.

Dr. Mogielnicki commented about there being ten patients in the waiting room queue, while four staff members were idle. Clearly, an imbalance among the resources was highly probable. We mentioned earlier that a program such as TIGERS is especially useful for making the user aware of the necessity of maintaining balance among the resources of a complex system. Here the two observers (Dr. Mogielnicki and the author) were confronted with an example.

The next step was to try to alleviate the imbalance. It was clear that adding more staff would prove fruitless: already there appeared to be more than could be utilized effectively in such a system. Also there were no unreasonable queues at x-ray or in the lab. The bottleneck appeared to be that there was not enough bed space to accomodate the patients. Thus the next step was to try introducing more bed space into the system.

Two more beds were added; nothing else was changed. The simulation was started again from time zero. The extra beds apparently solved the problem. Figure 5-6 shows the system after two hours and three minutes of elapsed simulated time. Recall that even with the five-bed system, the simulated E.R. appears to function smoothly at first because at time zero all beds are empty. Therefore the simulation was allowed to keep running, and the two observers watched.

Several interesting aspects of the system became apparent as the simulation ran. First, the characteristics of the various types took on special meaning with respect to their respective demands upon available resources. Type 2 patients, for example, were interesting in that they were not seen in the system for nearly as much time as the others, in spite of the fact that the arrival rate of Type 2 patients was 2.0 patients per hour compared with Type 3 at 2.0 and Type 4 at 0.8. Type 4 patients~~,~~ on the other hand, were seen in the system a surprisingly high proportion of the time, considering their low arrival rate. Type 3 patients were clearly the main drain on the system: unlike Type 2 patients, they used a significant amount of available resources, while their arrival rate was as high as that of Type 2. Although the arrival rates of the Types 2 and 3 patients were equal, there seemed to be more Type 3 patients because they were in the system so much longer. Dr. Mo-gielnicki remarked that if he were considering triaging certain types of patients to a clinic in order to reduce the load on the system, Type 2 patients would not appear to be the type to choose, since they are in and out of the system so quickly.

The system as it ran seemed reasonably stable. There seemed to be an inordinate number of patients in nurse-blocked states (see Figure 5-6 for example), but the problem did not appear serious enough to impede the functioning of the system. The waiting room queue held as many as four patients (at 5:26--see Figure 5-9), but within thirty minutes had decreased to zero (Figure 5-10). In fact, at 6:13 (Figure 5-11), the E.R. was practically empty. In spite of the fact that the number of patients in the nurse-blocked state seemed to indicate that another nurse would not have been wasted, the two extra beds solved the overcrowding problem of the five-bed system.

At this point Dr. Mogielnicki, considering the rate at which arrival rates were increasing at the Cambridge Hospital Emergency Room, wondered whether this seven bed system would be able to support a heavier load. Switching to Modify Mode (Figure 5-12), he changed the arrival rate of Type three patients from 2.0 to 4.0 patients per hour, thus increasing the arrival rate of patients of all three types from 4.8 to 6.8 patients per hour.

The CONTINUE button was then hit, and the simulation continued with the higher arrival rate. By 8:16, all seven beds were full (Figure 5-13). Nurse-blocked states continued to be evident. By 9:20, there were four patients in the waiting room queue (Figure 5-14). The two observers began to suspect that the higher arrival rate might be more than the system could comfortably cope with. At 10:10 the waiting room queue had reached a length of 8 (Figure 5-15), and the observers decided that their hypothetical emergency room would not handle the increased load.

The question of how to bring the system back into balance once again arose. Dr. Mogielnicki decided that the present configuration might simply not be large enough, but he was not convinced that the two extra beds had brought the system into balance. Recalling the inordinate number of nurse blocked states that had been observed throughout the simulation, he decided to alleviate that problem by adding two extra nurses, and then to continue the simulation.

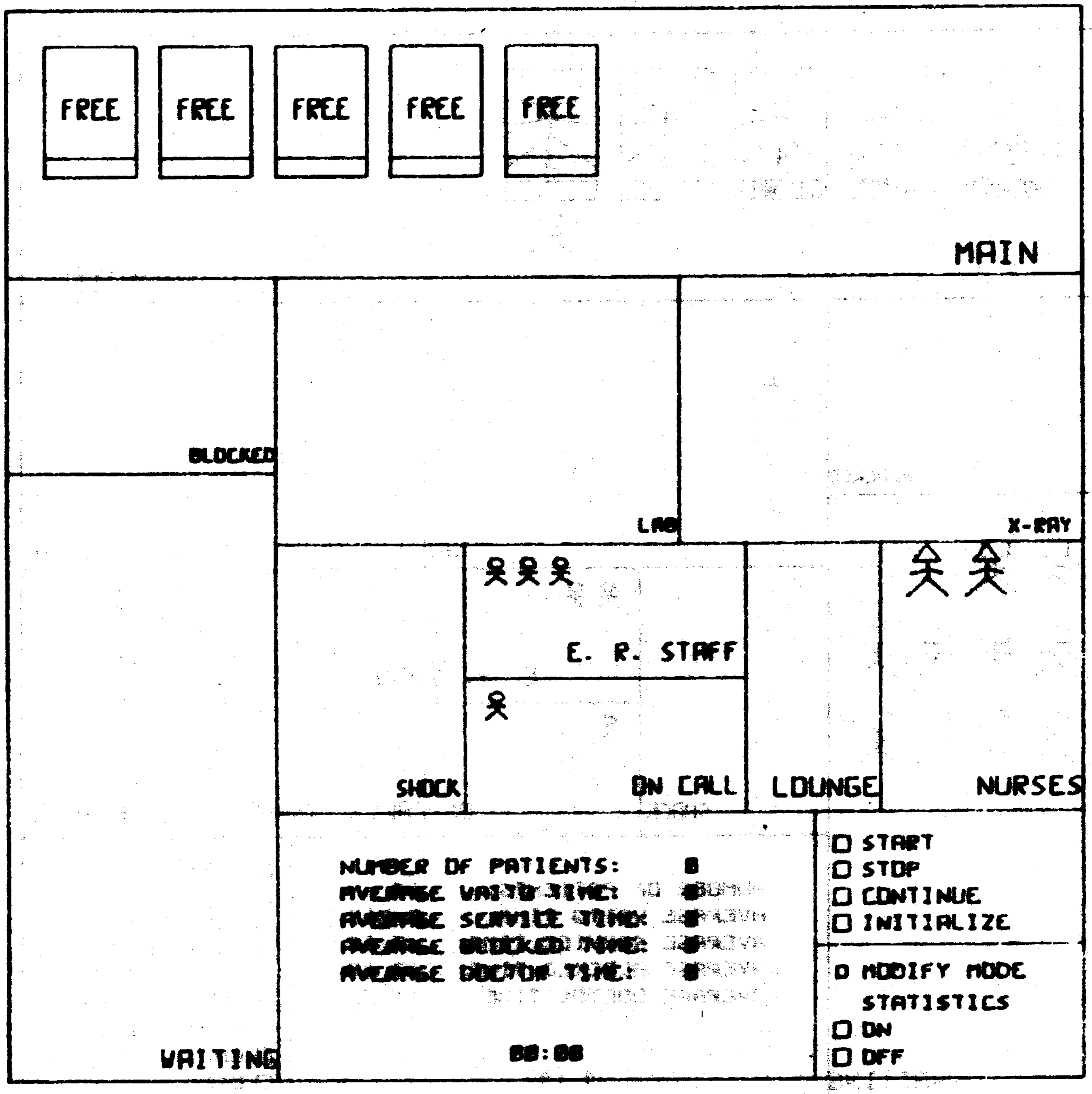
The two extra nurses clearly made a difference. Nurse-blocked states were no longer a problem. In fact the system seemed to be running smoothly again. By 11:34 the waiting room queue length was down to three (Figure 5-16), and by 14:03 (Figure 5-17) there was only one patient in the waiting room.

5.5 Statistics

Recall from Chapter IV that one of the display buttons controls the display of statistics in SIMULATE mode. For the current version, it has been decided that the following statistics be displayed:

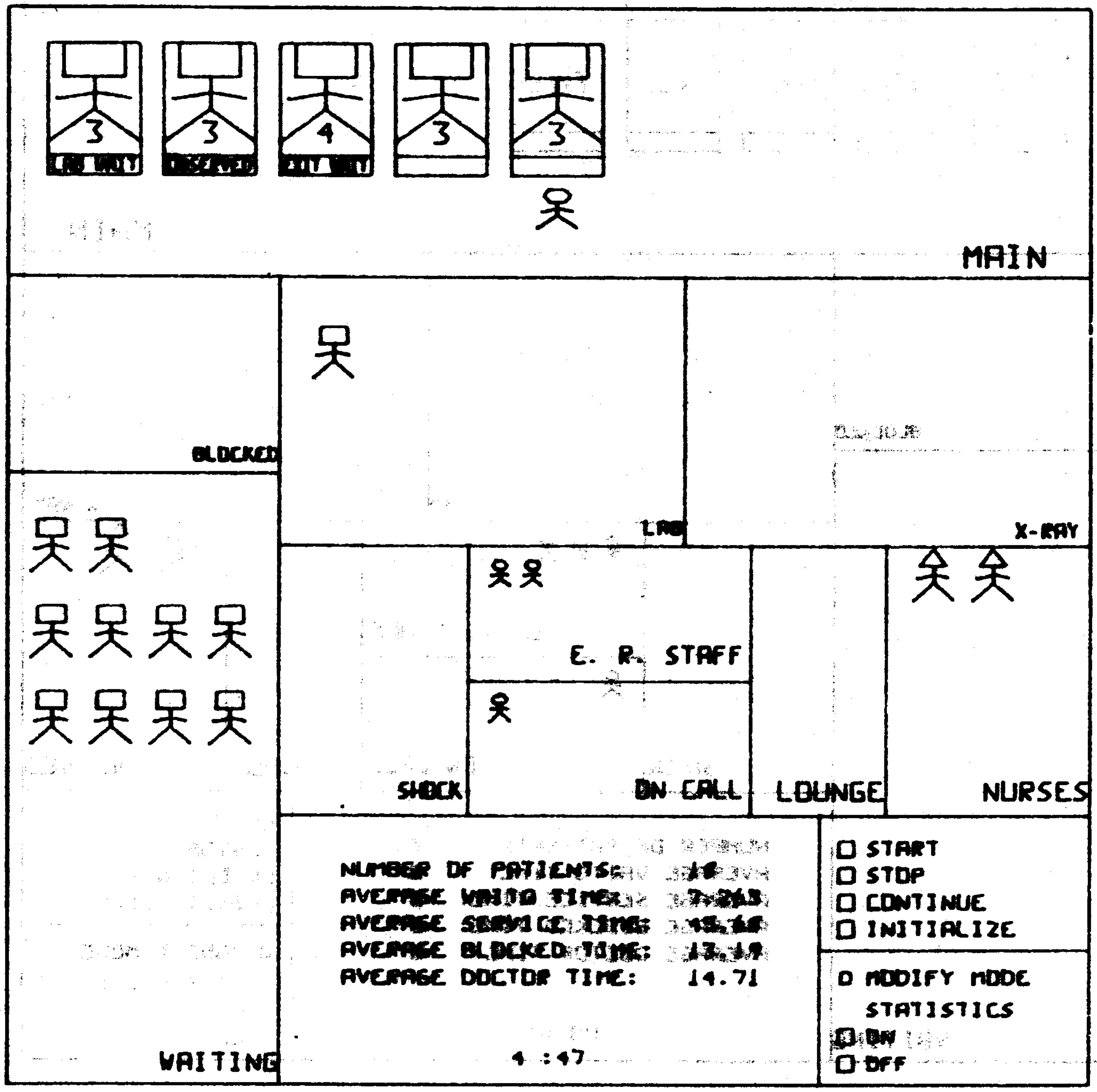
1. number of patients who have been treated
2. average time in waiting room queue
3. average service time per patient treated
4. average blocked time
5. average doctor time per patient treated.

As described in Section 5.1, items three and four posed some difficulty in their definition. The relevance of these five statistics is discussed in depth in Chapter III.



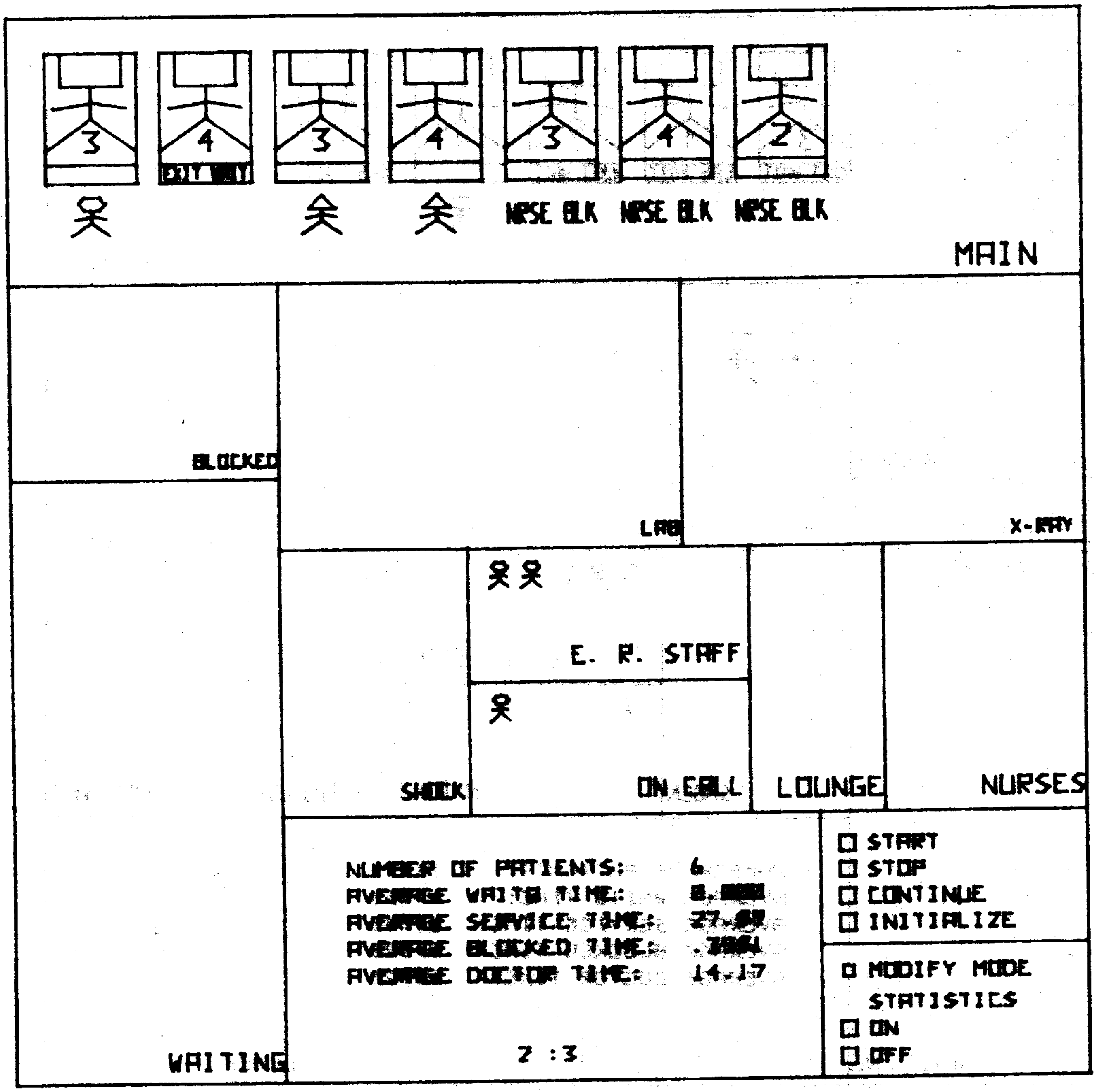
```
NUMBER OF PATIENTS:      0
AVERAGE WAIT TIME:       0
AVERAGE SERVICE TIME:    0
AVERAGE BLOCKED TIME:    0
AVERAGE DOCTOR TIME:     0

          00:00
```

Figure 5-4 Scene 1



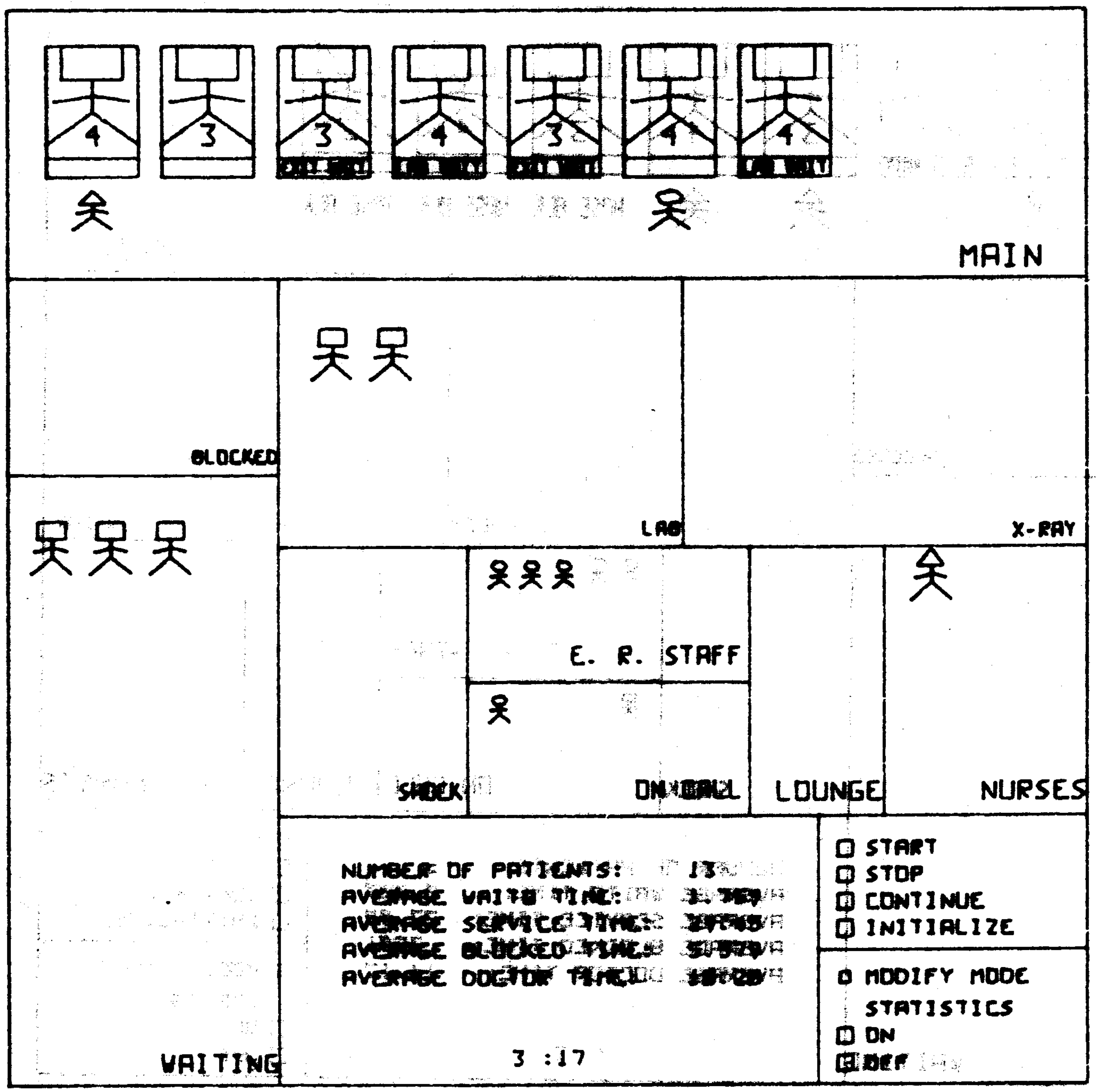
```
NUMBER OF PATIENTS:      18
AVERAGE WAIT TIME:       7.263
AVERAGE SERVICE TIME:    15.16
AVERAGE BLOCKED TIME:    13.19
AVERAGE DOCTOR TIME:     14.71

4 :47
```

Figure 5-5 Scene 2



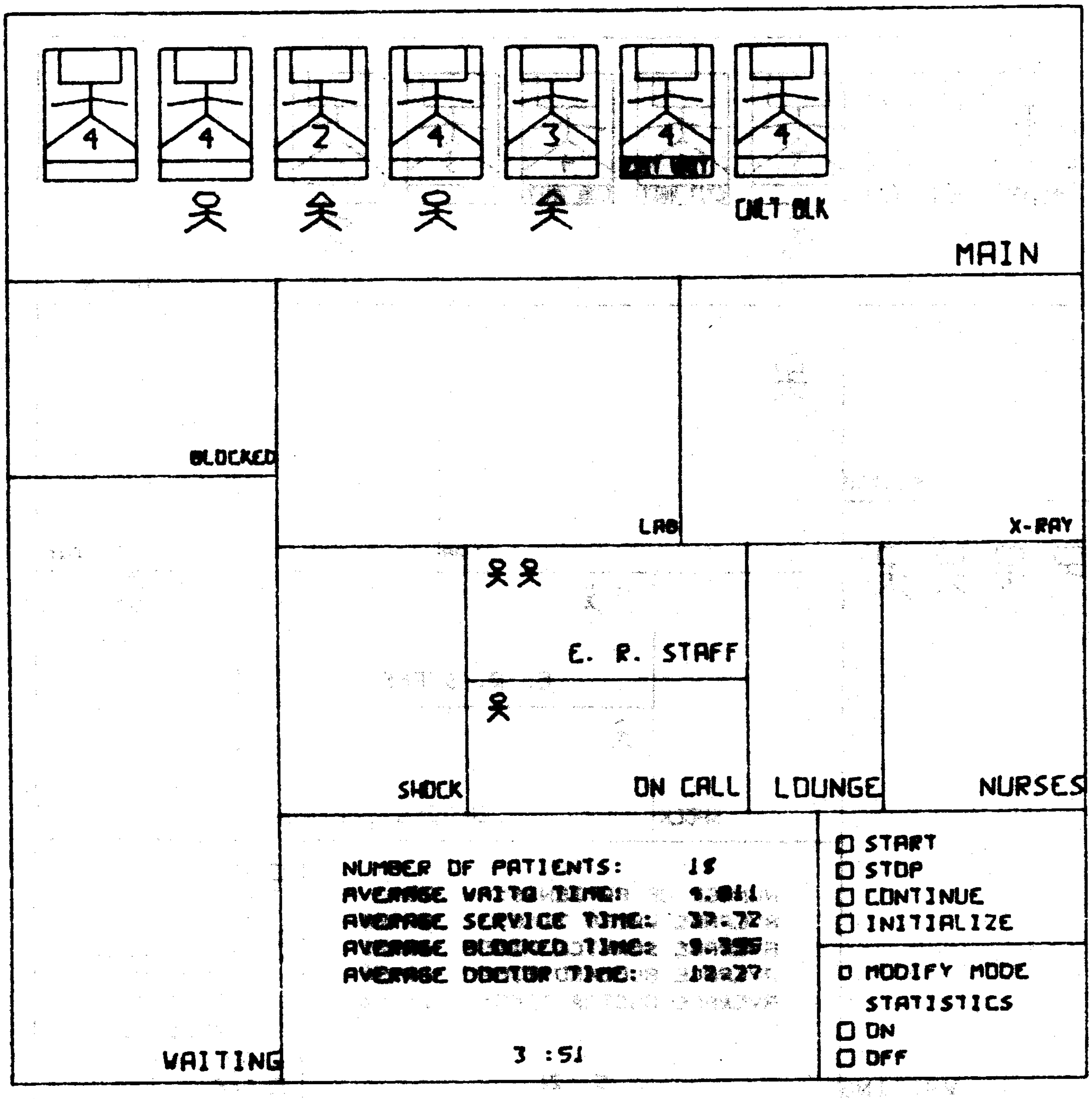
```
NUMBER OF PATIENTS:       6
AVERAGE WAIT  TIME:       0.[illegible]
AVERAGE SERVICE TIME:    27.[illegible]
AVERAGE BLOCKED TIME:     .[illegible]
AVERAGE DOCTOR TIME:     14.17

              2 :3
```

Figure 5-6 Scene 3



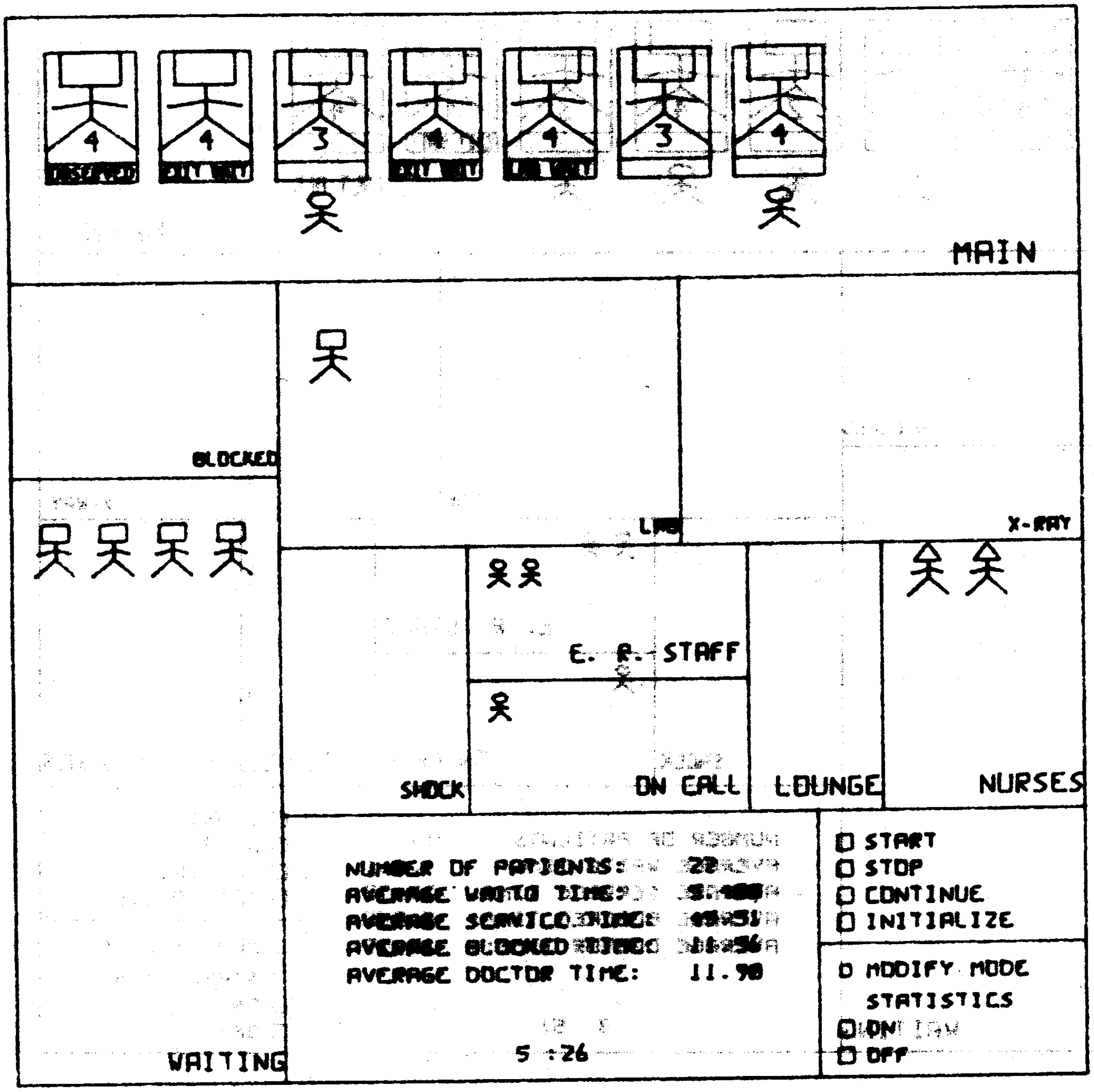
```
NUMBER OF PATIENTS:      13
AVERAGE WAIT TIME:       [illegible]
AVERAGE SERVICE TIME:    [illegible]
AVERAGE BLOCKED TIME:    [illegible]
AVERAGE DOCTOR TIME:     [illegible]

3 :17
```

Figure 5-7 Scene 4



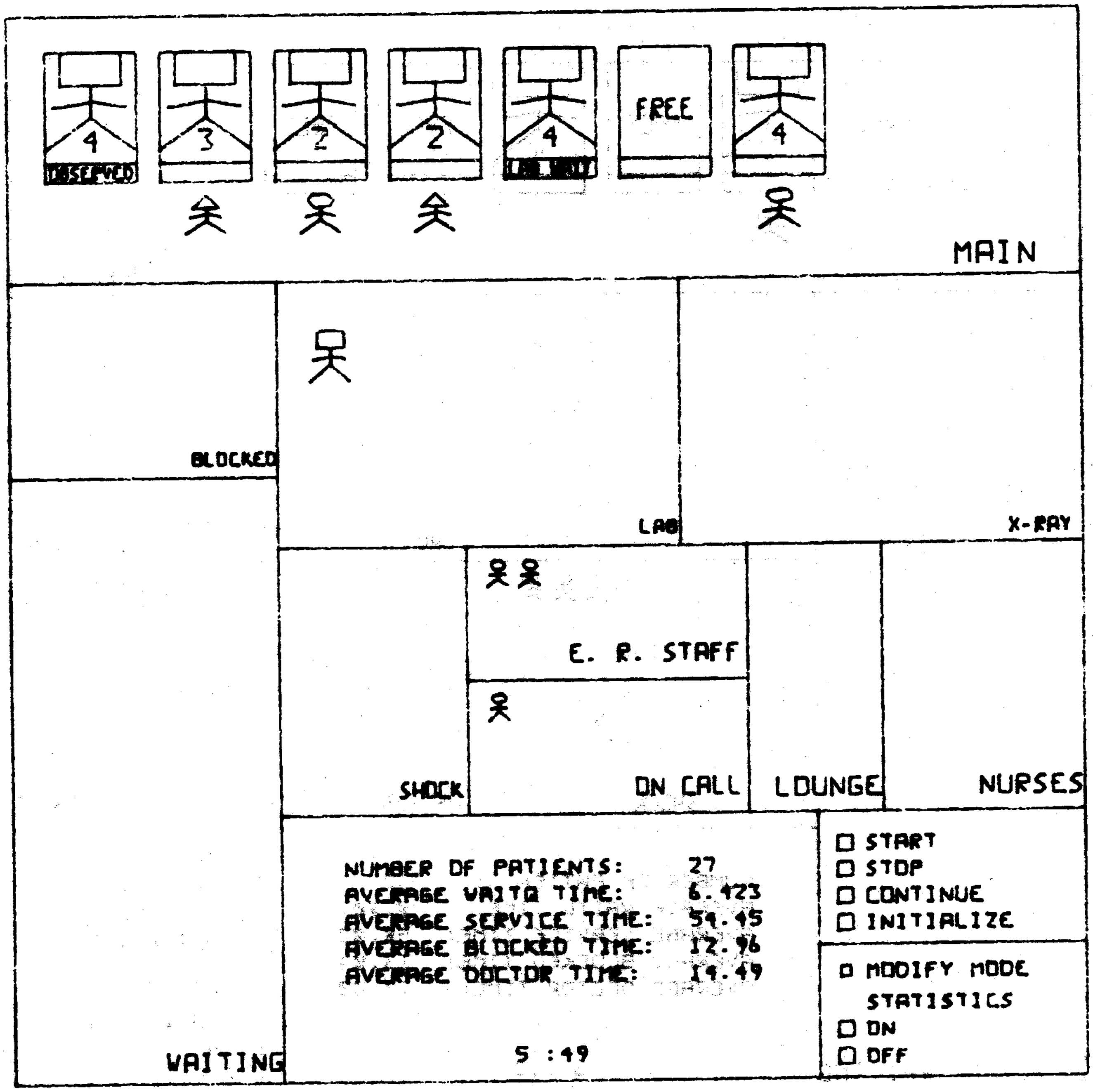
```
NUMBER OF PATIENTS:        18
AVERAGE WAITG TIME:        4.811
AVERAGE SERVICE TIME:      37.72
AVERAGE BLOCKED TIME:      9.355
AVERAGE DOCTOR TIME:       12.27

            3 :51
```

Figure 5-8 Scene 5



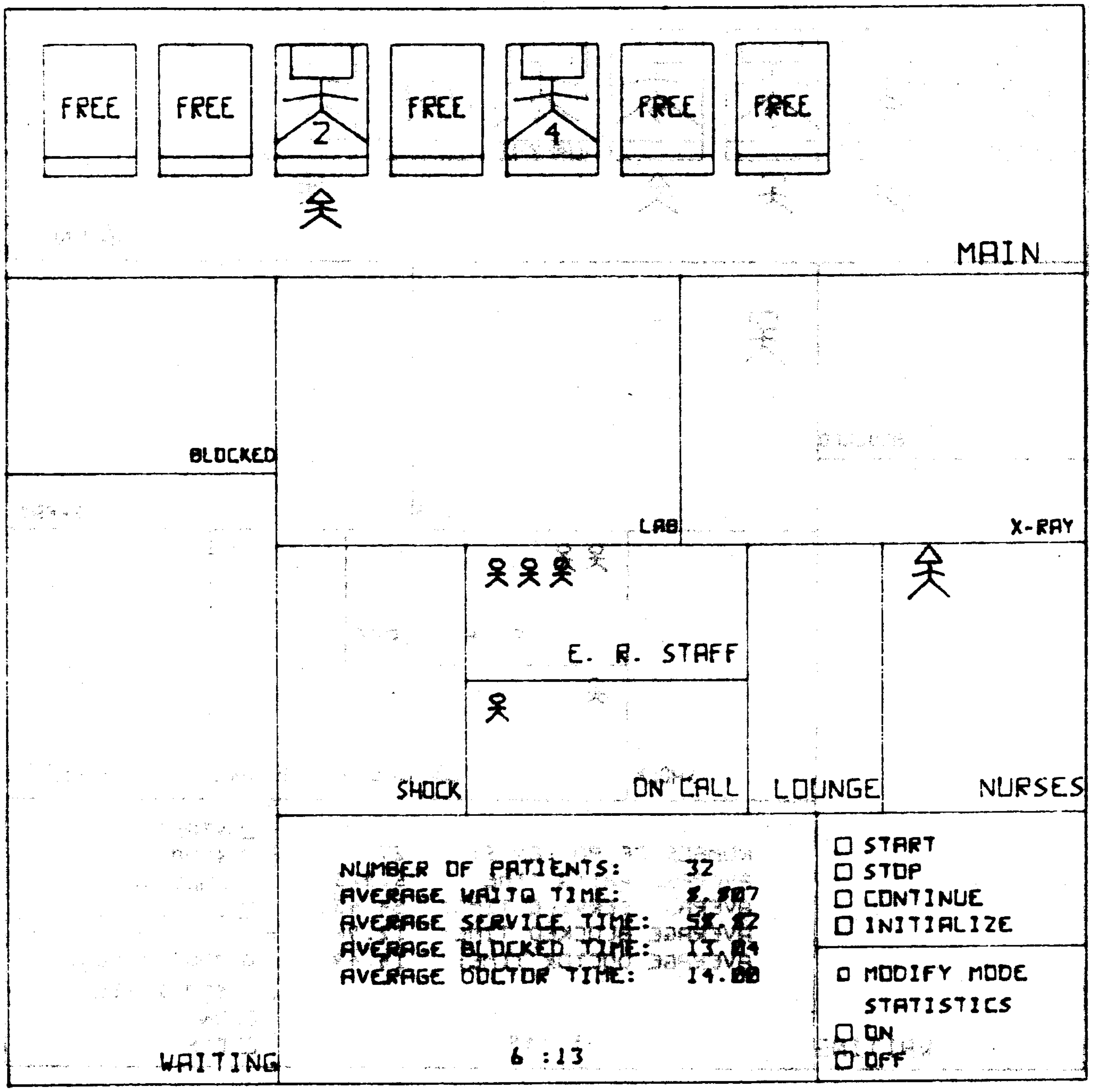
```
NUMBER OF PATIENTS:      22
AVERAGE WAITG TIME:      [illegible]
AVERAGE SERVICE TIME:    [illegible]
AVERAGE BLOCKED TIME:    [illegible]
AVERAGE DOCTOR TIME:     11.90

            5 :26
```

Figure 5-9 Scene 6



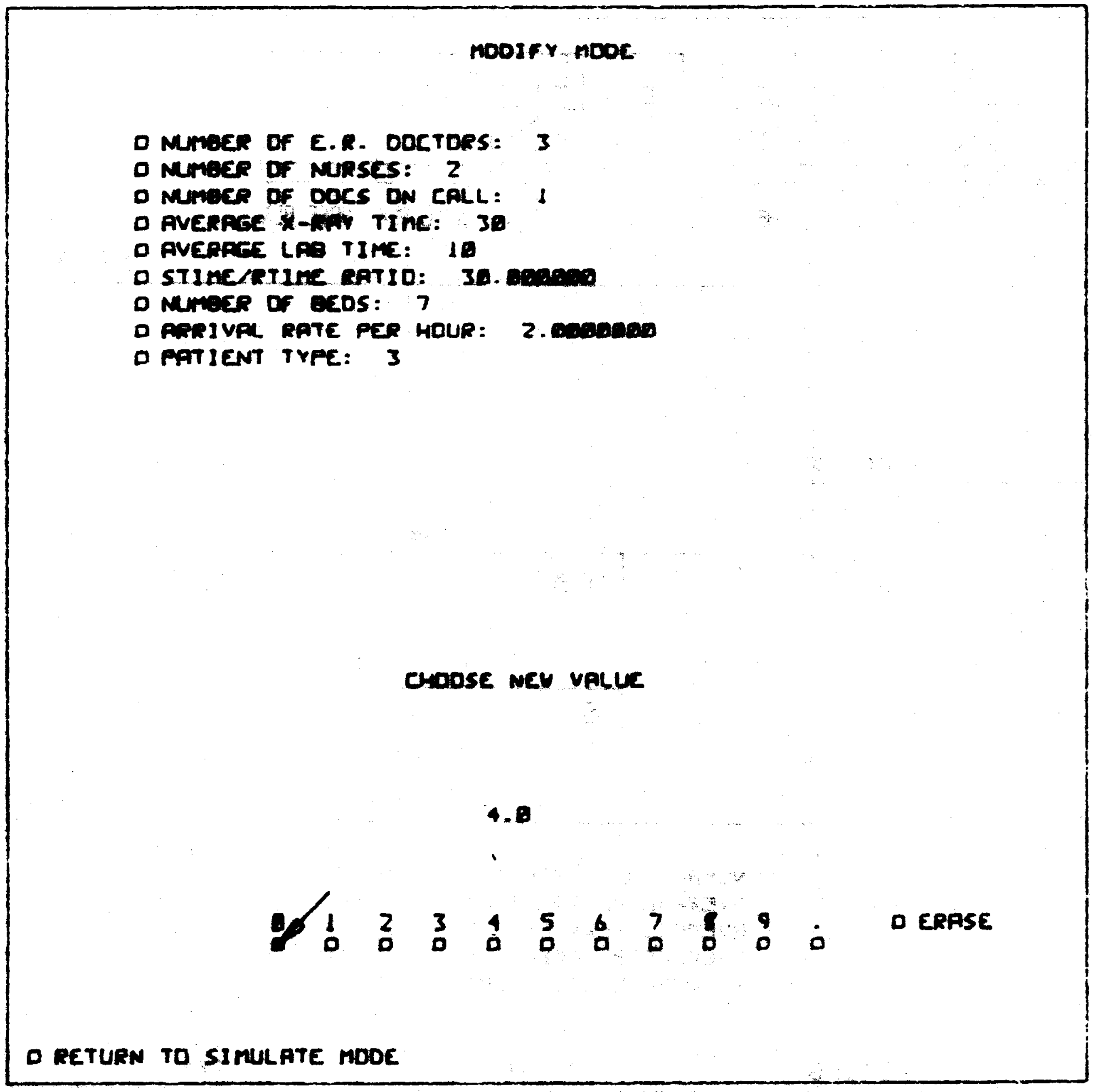
```
NUMBER OF PATIENTS:        27
AVERAGE WAITQ TIME:        6.423
AVERAGE SERVICE TIME:      54.45
AVERAGE BLOCKED TIME:      12.96
AVERAGE DOCTOR TIME:       14.49

            5 :49
```

Figure 5-10 Scene 7



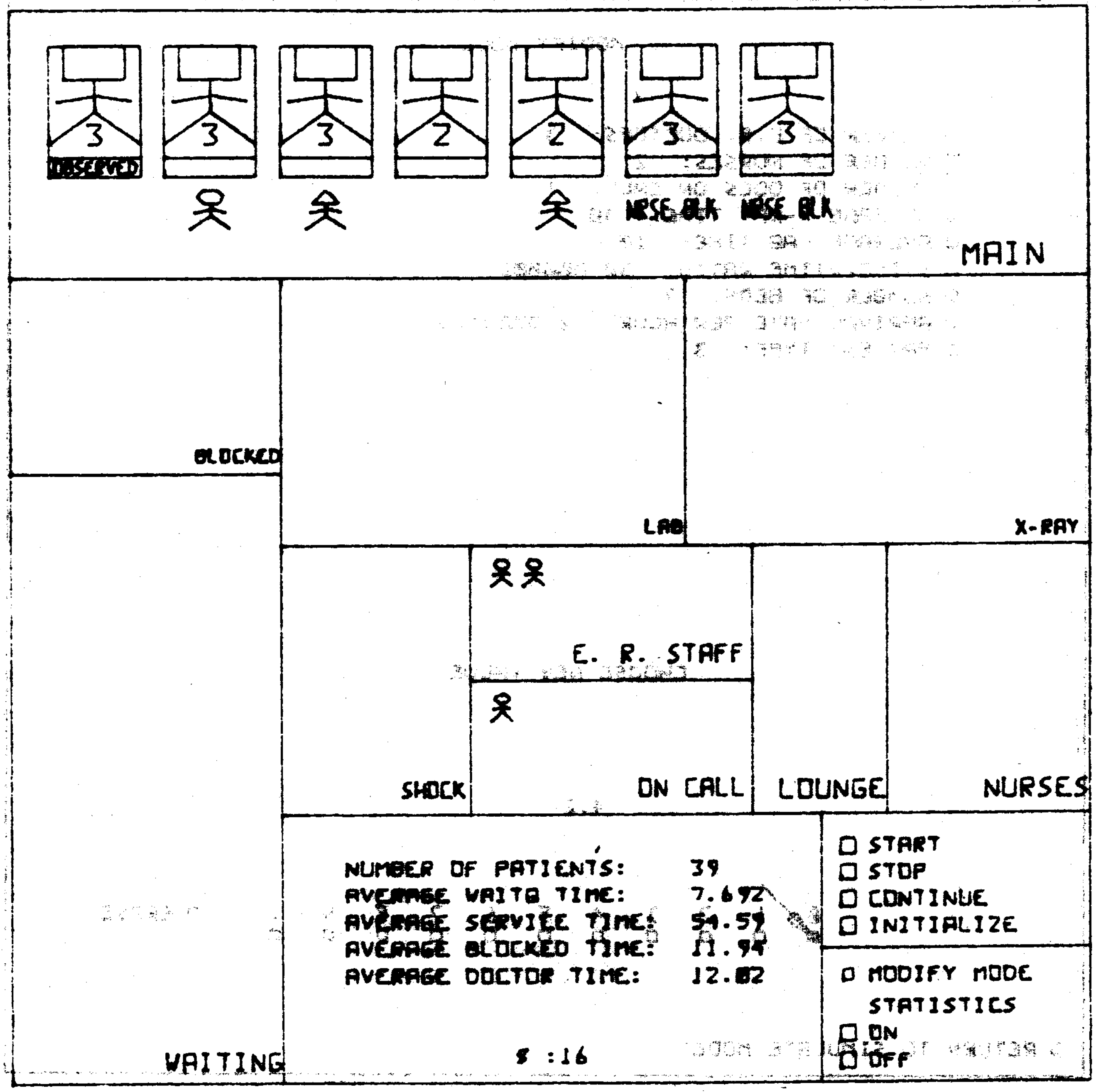
```
NUMBER OF PATIENTS:      32
AVERAGE WAITQ TIME:      8.007
AVERAGE SERVICE TIME:   58.82
AVERAGE BLOCKED TIME:   13.04
AVERAGE DOCTOR TIME:    14.00

            6 :13
```

Figure 5-11 Scene 8



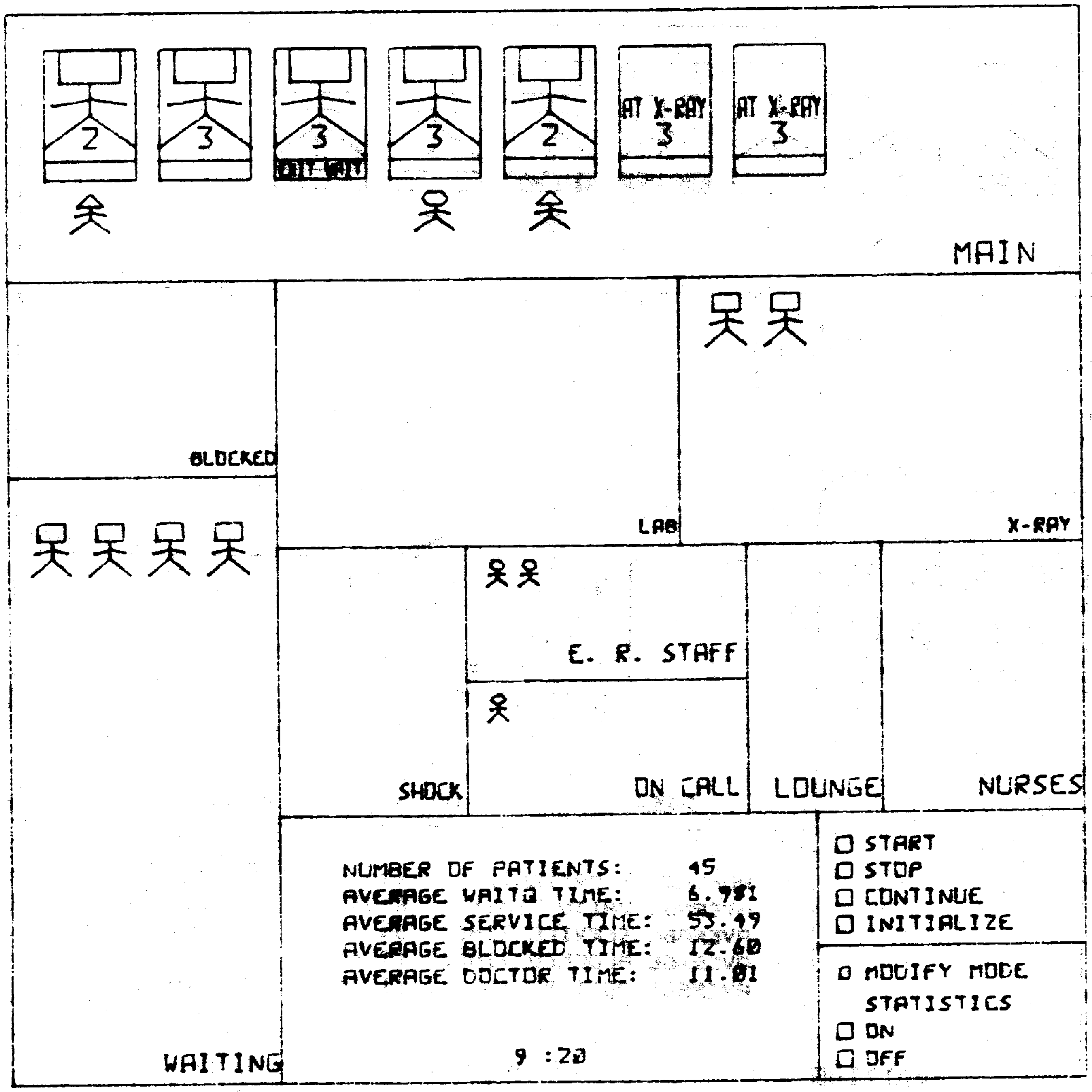
```
                         MODIFY MODE

     □ NUMBER OF E.R. DOCTORS:  3
     □ NUMBER OF NURSES:  2
     □ NUMBER OF DOCS ON CALL:  1
     □ AVERAGE X-RAY TIME:  30
     □ AVERAGE LAB TIME:  10
     □ STIME/RTIME RATIO:  30.000000
     □ NUMBER OF BEDS:  7
     □ ARRIVAL RATE PER HOUR:  2.0000000
     □ PATIENT TYPE:  3


                      CHOOSE NEW VALUE


                            4.0

          0  1  2  3  4  5  6  7  8  9  .       □ ERASE
          ■  □  □  □  □  □  □  □  □  □  □

□ RETURN TO SIMULATE MODE
```

Figure 5-12 Scene 9



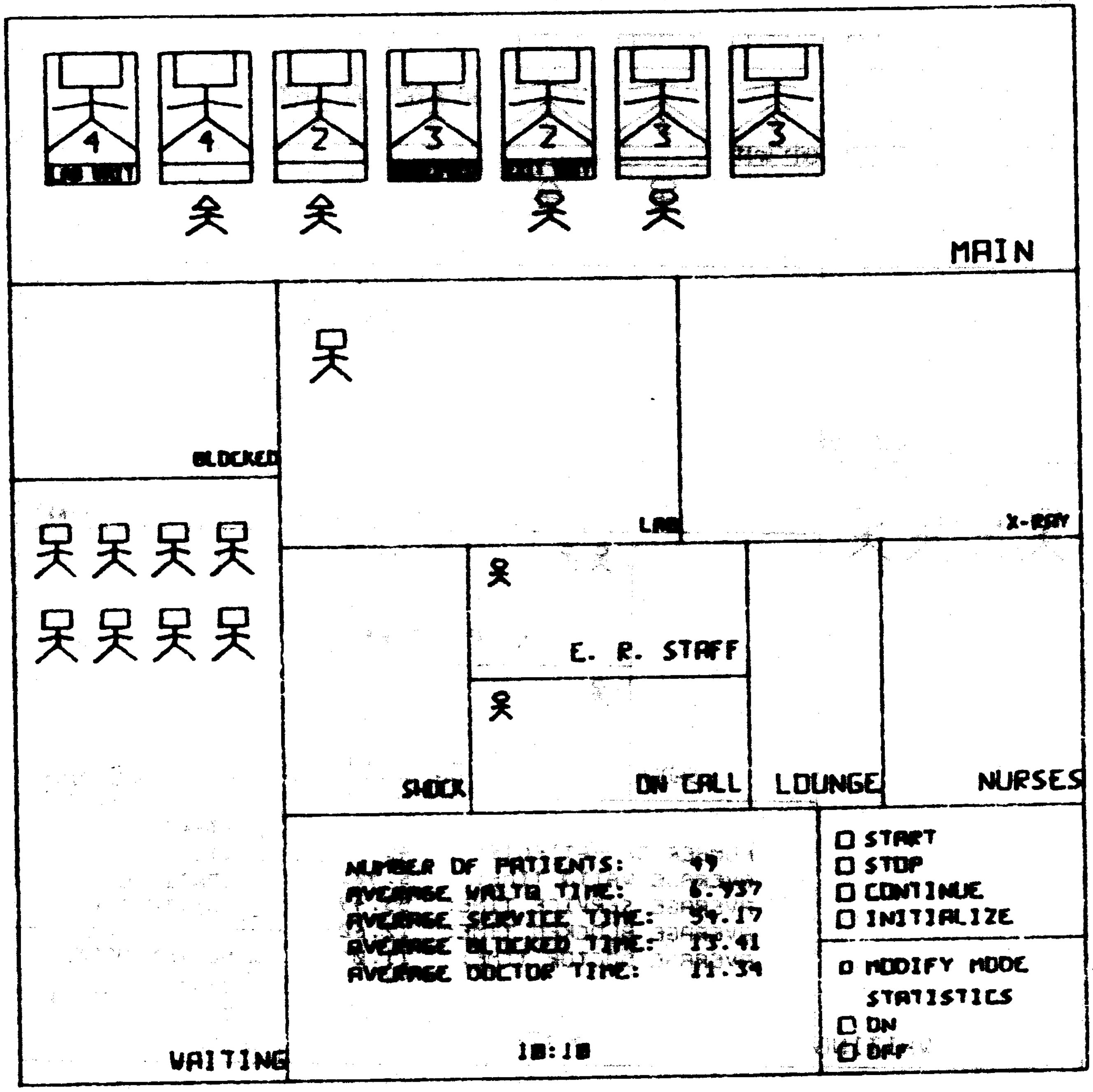
```
NUMBER OF PATIENTS:       39
AVERAGE WAITQ TIME:       7.692
AVERAGE SERVICE TIME:     54.59
AVERAGE BLOCKED TIME:     11.94
AVERAGE DOCTOR TIME:      12.02

            8 :16
```

Figure 5-13 Scene 10



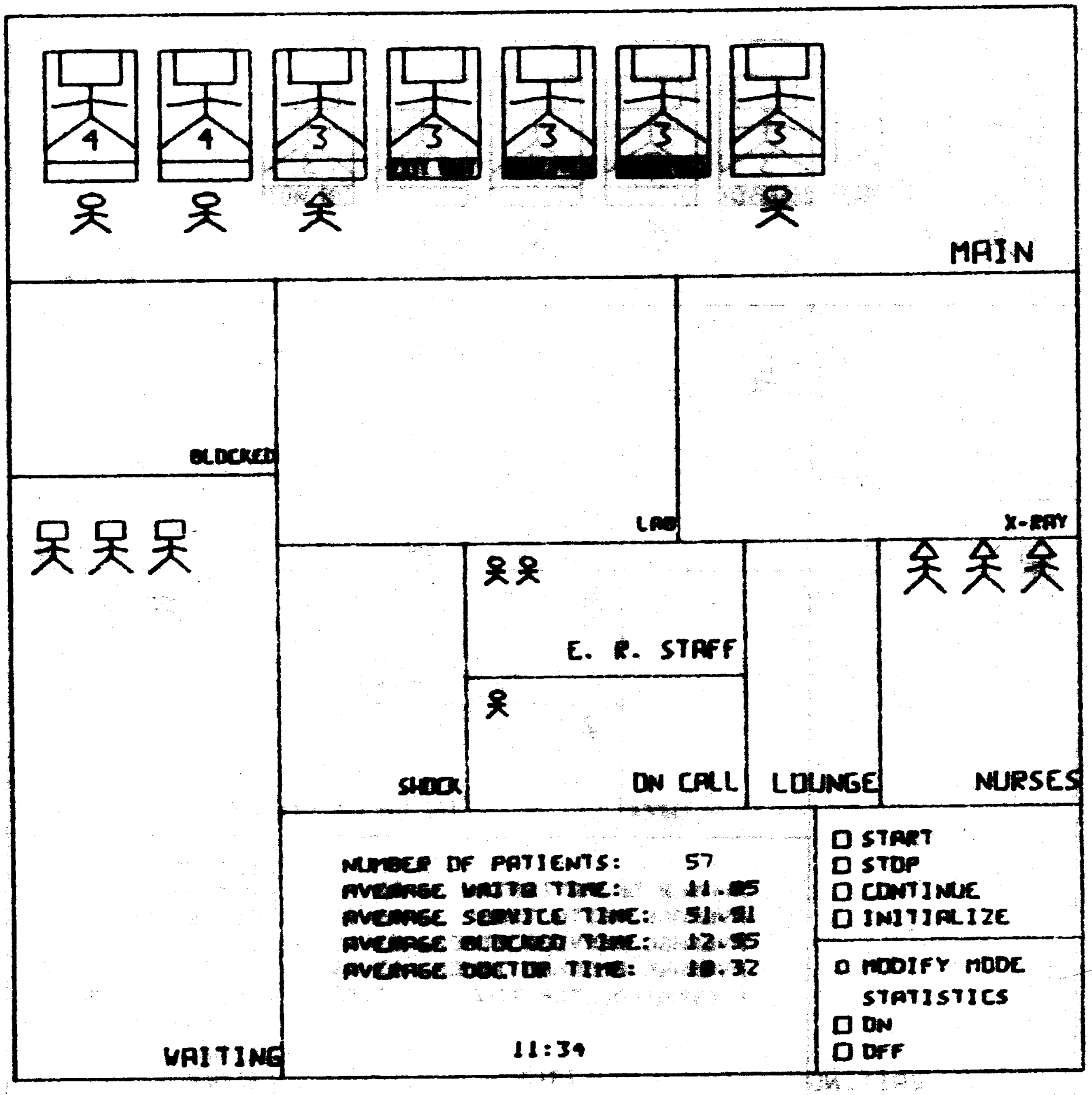
```
NUMBER OF PATIENTS:       45
AVERAGE WAITG TIME:       6.791
AVERAGE SERVICE TIME:     53.49
AVERAGE BLOCKED TIME:     12.60
AVERAGE DOCTOR TIME:      11.01

            9 :20
```

Figure 5-14 Scene 11



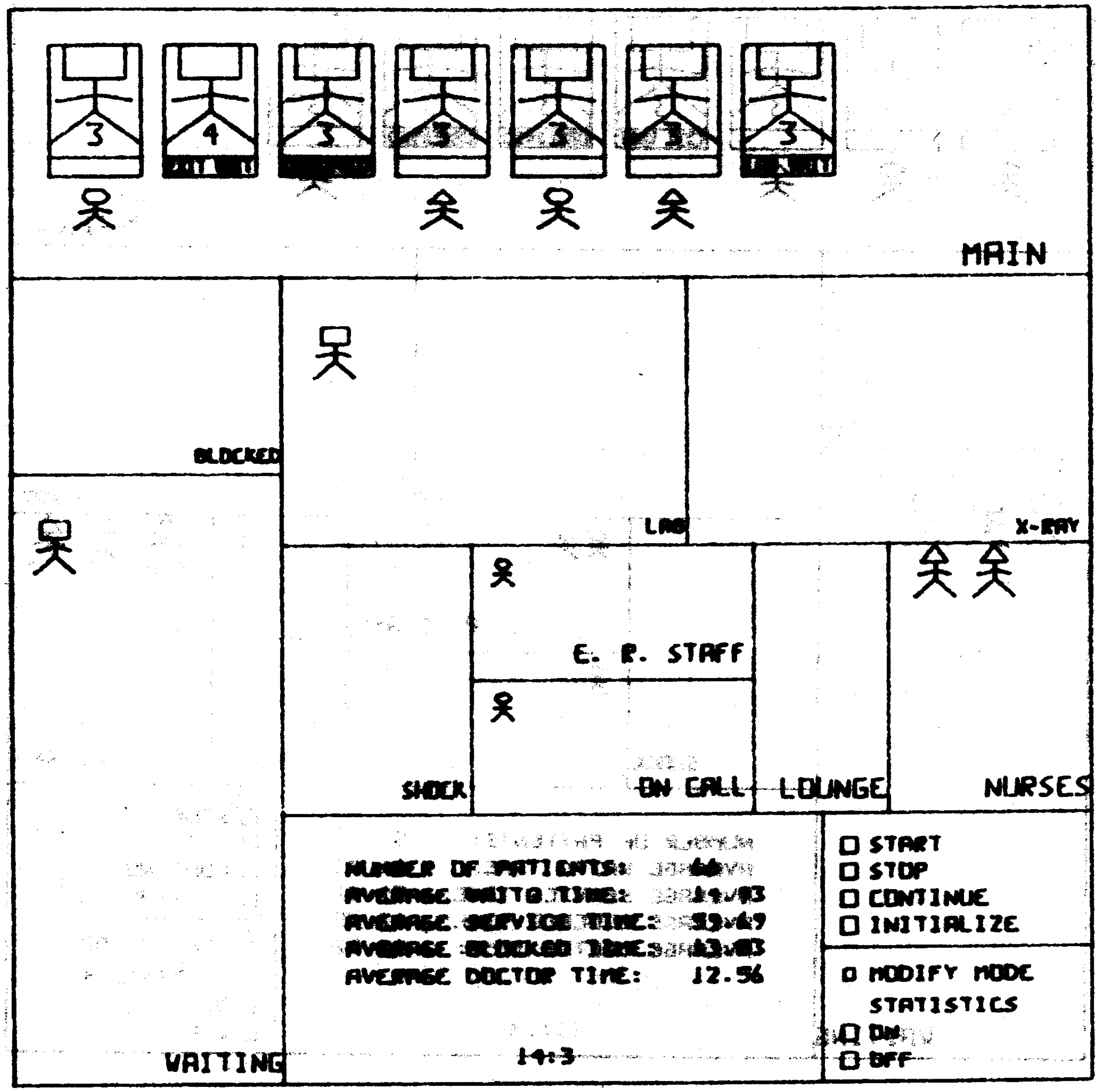
```
NUMBER OF PATIENTS:        49
AVERAGE WAITG TIME:        6.937
AVERAGE SERVICE TIME:      59.17
AVERAGE BLOCKED TIME:      13.41
AVERAGE DOCTOR TIME:       11.39

             10:10
```

Figure 5-15 Scene 12



```
NUMBER OF PATIENTS:        57
AVERAGE WAITG TIME:     11.05
AVERAGE SERVICE TIME:   51.91
AVERAGE BLOCKED TIME:   12.95
AVERAGE DOCTOR TIME:    10.32

              11:34
```

Figure 5-16 Scene 13



```
NUMBER OF PATIENTS:        66
AVERAGE WAIT TIME:      14.93
AVERAGE SERVICE TIME:   53.49
AVERAGE BLOCKED TIME:   13.03
AVERAGE DOCTOR TIME:    12.56

14:3
```

Figure 5-17 Scene 14

Dr. Mogielnicki has suggested that it would be constructive to add two additional sets of displayable statistics to the package (also adding two corresponding buttons to SIMULATE mode). The first would display the idle time (i.e. time not busy) of the various resources. This would be expressed as a percentage of total time. Included would be such information as doctor idle time and bed utilization. The second set of statistics would itemize the causes of blocked time. Thus instead of simply displaying average blocked time (as in the current implementation); average doctor blocked time, nurse blocked time, consultant blocked time and exit blocked time would be displayed. As we have stressed throughout our description of the TIGERS system, the program is intended to evolve with changing needs. This change suggested by Doctor Mogielnicki is one example.

CHAPTER VI: PUTTING THE RESEARCH IN ITS PROPER PERSPECTIVE

6.0 Introduction

This concluding chapter attempts to summarize what TIGERS is, what it is not, and what it might become. It begins with some warnings intended to preclude possible misinterpretation of the program and its displayed output. It then looks to the future, pointing out possible paths of future research suggested by the current work. Finally it looks at the present, and the potential offered by TIGERS-like programs.

6.1 Warnings

A program that communicates with the user through intuitive channels in addition to more rigorous ways can be valuable, but associated with such a program are inherent dangers of which the user should be wary. A TIGERS-like simulation carries pitfalls along with its blessings. The purpose of this section is to point out explicitly some of the more important for which one should be on guard.

The following point was made in Chapter IV, but is reemphasized here because instinct tends to make it easy to forget: as of this writing the TIGERS display in simulate mode has nothing to do with the actual layout of the emergency room's facilities. That is, the display does not necessarily bear any resemblance to the actual floor plan of the facility, nor do any of the routines which manipulate the various resources take the architecture directly into the simulation. The fact, for example, that the x-ray facility is located three minutes' walking distance from the main treatment room is reflected in the model only in that that time is a factor in calculating the time that a patient spends at x-ray. Such a model could be built -- a model which would allow the user to manipulate the possible arrangements of a facility. But the author contends that the effects of the architecture related parameters on the performance of the system are secondary when compared with the effects of parameters considered in the present research.

Another critical point which we have mentioned before but bears reemphasizing is the importance of basing decisions on a valid model. It is easy to convince oneself that the figures on the CRT automatically model the real world. We must continually remind ourselves that this is so only if we take the trouble to obtain valid data. This can be the most tedious task of the modeling process, but it must not be avoided if the model is to have value.

The TIGERS simulation lends intuition and therefore understanding -- information it communicates is not obscure. On the other hand it is easy to be misled by its inherent credibility: a situation might seem so obvious or real on the screen that it becomes too easy to avoid questioning the validity of the model. The simplicity and clarity of graphical communication is a mixed blessing. It becomes natural to simply assume that what is being observed represents a real life situation. The phenomenon is something of an extension to the well known "the-computer-said-it-therefore-it-is-true" phenomenon, in which we tend to accept facts as gospel simply because they are printed on computer print-out. Granted, this phenomenon is observed mostly in its numerous exploitations by the advertising industry; but it is real, and one should be wary of it.

6.2 Future Work

The current research touches several fields in which relatively little has been done. The idea of modeling systems similar to the Emergency Room characterized by small size (i.e. in one building) and great complexity has not been as popular as modeling more expansive systems. Also, the idea of interactive graphical simulation has been used hardly at all, especially in the world of public systems. In exposing the tip of several icebergs, TIGERS raises many questions about the parts still hidden.

6.2.1 A Valid Model

Before a tool like TIGERS can contribute towards making a hospital run more efficiently, it must be based upon a world line tree and parameter values which have been proven beyond reasonable doubt to constitute a valid model of the emergency room. Such research, although it would involve many hours of data gathering and analysis, would allow the program to become a practical tool instead of an academic demonstration.

A large class of unanswered questions has to do wth the nature of models of the E.R. Little is known, for example, about the marginal utility of detail. The whole issue of the (negative and/or positive) marginal utilities of aggregation with respect to model validity needs to be investigated. The example model implemented by the author was motivated by the hypothesis that an E.R. model can be significantly more accurate if it takes into account individually such services as x-ray, lab, and outside consultation, rather than aggregating them all together under the category "non-doctor service." This is a reasonable hypothesis, but it has not been proven. The more detailed model does not avoid the need for aggregation, even though fewer assumptions are needed for the more detailed than for the simpler one. It is conceivable that the emergency room is so complex that the additional detail introduced buys little accuracy over a model incorporating only two classes of service. Research into which factors increase the validity of models of the emergency rooms, especially research investigating the marginal utility of detail, would be worthwhile.

One aspect of the investigation mentioned above would be to construct the "ultimately detailed" model involving a world line tree with hundreds of nodes and branches. The development of such a detailed model would, of course, necessitate a large effort, but might offer the best solution to the problem of modeling the emergency room.

It seems reasonable to believe that the "optimum" model would fall somewhere between the two stage Markel et al. model and the ultimately detailed model described above. It would be interesting to construct several models at various intermediary levels of detail. Validity checks could be run and results could be compared. Conceivably one might notice a level of aggregation at which additional detail made no significant difference in results.

6.2.2 Analysis Routines

Another worthwhile addition to TIGERS would be a set of analysis and optimization routines. At present the user viewport serves to indicate imbalance in the system, and the program allows the user to vary his trial parameters to attempt to overcome the imbalance. But the program does not of itself make any analysis or suggestions. It would be useful if TIGERS did have analysis capability.

For example, it might prove useful for the program to be able to decide whether a user-implemented change has had a statistically significant effect on the system. Consider the following: Suppose that the waiting room queue is seen through the viewport as obviously too long, and the experimenter postulates that adding a nurse to the system will alleviate the problem. He notices a small change in the queue length, but does not know whether the change is large enough to be statistically significant. If an anlysis package existed as part of TIGERS, it might incorporate the necessary tools for determining the answer to this significance question.

A more sophisticated analysis package might even be able to make suggestions. Such a program would not only decide whether a change were significant, but it might actually suggest the change. Thus in the example above it might suggest that the experimenter try adding a nurse to the system. Implementation of such a package would involve design of a set of heuristics which make use of such statistics as accumulated patient nurse blocked time, accumulated doctor blocked time, etc.

6.2.3 The Analyst's Tool

The main research effort of the work reported here was directed towards development of TIGERS as an administrative tool and communications medium. One can communicate with a *particular model* on a highly interactive level. One logical next step is to create a tool which allows design of models at the same level of interaction. The present implementation of TIGERS is based upon a single world line tree which the author designed as a reasonable example. A number of other world line trees were possible, but since the present research was not primarily concerned with the design of a model, only one tree was actually implemented (i.e. used to generate a simulation). Although the analyst might wish to implement a number of trees in the TIGERS environment, such implementations are not necessarily easy. To some extent, the ability to set parameters to zero for various types of patient allows limited ability to experiment with different world line trees; but complete flexibility to change the tree does not exist. TIGERS provides a useful set of subroutines and a highly functional man-machine interface, but there is no escaping the fact that changing a world line tree into a working simulation can sometimes be a lot of work.

The proposed "analyst's tool" would do this work. Essentially, it would accept as input a formulated world line tree. It would then generate the appropriate subroutines and actually write the simulation program to bring to life the given tree. Thus the black box in TIGERS, which now must be filled by the modeler(s), would be filled automatically by this proposed program. There is good reason to believe that such a program is well within the realm of possibility, but we must realize that we are discussing a project which would probably represent at least as much effort as has already been expended thus far on the current research. If it were implemented, it would extend to the experimenter the same flexibility to experiment with various trees as TIGERS now allows one to experiment with a given model.

6.3 Why TIGERS?

In this report, we have introduced the idea of the graphical interactive simulation of a public system. Although the idea has as yet hardly been explored, it seems to show promise as a bridge over the gap between the administrator and the technical analyst. TIGERS is an analytical tool which can interact directly with the administrator as well as the analyst, and which hopefully can therefore serve as a focus in co-ordinating the insights that both have to offer.

The simplicity of graphical communication is especially important in the world of public systems. No special training is needed to understand the language of graphics, and thus it might serve to communicate where other media might prove less fruitful. A hospital administrator, for example, who must explain to officials in city government why he will need to expand his facilities to a certain size before the end of the next five years can lucidly make his point with a model that has graphical output. Furthermore -- and this point becomes very significant in the public sector -- graphics is dramatic. It can provide persuasive evidence for demonstrating a need. Weak points and bottlenecks in the system become obvious as crowds of figures start to overflow the screen.

The intuition added by graphics becomes especially significant in modeling a system such as the hospital emergency room in which balance is so critical. It was discussed in Chapter III that in the E.R., an optimum amount of any resource is optimum only when in proper balance with the other critical resources and with the demands made on the system. A thousand beds is probably no more useful than ten if there are only two doctors available. Graphics is especially useful in a system where a balance must be struck among many subsystems of a complex system; the user viewport allows a view of the entire system.

The human engineering aspects of a TIGERS-like program are important. The nature of the medium introduces psychological aspects to the analytical process which are usually insignificant or do not exist.

For example, interactive graphics can add an element of flexiblity which is perhaps unobtainable through other media, flexibility which is desirable for a number of reasons. First the user can do much more useful work per unit time. When he observes that the execution of the simulation is no longer interesting, he can "flush" it immediately, and waste no more time with it. Not surprisingly, flexibility also encourages experimentation that he might otherwise not consider worth the trouble. The user is encouraged to be creative because it becomes easy to try new ideas. Also significant is the immediacy of reinforcement, which also encourages creativity. A user often will not bother trying out more far-fetched ideas if he has to go to any trouble to implement them or wait for results, and yet it is well known that occasionally a far-fetched idea will turn out to be the beginning of an exciting new way of looking at something.

Another human engineering aspect of the TIGERS-like program is the very fact that such programs are more interesting to work with than the relatively dry non-graphical packages. Administrators are often loath to become deeply involved in rigorous analytical methods. The graphics medium makes the subject considerably more palatable.

Dr. Mogielnicki has pointed out that hospital administrators are often mistrustful and/or uninterested in the more technical methods of analysis, even though such methods might sometimes be applicable to problems facing them. He suggests that a TIGERS-like program might stimulate interest in such methods, because such a program cannot answer all the questions that it raises, and thus stimulates an interest in tools that can.

The hardware upon which the present system is implemented is currently too expensive for practical application in most situations, but graphics technology is developing rapidly and is fast entering the realm of practicability for smaller installations. Both from the points of view of the analyst and of the public agency decision maker, the medium represents a potentially constructive addition to the field of public systems analysis.

BIBLIOGRAPHY

Berki, S. E. and Heston, A. W., Introduction, *The Annals of the American Academy of Political and Social Science*, vol. 399, January, 1972, pp. iv-xiv.

Drake, Alvin W., *Fundamentals of Applied Probability Theory*. New York, McGraw-Hill, 1967.

Lee, R. I. and Jones, L. W., *The Fundamentals of Good Medical Care: An Outline of the Fundamentals of Good Medical Care and Estimate of the Service Required to Supply the Medical Needs of the United States*, Hamden Conn., Archon Books, 1962.

Markel, Purks, Shields, and Weissberg, *A Study of the Cambridge Hospital Emergency Room*, unpublished.

Mechanic, D., "Human Problems and the Organization of Health Care," *The Annals of the American Academy of Political and Social Science*, vol. 399, January, 1972, pp. 1-11.

Pfister, Gregory, *A Muddle Primer?*, Programming Technology Division Document SYS.11.01, M.I.T. Project MAC.

Report of the National Advisory Commision on Health Manpower, vol. 1, November, 1967 (Washington, U.S. Government Printing Office).

"Research and Statistics Note 18-1970," Office of Research and Statistics, Social Security Administration, U.S. Dept. of Health, Education, and Welfare. Washington, D.C., October 30, 1970.

Rutstein, D. D., *The Coming Revolution in Medicine*, Cam bridge, Massachusetts, M.I.T. Press, 1967.

U.S. Dept. of Health, Education, and Welfare, *A Study of National Health Insurance Proposals Introduced in the 91st Congress*, A Supplementary Report to the Congress, July, 1971.

U.S. News and World Report, vol. 72, June 26, 1972, pp. 78-81.

Time, vol. 97, June 7, 1971, pp. 86-88.

Weinerman, E. R. et al., "Yale Studies in Ambulatory Medical Care, V: Determinants of Use of Hospital Emergency Services," *American Journal of Public Health*, vol. 56, July, 1966, p. 1044.

GLOSSARY

1. *Poisson Distribution*
   A distribution for random events which assumes that the intervals between events are independent and exponentially distributed. Events occurring according to a Poisson distribution are completely random, unscheduled events. In models described in this report, patient arrivals are assumed to be Poisson.

2. *Primary Care*
   That part of the health care system which represents the patient's first level of contact with medicine.

3. *Queueing Theory*
   A branch of mathematics which deals with waiting line problems. In a typical queueing problem, a service facility provides service to customers who arrive in some random manner and require some variable amount of time to be served. Queueing theory describes such features of the service process as the queue sizes, queue delays and server idle time.

4. *Service Time*
   In queueing theory, the length of time required to serve a customer. In the emergency room, this corresponds to the time a patient spends in treatment after leaving the waiting room queue.

5. *Triage Nurse*
   A nurse stationed in the emergency room who directs incoming patients to sources of treatment appropriate to the urgency of their needs. A triage nurse might, for example, order that x-rays or a blood sample be taken before the patient enters the main treatment room.

APPENDIX A:
DESCRIPTION OF THE TIGERS PROGRAM

A.0 Introduction

In this appendix, the actual programming of TIGERS is discussed. The data bases are described, and the routines that manipulate them are explained in depth. Although the appendix is rather complete in its discussion of the various subprograms which comprise TIGERS, it does not, of course, replace the listing of the program itself. But except for the reader who intends to modify the program, a copy of the listing is not a necessity.

Describing a program is difficult in that it is almost impossible to avoid alluding to topics not yet covered, but forward referencing has been avoided to as large an extent as possible. The structure of this appendix is such that the broader, and generally more important, topics are discussed first. The reader who is interested only in getting a general feel for the program need read only the first part, although he may wish to skip afterwards to the description of the graphics handling routines. Section A.7.2 might also hold special interest in that it describes, among other things, the algorithm upon which the currently implemented model is based. The reader who is also interested in details of the program will want to read the whole appendix.

This appendix serves two purposes: 1) it allows the interested reader to examine to almost any depth he chooses how the TIGERS program works, and 2) it allows the analyst who is interested in working with

TIGERS to become familiar with the protocols of how data structures are set up and manipulated, and to discover resources that the program offers him.

A.1 MUDDLE

Every effort has been made to keep the discussion which follows independent of the MUDDLE language in which the program is written. But occasionally it becomes necessary to use a phrase or two of the language to clarify a point. We therefore outline here a few essentials which will clarify the references to MUDDLE which are made.

MUDDLE is a so-called *list processing language* based upon certain types of data structures. A group of objects enclosed in parentheses forms a *list*:

(<object> <object> . . . <object>)

A list can be comprised of no objects (the empty list) or many objects, and the objects can be of various types such as numbers, variables, vectors, or other lists.

A *vector* is similar to a list, but it is enclosed by brackets instead of parentheses:

[<object> <object> . . . <object>]

The differences between the two types of structures have to do with the manner in which they are stored in the computer. The objects in a vector can be changed easily, whereas the *number* of objects in a list can be changed easily.

The structure which is used to indicate function application is the *form*, which is delineated by angle brackets (<>). The first element of the form is taken to be the name of the function being applied, while the remaining elements of the form are taken as arguments. Thus the form

```
<FUNCTB 1 2 A>
```

would apply the function FUNCTB to the arguments 1, 2 and A. The form

```
<SIMULATE>
```

would execute the function SIMULATE which takes no arguments.

Variables in MUDDLE can take on two values, the local value (LVAL) and the global value (GVAL). The LVAL is assigned by the function SET, while the GVAL is assigned by the function SETG. Thus

```
<SET V1 4>
```

would assign the local value 4 to the variable V1, and

```
<SETG V1 DOG>
```

would assign V1 the global value DOG. Note the values of a variable might be anything -- other variables, numbers, vectors, lists, etc. -- they are not restricted to being numbers. The local value of V1 can be referred to as ".V1"; and the global value, as ",V1". Thus after the call to SET above, "V1" evaluates to V1, but ".V1" evaluates to 4.

Certain functions in MUDDLE exist to manipulate lists. This set of functions lends great power to the language, but we discuss here only a very small subset. The function NTH applied to a list returns the n'th element of the list, where the list is the first argument to NTH, and n is the second argument. NTH is generally called using a shorthand method: by "applying" a number to a list, the n'th element of the list is obtained. Thus if we said to MUDDLE

```
<SET LISTA (79 CAT 2.5)>.
```

Then

<1 .LISTA>

returns 79, and

<2 .LISTA>

returns CAT.

The reader should bear in mind that this discussion of MUDDLE is non-rigorous and serves only to clarify certain phrases used in Appendix A. The reader interested in MUDDLE should consult Gregory Pfister's *A MUDDLE Primer?*, in which the language is explained in depth.

A.2 The Events Scheduler

The heart of TIGERS is the events scheduler, which is named, surprisingly enough, SCHDLR. It is discussed in greater detail in Section A.6.3, but we introduce it here. In order to understand the scheduler, one must first examine its associated data bases. Its three key structures are the patient, the event, and the schedule.

The *patient* in the TIGERS environment is represented by a vector of the following eight components:

1. Number of doctor visits
2. Pointer to bed location
3. Conflict flag (Section A.7.2)
4. Time of arrival
5. Time of entrance to main treatment room
6. Minutes of doctor time
7. Minutes of blocked time
8. Patient type

The *event* is a vector of length three with the following format:

[<time> <routine> <patient>].

The <time> is the time at which the scheduler is to execute the routine <routine>. <patient> is the patient that is of primary concern to <routine> as it executes. This patient is generally referred to as the *current patient.*

The *schedule* is the list of events to be executed; there is only one schedule, so it has been given a name, SCED. SCED is an ordered list; the elements (i.e. events) are arranged such that the events whose associated routines are to be executed first are first on the list.

It was stated in Section 4.1 that TIGERS is an *event paced simulation* in which the scheduler executes routines associated with *events* which manipulate the data bases and generate more events. A simulation run under TIGERS is basically a loop which repeatedly calls SCHDLR. (This loop is desribed in Section A.6.1.) SCHDLR does the following:

1) Delete the next event from SCED.

2) Wait (if necessary) until the simulated time is equal to the time of the event.

3) Execute the associated routine.

4) Return to the routine (not executed by SCHDLR) which called SCHDLR.

The "wait" in step 2 is not characteristic of the event paced simulation. But because one of the *raisons d'etre* of a TIGERS-like simulation is to lend intuition, it is desirable not only to maintain the proper order of execution of events, but also to maintain a simulated temporal flow. The program therefore maintains its own simulated time stream. The ratio of the speeds of simulated time to real time is one of the parameters whose value is set by the user. (The speed of the computer limits this ratio to a maximum of approximately one hundred.)

To say that a routine *generates another event*, is to say that at some point in some routine it is decided that some other event is going to happen at some future time. In order to realize execution of this future event, the routine must inform the scheduler. Two steps are necessary: 1) Create the event; i.e. create the vector containing the time, routine, and patient. 2) Place this event in the list SCED *in its proper place in order*. The utility function which any routine calls to send an event to the scheduler is called ADSCD (ADd to SCheDuler). ADSCD accepts the three arguments of time, routine, and patient; generates the vector, and places the event in its proper place in the list. ADSCD and SCHDLR are discussed in detail in Section A.6.3.

A.3 Other Key Data Bases

In addition to SCED, there are eight other important data bases:

1) queue of patients in waiting room (WAITQ)
2) queue of patients in beds waiting for a doctor (DRQ)
3) queue of patients in beds waiting for a nurse (NURSQ)
4) queue of patients waiting for x-ray facility (XRQ)
5) BEDSTR
6) CNTSTR
7) PDFSCH
8) PROBSCH

The four queues all operate in a first-come-first- served manner. In the TIGERS program they are structured as lists of patients, and they are manipulated by two utility functons: The routine ADTOQ accepts two arguments, a patient and a queue, and appends the patient to the end of the given queue. The routine LEAVEQ accepts one argument, a queue, and returns the next patient in line. These routines are discussed in greater detail in Section A.6.4.

Note that the queue of patients waiting for lab test results is not on the above list of data bases. This is because in the current implementation of TIGERS, the lab queue is not first-come-first-served. Rather, the lab is viewed simply as a black box to which requests are made for service. (The time that it takes to honor such a request, however, is a function of the number of requests outstanding.)

BEDSTR and CNTSTR are vectors in which are stored several key parameters of the program. They contain all of the information which is dynamically displayed on the TIGERS display during a simulation. These two structures are of such a form that the information that they contain is accessible by both the simulation routines and the graphics routines. The form of CNTSTR is:

$$[\underline{a}\ \underline{b}\ \underline{a}\ \underline{b}\ \underline{a}\ \underline{b}\ \ldots],$$

where $\underline{a}$ is always 0 or 1, and $\underline{b}$ contains a parameter value. $\underline{a}$ is strictly for use of the graphics routines: when $\underline{a}$ is 1 it indicates that the associated $\underline{b}$ has been changed since the screen was last updated, and that the value should be updated on the display. CNTSTR in the present implementation of TIGERS, contains six parameter value pairs ($\underline{a}$ $\underline{b}$):

1) length of x-ray queue
2) length of waiting room queue
3) number of nurses available
4) number of E.R. doctors available
5) number of doctors on call
6) number of patients waiting for lab reports

BEDSTR, which contains information about the main treatment room, is a vector of the form

$$[\underline{a}\ \underline{b}\ \underline{c}\ \underline{d}\ \underline{e}\ \underline{a}\ \underline{b}\ \underline{c}\ \underline{d}\ \underline{e}\ \ldots]$$

where $\underline{a}$ is as in CNTSTR, and $\underline{b}$, $\underline{c}$, $\underline{d}$, and $\underline{e}$ define the state of one bed.

Recall from Section 4.2.2 that each bed has associated with it four fields. b is associated with field four. In the present implementation of TIGERS, field 1 can be in one of three states; field 2, in one of four; field 3, in one of seven; and field 4, in one of four. Thus b can be either 1, 2, or 3; c and e, an integer between 1 and 4, inclusive; and d, an integer between 1 and 7, inclusive (see Figures 4-4, 4-5, 4-6). Note that TIGERS knows the state of a patient by examining the states of the fields of his associated bed.

The structures PDFSCH and PROBSCH are associated with the world line trees of the various types of patients in the TIGERS system. Section 4.3.1 describes world line trees and explains how each node is associated with a probability distribution function and each branch with a probability. The program manages this information by associating with each world line tree two vectors, one containing the probabilities associated with all the branches, the other containing the *means* of the probability distribution functions. Thus each node and each branch of each world line tree is associated with a position in a vector.

Recall that one world line tree is associated with each of the n types of patient in the TIGERS model. PDFSCH is a vector of n vectors, the n vectors containing the branch probabilities of the n types of patient. PROBSCH is also a vector of n vectors, the elements of each of which are the means of the pdfs. Thus PROBSCH and PDFSCH store all the numerical information associated with any world line trees in the model.

A.4 Classes of Functions in TIGERS

The purpose of this section is to introduce the various types of subroutines that comprise the TIGERS program. The sections following will examine these subroutines one by one.

Before discussing the various functions and types of functions in the program, however, a word on notation is in order: for ease of discussion, we may speak of "executing an event," which is a simpler way of saying "executing the function associated with an event." Also the reader should regard the words "function," "routine," and "subroutine" as synonyms. Routines in TIGERS take various numbers of arguments, and some return values. But function characteristics will be made evident by explicit explanation or from context -- not by any naming convention which differentiates among the three terms.

Within the TIGERS program can be found the following nine classes of functions:

1. *Event functions* are routines which are a part of the simulation per se. These are the routines that manipulate the data bases which represent aspects of the emergency room. It is event routines which send events to the scheduler, and it is also event routines which are executed by the scheduler.

2. *Random variable functions* (r.v. functions) are functions which use random number generators to generate random values of random variables. They are used by event functions to decide which branch of a world line tree is to be followed by a given patient at a given time. It is r.v. functions which introduce randomness into the simulation.

3. *Simulation functions* are functions such as SCHDLR which, although not event funcions, are an integral part of the simulation.

4. *Utility functions* are useful aids to the programmer who is constructing event functions. They provide something of a meta-language within MUDDLE which facilitates the writing of event routines.

5. *Time functions* maintain the simulated time flow.

6. *Graphics functions* manage the pictures on the scope and maintain the dynamic display.

7. *Button handling functions* are routines which are called whenever their associated buttons are hit by the user.

8. *Interrupt handling functions* are routines which are called whenever a clock interrupt is generated by the computer's interrupt system. These functions 1) update the clock and 2) decide which button handling routines to call when buttons are hit.

9. *MUDDLE functions* are standard arithmetic and structure manipulating functions which are "built in" to MUDDLE. They are used to construct other functions of all types. Although they are absolutely essential to programming within TIGERS, they are not discussed further in this report, and are included in this list only for completeness. [1]

Not all TIGERS functions fall uniquely into one of these nine classes, but generally the classes are distinct. In the following sections we examine TIGERS subroutine by subroutine. All of the major functions are discussed, but for some of the most minor, the program listing should be consulted. In the sections below the functions will be organized by class and will be discussed in the order of the above list. Where useful, flowcharts are provided and/or an example of how each routine is invoked is given. Also, where appropriate, an example of the effects of calling the routine is also given. These sections are written as an aid to anyone planning to write programs in the TIGERS environment.

A.5 Manipulation of CNTSTR and BEDSTR -- Utility Functions UPD, VAL, ADD1, and SUB1

BEDSTR and CNTSTR are a very important part of TIGERS. Any simulation within TIGERS references them frequently; it is therefore highly desirable that manipulation of these critical data bases be as simple for the analyst as possible. The routines UPD, VAL, ADD1 and SUB1 exist for this purpose.

UPD is a function for manipulating BEDSTR. It takes three arguments, the third of which is optional. The first two are integers, while the optional third is a patient (vector of length eight). If the third argument is not supplied, the current patient is assumed. The effect of a call to UPD is to set the field indicated by arg1 in the bed associated with arg3 to the value of arg2. Also UPD sets the bit which informs the graphics routines that they should indicate the change on the display. Thus if BEDSTR currently looks like this

[0 2 1 5 3 0 2 1 1 3 0 2 1 5 2 0 2 2 6 3 0 2 1 5 4],

and UPD is called:

<UPD 3 4 .CPAT>

where .CPAT is the patient in bed 3, then BEDSTR is changed to:

[0 2 1 5 3 0 2 1 1 3 $\underline{1}$ 2 1 $\underline{4}$ 2 0 2 2 6 3 0 2 1 5 4].

Thus the status of the patient in bed three is changed from being treated by a nurse to waiting for a doctor. Furthermore, the next time the graphics routines look at BEDSTR, they will notice the 1 (underscored above) and then reset the 1 to 0 and make the correct change in the displayed status of bed 3.

VAL, ADD1, and SUB1 are functions for manipulating CNTSTR. Recall that CNTSTR contains the values of certain parameters, and that CNTSTR also contains information for the graphics updating routines. The routines VAL, ADD1, and SUB1 make it easy to refer to the elements of CNTSTR by separate names without being concerned about the graphics information. Instead of XQL referring to the actual value of the length of the x-ray queue, it actually points to the position in CNTSTR where this value resides. The call

<VAL .XQL>

returns the length of the x-ray queue. Similarly <VAL .WQL> returns the length of the waiting room queue, and so on for NNURS, NDR, NONCL, and LABQL. (See the description of CNTSTR in Section A.3.) ADD1 is used for incrementing any of these values, and SUB1, for decrementing. Thus if CNTSTR looks like this:

[0 0 0 -2 0 1 0 2 0 1 0 1],

and the analyst wishes to increment the number of nurses available, he uses ADD1:

<ADD1 .NNURS>.

CNTSTR then looks like this:

[0 0 0 -2 $\underline{1}$ $\underline{2}$ 0 2 0 1 0 1].

As with the change to BEDSTR, the graphics routines will make the appropriate change in the display, and reset the 1 (underscored above) to 0.

A.6 Simulation Routines

A.6.1 SIMULATE

SIMULATE is the main simulation loop mentioned in Section A.2. It is fully explained by the flowchart in Figure A.1. STOPBIT is a bit, which, if set, causes the program to exit from the loop. ECOUNT is a count of the number of events executed.

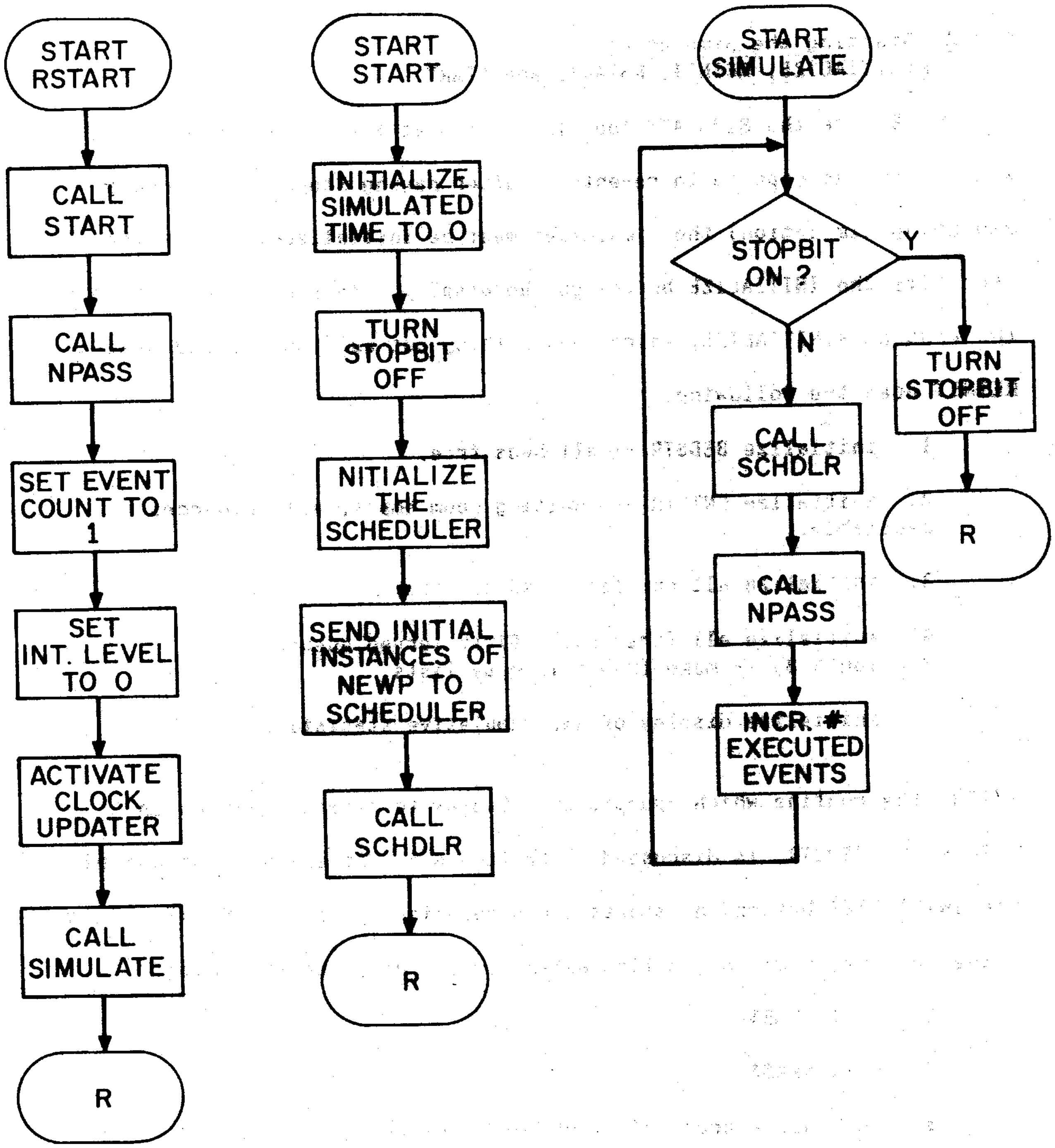


Figure A-1 Flowcharts of RSTART, START, and SIMULATE

A.6.2 Starting the Simulation -- RINITIALIZE, REINIT, RSTART, and START

Before the SIMULATE loop is entered at the beginning of a simulation (as opposed to re-entered after having stopped an already executing simulation) the simulation must be initialized. First the user hits the INITIALIZE button on the display. This cause execution of the routine RINITIALIZE, which does nothing but call REINIT and NPASS. REINIT does the following:

1. initialize BEDSTR -- all beds free,
2. initialize CNTSTR -- waiting room empty, all resources available,
3. initialize all cumulative statistics,
4. initialize all first come, first served queues (see Section A.3) -- make them all empty lists,
5. initialize display of all cumulative statistics.

NPASS, the routine which updates the display to reflect changes in CNTSTR and BEDSTR, is discussed in Section A.9. After the user has hit the INITIALIZE button, he starts the simulation by hitting START. This causes execution of the routine RSTART, which does the following:

1. Call START.
2. Call NPASS.
3. Initialize count of events executed (ECOUNT) to 1.
4. Initialize the interrupt handler which updates the clock in the display. This clock updating routine is called UPCLOCK, and is described in Section A.11.
5. Enter the simulate loop by calling SIMULATE .

The routine START, which is the first routine called by RSTART, does the following five steps:

1. Initialize the simulated time to 0.

2. Turn off STOPBIT (in case it has been left on).

3. Initialize the events scheduler by setting the initial value of the schedule. The statement

```
<SET SCED
  ([<+ .NOW
      <EXPDIS </ 60.0 <2 .LAMBDAS>>>>
    '<NEWP 2>
    [0 0 0 0 0 0 0 2]])>
```

   declares that SCED is a list of length one. Its one element is the event vector whose elements are the time that the event is to be executed (calculated by calling EXPDIS); the routine NEWP, which generates a patient arrival (Section A.7.1); and the dummy patient vector [0 0 0 0 0 0 0 2]. (NEWP is one of the few routines which has no "associated patient".)

4. Generate and send to the scheduler instances of the event NEWP, so that patients of all possible types will be generated.

5. Call the scheduler. This executes the event NEWP just sent to the scheduler. NEWP of course generates other events, and so the simulation is on its way.

Flowcharts summarizing the entire initialization procedure (RINITIALIZE, RSTART and routines that they call) are given in Figure A-2.

A.6.3 SCHDLR and ADSCD

The events scheduler was introduced in Section A.2. The purpose of this section is to describe the routines SCHDLR and ADSCD in greater detail.

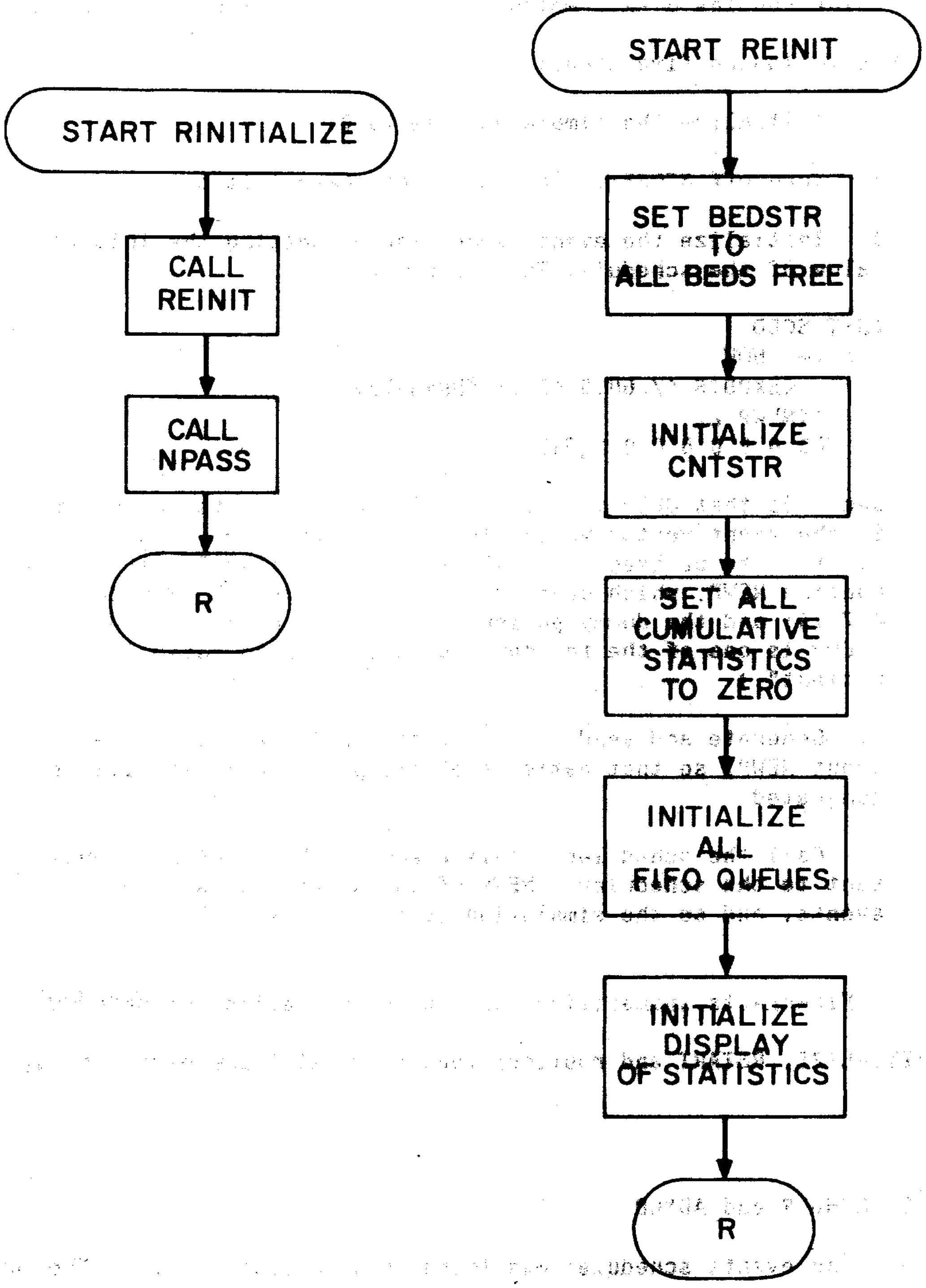

START RINITIALIZE
CALL REINIT
CALL NPASS
R
START REINIT
SET BEDSTR TO ALL BEDS FREE
INITIALIZE CNTSTR
SET ALL CUMULATIVE STATISTICS TO ZERO
INITIALIZE ALL FIFO QUEUES
INITIALIZE DISPLAY OF STATISTICS
R

Figure A-2 Flowchart of RINITIALIZE and REINIT

A flowchart for SCHDLR is given in Figure A-3. Recall that if .L is a list, then <1 .L> is the first element of the list, <2 .L> is the second element, etc.

SCHDLR takes no arguments. When it is called, it removes the next event from the list SCED; waits, if necessary; and executes the routine associated with the event. Consider for example, a state of a simulation in TIGERS where SCED is as follows:

```
([37.685173 <SNDNS PRFD> [0 6 0 37.685173 37.685173 0 0 3]]
 [92.209408 <NEWP 3> [0 6 0 37.685173 37.685173 0 0 3]]
 [96.171992
  <FRDR>
  [1 1 0 18.469403 18.469403 66.055544 0 3]]
 [97.171992
  <SNDXR>
  [1 1 0 18.469403 18.469403 66.055544 0 3]]
 [107.17436
  <NEWP 2>
  [1 6 0 24.505186 24.505186 .69831634 0 2]]
 [483.71264 <NEWP 4> [0 0 0 0 0 0 0 2]])
```

If at this point the program calls SCHDLR

```
<SCHDLR>
```

the program waits until the simulated time is greater than 37.7 minutes (if waiting is necessary), and then the event

```
[37.685173 <SNDNS PRFD> [0 6 0 37.685173 37.685173 0 0 3]]
```

is removed from SCED, and the routine NEWP is executed. When SCHDLR returns to the calling function, SCED is changed to something like this:

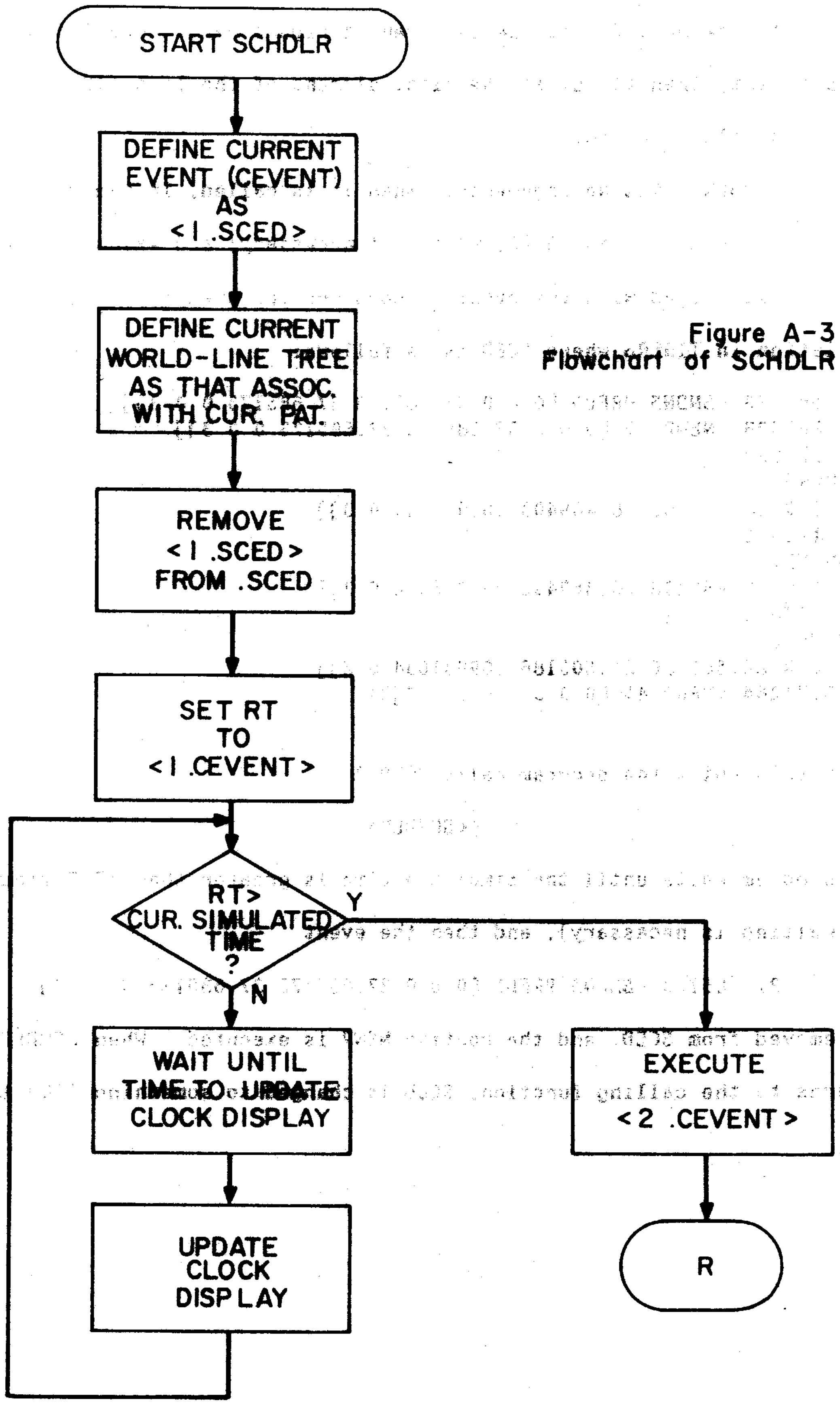

START SCHDLR
DEFINE CURRENT EVENT (CEVENT) AS <1 .SCED>
DEFINE CURRENT WORLD-LINE TREE AS THAT ASSOC. WITH CUR. PAT.
REMOVE <1 .SCED> FROM .SCED
SET RT TO <1 .CEVENT>
RT> CUR. SIMULATED TIME ?
Y
N
WAIT UNTIL TIME TO UPDATE CLOCK DISPLAY
UPDATE CLOCK DISPLAY
EXECUTE <2 .CEVENT>
R

Figure A-3
Flowchart of SCHDLR

```
([41.656325 <FRNRS> [0 6 0 37.685173 37.685173 0 0 3]] *
 [41.656325 <PRFD> [0 6 0 37.685173 37.685173 0 0 3]] *
 [92.209408 <NEWP 3> [0 6 0 37.685173 37.685173 0 0 3]]
 [96.171992
  <FRDR>
  [1 1 0 18.469403 18.469403 66.055544 0 3]]
 [97.171992
  <SNDXR>
  [1 1 0 18.469403 18.469403 66.055544 0 3]]
 [107.17436
  <NEWP 2>
  [1 6 0 24.505186 24.505186 .69831634 0 2]]
 [483.71264 <NEWP 4> [0 0 0 0 0 0 0 2]])
```

Notice that in addition to the fact that the first event from before the call to SCHDLR is no longer there, certain other events (indicated by *) have been added to the list. These new events were created during the execution of NEWP; ADSCD was called to create these new events and send them to the scheduler.

ADSCD does two things: create an event vector, and insert this vector in SCED. The function takes three arguments, a time, a routine, and a patient. The third argument is optional, and if it is not supplied, the current patient is assumed. The call

```
<ADSCD 41.656325 '<FRNRS> [0 6 0 37.685173 37.685173 0 0 3]>
```

would cause the the first event marked by * above to be generated and added to SCED as shown.

A.6.4 ADTOQ and LEAVEQ

Recall that first-come-first-served queues are simply ordered lists in TIGERS. ADTOQ and LEAVEQ, introduced in Section A.3, are the routines used for manipulating these queues. The routines are simple, but it will be constructive to give examples of their use and of effects of their use. Consider a point in a simulation in which all beds are occupied and there is a queue of length four in the waiting room. The TIGERS waiting room queue, WAITQ, might look like this:

```
([0 0 0 47.946094 0 0 0 2]
 [0 0 0 49.379954 0 0 0 3]
 [0 0 0 53.499452 0 0 0 4]
 [0 0 0 59.360199 0 0 0 3])
```

Say it is time 63.740902 and the event NEWP, which generates new patients, is executed. NEWP notices that there are no free beds, and adds the new patient to the queue by calling ADTOQ

```
<ADTOQ WAITQ [0 0 0 63.740902 0 0 0 3]>.
```

After the call, WAITQ is larger:

```
([0 0 0 47.946094 0 0 0 2]
 [0 0 0 49.379954 0 0 0 3]
 [0 0 0 53.499452 0 0 0 4]
 [0 0 0 59.360199 0 0 0 3]
 [0 0 0 63.740902 0 0 0 3])
```

Say a few minutes later a bed becomes free; then the routine CALLIN is called to "call in" a patient from the waiting room. CALLIN uses the function LEAVEQ to get the "next in line" from the queue:

```
<SET NP <LEAVEQ WAITQ>>
```

At this point, .NP is equal to [0 0 0 47.946094 0 0 0 2], and CALLIN has its new patient. Furthermore WAITQ has been updated to

```
([0 0 0 49.379954 0 0 0 3]
 [0 0 0 53.499452 0 0 0 4]
 [0 0 0 59.360199 0 0 0 3]
 [0 0 0 63.740902 0 0 0 3])
```

A.7 Event Functions

In this section we discuss the event functions currently available in the TIGERS environment. It is important to keep in mind while reading it that most of these functions are designed around the model described in Chapter V. That is, the event functions, at least the less general ones, are necessarily related to the world line tree being implemented. It is difficult to say which of the existing functions would be obsolete -- or which would need to be rewritten, or which new functions would need to be added -- if some other model were being implemented. It suffices to remember that the program is intended to evolve to meet changing needs.

There are some definitions and phrases that should be kept in mind in studying these events. First recall that all event functions are executed only by the scheduler, and that associated with each event function executed by the scheduler is a patient. The event functions are written assuming the existence of this patient; in the context of discussing an event we shall refer to this patient as the *current patient* and to the associated bed as the *current bed*.

Often we shall speak of "sending an event to the scheduler." Recall that this process involves two steps:

1. Decide when the event is to be executed. This decision is made using r.v. functions to generate a random value for the time increment involved.

2. Use ADSCD to create the event and place it in its correct position in the ordered list SCED.

Also we shall sometimes speak of sending an event to the scheduler "tr be executed immediately." This simply means that the scheduled time of execution of the event will be the current time; thus the event will be executed immediately after the current event. Sending an event to the scheduler to be executed immediately thus avoids having an event function called by other than the scheduler.

Finally recall that four fields are associated with each bed. The utility function UPD, which updates the values of these fields, will occasionally be used in this discussion, especially in the flowcharts.

A.7.1 NEWP

NEWP (NEW Patient) is the event function which generates arrivals to the system. Unlike most event functions it takes an argument, the patient type of the patient to be generated. Execution of NEWP effects the simulated world in two ways. First a new patient of the specified type appears in the simulated emergency room requesting treatment. The patient is either assigned to a bed and a nurse, or, if one or both of these are not available, the patient enters the waiting room queue. The second major job of NEWP is to perpetuate itself so that more patients will arrive in the future. This it accomplishes by generating a new NEWP and sending the event to the scheduler.

The algorithm of the routine can be described by the following steps:

1. Generate a random value of an exponential random variable. Use this value to determine the arrival time of the next patient of the current patient type.

2. Call ADSCD to create the next instance of NEWP for the current patient type.

3. Create a new patient (recall that a patient is described by a vector of length eight).

4. Increment the number of patients in the system.

5. If there is a waiting room queue, add the new patient to queue. Then return to the scheduler. If there is no queue, proceed to steps 6 and 7.

6. Find an empty bed, and assign the patient to that bed.

7. Generate (send to the scheduler) the event to assign a nurse to that patient.

Since each NEWP perpetuates the arrivals of only one type of patient, it is necessary for the initialization routines to initialize SCED such that one instance of the arrival of each type of patient is scheduled. Thus for a model with three patient types, the initial schedule might look like this:

```
([3.5183036 <NEWP 3> [0 0 0 0 0 0 0 2]]
 [53.986576 <NEWP 2> [0 0 0 0 0 0 0 2]]
 [347.16065 <NEWP 4> [0 0 0 0 0 0 0 2]])
```

After SCHDLR executes the first event on the schedule at time 3.5183036, the schedule might appear as follows:

```
([3.5183036 <SNDNS PRFD> [0 1 0 3.5183036 3.5183036 0 0 3]]
 [7.1731240 <NEWP 3> [0 1 0 3.5183036 3.5183036 0 0 3]]
 [53.986576 <NEWP 2> [0 0 0 0 0 0 0 2]]
 [347.16065 <NEWP 4> [0 0 0 0 0 0 0 2]])
```

Note how the event NEWP executed at that time perpetuates itself by creating a new similar event to be executed at time 7.1731240. The event <SNDNS PRFD> was generated to call in a nurse to treat the new patient. (We might remark as an aside that Type 4 is apparently a very rare type, since the first Type 4 arrival is not scheduled until almost six hours into the simulation.)

A.7.2 PRFD and ASSIGN

PRFD (Patient Ready For Doctor) is the event that is executed whenever a patient is ready to be seen by a doctor. This might occur just after a patient has been admitted to the emergency room, or upon a patient's return from x-ray, or upon any number of other occasions. The exact time is, of course, a function of the model that is implemented (hereafter referred to as the *current model*). PRFD, more than any other event, is dependent upon the current model, for it is PRFD, or at least its subfunction ASSIGN, which is always rewritten with each implementation of a world line tree. Oversimplifying only a bit, one might say that PRFD in a TIGERS simulation has the following tasks:

1. If a doctor is available, assign this doctor to the patient; if not, place the patient in a queue of patients waiting for doctor.

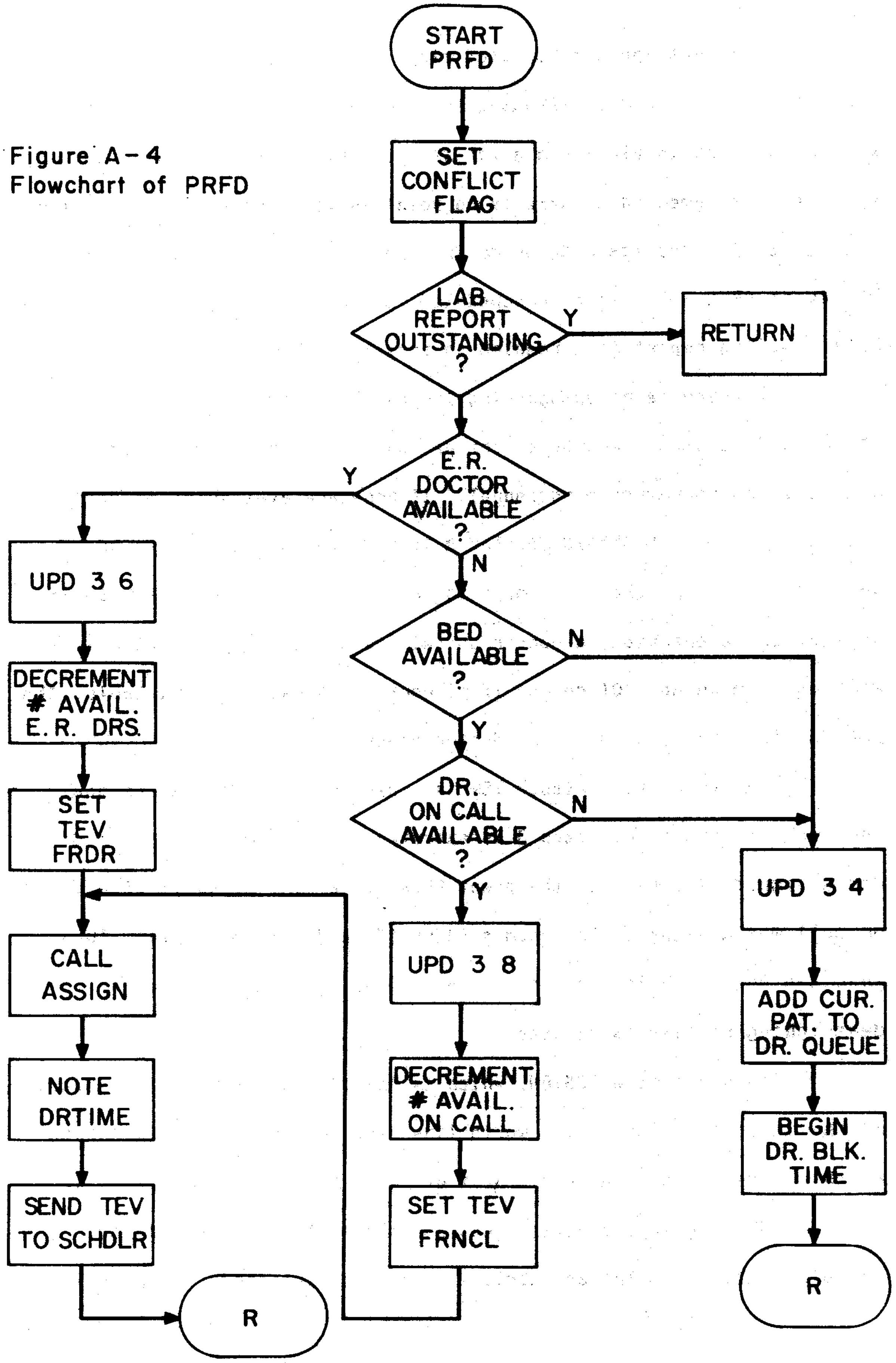
2. Decide duration of visit with doctor, and determine next event in world line of patient.

3. Generate the event to deassign the assigned doctor at the appropriate time.

PRFD generates all events which represent items for which a doctor is usually responsible. Laboratory analyses, x-rays, and consulting physicians from the floor are ordered by the event PRFD. The flowchart for PRFD appears in Figure A-4; although it is largely self-explanatory, some discussion is necessary. Step 2 above is performed by the subroutine ASSIGN which will be discussed below, but first we examine the implementation of steps 1 and 3, referring to Figure A-4.

First the conflict flag is set to 0. This is the heretofore unexplained flag, mentioned in Section A.2, which is the third element of the patient vector. This flag is used by the program to avoid generating redundant instances of PRFD for any given patient. If it were not for this flag, any patient for whom both x-ray and lab tests were ordered might find himself seeing the doctor twice at the same time, since the events associated with both return from x-ray and return from lab generate the event PRFD.

Figure A-4
Flowchart of PRFD



After setting the conflict flag, the program checks to see if there is a lab report outstanding for the current patient. If there is, the patient is placed in a *lab blocked* state until the report comes in. It is assumed that there is no point in visiting a doctor a second time until lab results ordered during the first doctor visit are in. The event PRFD will again be scheduled for the patient for the time just after the lab report is scheduled to arrive.

If there is no outstanding laboratory report, the program attempts to assign a doctor to the patient. It first checks for available emergency room personnel. If none are available, it decides whether the patient should be placed in a *doctor blocked* state or whether an attempt should be made to call in a doctor on call. In the present implementation, a doctor on call is sought only if there is a waiting room queue. Of course if no doctors on call are available, the patient is still placed in the doctor blocked state.

The temporary variable TEV is a device which is used to deassign the correct doctor when step 3 is executed. If an emergency room doctor was assigned, TEV becomes the event FRDR (free E.R. doctor); if a doctor on call was assigned, TEV becomes FRNCL (free doctor on call). Thus when the event TEV is executed at the end of PRFD, the correct deassigning routine is executed.

The subroutine ASSIGN, which is internal to PRFD, is the heart of the implementation of the world line tree in TIGERS. It is ASSIGN which decides what a patient's next event is to be and when it is to occur. ASSIGN generates this event and sends it to the scheduler. It therefore changes with each world line tree implemented. Chapter V discusses the world line tree which has been implemented as a part of this research. Figure A-5 gives a flowchart for ASSIGN which is based upon that tree.

Before further discussing ASSIGN, we digress at this point to introduce random variable functions, the use of which is necessarily understood before ASSIGN can be explained. Random variable functions (r.v. functions) constitute the mechanism employed by TIGERS to introduce randomness into the simulation. As of this writing there are two types of random variable functions. The *boolean* type is called with no arguments, and returns simply *yes* or *no* (actually T or FALSE. As mentioned in Chapter V, the model is structured such that associated with certain key questions are probabilities of the likelihood of certain events occuring which are set by the experimenter. Each boolean r.v. function is associated with one of these probabilities. The function, when called, generates its yes or no value by considering the probability and calling a random number generator. That is, it is as if the function were to precisely weight (bias) a coin, and then flip the coin to generate a value. The test model is so structured that each of the key questions listed in Chapter V has associated with it a probability and a boolean r.v. function. In Figure A-5 each of the decision boxes 6, 17, 18, 20, and 24 represents a call to a boolean r.v. function, and thus also represents a point at which the experimenter assigns a probability.

The second type of r.v. function is the time (t.r.v.) function. T.r.v. functions are used to determine values for random time intervals when they are called for. They are called with no arguments and, like

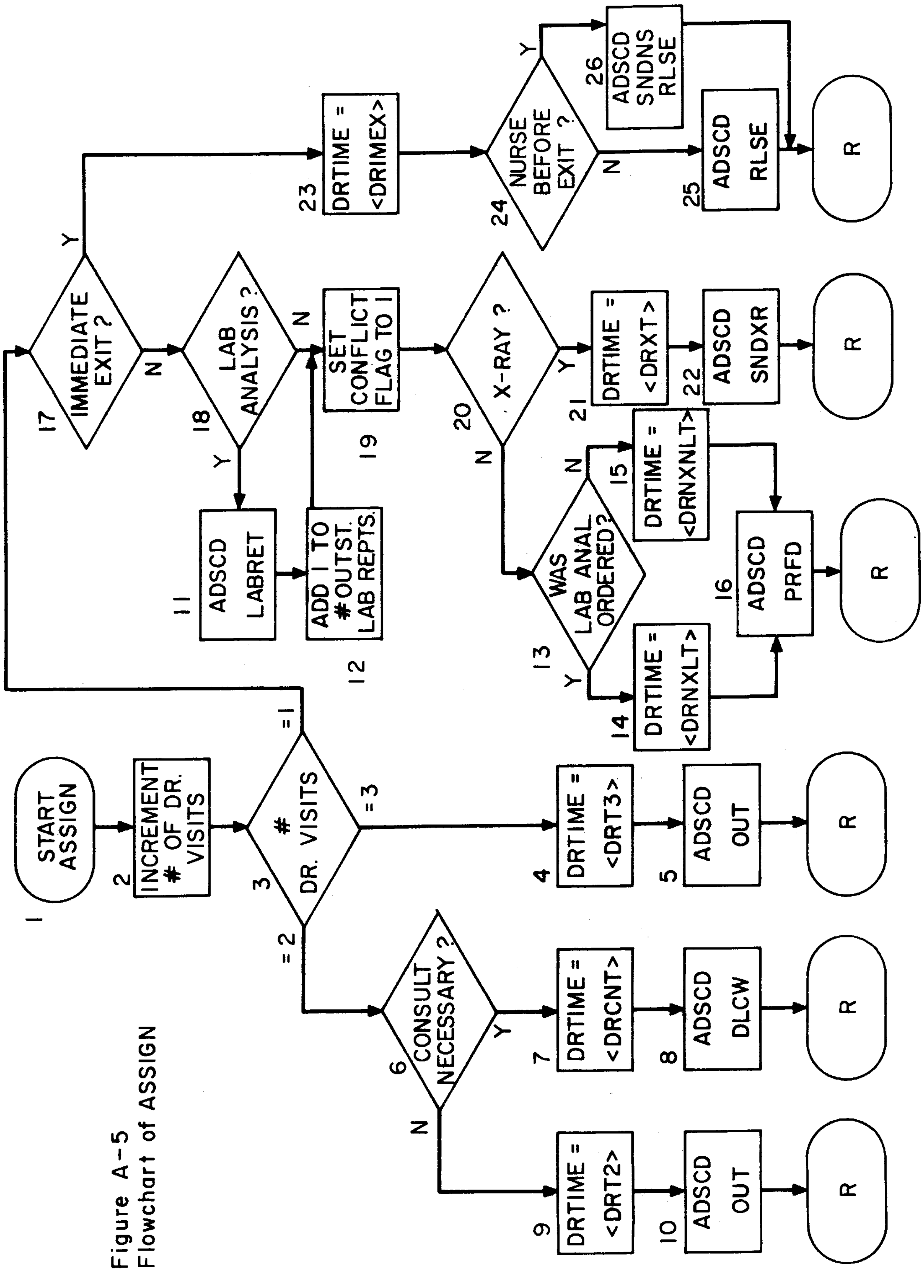


Figure A-5
Flowchart of ASSIGN

boolean r.v. functions, use a random number generator and a parameter set by the experimenter to generate a value. This user-set parameter is usually the mean of the probability distribution function associated with the value being generated. Thus, if the associated parameter of the t.r.v. function DRXT were, say, 30.0; a likely result of the statement

<SET TIMEVALUE <DRXT>>

would be to cause the variable TIMEVALUE to be set to, say, 37.61438. A repetition of the statement would set TIMEVALUE to some entirely different value. For more information on random variable functions, see Section A.8.

We now return to our discussion of ASSIGN and Figure A-5. Recall from Chapter V that the model assumes one, two, or three visits with the E.R. doctor, depending on the complexity of the treatment necessary. This is reflected in the three main subsections of the flowchart, each of which is associated with one of the branches emanating from the decision box which examines the number of doctor visits (box 3). Generally speaking, ASSIGN does the following:

1. Increment the number of doctor visits.

2. Decide which event(s) should be scheduled next for the patient.

3. Call the appropriate r.v. function to decide how long the current doctor visit will last, and therefore when the next event is to be scheduled.

4. Pass this information (time, event, patient) to ADSCD, which creates the event and puts it in its proper place on the schedule.

Let us examine in more detail the three main branches of Figure A-5. Clearly branch one (first doctor visit) is the most complicated. The first decision (box 17) is whether the patient is relatively easy to treat, in which case an immediate exit is scheduled. If this is the case, the appropriate t.r.v. function is called (box 23), and then another decision point is reached (box 24). The question here is whether or not the patient needs to see a nurse before leaving. Once the appropriate event is scheduled, the function returns to the calling function, PRFD.

If an immediate exit is not scheduled, more questions remain. First, is a lab report necessary (box 18)? If so, make the appropriate arrangements (boxes 11 and 12). Then set the conflict flag and continue. Is an x-ray to be taken (box 20)? If so, call the appropriate function to generate doctor time (box 21) and send SNDXR to the scheduler. Then return. If no x-ray is necessary, set the appropriate doctor time (boxes 13, 14, and 15), and send the event to the scheduler (box 16). Then return. Different t.r.v. functions are called, depending upon whether a lab test was scheduled or not.

Branch two is considerably simpler than branch one. On the second doctor visit, the only major decision is whether a consultant physician is to be called or not (box 6). If the consultant is called, the event of his arrival is scheduled (boxes 7 and 8); otherwise the patient is scheduled to leave (boxes 9 and 10).

Finally, branch three is the simplest. This branch is only reached when a doctor sees the patient after the consultant leaves. This calling in a doctor for a third time signifies that this patient has needed much of the emergency room resources. Branch three simply schedules the patient's leaving after seeing the doctor the third time.

A.7.3 SNDXR, XRAYFR, and LABRET

SNDXR is the function which transfers the current patient from the main treatment room to the x-ray facility. This involves the following steps:

1. Increment the number of patients at x-ray.

2. Update field 1 of the current patient to 3 (at x-ray).

3. If the facility is already in use, add (using ADTOQ) the current patient to the end of the queue of patients waiting for x-ray, and return to the scheduler.

4. Otherwise, start the patient's x-ray service. This involves calling the routine XRAYT to generate a random value for the time spent at x-ray, and using ADSCD to send to the scheduler the routine (XRAYFR) to return the patient to the main treatment room.

5. Set the conflict flag to 1.

6. Send to the scheduler the event PRFD to be executed just after XRAYFR.

XRAYFR is the event alluded to above which releases the current patient from the x-ray area. Its algorithm is as follows:

1. Decrement the number of patients at x-ray.

2. Update field 1 of the current patient to 2 (in main treatment room).

3. If there is no x-ray queue, simply return to the scheduler.

4. Otherwise, remove the next in line from the queue, and make this patient the current patient. Then for this new patient repeat steps 4, 5, and 6 of SNDXR above.

LABRET is the event which returns a lab report on a patient to the main treatment room. The actual request for the report is made by the doctor in PRFD (in ASSIGN). When the request is made, the number of reports pending in the lab is incremented, the time until the return of the lab report is calculated (an r.v. function is called), and the event LABRET is sent to the scheduler. LABRET does the following operations:

1. Decrement number of lab reports pending.

2. If the conflict flag is set to 1, then return immediately to the scheduler.

3. Otherwise check whether the patient is at x-ray. If so, return immediately to the scheduler.

4. If the conflict flag is set to 0 and the patient is not at x-ray, send the event PRFD to the scheduler to be executed immediately.

A flowchart of LABRET appears in Figure A-6.

A.7.4 DLCW, CLTARV, CLVDR, AND CLVOUT

DLCW, CLTARV, CLVDR, and CLVOUT are the functions that handle that part of the patient's world line tree related to the visit of a consultant physician (if necessary). All are extremely simple. DLCW (Doctor Leave, Consult Wait), if it is scheduled, is scheduled during the second doctor visit in ASSIGN. It does the following tasks:

1. Update field 3 to 3 (waiting for consultant).

2. Call the r.v. function CNSLTT to find out how long the patient will be in the consult blocked state.

3. Send the event CLTARV to the scheduler.

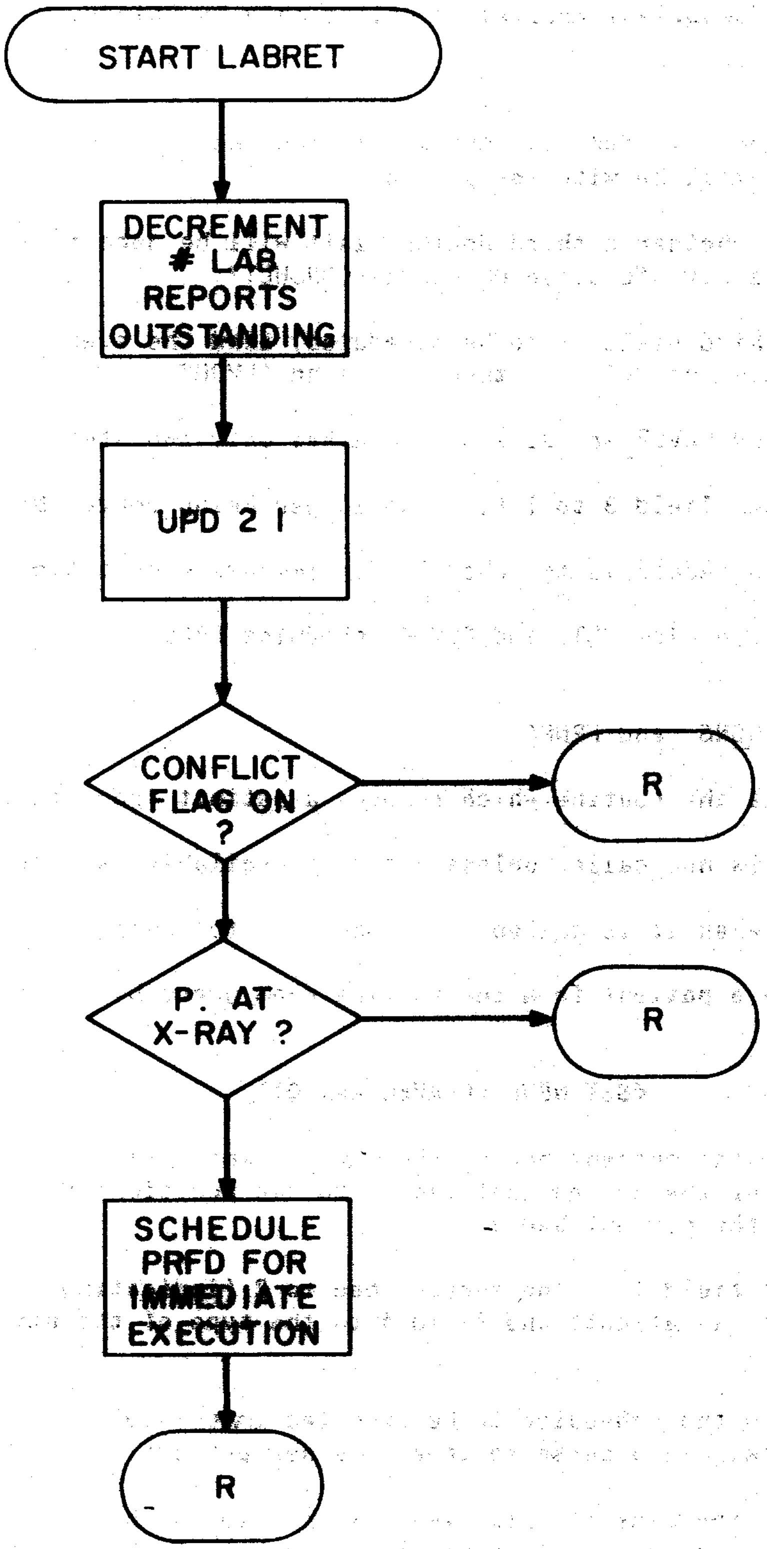

START LABRET
DECREMENT # LAB REPORTS OUTSTANDING
UPD 2 I
CONFLICT FLAG ON ?
R
P. AT X-RAY ?
R
SCHEDULE PRFD FOR IMMEDIATE EXECUTION
R

Figure A-6 Flowchart of LABRET

CLTARV (ConsuLTant ARriVe) is called only by CLTARV. Its algorithm:

1. Call the r.v. function CNSDRT to find out how long the consultant will be with the patient.

2. Decide whether a third doctor visit will be necessary by calling the r.v. function OUT-AFTER-CONSULT?.

3. If a third visit is to be scheduled, send the event CLVDR to the scheduler. Otherwise, send CLVOUT.

CLVOUT and CLVDR are similar; each has only two steps. The first is to update field 3 to 1 (i.e. no longer being served by the consultant). The second is to schedule for immediate execution the next event: CLVOUT schedules OUT, and CLVDR schedules PRFD.

A.7.5 CALLIN, SNDNS, and FRNRS

CALLIN is the routine which removes a patient from the waiting room queue. It is not called unless a bed is available, and this bed is the current bed when it is called. It does the following:

1. Remove a patient from the waiting room queue by using LEAVEQ

```
<SET NEWP <LEAVEQ WAITQ>>
```

2. Assign the current bed to the new patient. (i.e. set element 2 of the new patient vector to the position in BEDSTR of the current bed.)

3. Update field 1 of the current bed to 2 (indicating presence of a patient) and field 4 to the type of the new patient.

4. Send to the scheduler to be executed immediately the event to call in a nurse to treat the new patient.

5. Record the time of this event as the time of beginning of service for the new patient (in element 5 of the patient vector).

SNDNS is the event which assigns a nurse to a patient. It takes one argument, an event, which informs the routine which event is to be sent to the scheduler when the nurse's treatment of the current patient is over. Thus

<SNDNS PRFD>

would cause a nurse (if available) to be assigned to the current patient, and it would cause SNDNS to send the event PRFD to the scheduler to be executed whenever current nurse treatment ended. The call

<SNDNS RLSE>

would similarly cause the event RLSE to be executed when nurse treatment was over.

FRNRS is the routine which deassigns a nurse from a patient and reassigns the nurse to a patient waiting for a nurse (if any such patients exist). This procedure is accomplished by the following steps:

1. Update field 3 of the current patient to 1 (no server).

2. If no other patients are waiting for a nurse, simply increment the number of available nurses and return to the scheduler.

Otherwise:

3. Use LEAVEQ to remove the first patient from the queue of those waiting for nurse service, and assign the nurse just freed to the new patient. (Update field 3 of the new patient to 5).

4. Decide how long this new patient will be treated by the nurse, and send the event FRNRS to the scheduler to be executed at the appropriate time.

5. Send to the scheduler the event representing the next event in the new patient's world line after the nurse leaves. (It knows this event because the event which placed the patient on the queue of patients waiting for a nurse also placed on the queue information describing this event.)

A.7.6 RLSE, OUT, and PDWO

RLSE (ReLeaSe) releases a patient from the system when all treatment is over. It involves four tasks:

1. Incorporate statistics of the current patient into the cumulative statistics.

2. Decrement the number of patients in the system.

3. Remove all traces of the current patient from the current bed. (i.e. Update all four bed fields to 1.)

4. If there is a waiting room queue, send the event CALLIN (to call a patient from the waiting room into the free bed) to the scheduler to be executed immediately.

OUT is the function which prepares the patient to leave the emergency room system. The flowchart for OUT appears in Figure A-7. The routine first decides whether the patient is to undergo observation or not. In calculating the value of the boolean random variable on which the decision is based, the probabilities of the branches at this point in the world line tree of the model are a factor. If the patient is to undergo an observation period, the duration of this period is calculated, and the event PDWO (Patient Done With Observation) is sent to the scheduler. Otherwise, the event RLSE is sent to the scheduler. PDWO does nothing more than generate the event RLSE in a similar manner to OUT when PDWO is not generated. Note that the probability distribution function associated with the time before RLSE is scheduled to be executed varies, depending upon whether the patient is to be admitted to the hospital or released to the outside world.

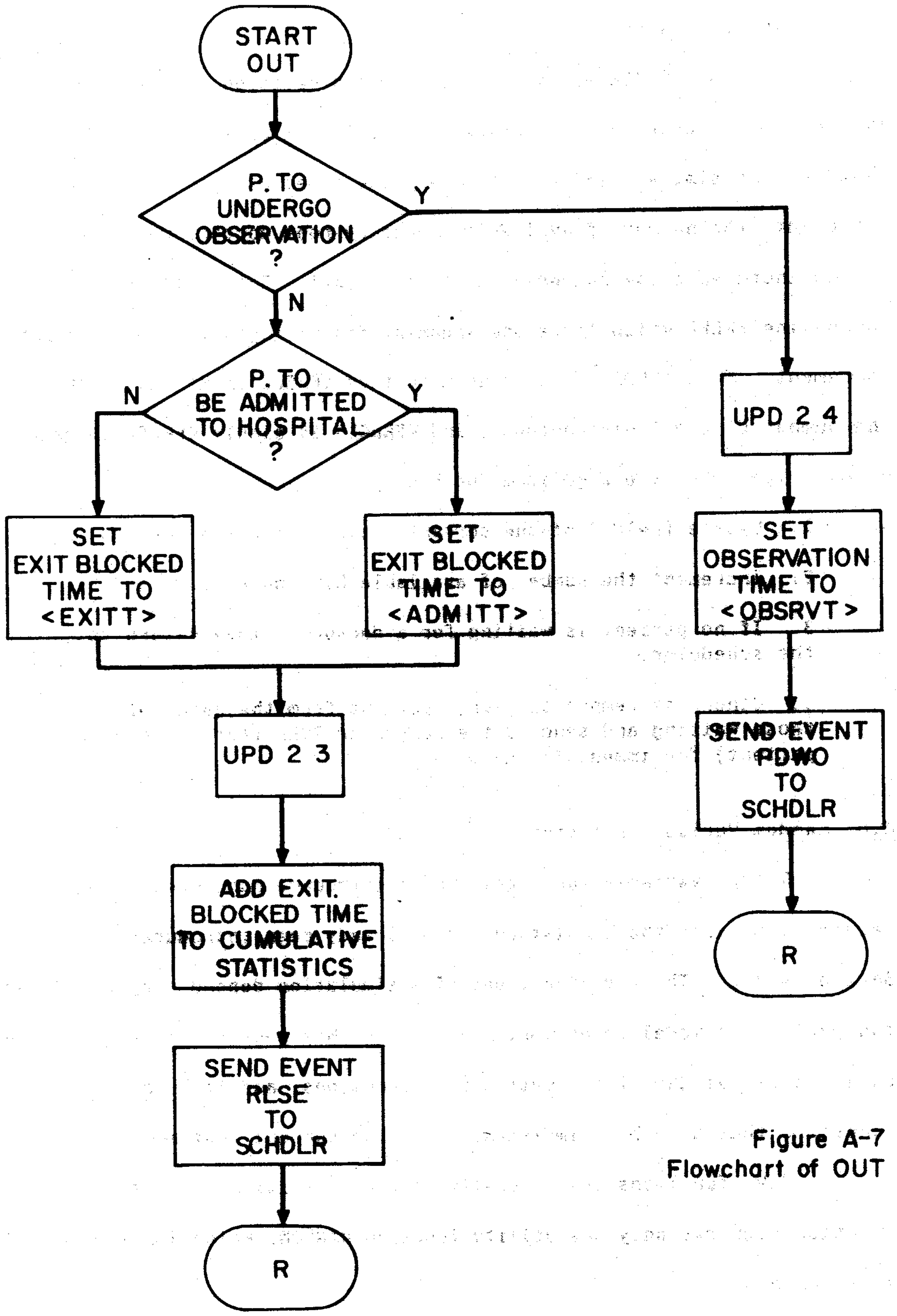

START OUT
P. TO UNDERGO OBSERVATION ?
Y
N
P. TO BE ADMITTED TO HOSPITAL ?
N
Y
SET EXIT BLOCKED TIME TO <EXITT>
SET EXIT BLOCKED TIME TO <ADMITT>
UPD 2 3
ADD EXIT BLOCKED TIME TO CUMULATIVE STATISTICS
SEND EVENT RLSE TO SCHDLR
R
UPD 2 4
SET OBSERVATION TIME TO <OBSRVT>
SEND EVENT PDWO TO SCHDLR
R

Figure A-7
Flowchart of OUT

A.7.7 FRDR and FRNCL

FRDR and FRNCL are the routines which deassign an emergency room doctor and a doctor on call, respectively, from a patient. The two routines are similar; in fact, they are the same, except the first increments the number of available emergency room doctors while the second increments the number of doctors on call. Both call the subroutine FREED which takes one argument describing which data base to increment. Thus <FRDR> is nothing more than <FREED .NDR> (where NDR is the number of available doctors), and <FRNCL> is simply <FREED .NONCL>. We describe below the algorithm for FRDR:

1. Update field 3 of the current patient to 1 (no server).
2. Increment the number of available E.R. doctors.
3. If no patient is waiting for a doctor, simply return to the scheduler.
4. Otherwise remove the first patient from the queue of those waiting and send to the scheduler PRFD (for the new patient) for immediate execution.

A.8 Random Variable Functions

Random variable functions, which introduce randomness, non-determinism, into the simulation, have already been introduced in Section A.7.2. The r.v. functions of a simulation depend very much upon the particular model being implemented. In this section we describe the facilities available for creating r.v. functions, and list those functions that have been implemented for the current test model.

R.v. functions are generally trivial. Writing a boolean r.v. function requires only the utility function RANDOM, which is called with no arguments and returns a number between 0 and 1. Let us say that the probability of a *yes* is P; and of a *no*, 1-P. Then the algorithm for the desired boolean r.v. function is as follows:

1. Call RANDOM.

2. If the value returned is less than P, then return the value T.

3. Otherwise, return FALSE.

For time random variable functions, the algorithms can get a bit more complicated but are still basically simple. In the presently implemented model all time increments are assumed to be characterized by exponentially distributed random variables. Thus for the present model, the only necessary utility function is EXPDIS, which accepts one argument representing the mean of an exponential distribution, and returns a random value of the random variable described by the distribution. Analogous utility functions for other probability distribution functions are being written.

The present t.r.v. functions do nothing more than call EXPDIS with a given mean, but it is conceivable that more complex functions might be more accurate, and therefore desirable. For example, one might want to introduce a fixed delay plus an increment characterized by an exponential or gaussian distribution. Such routines are not difficult to create.

The following eight boolean r.v. functions exist in the present configuration:

XR? -- Is an x-ray necessary?

LAB? -- Is a laboratory analysis called for?

EXIT1? -- Does the patient exit immediately after the first doctor visit?

NUREX? -- If the patient does leave immediately, does he see a nurse first?

ADMIT? -- Is the patient to be admitted to the hospital proper?

CNSLT? -- Does the patient see a consultant?

OUT-AFTER-CONSULT? -- If the patient does see a consultant, does he exit immediately after the consultation.

OBS? -- Does the patient undergo a period of observation before he leaves the system?

There are also seventeen time random variable functions:

ADMITT -- time spent in administrative "red tape" waiting to be admitted to the hospital

ADMIT2 -- time spent in administrative "red tape" waiting to be admitted to the hospital, after having undergone a period of observation

CNSDRT -- time spent with a consultant

CNSLTT -- time spent waiting for a consultant

DRCNT -- time spent on second doctor visit if consultant is scheduled

DRIMEX -- time spent with doctor before immediate exit

DRNXLT -- time spent with doctor on first doctor visit if lab report but no x-ray is scheduled

DRNXNLT -- time spent with doctor on first doctor visit if no lab report and no x-ray are scheduled

DRT2 -- time spent with doctor on second doctor visit if no consultation is scheduled

DRT3 -- time spent with doctor on third doctor visit

DRXT -- time spent with doctor on first doctor visit if patient is sent to x-ray

EXITT -- time spent in exit blocked state before leaving emergency room

EXIT2 -- time spent in exit blocked state before leaving emergency room, if patient has undergone period of observation

LABT -- time before laboratory analysis results are returned

NURST -- time spent with nurse

OBSRVT -- time spent under observation

XRAYT -- time spent at x-ray

A.9 Graphics Functions

In this section we discuss the graphical updating of the display in Simulate Mode. Maintaining the buttons (most of which are in Modify Mode) also involves graphics, but we discuss button handling in a separate section.

Aside from the button handling routines and the functions which actually create the pictures (which are not described in this document), there is only one main graphics function, NPASS. Recall that a simulation in TIGERS is a loop which repeatedly calls SCHDLR. In this loop each call of SCHDLR is immediately followed by a call to NPASS (Figure A-1). NPASS is the routine which looks at BEDSTR and CNTSTR and updates the display to reflect any changes made during the execution of the last event.

The display in Simulate Mode is composed of three types of pictures:

A. The static framework. This includes the lines delineating the various sectors, and the identifying labels.

B. The beds in the main treatment area and their associated fields. Relevant information is in BEDSTR.

C. The collections of stick figures which represent groups of various types of persons. Relevant information is in CNTSTR. As of this writing these collections include

1. patients awaiting lab analysis results
2. patients in x-ray area
3. available emergency room doctors
4. available doctors on call
5. available nurses
6. patients in waiting room queue

Clearly NPASS need be concerned only with types B and C. The routine updates the two types of pictures in two distinct phases, type C followed by type B. We discuss the two phases separately.

For maximum execution speed, each of the six collections of type C is updated by its own updating routine. These routines comprise the elements of the vector RFQS. Thus each element of RFQS corresponds to one of the element pairs of CNTSTR (Section A.3). The type C pictures are updated using the following algorithm:

1. Check item i (i.e. the i'th pair) of CNTSTR. Was it changed during the execution of the last event? (I.e. is it flagged with a "1"?)

2. If no, increment i, and go back to 1 to check the next item of CNTSTR.

3. If yes, call (i.e. execute) element i of RFQS, thus updating the displayed value of the associated value in CNTSTR. Then increment i, and go back to 1 to check the next item of CNTSTR.

When i exceeds the number of items (currently six), the updating of the type C pictures is complete.

The type B pictures are updated in a similar manner, except BEDSTR is the main data base instead of CNTSTR, and only one routine is necessary, not six. All the beds can be updated by the same routine. Essentially the algorithm consists of checking each bed, updating those that have been changed and skipping those that have not. The programming is such that the numbers in BEDSTR associated with each bed field are used to call the approriate picture.

A.10 Time Functions

An important part of the TIGERS environment is the provided simulated time flow. There are two time functions: STIME, which returns the present simulated time in minutes, and CHANGESPEED, which is used to alter the rate of simulated time flow.

There are four internal variables used by these time functions:

TSCALE is the number of thirtieths of a real second in one simulated minute.

LASTIME is the absolute real time that CHANGESPEED was last called.

LASTMIN is the simulated time that CHANGESPEED was last called.

TTEMP is the absolute real time that STIME was last called.

The "absolute real time" is the time which is supplied by the computer's clock. This time is given as the number of thirtieths of a second since the operating system was last started.

STIME takes one argument, the current value of TSCALE. It returns a floating point number which is the simulated time in minutes since the simulation was initialized. It does the following steps:

1. Read the absolute real time from the computer's clock. Call it T.

2. Calculate the simulated time as

   LASTMIN + ((T - LASTIME) / TSCALE).

STIME is written in assembler language rather than MUDDLE because MUDDLE cannot access the clock and because the function is called so often that execution speed is critical.

CHANGESPEED also takes one argument, the new value of TSCALE. It does the following:

1. Set LASTMIN to the current simulated time
2. Set TSCALE to the given new value
3. Set LASTIME to the present absolute real time

CHANGESPEED also sets the value of the variable OFTEN, which is related to the frequency that the displayed clock is updated.

A.11 Interrupt Handling Functions

MUDDLE has a facility through which MUDDLE programs can access the PDP-10'S half second clock interrupt. TIGERS has two interrupt handling functions which use this facility: UPCLOCK, which updates the displayed simulated time, and CHECKBI, which makes the program interactive. *The reader should bear in mind as he reads the descriptions below, that these functions are executed every half second.*

The variable OFTEN, whose value is set by CHANGESPEED, defines how often the display will be refreshed. It is set such that the displayed clock is refreshed approximately as often as it changes. Thus if a simulated minute were equal to five real seconds, the clock would be updated about every five seconds.

UPCLOCK (UPdate CLOCK) does the following:

1. Decide whether it is time to update the displayed clock. If not, return to the simulation.

2. Decide whether the hour needs to be updated as well as the minute. If so, update the hour and the minute on the displayed clock.

3. Otherwise, update only the minute on the displayed clock.

4. Return to the simulation.

CHECKBT (CHECK BuTtons) employs the following algorithm:

1. Check: have any buttons been hit in the past half second?

2. If not, return to the simulation.

3. If so, change the displayed square associated with the button to a triangle. (This indicates to the user that the button has been hit and that the routine associated with the button is being executed.) Then desensitize the button so that the button cannot be hit while its associated routine is running. Then proceed to execute the routine associated with the hit button. Finally, when execution of the routine is terminated, restore the square and resensitize the button. Return to the simulation.

A.12 The Button Protocol

Since button handling is critical in making TIGERS interactive, a button protocol has been implemented which makes it relatively easy to create buttons of various types. Currently three types of buttons exist:

1. Unlabeled buttons
2. Labeled buttons
3. Labeled buttons with value

Every button -- regardless of type -- is a single MUDDLE picture, and every button has associated with it a single routine. Whenever the button is hit the associated button servicing routine is executed. Two main data bases are associated with the buttons: PICVECT is a vector of all the buttons, and SERVECT is a vector of all the button handling routines. Each element of PICVECT is associated with an element of SERVECT.

Unlabeled buttons appear simply as squares. No text string is associated with the button as part of the same picture, although external labels are generally added. Examples of this type are the squares above the numbers used for creating text strings in Modify Mode.

Labeled buttons have a text string associated with the square as part of the same picture. Examples of this type of button are the buttons used for changing modes.

Labeled buttons with value have text strings representing values of some variable associated with the button. Examples of this type are all the buttons in the top half of the screen in Modify Mode.

The function BUTTON creates buttons of type 2; the function BUTTON2, buttons of type 1; and the function BUTTON3, buttons of type 3. A naming protocol has been implemented which aids in associating text strings, values, and service functions with their associated buttons. We will not discuss the details of this protocol in this document, but we will give an example of a call to a button creating function. This should communicate a general idea of how the protocol is useful. The call

<BUTTON3 -375 200 BLAMBDA ,QLAMBDA ,QVLAMBDA 28>

creates the button in Modify Mode which is associated with the arrival rate LAMBDA. (-375 200) are the (x y) co-ordinates of the center of the square; BLAMBDA is the name of the button; QLAMBDA is the name of the identifying text string; QVLAMBDA is the name of the text string associated with the value of LAMBDA; and 28 is the index of BLAMBDA's position in PICVECT and RLAMBDA's position in SERVECT. RLAMBDA is of course the associate button servicing routine.

A.13 The Button Servicing Functions

As of this writing there are twenty-nine buttons in TIGERS. It is the purpose of this section to describe their functions by discussing their associated handling routines. We will not describe every routine individually, since there are groups of similar routines. We will, however, discuss in detail at least one member of each "equivalence class."

A.13.1 SIMULATE Mode -- RSTART, RSTOP RINITIALIZE, RSTATON, RSTATOFF

RSTART was discussed in Section 4.2.2., RCONTINUE, [illegible] following:

1. Call START (Section A.6.2).

2. Call NPASS (Section A.9).

3. Initialize the count of executed events.

4. Enable the clock updating routine. (i.e. cause it to be executed at every half second clock interrupt.)

5. Call SIMULATE (which is an infinite loop).

RSTOP stops the simulation. It has three steps:

1. Set STOPBIT to T. This causes an exit from the infinite loop in SIMULATE.

2. Disable the clock updating routine. (I.e., disassociate it from the clock interrupt.)

3. Set the variable STOPSTIME to be equal to the current simulated time.

RCONTINUE continues the simulation after its having been stopped by RSTOP. It does the following:

1. Set LASTIME to the current absolute real time.

2. Set LASTMIN to the value of STOPSTIME set by RSTOP.

3. Enable the clock updating routines.

4. Call SIMULATE.

RINITIALIZE resets the simulation to time zero. It does nothing more than call REINIT (Section A.6.2) and NPASS (Section A.9).

RSTATON displays the cumulative statistics.

RSTATOFF erases the cumulative statistics from the display.

A.13.2 Changing Modes -- RMODIFY and RRETURN

RMODIFY changes modes from Simulate to Modify. This involves the following:

1. Call RSTOP (Section A.13.1).

2. Call RERASE (Section A.13.3).

3. Erase the display.

4. Display the pictures of Modify Mode.

RRTSM (Return To Simulate Mode) changes modes back to Simulate mode:

1. Set STOPBIT to FALSE. This will cause SIMULATE, when started, to loop until STOPBIT is reset to T by RSTOP.

2. Call NPASS (Section A.9).

3. Erase the display.

4. Display the pictures of Simulate mode.

A.13.3 Creating Text Strings -- R1, R2, R3, ..., R0, R., RERASE

This section discusses the buttons which comprise the "blackboard" in Modify Mode. The following variables are relevant:

CURSTR - Current text string. This is the string that is displayed above the "blackboard" buttons. The user builds this string using the blackboard.

CURVAL - Floating point value represented by the current text string

The routines for R1 through R0 and R. are similar. We describe here the algorithm for R1:

1. Append "1" to the current string.

2. Update the display to reflect the change.

3. Set CURVAL to the floating point number represented by the new current string.

RERASE sets CURSTR to "" (the empty string), updates the display to reflect the change, and sets CURVAL to FALSE.

A.13.4 Buttons With Values

This section is concerned with the buttons which are used for altering the parameters of the TIGERS model. As of this writing there are nine -- which buttons exist depends upon which parameters the experimenter wishes to vary. The buttons currently implemented are as follows:

BDRS - used to choose number of emergency room doctors in the model system.

BNRS - used to choose number of nurses.

BONCL - used to choose number of doctors on call.

BTYPE - used to select the *current type* of Modify Mode. Certain parameters vary with type of patient, the best example being the arrival rate of patients to the system. The value displayed of any of these type dependent parameters is the value associated with the current type. Whenever the current type is changed, the displayed values of these parameters are changed appropriately.

BXRT - used to choose mean time spent by patient of current type at x-ray.

BLT - used to choose mean time for return of lab analysis report for patient of current type.

BSCL - used to choose ratio of simulated time to real time.

BBEDS - used to choose number of beds in the main treatment room.

BLAMBDA - used to set arrival rate of patients of the current type to the system in persons per hour.

The functions associated with all these buttons are similar in that the first thing they do is check to see if CURVAL has a value other than FALSE. If it does not, the phrase "CHOOSE NEW VALUE" is displayed to indicate to the user that he should use the blackboard to assign a value to CURVAL. In our descriptions below of the algorithms, we assume that CURVAL has in fact been assigned a value.

RDRS, RNRS, RONCL, and RTYPE are similar. We choose a representative example, and elucidate below the algorithm for RDRS:

1. Set the flag in CNTSTR which indicates that a change has been made in the number of emergency room doctors.

2. In CNTSTR, set the number of doctors equal to the value of CURVAL (rounded off to the lowest integer).

3. Reset CURSTR to the empty string.

4. Reset CURVAL to FALSE.

5. Update the displayed values of CURSTR and the number of doctors to their new values.

RXRT, RLT, and RLAMBDA form another equivalence class. The displayed values associated with these three functions all depend upon the current type (selected using BTYPE). Except for the fact that the current type affects the actual variable that is changed, these three routines are similar to the four discussed above. RBEDS differs from the four only in that instead of merely changing a certain value, a whole new picture has to be compiled based upon the new number of beds. RSCL differs only in that instead of changing a value directly, it calls CHANGESPEED (Section A.10).

References

1. For a complete description of MUDDLE, see G. Pfister, *A Muddle Primer?*, Programming Technology Division Document SYS.11.01, M.I.T. Project MAC.

APPENDIX B:
SAMPLES OF MUDDLE SOURCE CODE

The purpose of this appendix is simply to give the reader an idea of what MUDDLE looks like. Following are listings of the two functions PRFD and ASSIGN. They are flowcharted in Figures A-6 and A-7, respectively.

```
<DEFINE PRFD ()
        <PRINT "PRFD">
        <PROG ()
              <PUT .CPATNT 3 0>
              <COND (<==? 2 <<+ <2 .CPATNT> 2> .BEDSTR>>
                     <RETURN 0>)
                    (<0? <VAL .NDR>>
                     <COND (<LE? <VAL .WQL>>
                            <UPD 3 4>
                            <ADTOQ DRQ .CPATNT>
                            <ADTOQ DRQ .RT>
                            <RETURN 1>)
                           (<LE? <VAL .NONCL>>
                            <UPD 3 4>
                            <ADTOQ DRQ .CPATNT>
                            <ADTOQ DRQ .RT>
                            <RETURN 2>)
                           (ELSE
                            <UPD 3 8>
                            <SUB1 .NONCL>
                            <SET TEV FRNCL>))>)
                    (ELSE
                     <SUB1 .NDR>
                     <UPD 3 6>
                     <SET TEV FRDR>))>
              <ASSIGN .CPATNT>
              <SET TTIME <- .TTIME 1>>
              <PUT .CPATNT 6 <+ <6 .CPATNT> <- .TTIME .RT>>>
              <ADSCD .TTIME <FORM .TEV>>>>


<DEFINE ASSIGN (CPAT "AUX" LAB)
        <PRINT "ASSIGN">
        <PROG ()
              <PUT .CPAT 1 <+ 1 <1 .CPAT>>>
              <COND (<==? <1 .CPAT> 1>
                     <COND (<EXIT1?>
         <SET TTIME <+ .RT <DRIMEX>>>
         <COND (<NUREX?>
                 <ADSCD .TTIME
                         '<SNDNS RLSE>
                         .CPAT>
                 <RETURN 1>)>
         <ADSCD .TTIME '<RLSE> .CPAT>
         <UPD 2 3>
         <RETURN 2>)>
 <COND (<SET LAB <LAB?>>
         <ADSCD <+ .RT <LABT>>
                 '<LABRET>
                 .CPAT>
         <UPD 2 2 .CPAT>
         <ADD1 .LABQL>)>
 <COND (<XR?>
         <PUT .CPAT 3 1>
         <ADSCD <SET TTIME <+ .RT <DRXT>>>
                 '<SNDXR>
                 .CPAT>
         <RETURN 7>)>
 <PUT .CPAT 3 1>
 <ADSCD <SET TTIME
                 <+ .RT
                  <COND (.LAB <DRNXLT>)
                         (<DRNXNLT>)>>>
         '<PRFD>
         .CPAT>
 <RETURN 3>)
((<==? <1 .CPAT> 2>
 <COND (<CNSLT?>
         <ADSCD <SET TTIME <+ .RT <DRCNT>>>
                 '<DLGU>
                 .CPAT>
         <RETURN 4>)>
 <ADSCD <SET TTIME <+ .RT <DRT2>>>
         '<OUT>
         .CPAT>
 <RETURN 5>)
((<==? <1 .CPAT> 3>
 <ADSCD <SET TTIME <+ .RT <DRT3>>>
         '<OUT>
         .CPAT>
 <RETURN 6>)
(ELSE <ERROR TOO-MANY-DOCTOR-VISITS>)>>>
```