LEVEL II

INVENTORY 1

AD-A150 973

DTIC ACCESSION NUMBER

AFIT/CI/NR-85-11T

DOCUMENT IDENTIFICATION

1984

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR

NTIS	GRA&I	☒
DTIC	TAB	☐
UNANNOUNCED		☐

JUSTIFICATION

BY
DISTRIBUTION /
AVAILABILITY CODES

DIST	AVAIL AND/OR SPECIAL
A-1	

DISTRIBUTION STAMP

DTIC
COPY
INSPECTED

DTIC
SELECTED
MAR 12 1985
D

DATE ACCESSIONED

DATE RETURNED

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NO.
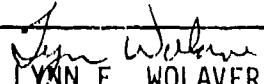
DTIC FORM 70A
DEC 83

DOCUMENT PROCESSING SHEET

PREVIOUS EDITION MAY BE USED UNTIL
STOCK IS EXHAUSTED.

85 03 11 059

AD-A150 973

	READ INSTRUCTIONS BEFORE COMPLETING FORM

AFIT/CI/NR 85-11T

	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
Development of a Hospital Information System	THESIS/DISSERTATION
	6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
Larry L. Cobler	

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
AFIT STUDENT AT: Trinity University	

11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE
AFIT/NR WPAFB OH 45433	1984
	13. NUMBER OF PAGES
	159

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	UNCLASS
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1

LYNN E. WOLAVER 21 Feb 85
Dean for Research and
Professional Development
AFIT, Wright-Patterson AFB OH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1473 1 JAN 73   EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

85 03 11 059

DEVELOPMENT OF A HOSPITAL

INFORMATION SYSTEM

ABSTRACT OF THESIS

Presented to the Faculty of Trinity University
in Partial Fulfillment of the Requirements

For the Degree of

Master of Science

in

Computing and Information Sciences

By

Larry L. Cobler, B.B.A.

This thesis will explore Hospital Information Systems and
explain an approach for the successful development and im-
plementation of an information system. Tracing the develop-
ment of Hospital Information Systems to the present will
reveal the inadequacies of many of the currently installed
systems. The enactment of recent regulations affecting the
way hospitals are reimbursed for treatment costs has created
the need for more sophisticated information systems.
Whereas prior automated systems where primarily used in
transaction processing applications, new systems must be
developed which can provide comprehensive information for

decision-making activities.

The implementation of hospital information systems has heretofore been relegated to vendors who provided a "canned" system which was sufficiently broad enough to be used at many hospitals. The more sophisticated needs demands more individualized sytems. The administrator will have to become more involved in the management of the development process. The Systems Development Life Cycle is an approach which can be used to successfully produce the needed information system. In particular, the use of the Structured Techniques for systems development will be explained. It will be shown that the major costs of an information system are not in the development process but in the maintenance of the system after delivery and the benefit of using the Structured Tools is in reducing the maintenance costs by developing systems that are easily maintained.

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (AU). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: Development of a Hospital Information System

AUTHOR: COBLER, LARRY L.

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES   ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES   ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS _____   ( ) b. $_____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY   ( ) b. SIGNIFICANT   ( ) c. SLIGHTLY   ( ) d. OF NO
        SIGNIFICANT   SIGNIFICANT   SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME	GRADE	POSITION
ORGANIZATION	LOCATION	

STATEMENT(s):

DEVELOPMENT OF A HOSPITAL

INFORMATION SYSTEM

ABSTRACT OF THESIS

Presented to the Faculty of Trinity University
in Partial Fulfillment of the Requirements

For the Degree of

Master of Science

in

Computing and Information Sciences

By

Larry L. Cobler, B.B.A.

This thesis will explore Hospital Information Systems and
explain an approach for the successful development and im-
plementation of an information system. Tracing the develop-
ment of Hospital Information Systems to the present will
reveal the inadequacies of many of the currently installed
systems. The enactment of recent regulations affecting the
way hospitals are reimbursed for treatment costs has created
the need for more sophisticated information systems.
Whereas prior automated systems where primarily used in
transaction processing applications, new systems must be
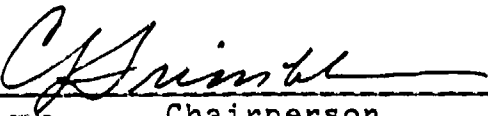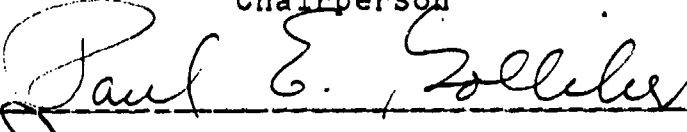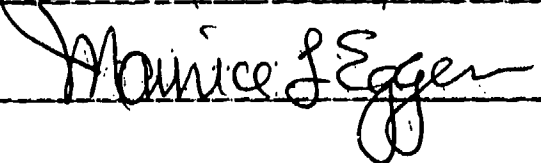developed which can provide comprehensive information for
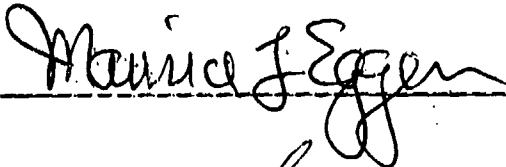
decision-making activities.

The implementation of hospital information systems has heretofore been relegated to vendors who provided a "canned" system which was sufficiently broad enough to be used at many hospitals. The more sophisticated needs demands more individualized sytems. The administrator will have to become more involved in the management of the development process. The Systems Development Life Cycle is an approach which can be used to successfully produce the needed information system. In particular, the use of the Structured Techniques for systems development will be explained. It will be shown that the major costs of an information system are not in the development process but in the maintenance of the system after delivery and the benefit of using the Structured Tools is in reducing the maintenance costs by developing systems that are easily maintained.

DEVELOPMENT OF A HOSPITAL

INFORMATION SYSTEM

Larry L. Cobler
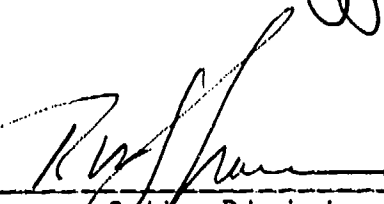

APPROVED BY THE THESIS COMMITTEE:

_____
Chairperson

_____

_____

APPROVED BY THE CHAIRPERSON OF THE DEPARTMENT:

_____

APPROVED BY THE DEAN:

_____
Dean of the Division of
Business and Administrative Studies

December 16, 1984

iii

DEVELOPMENT OF A HOSPITAL

INFORMATION SYSTEM

By

Larry L. Cobler, B.B.A.

THESIS

Presented to the Faculty of Trinity University
in Partial Fulfillment of the Requirements

For the Degree of

Master of Science

in

Computing and Information Sciences

TRINITY UNIVERSITY
December 16, 1984

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER I

INTRODUCTION

Objective and Scope

The age of the computer is upon us. For some disci-
plines the advancing use of the computer has been faster and
more all-encompassing than for others. The business sector
and the military must be regarded as among those in the
forefront, while the social sciences and most hospitals have
lagged behind. It is the intent of this thesis to examine
the historical development of hospital information systems
(HIS) with the hope that from its history one can learn
those ideas that have worked and lead to further progress
and also those ideas that have been less than successful.
In addition, this study will provide a thorough look at an
accepted approach to developing an automated information
system. This approach involves using the Systems Develop-
ment Life Cycle by first doing a Systems Analysis, a Systems
Design, coding and testing and then finally, systems imple-
mentation. The study will lean heavily on discussion of
structured tools as a technique for performing the steps of
the life cycle; however, space will also be given to discuss
traditional techniques. Given the lessons of history and a
proven successful process for developing an information
system, the reader should then be able to knowledgeably

1

participate directly or indirectly in the evaluation of an existing hospital information system or in the development of a new one.

What is an HIS

"What does 'hospital information system' mean? 'It means just what I choose it to mean -- neither more nor less,' said Humpty Dumpty."[1, p. xv] Here is the first problem to be encountered. The typical hospital administrator is either deeply entrenched in the process of determining if an automated hospital information system is needed, in trying to maintain a system, or in trying to update the system currently in place. It is the administrator who is in the middle, besieged by demands from the board of directors to "modernize", frustrated by staff who can't understand why things can't be done like they have always been done, and inundated by vendors who have "the perfect system" for the hospital. This pressure-cooker atmosphere has often lead to short-sighted and disasterous decisions. It is imperative in situations like this to be fully informed before a decision is made. The first step to being knowledgeable about a subject is to insure that when discussing it, all involved are understanding the same message. The way to do this is to define terms.

Definitions

A <u>System</u> is any set of objects and   eas, and their
interrelationships which are ordered t   ommon goal or
purpose[2, p. 9]. This broad definition gives room for the
concept of systems within systems or subsystems. In a
hospital setting this could be seen as the hospital being
the overall system and each department or division within
the hospital being another system or subsystem. A typical
list of systems within a hospital is included in Figure 1-
1[9, p. 4].

Information is in the eyes of the beholder. A trite
and overused statement but true nonetheless. And, if some-
thing is information for one and not for another, what is it
for the latter? It is data. An important distinction must
be made between data and information. Data are raw facts.
Information is data placed into a meaningful context for its
recipient[2, p. 4]. For those who doubt there is a dif-
ference, visit a manager who frequently receives "management
reports" and ask him how much of what he reads is useful.
Even more revealing is to look in his trash can to find
those "indispensable" reports. What's in the trash is data
and what's on his desk being used in making management
decisions is information. The tragedy with this possibly
lighthearted poke at information versus data is that the
consequences of developing an information system that pro-
duces data and not information can easily be converted to

PATIENT SERVICES

Inpatient Services Systems

* Admissions, Discharge,
   Transfer/Census Control
* Medication Distribution
* Nursing Services
* Support Services Systems
   + Patient Food Services
   + Linen/Laundry
   + Patient Transportation
   + Housekeeping
   + Social Services
   + Patient Information
      Services

Ambulatory Care Services
Systems

* Emergency Services
* Referred Outpatient
   Services
* Clinics

General Patient Services
Systems

* Diagnostic Services
   Systems
   + Laboratory
   + Diagnostic Radiology
   + Therapeutic Radiology
   + Nuclear Medicine
   + Respiratory Services
   + EKG/EEG Services
* Rehabilitation Services
* Surgical Services

Patient Records Management
Systems

* Transcription
* Indexing, Storage, and
   Retrieval
* Quality Assurance

MANAGEMENT SERVICES

Financial Management Systems

* Patient Charging, Billing,
   and Accounts Receivable
   Systems
   + Charging
   + Credits
   + Billing
   + Accounts Receivable
* Budgeting
* Accounts Payable
* General Accounting
* Cash Management

Personnel Management Systems

* Timekeeping/Payroll
* Position Control
* Evaluation and Training

Materials Management Systems

* Capital Equipment
* Purchasing and General
   Store
* Central Supply

Facilities and Equipment
Management Systems

* Work Orders
* Scheduled Maintenance

Management Planning and
Control Systems

* Internal Management
   Reporting
* External Reporting

Figure 1-1: Typical List of Systems Within a Hospital

millions of dollars and thousands of man hours wasted.

This cannot be better illustrated than by a situation
reported in Computerworld newspaper on February 21, 1983.
The article, titled "Proprietary Language Snags $3 Million
Budget System", describes the frustration encountered by the
Santa Clara County, California, government while trying to
implement the Comprehensive Budget and Management Informa-
tion System (CBMIS) produced by American Management Systems
of Arlington, Virginia. Granted, this product is probably
very well written and could work well for the right company,
but after three years of trying to install and modify this
$3 million dollar program the results were very disappoint-
ing. The following quote shows one of the major problems
Santa Clara County had with the system but shows even more
the importance of differentiating between data and informa-
tion.

> Another of CBMIS' major purported shortcomings is
> that the management reports it produces are typically ill
> formatted, overly complicated and just downright useless,
> Rixman [the county's fiscal services manager] said. "We
> receive a voluminous number of reports from the system,
> but most of them I simply throw away," said Maya
> Bernardo, a senior analyst with the county's Revenue and
> Systems Agency. "Only two or three of the reports that
> cross my desk contain just the right level of detail for
> someone in a position like mine," Bernardo said.[47]

Hopefully one of the results of correctly developing a
formal HIS is that everyone with information needs will get
information and not data.

An Information System, in the context of previous
definitions, is really a subsystem of a larger total system.

One can look at the hospital as being logically divided into three subsystems:

1) the operations subsystem which includes all the activities, material flow, and people directly related to performing the primary functions of the hospital, i.e., the doctors, nurses and the other health care providers;

2) the management subsystem which includes all the people and activities directly related to determining the planning, controlling, and decision-making aspects of the operations subsystem. This would be the CEO, the hospital administrator and all the administrative staff; and,

3) the information subsystem which is the collection of people, machines, ideas, and activities that gather and process data in a manner that will meet the formal information requirements of an organization[4, p. 26]. The pervasive nature of the information system can be expressed as a network reaching into all parts of the organization -- the connective tissue which links all other systems[5, p. 36]. Figure 1-2[4, p. 27] shows the interrelationships between the three major subsystems of the organization.

A problem which arises when defining a Hospital Information System (HIS), is that there are so many differing views as to what an HIS is and what it should do. There tends to be a plethora of definitions. Each author slants his definition to fit the emphasis of his writings. Another controversy is whether to call this as yet undefined system

LEGEND:

——→ Flow of data and information

▨▨▨▷ Flow of people, work, material, etc.
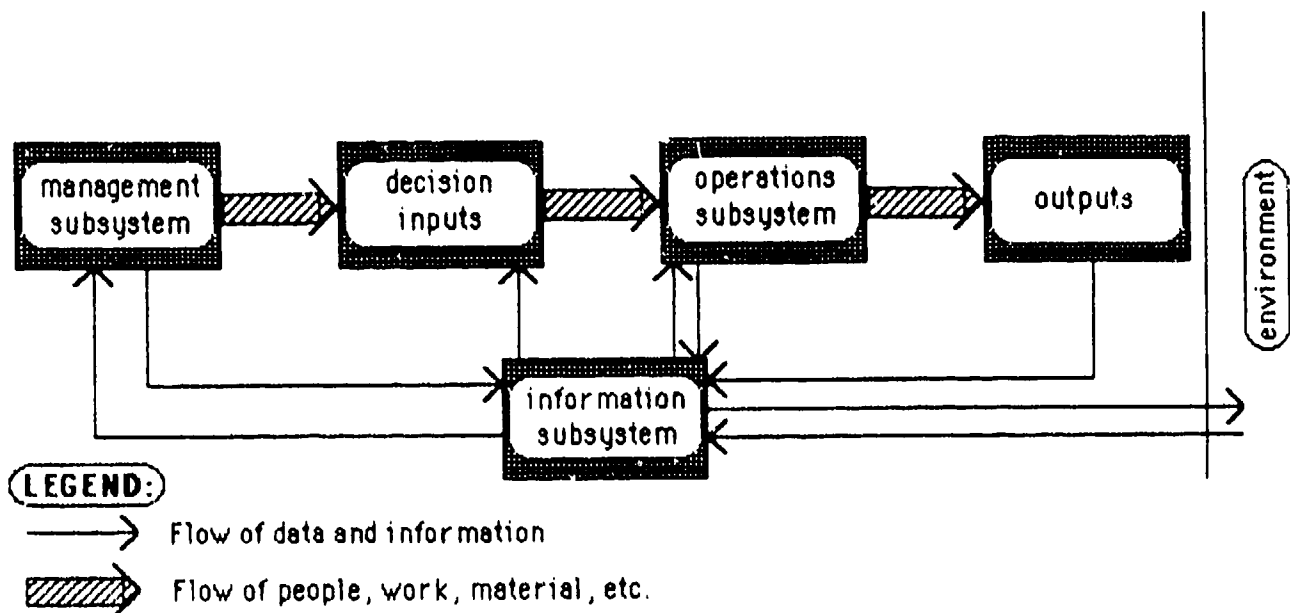
Figure 1-2: Relationship of the Information Subsystem to the Total System

a Medical Information System (MIS; not to be confused with Management Information Systems) or a Hospital Information System. One author states his preference for MIS because it emphasizes the patient care component of a computer based system which is able to provide all the information pertinent to a patient's care throughout the hospital[6, pp. 7,8]. Still another author goes so far as to divide HISs into classes and levels within classes based on whether the HIS is composed primarily of individual stand-alone systems in various departments or whether the systems in the departments are all tied together. Furthermore, he makes a distinction whether the system is administratively oriented or patient oriented[7, p. 13]. And finally, another author defines an HIS very succinctly as, "A set of formal arrangements by which facts concerning the health or health care of

individual patients are stored and processed in computers."[18, p. 9] Another factor is added here to the problem of defining an HIS -- the computer.

The question arises -- is a computer needed to have an HIS? The answer to this question will be developed more fully later, but for the moment the answer is "no." The temptation here is to provide a lengthy definition that encompasses everything anyone has defined as an HIS or MIS. Unfortunately, neither time nor space permit this, so logic is used.

First, since in the research for this paper, most references are to Hospital Information Systems, the author will defer to the majority of the writers. The reader should keep in mind however, that although some authors prefer another title for the information system discussed here, most of the time there will be agreement on the substantial points as to what that system is and the function it performs. Secondly, since information and system have been previously defined, those definitions should be utilized here realizing they are combined in the context of a hospital setting. So the definition of a Hospital Information System follows:

> a system in a hospital that collects data and
> transforms it into information.

Notice no assumptions are made about what method is used to perform this task, whether it be automated or manual and no presumptions are made as to the final destination of the

information, whether it be primarily for clinical use or administrative. Correctly, the only assumptions made are about the terms already defined.

So, what is the distinction between an information system at General Electric Co. and the one in a hospital? The answer is, there is none. Both fulfill the same objective, turning data into information. The distinction occurs not in the definition nor the objective but in the implementation. Later the distinctives of an HIS will be discussed, but it should be noted that much of the research for this paper, especially that of Chapters 2, 3, 4, and 5 was done in the field of information systems proper. And, the reason this study can still be useful to a hospital is because an information system is an integral concept of every business and the techniques used to develop an information system can be used universally.

Areas of Information Needs

The definition of a hospital information system is like a shell which gives form to the thoughts, but the actual information needs of each area within a hospital is what gives life and meaning to the HIS.

The value of information to a hospital's operation cannot be overstressed. The speed and accuracy with which information is handled bears directly on the quality of care a patient might receive. One reason for a hospital's exist-

ence is to allow health care providers to gather information about their patients and, then, to diagnose and provide therapy based on that information[6, p. 4]. It is obvious that inaccurate information or information that is not available on a timely basis can have catastrophic results, not only in terms of human life but also in terms of the financial impact a malpractice claim might have on the hospital and practitioner.

But the value of information is not just internal to the organization. There are many outside agencies that demand accurate and timely information. Some of these external demands for information are needed to provid accreditation of hospitals or departments of hospitals such as the accreditation issued by the Joint Commission on the Accreditation of Hospitals. However, a possibly greater demand for this information is placed on hospitals by governmental agencies, third party payers, and the public[6, p. 6]. These latter groups are very interested in the fiscal performance of the hospital as reflected in reports provided from accurate information and can have much to say about the future well being of a hospital.

While understanding the increasing importance of information, many hospitals have the tendency to overcollect data. This results from not knowing exactly what to collect[8, p. 15]. Considering that between 23 and 39 percent of a hospital's expenses relate to the collection of data

and production of information[6, p. 16], it is not difficult
to see that the overcollection of data can really tax an
already overstretched budget.

Considering the value of information to the hospital,
it is imperative that the information needs throughout the
hospital are precisely determined. An excellent method for
accomplishing this task is to use a book called, Analysis
Manual For Hospital Information Systems, which is designed
(through the use of a multitude of questionnaires) to aid
hospitals in the analysis of information needs in each
department[9, p. 1]. This step will be discussed later as
an integral part of the System Analysis phase of the Systems
Development Life Cycle.

The information needs of departments within the hospi-
tal seem to fall logically into two categories -- informa-
tion needs concerned with patient services and those related
to management services.

Patient Services

Inpatient services

Admissions, discharges, transfers/Census control.
This service receives information from the physician who
will admit the patient, as to the purpose for the admission.
The service will collect demographic and payment information
from the patient and will also receive information regarding
any transfers of the patent from one bed to another, includ-

ing whether the patient is discharged.

Medication distribution. Information from the doctor as to the name, quantity, and the method of administration of drugs is required by this service. Also, the matching of patient information to insure that the correct drug gets to the correct patient is very important. Drug information from the wards is also necessary to issue floor stock. The pharmacy must also have a method of collecting information needed to prepare purchase orders for replenishing their supplies. To prevent the possible administration of drugs whose interactions may not be desired, a patient profile should be maintained showing the drug history for each patient.

Nursing service. This service requires information from many sources. Beginning with admissions, nursing service must know when and whom they are to receive. This may come from another ward if the patient is being transferred, from the emergency room, or from the admitting office. The nurse must gather a history from the patient and also determine the patient's current condition. Information from the doctor in the form of orders must also be recorded. Ongoing information such as additional orders from the physician, test results, patient progress, and vital signs is necessary. When pertinent, surgery must keep the nurse informed as to schedule changes and information pertinent to scheduling of nurses must also be available.

Support services system. This system typically includes Patient Food Services, Linen/Laundry, Patient Transportation, Housekeeping, Social Services, and Patient Information Services. All of these support services require information about the patient pertinent to their service. Most important to several of them is notification of the patient's discharge or transfer which then triggers action in their departments. Most of this information is received from nursing service. Patient information services receives initial information about the patient from the admissions office and updates from nursing service.

Ambulatory care services

Emergency services. The information required here is similar to that needed for admissions and for nursing service. Patient demographic data and payment information must be determined from whatever source is available. Physicians orders and results from tests are also needed. If a patient is not admitted, information needed to determine appropriate charges is gathered and coupled with the patients payment information and is used to prepare a billing.

Referred outpatient services. This situation occurs when a physician determines that treatment is required in the hospital but does not require the patient to remain overnight. This might be for services such as X-ray or physical therapy. The services may be required once or as an ongoing treatment. The clinic involved is usually

informed by the referring physician of the basic patient information and the type of service to be performed. Additional information is gathered from the patient on the scheduled day and includes demographic data as well as billing information.

Clinics. A clinic operates essentially as described in referred outpatient services except that the patient utilizes the clinic instead of a "family physician." In this case there is no outside physician so all patient information and billing information comes entirely from the patient and information regarding treatment of the patient comes from the staff physician in that clinic. Test results from other services are sent to the requesting clinic for file in the patient's record. Charges are determined based on services rendered.

General patient services systems

Diagnostic services system. Typically, information flows into the sections that make up this system; Laboratory Services, Diagnostic and Therapeutic Radiology, Nuclear Medicine, Respiratory Services, and EKG/EEG Services in the form of a requisition from nursing service or perhaps the outpatient clinic as a result of a doctor's orders. Required information often includes the name of the attending physician, patient identification, hospital identification, admitting diagnosis, and the diagnostic service requested.

Rehabilitative services. As in the previous section, the primary source of information is based on a requisition for services from nursing services or the outpatient clinic. Included on the requisition is patient identifying information and the requested services. Due to the ongoing nature of these services, usually more complete information from the requesting physician is obtained.

Surgical services. Here the flow of information begins with the physician ordering surgery. Patient information as well as hospital identification will be provided along with the type of procedure to be performed. Information is also gathered during the time of surgery as to the condition of the patient as well as notes on results of the surgical procedure. Usually prior to surgery the anesthesiologist will gather information from the patient and with other facts gathered from the physician and nursing staff, will determine the type of anesthesia to administer.

Patient records management systems

Transcription. Of primary importance in this department is the actual information that will be transcribed from the dictation equipment. The only other information needed is the type of report being dictated, history, physical, operation report, or discharge summary. This information will determine what type of form is used for the report.

Indexing, storage, and retrieval. As its name implies, this section's main interest is the disposition of

information about the patient gathered throughout the hospital. Usually this section, Medical Records, will receive a patient's record after discharge. Their function at that point is to insure the completeness of the record and to extract information from the record to be used for indexing or for preparing an abstract which might be used for quality control. They also will gather information from the record to prepare statistical reports. Most likely the record will be indexed by patient name, physician, diagnosis, and surgery performed. The only other information received is in the case of a retrieval, in which case information that will match an index key as well as verification of authorization for the retrieval are required.

Quality assurance. A very necessary function is performed through quality assurance. The quality and appropriateness of care provided a patient is evaluated by review of information extracted from the medical record. In addition, if the case warrants, direct testimony from the health care provider may be used. Informational directives from outside agencies such as Professional Standards Review Organizations (PSROs) and the new Professional Review Organizations (PROs) are needed to evaluate the standards of care provided by the hospital.

Management Services

Financial management services

Patient charging, billing, and accounts receivable
systems. Normally co-located in the Business Office, these
functions depend on information provided from the entire
hospital. Initial records are established at admission as
to patient and payment information. Verification is often
done on insurance coverage claimed by a patient. Nursing
services must inform the business office of all services and
supplies used as well as any bed transfers. Any credits for
unused supplies and services must also be provided by the
apporpriate department. Discharge information including the
discharge diagnosis must be provided to initiate billing of
the patient and third parties. Notice of payments must also
be received to adjust the patient's balance correctly.

Budgeting. The budgeting process involves gathering
information from every department in the hospital as to
projected revenues, personnel needs, supplies, and equipment
needs. After the budget is in place, this office monitors
the budget by receiving reports from various departments
such as Personnel or Purchasing as to the current expenses
and compares these against budgeted expenses, producing
periodic reports.

Accounts Payable. When purchases are made, a purchase
order is sent to this department to initiate an account
payable. When items are received, notice is sent here so

the payment process can begin. When payment is actually made, the accounts payable section is notified and the account adjusted.

General accounting. Probably the most interactive department, General Accounting receives data from almost every department in the hospital in order to make appropriate accounting entries. From these data, periodic reports and financial statements are prepared.

Cash management. This section must have information on cash receipts and disbursements, projected cash receipts and disbursements and cash account balances. This is necessary to insure the best use of available cash and to provide for ongoing cash needs.

Personnel management systems

Timekeeping/payroll. Initially data are gathered on each employee which include demographic data, job title, starting salary, and starting date. Periodically, updates of pay rate are made as information is received from supervisors. For hourly workers, information at the end of each pay cycle must be received as to number and types of hours worked (regular or overtime) and any vacation time used. In the preparation of paychecks, deduction information must be known for each employee.

Position control. Executive management informs this section what types of jobs are needed and the number of each

needed to run the hospital. Job descriptions must be deter-
mined for each type of job. Employee information is matched
against each position. Supervisors notify the section when-
ever an employee will be leaving the position or when anoth-
er employee is hired so each position will accurately re-
flect a status.

Evaluation and training. When an employee is first
hired, this section must be notified to provide an orienta-
tion. Specific training requirements must be established
for employees so this department can schedule appropriately.
This section also must be told how often an employee will be
evaluated so they may schedule evaluations to be done by
supervisors.

Materials management systems

Capital equipment. Much work is done prior to a re-
quisition for purchase of capital equipment. This often
long process includes the initial request and justification
followed by several levels of review. Depending on the cost
of the item, additional procedures involved with filing a
Certificate of Need may be required. Lease/purchase deci-
sions and vendor decisions must also be made. However,
after all this has been accomplished, the requisition in-
cluding pertinent equipment information and vendor informa-
tion is received in this department to initiate the pur-
chase. When equipment is received, information as to the
condition of the equipment, the identification information

and other receiving information must be sent to this office
so payment may be initiated.

Purchasing and general stores. Information about sup-
plies needed throughout the hospital are sent to this de-
partment so inventories may be maintained and restocking
procedures followed. It is important to have information on
the projected usage of supplies, normal delivery times for
reorders, and space available to store supplies in order to
effectively manage this section. Requisitions from depart-
ments are used to supply the hospital and purchase orders
are used to buy from vendors. Accurate information about
each line item must be maintained to provide an accounting
of this department.

Central supply. Patient care supplies are stocked
here as well as sterile equipment and supplies. Requests
for these items are made by nursing service and surgery.
Patient information must be received so charges can be made.
This section is responsible for training on new equipment so
they must also be informed as to who will be using the
equipment.

Facilities and equipment management systems

Work orders. Requests for work are received from
departments describing the work needed and when it should be
completed. In order to schedule work this department must
know what supplies and equipment and manpower will be needed

for the job. As a job is started they must also be kept
informed as to the progress and consumption of resources so
appropriate charges can be made.

Scheduled maintenance. When equipment is received at
the hospital the maintenance department is informed about
the receipt and it is determined how often preventive main-
tenance is to be performed. A spare parts inventory must be
determined based on projected need, critical nature of the
equipment, and projected resupply time. When maintenance is
performed workers must inform this shop how much time and
what supplies were used so proper charges can be made and
inventories adjusted.

Management planning and control systems

Internal management reporting. The amount of informa-
tion needed for this function is determined by many factors.
These factors determine what reports are generated. Infor-
mation might be required about personnel, resource utiliza-
tion, accounting transactions, quality assurance, patient
data, case-mix information, or about other hospitals in the
area or across the nation. The important thing to remember
here is that the temptation is to report too much. As in
the previous discussion regarding information versus data,
it must be determined precisely what information is needed
for this function to be performed properly. Currently, one
of the more important areas of concern for hospitals is
management of the product line, the services most frequently

provide to patients. In this case an internal report listing cases by Diagnosis Related Group (DRG) as well as summarized clinical statistics about each group and the revenues and expenses for each group would be helpful. The key -- know what information is needed beforehand so information, not data is produced.

External reporting. The information here will be related to the type of hospital, the location in the country and the particular regulating agencies that are demanding information. A survey of the names of the reports required, the type of agency needing the report, the frequency of the report, and the source of the information within the hospital will aid in efficiently organizing and reporting the information required.

Unique Characteristics of an HIS

Although every organization has some sort of information system, differences arise in the extent to which they have been developed and the purposes to which they are applied. It is generally accepted that "hospital information systems lag well behind their counterparts in the profit oriented sector."[10, p. 95] There are two reasons which can help to explain this as well as point out the uniqueness of hospital information systems.

First, until recently, hospitals have not been too concerned with cost containment. This is because they have

been able to recover costs, especially from third parties,
without much justification of their expenses. The emphasis
has always been quality of care without regard to cost[11].
Without the regard for tight controls on cost, the informa-
tion system was not forced to mature.

Secondly, the medical field is still very much an art.
Because a physician often cannot define the logical steps
leading to his conclusions, it is very difficult to trans-
late that process into a very logical algorithm. Many times
the key to a breakthrough of a medical dilemma is from a
very obscurely related or seemingly totally unrelated piece
of data. For example, when a physician is perplexed about
the cause of a particular ailment, he will leaf through the
patient's record looking for a clue to the solution. This
undefined search is not well suited to automation[18, p.
118]. Therefore, one of the reasons for automation, to be
able to do something the physician cannot, is often elimi-
nated.

The complexity of the hospital information systems and
the heretofore lack of concern for cost containment are two
unique aspects of an HIS that have limited its movement into
more sophisticated information handling systems.

Automated Systems versus Manual Systems

The development of an information system is not depen-
dent on its being a computer based system. Although the
discussion has centered around automated information sys-

tems, many of the concepts and methodologies can be very useful in the development of manual systems.

This is said tongue-in-cheek because all indications point to a greater use of automated HISs in the future. With the advent of new reimbursement policies by the government and other private insurance agencies, hospitals are being forced to keep detailed records which would not have been feasible ten years ago. However, the continuing reduction in the cost of processing information is now forcing hospitals to use computers[11].

History of HISs

Comprehensive information systems are, perhaps, the single most critical factor in dealing with the complex problems facing health care executives in the decade to come. Only institutions that have the human productivity and the technical ability to process data quickly and easily will be able to adapt to the changes of the future.[12]

This indictment is being echoed by many others. Perhaps the panic would not be so widespread had the development of HISs occured much faster. Yet, while looking at the historical development of HISs one must remember there has never been the motivation, until now, to progress more rapidly.

Development of HISs

Prior to the 1960's, computers were not used in hospitals. Most of the activity in the hospital was grouped into departments with much duplication of data gathering and not

much of an attempt to integrate[13, p. 13]. If you had
visited the hospital then, it would not have been unusual to
be asked the same information every time you turned around.
During that decade, as overall use of computers increased,
some hospitals began to utilize computers primarily for
accounting and other typical business applications. At the
same time others saw the potential for applications in
clinical use. Vendors noted the interest in these areas and
began to produce "packages" for the automation of hospital
functions; however, the promises made by the companies often
fell far short of actual performance[14, p. 5]. Due to the
cost of equipment and personnel required to support an in-
house developed system or even a vendor produced system,
some hospitals decided to operate in a shared-system en-
vironment such as Shared Medical Systems, operated by the
McDonnel Douglas Corporation which began in 1969. In this
situation, the vendor maintained the computer and programs,
usually off-site, while the hospital supplied the data to be
processed either by batch or over terminals tied to the main
computer.

The progress in the 1970's saw the proliferation of
minicomputers and continued emphasis on the development of
commercially prepared packages. There appeared to be a
segregation of applications into three areas: (1) financial-
/administrative information systems; (2) patient information
systems; and, (3) departmental information systems. The

failure of vendors initially to produce integrated packages
that would service all functions in the hospital, led to
emphasis of non-integrated, stand-alone systems spread
throughout the hospital. The problem was that there was
still duplication of data collection and an inability to
relate information between systems easily. Shared-systems
continued to expand during this time. An encouraging sign
during the latter part of the 1970's was the growing concern
for developing information systems that would not only be
useful for the management of every day operations but also
for management and planning purposes.

Overall, progress up to this time had centered around
the use of computers primarily for transaction-oriented
tasks such as financial applications. However, as these
applications have grown in sophistication, accuracy, and
speed, the one area that has been lacking is the use of
computers for decision making support systems.

State-of-the-Art

"The challenge of the 1980's is to develop flexible
systems that integrate data from diverse systems and to
utilize these data effectively and in decision-making."[12]
With the passage of the Tax Equity and Fiscal Resposibility
Act (TEFRA) of 1982, the direction of the development of
HISs has changed dramatically. In order for hospitals to
survive, they are seeing the need to be able to integrate

information from all their previously separate systems.
With the advent of the microcomputer and the continued
decrease in cost of minicomputers, many hospitals find them-
selves with a proliferation of independent systems through-
out the hospital, and no plan for integration or compatibil-
ity[28].

This clearly must change if a hospital is going to be
able to gather information that will report its costs for
each DRG. Under DRGs, instead of retrospective reimburse-
ments for whatever their costs might have been, hospitals
are reimbursed on a prospective basis. "For the first time,
hospitals are being forced to collect data and allocate
costs the way other businesses do. Disease groups become
'product lines' and hospitals have to know precisely what
their costs are for each."[11] If hospitals cannot pull
together patient clinical data to determine the DRG, patient
billing data to determine revenues, and institutional cost
data to see what it cost to treat that patient, they may
find themselves being reimbursed for much less than their
costs. If this persists, it isn't difficult to see that a
hospital will fold.

Currently, state-of-the-art HISs are developing around
the need to provide decision-making information. As one
writer put it, "Decision-making is perhaps the most signifi-
cant new challenge facing health care information
systems."[12]

Outlook for the Future

In the future, as today, Canada's health care industry
must deal with capital shortage, more pervasive and ever-
changing government regulations, continuing technological
change, more demanding professional and community needs,
and increasing pressures for cost control. Effective
information systems are imperative to meet these growing
demands."[15]

Although written about Canada's health care industry,

the same goes for the United States. Indeed, "hospitals can

no longer afford the luxury of a laissez-faire, evolutionary

approach to the use of information."[13, p. 41] Although

technology continues in the direction of diagnostic-aids for

health care practitioners, the use of artificial intel-

ligence, the further automation of medical records, and

continued development of paperless claims, it appears that

the most pressing needs are in the area mentioned above,

that of integrated information systems capable of providing

decision making assistance.

Progress in the future will not be without barriers.

The public perceptions have changed concerning the belief

that "all technological advances are good regardless of the

cost." With the escalation of cost has come the backlash

that causes many to question this earlier standard. The

problem is that computers are now lumped in with "all tech-

nological advances" and the very tool that can aid in cut-

ting costs is now questioned as being too expensive. Along

with the development of these integrated HISs has got to

come public education.

In looking at the past development of HISs as well a looking into the future, a glaring need stands out, that of a systematic way to develop hospital information systems. "Hospital administrators must take responsibility for a careful, orderly process of planning to insure that hospital information requirements are satisfied."[13, p. 41] The next section will describe such a systematic approach that will insure those requirements are met.

Description of Methods for Establishing an Automated HIS

The pressure is on. Whether or not a hospital is still around five or ten years from now depends on the hospital's ability to develop an advanced information system[17]. Some of the hospital's problems have been caused partially by an abandonment of the hospital adminis-trator's responsibilities to the technocrats[28]. The result has been a welcomed response by vendors to provide what they think is desired, but as often turns out in reali-ty is not what's needed at all. "We have today a supply push, not a demand pull . . . from vendors, [consequently] hospitals must guard against being steamrolled into a pur-chase that may not be appropriate."[28] As business learned long ago, a systematic approach to development of an infor-mation system along with involvement from upper level man-agement will help insure a successful implementation.

Two Approaches

In the late 1950's and early 1960's the concept of the Systems Development Life Cycle came about[19, p. 103]. The exact definition of the life cycle varies depending on which author is read. Some are five steps and others more. However, when looked at as a whole, the progression is the same, from systems analysis through maintenance of the implemented system, the difference being in the divisions among steps. The importance, though, is not how many steps there are, but the sequential fashion through which the life cycle is traversed. The stepping through the life cycle insured the customer's visibility of the progress and provided decision points along the way before committing to a full-scale development of the project[20, p. 393]. The steps of this life cycle are:

1) Systems Analysis -- identifying the information needs

2) General Systems Design -- a broad design of the system which includes several alternatives

3) Systems Evaluation and Justification -- a look at the impact of the system on the organization and cost/benefit analysis

4) Detail System Design -- a formalization of the design and coding and testing

5) Systems Implementation -- the installation, training, and maintenance of the system.

The way a user would evaluate the progress at each
step was through the delivery of written reports and de-
tailed written specifications. Although much better than
earlier methods, the production of extensive narrative ex-
planations of what the information system was supposed to
do, was also its primary inadequacy[19, p. 103]. The re-
quirement for an analyst or designer to use English text to
describe the technically complex workings of a system not
only produced frustration on the part of the user, but it
also did not lend itself to transferring easily between
analyst and designer. This plus the fact that it forced the
analyst to get too detailed (overlapping into the design
area) created a need for some new methodologies.

In the mid to late 1970's these new methodologies came
on the scene in the form of what was called structured
techniques. The use of these techniques or tools, centered
primarily in the areas of structured analysis and design,
still accomplished the purpose of the Systems Development
Life Cycle. Their use, however, was intended to involve the
user more fully at each step along the way by producing
products that were realistically understandable. The most
noticeable changes were the use of graphic representations
of the system instead of voluminous reports and the use of
coding techniques which allowed the user to see working
models of the system very early in the coding process in-
stead of only at the end. A benefit of these new tools was

the ability to request changes to the system early when the
cost was not nearly so high [21, p. 7], because the user
could actually know what the proposed system was going to
do. And if they knew early what the proposed system would
do, they could be somewhat guaranteed of receiving the
system they wanted, not what the vendor "thought" they
wanted.

Through the use of the Systems Development Life Cycle
and the new structured tools (and to a lesser degree the
earlier traditional methods), a hospital administrator can
now have a systematic way of developing much needed informa-
tion systems. The approach is not difficult to grasp and
does not require a mastery of the tools, only an awareness
of the process; the goal of this thesis.

Project Estimation

Perhaps the best context in which to put this section
is served by quoting Edward Youdon.

> This is the one area about which I have to admit to
> being a complete cynic. I honestly don't know how people
> estimate projects or how they determine when a project
> can be finished. I am aware that there are very complex
> formulas for estimating how long a project will take, and
> how many people will be required to complete it in the
> allotted time*. And I am aware that there is a body of
> knowledge on scheduling manpower for large projects**.

Nevertheless, I remain a cynic. Perhaps this is be-
cause of my experience as a consultant. I have seen too
often that people cannot devise reasonable estimates
because they are working on a programming project of a
type never before experienced.

----

* See, for example, the discussion in George Weinwurm's
On the Managing of Computer Programming (Philadelphia:
Auerbach Publishers, 1970)

** See Philip Metzger's Programming Project Management
(Englewood Cliffs, N.J.: Prentice-Hall, 1975)[22, p. 222]

A quote like this from such a well known consultant

does not give one much confidence in the area of estimating

projects. However, as one looks at the people who are

experts in the field, the most often recommended procedure

for estimating involves keeping track of past projects.

Even then, the figure estimated for cost and duration is

only accurate to plus or minus 20 - 30%[21, p. 173].

One of the real temptations for the manager of a

systems development team is to be pressured by the demands

of the user for quick delivery. This false scheduling makes

it "very difficult to make a vigorous, plausible, and job-

risking defense of an estimate that is derived by no quanti-

tative method, supported by little data, and certified

chiefly by the hunches of the manager."[23, p. 21] Too

often the response to a bad estimate is to add manpower.

But as F. P. Brooks has said, adding more manpower tends to

lengthen, not shorten the schedule[23, p. 19].

Indeed it may seem that there is no sense then trying

to estimate a project; however, this conclusion will not be

tolerated by business which must have figures in order to

make decisions. The implications of this dilemma are two-
fold. For the user, it must be understood that the estimate
is at best a method of comparative bracketing based on the
vendor's best guess tempered by their historical data. In-
deed "such comparative bracketing may be the only method of
estimating the scope of the project, other than 'sticking a
wet finger in the air.'"[21, p. 173]

The onus is on the software developer to be as realis-
tic as possible and up front with the accuracy of his esti-
mates, and not to be pressured by the user into making
unrealistic projections.

Estimating must be done. Even Mr. Yourdon realizes
that. It is only fair to balance his cynicism with his
later advice to "Continue estimating your projects just as
before. If you have a scientific method of scheduling your
projects, fantastic! Keep doing it! If you schedule your
projects according to a combination of your horoscope, the
stock market, and the phase of the moon, keep doing it!"[22,
p. 224]. My only advice to the administrator -- let the
buyer beware.

CHAPTER II

SYSTEMS ANALYSIS

The first step in the Systems Development Life Cycle
is Systems Analysis. Normally, however, there will have
been several events occur prior to the initiation of the
Systems Analysis.

First, there must be some reason an organization has
arrived at this point. Normally it is either the result of
reacting to a pressure or taking an opportunity[21, p. 155].
Perhaps a hospital has decided to take advantage of some new
technology or just decided to do an overhaul on the present
system. These reasons will have different implications on
the development of a system than will those caused by pres-
sures on the hospital. In the previous chapter, it was
apparent that hospitals are under great pressures to utilize
information systems to reduce costs. The pressures that the
hospital feels are no doubt going to carry over into the
Systems Analysis phase in the form of tighter scheduling
demands and a more inhibited flow of information due to the
stresses involved.

Hopefully, the hospital's management realizes the
importance of this phase of the development. If they have,
they will have committed beforehand the resources necessary
to carry out the Systems Analysis. This will include the
necessary committees and other personnel required to make

policy and oversee the day to day operations of the project.
The wise hospital administrator, realizing the time demands
on the people involved in the project, will schedule the
time needed for each individual so there will not be a
conflict with their other daily duties. Of course the size
of the hospital and the complexity of the project antici-
pated will determine the makeup of these committees; how-
ever, it is imperative that the administration be fully
committed. As one hospital put it,

> Willingness on the part of the administration to take
> an active role in the planning was important, because
> hospital leaders not only provide direction needed to
> achieve the desired goals but also demonstrate attitudes
> toward the project that have widespread influence on the
> opinions of others.[24, p. 131]

Another integral part of the project is the complete invol-
vement of the users. Another hospital goes so far as to say
that,

> Because the failure of an HIS can almost always be
> attributed to the noninvolvement of user personnel, the
> steering committee was asked to actively involve hospital
> personnel from all areas of the hospital in all stages of
> the system's planning and implementation.[25, p. 144]

The success of the HIS is contingent on each.

Another step that must be done before the Systems
Analysis is the creation of a master plan for information
systems development. Too often the hospital's response to
pressures is just to react. They have a vendor come in and
tell them what's needed or rely on someone in the hospital
to come up with "the answer". This is somewhat like having
a builder construct a building for you without giving any

specifications[13, p. 42]. It is essential that a long and short range plan be created to provide the guidelines necessary for development of the HIS.

A question that must be decided before the Systems Analysis is, "Who will perform this analysis?" There are several options including using in-house capabilities, hiring an independent consultant, or contacting a vendor. All have their place but a comment on each is in order.

The use of in-house people is acceptable if they have the necessary expertise and the time. Usually a data processing shop is so involved with the daily operations that to take on a project of this proportion would cause significant degredation of the current system or require an investment in more manpower.

The use of an independent consultant is good but he should be somewhat knowledgeable of the complex hospital functions. It is true that knowledge of the tools used in the Systems Development Life Cycle can allow one to go into any setting and produce the desired information system, but the question must be answered, how much time is the hospital willing to take to familiarize the analyst with the intricate workings of the hospital?

The value of a vendor who knows the hospital business must always be balanced with the inherent bias toward his own product. It would be nice to believe a vendor would come in and, realizing his product was not sufficient,

recommend another, but that is highly unlikely.

Lastly, as has been mentioned before, it is necessary for the administration to understand the process that is about to evolve. They must realize that this first stage is designed to ask WHAT things are presently being done and the results of this first stage are not going to be an operational HIS - - yet. They must realize that this is the beginning of a process that is iterative in nature. The development of an idea, the feedback and the redesign of that original idea will continue throughout the life cycle. They need to be prepared for many hours of discussions. The administrator must realize that this whole process may change the way business is done in the hospital. This first phase may reveal bad policies or procedures that are currently in place. An added benefit of a Systems Analysis is getting to know the systems much better than before and having the opportunity to change those policies that aren't working. Too many times an HIS is looked at as the answer to all the problems. The truth is, the automation of bad procedures only produces bad results faster. Administrators must have a realistic view of what an HIS will do and about the process by which it is developed.

The Traditional Approach

As stated by Austin, "Systems analysis is the process of collecting, organizing, and evaluating facts about information system requirements and the environment in which the system will operate."[13, p. 163]

The difference between the traditional approach to Systems Analysis and the approach using structured tools is not in the purpose for conducting the analysis nor is it necessarily in the methods used for collecting or evaluating the facts about the system under analysis. While discussing more fully the Structured Systems Analysis, many of the steps of the process will also apply to the traditional approach. The primary difference is in the way the facts are presented and the extent to which the proposed system is developed.

The primary outcome of the traditional Systems Analysis is a document describing the proposed system which is often hundreds or thousands of pages of "computerese" which the user must interpret to determine if it will meet requirements. Realistically, the user often relies on the integrity of the analyst (not wanting to be considered ignorant) and signs off on the proposal only to be sorely disappointed when the system is implemented. When confronted with this frustration by the user the analyst falls back to his line of defense, "But you signed off on the specification."

The problem can best be seen by an analogy presented

by Gane and Sarson.

> Can you imagine spending five years' salary on a
> custom-built house [or hospital] on the basis of an
> exhaustive narrative description of how the house will be
> built? No pictures, no plans, no visits to a similar
> house - just the 150 page narrative. "The living room,
> which faces south-southeast, will be 27'x16' at its
> greatest width, with the western half taking a trapezoi-
> dal form, the west wall being 13'4" long (abutting the
> northern portion of the east wall of the kitchen). .
> ."[21, p. 4]

The problem now appears quite obvious.

> Yourdon lists five reasons these traditional documents

pose such difficulties for the user.

1) They're monolithic, and must be read from beginning to
   end. A user cannot easily find information about a
   particular part of the proposed system without search-
   ing the entire document.

2) They're redundant, giving the same information in
   numerous locations throughout the document, but with-
   out benefit of cross-reference.

3) They're difficult to modify and difficult to maintain.
   A simple change in the user's requirements may neces-
   sitate changes to several different parts of the func-
   tional specification - and, because the document is
   monolithic, it's exceedingly painful to change. Con-
   sequently, the specification may not be kept current.

4) They're often physical instead of logical, in that
   they describe the users requirements in terms of
   either physical hardware or the kind of physical file
   structure that will be used to implement the system.
   Such information often muddles the discussion about
   what the user wants his system to do by giving details
   about how the system will do things.

5) They are not a useful target for ongoing development
   of the system; indeed, as one of my company's clients
   said, the classical functional specification is "of
   historical significance only." Consequently, the
   system that is designed may differ considerably from
   the system that was specified.[22, pp. 37-38]

Hospitals cannot afford to pay for this service and receive from it a document they cannot completely understand with the consequence being the implementation of an HIS they cannot use. The development of structured tools for Systems Analysis goes a long way in preventing these problems with the traditional approach.

Structured Systems Analysis

Certainly you've heard it said, "A picture is worth a thousand words." If there is any one feature of the Structured Systems Analysis that stands out, this is certainly it.

A significant problem for the systems analyst is bridging the gap between the user and the systems designer. They must be able to define the system the user needs accurately and present it in such a way that both the user and the designer understand. Not only must they do this in an understandable way, but they must do it without becoming "prematurely physical" and limiting the options of the designer. While looking at the tools of Structured Analysis, it should be quite noticeable that the graphical approach (pictures if you will) is going to aid immeasurably in solving this problem.

The Tools of Structured Analysis

This section will look briefly at the tools of Structured Analysis. For a more in-depth description, several books, which are also listed in the Reference section of this paper, could be studied, including: <u>Structured-Systems Analysis: Tools and Techniques</u>, <u>Managing the Structured Techniques</u>, <u>Structured Analysis</u>, and <u>Structured Analysis and System Specification</u>.

Data Flow Diagrams

A Data Flow Diagram (DFD) is a graphic representation of the flow of data through a system, whether the system be manual, automated, or a combination of both.

The purpose for using a DFD is to represent in a logical way, all the facts about the current system and the requirements for the new system. This convenient method for summarizing these facts makes it easier for the analyst and user to communicate about the system.

Characteristically, the DFD will be <u>graphic</u>, using the symbols listed in Figure 2-1 to portray all the component parts of the system and its interfaces. It will be <u>partitioned</u> in that each function or working part of the system will be identified. The DFD will be <u>multidimensional</u>, taking each function and reducing it down to its lowest level, going from functions described in very little detail and being very abstract to functions that are very specific and detailed. <u>Flow of data</u> is emphasized and the <u>flow of con</u>
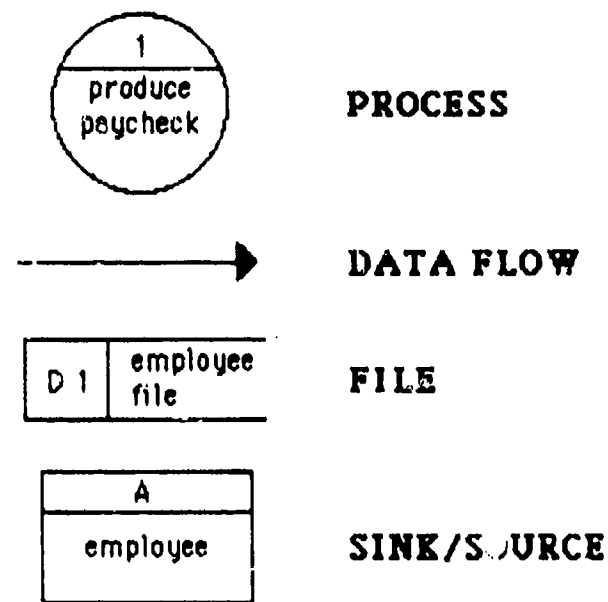
Figure 2-1: Symbols for the Data Flow Diagram (DFD)

trol is de-emphasized[26, p. 48].

The symbols, listed in Figure 2-1, used to construct the DFD are described below. This is only one representation of the symbols; others may be used also. For example, instead of using a rectangle for the sink/source, a triangle might be used. Or, instead of using a circle for the process, a rounded corner square might be used. The importance is placed on consistency of use throughout the DFD and conformation to the definitions below.

1) The process, takes data in, performs some operation to change them, and then sends them out,

2) The data flow is the path on which data travel throughout the system,

3) _files_ are places where data remain temporarily, and

4) _sources_ and _sinks_ are entities outside the system under study which either receive data from the system (sink) or input data to the system (source).

As a hospital administrator, it is not important that all the intricacies of producing a DFD be known; however, there are some conventions used in developing the DFD which, if known, would aid in reducing the confusion encountered upon seeing the first DFD.

First and foremost, the DFD is a _logical_ representation. Hence, it is not going to show control or timing of events. These are not important at this point in the development process.

The DFD is going to be presented in varying levels of detail. From the highest level, least detailed DFD called the Context Diagram which may be only one process, down to the lowest level, most detailed process called a Primitive (see Figure 2-2 for examples of both). Each level should contain about seven processes and be represented on a different page. This "leveling" process involves taking a process at a high level and exploding it into more detailed processes. Each process that cannot be exploded any further (a primitive) will have what is called a mini-spec (mini-specification) written for it. The mini-spec is a logical description of what the function does using decision trees/tables or structured English.
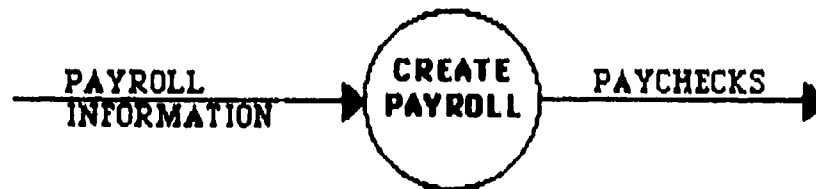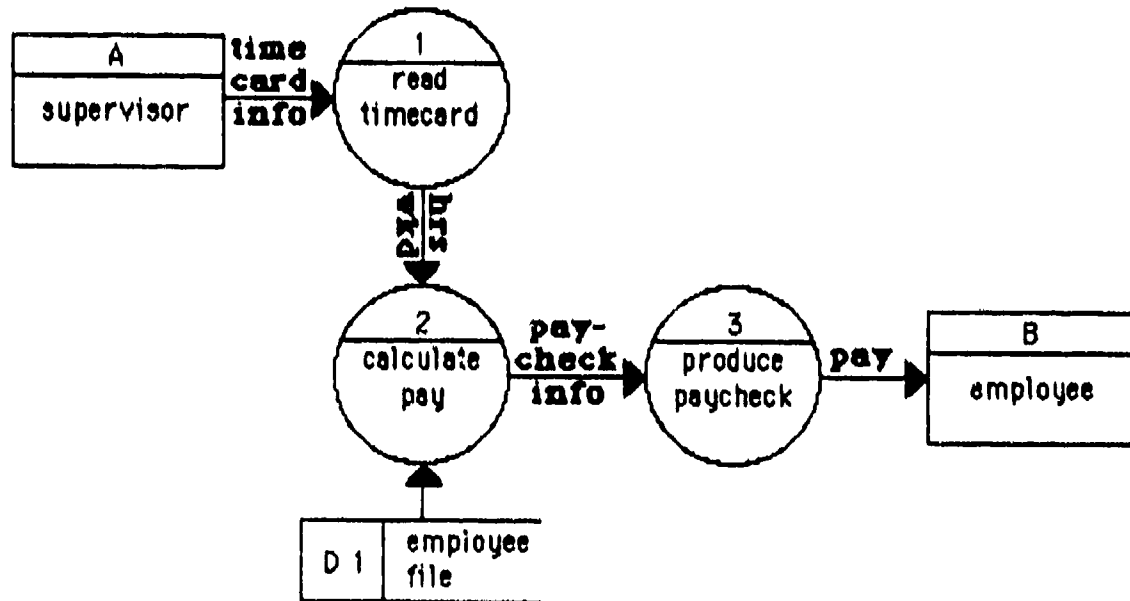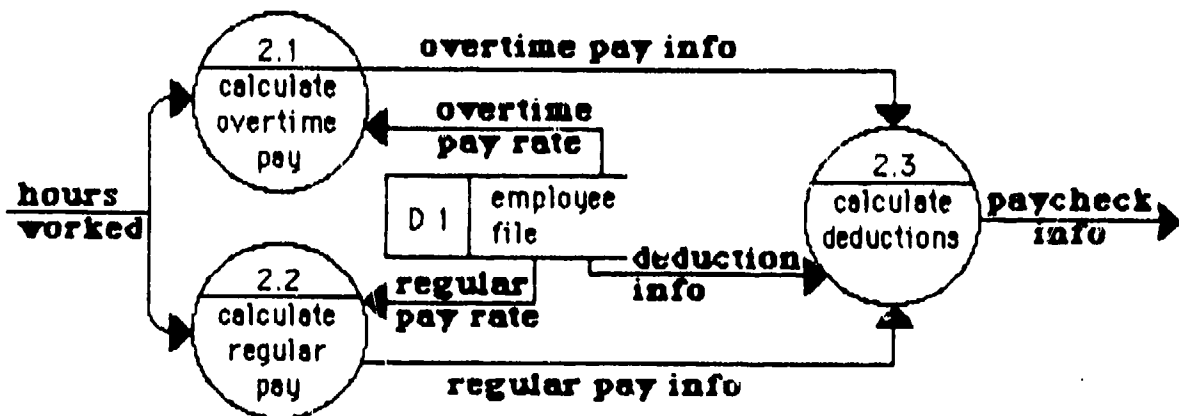
Figure 2-2: Levels of Data
Flow Diagrams

Reading a DFD is similar to reading a map. Usually the flow of data has a beginning point, a source, and flows in the direction of the arrows along data flows, through processes that change the data, into files, and possibly into a sink. Files could be thought of as any temporary storage place for data; a clip board, a physical file, a magnetic tape, a card index, or even a desk drawer.

The marking of symbols is significant also. Data flows should be named to represent all the data that flow on that path. Processes should use an action verb and an object meaningful to the user. Usually processes are numbered, files are labeled with a letter "D" followed by a number, and sources and sinks are identified by a capital letter. In the interest of clarity, it is often better to show the same process, source/sink, or file more than once on a page to prevent extensive crossing of data flow lines. In this case, additional hash marks are used to identify those symbols as being duplicates.

Although the DFD may seem strange at first, the definite advantages to the use of this tool will be seen in the long run. Often the strangeness of the DFD comes from trying to relate it to a flow chart. Even though there are some similarities, the differences end up being the strengths of the DFD. Whereas a flow chart is very physical in its use of certain symbols, the result of the DFD is a picture from the view of the data not the processors of the

data. This allows great flexibility and clarity in defining
the system which the detailed, physical nature of the flow
chart does not.

The flexibility of the DFD lets the analyst look at
possible man-machine boundaries within the system by segre-
gating functions that might be done manually and those that
could be done by a machine. The ability to experiment with
these boudaries allows the analyst to present several op-
tions for implementing the proposed system.

Data Dictionary

It is not difficult to imagine the nightmare of trying
to keep track of all the names of the elements identified
during the construction of the DFD. The solution to this
problem is the Data Dictionary (DD). The DD is a method for
organizing and defining all the data elements used in the
DFD.

Typically the DD is organized in one of two ways.
Either it is divided into alphabetized sections of data
flows, files, functional primitives, and sinks/sources or it
does not differentiate between these sections and lists all
elements together, alphabetically.

The DD can be maintained manually or commercially
developed computerized packages can be obtained. The bene-
fits of the automated system include ease of use and main-
tainability. However, whichever is used, the overall bene-
fits of a DD are significant.

Without the ability to keep track of names of data elements, duplication would creep in causing great confusion. Or, if an existing system is under analysis, the DD permits handling duplication or the same data known by another name (aliases), gracefully. In this case all the aliases of a data element would be listed for each definition, allowing for cross-referencing. The DD also allows you to locate unnecessary data or data whose sources are not defined. Data residing in a file but never leaving could indicate data that are not needed. On the other hand, data which are in a file but are not shown on an incoming data flow could indicate a missing process. Perhaps the most important aspect of the DD is that it is a place to go when you do not understand what is meant by an item on the DFD.

An added benefit of the DD is the manner in which it can be used as documentation for the system after implementation. A DD will aid immeasurably in the maintenance or upgrade of a system.

Structured English

As has been mentioned before, the functional primitives, or lowest level processes, of the DFD are referred to as mini-specs. They are mini-specifications which when combined make up the total functional specification. Each mini-spec will describe the process used to transform the incoming data into the outgoing data. Along with decision

trees and decision tables, structured English is a method used to describe that process.

Structured English uses action verbs, elements defined in the DD, and certain logical constructs (IF-THEN-ELSE; DO-WHILE; CASE) borrowed from structured programming to present the logic used in each mini-spec. It is intended to be readable for the u English uses action verbs, elements defined in the DD, and certain logical constructs (IF-THEN-ELSE; DO-WHILE; CASE) borrowed from structured programming to present the logic used in each mini-spec. It is intended to be readable for the users sake and yet rigorous enough to describe the process accurately.

A more detailed description follows in Chapter IV in the section titled, "Stuctured Programming".

Decision Tables/Trees

As DeMarco says, "Certain kinds of policy [processes] simply cry out to be described using a Decision Table [or Tree]."[26, p. 215] Some may not have ears attuned to the cry, so another guideline might be to use a Decison Table when there are several conditions acting in various combinations to produce differing results.

An example from Gane and Sarson might help clarify the use of the Decision Table. Suppose you had the following description of a policy:

> Customers who place more than $10,000 business per year, and in addition, either have a good payment history or have been with us more than 20 years are to receive

priority treatment.

The fact there are several conditions which can be combined
in different ways to produce different results, might dic-
tate the use of the Decision Table.

The table is constructed by assembling all the conditions
in rows at the top of the table and all the actions in rows
at the bottom (see Figure 2-5[21, p. 83]). All the possible
combinations (rules) are listed in columns at the top. To
determine the number of possible rules, take the product of
the number of possibilities for each condition. In the
example there are three conditions each with only two possi-
bilities, either yes or no. The number of possible rules
would then be 2x2x2 or 8 possible rules. Many of the rules
can often be eliminated because some combinations of condi-
tions are not feasible. However, for each realistic combi-
nation of conditions, the appropriate action(s) is(are)
marked. In the example, the action is marked with an "X" in
the column containing the applicable rule. When more than
one action results from an individual rule, each action, in
the order to be taken, is marked[21, pp. 82-83].

The Decision Tree is nothing more than a "tree" repre-
sentation of a Decision Table. It's use is often dictated
by the user who may be more comfortable with its form. Gane
and Sarson's example is shown in tree from in Figure 2-4[21,
p. 82].

(a) With conditions and actions filled in:

RULES

	1	2	3	4	5	6	7	8
c1: more than $10,000 a yr?								
c2: good payment history?								
c3: with us more than 20 yrs?								
a1: priority treatment								
a2: normal treatment								

CONDITIONS { c1, c2, c3 }
ACTIONS { a1, a2 }

8 rows:
3 conditions each of which has 2 possibilities
$2^3 = 8$ rules

(b) With all the possible combinations of conditions filled in:

RULES

	1	2	3	4	5	6	7	8
c1: more than $10,000 a yr?	Y	Y	Y	Y	N	N	N	N
c2: good payment history?	Y	Y	N	N	Y	Y	N	N
c3: with us more than 20 yrs?	Y	N	Y	N	Y	N	Y	N
a1: priority treatment								
a2: normal treatment								

Y = Yes
N = No

(c) With the combinations of conditions connected to the action:

RULES

	1	2	3	4	5	6	7	8
c1: more than $10,000 a yr?	Y	Y	Y	Y	N	N	N	N
c2: good payment history?	Y	Y	N	N	Y	Y	N	N
c3: with us more than 20 yrs?	Y	N	Y	N	Y	N	Y	N
a1: priority treatment	x	x	x		x	x		
a2: normal treatment				x			x	x

Figure 2-3: Example of a Decision Table
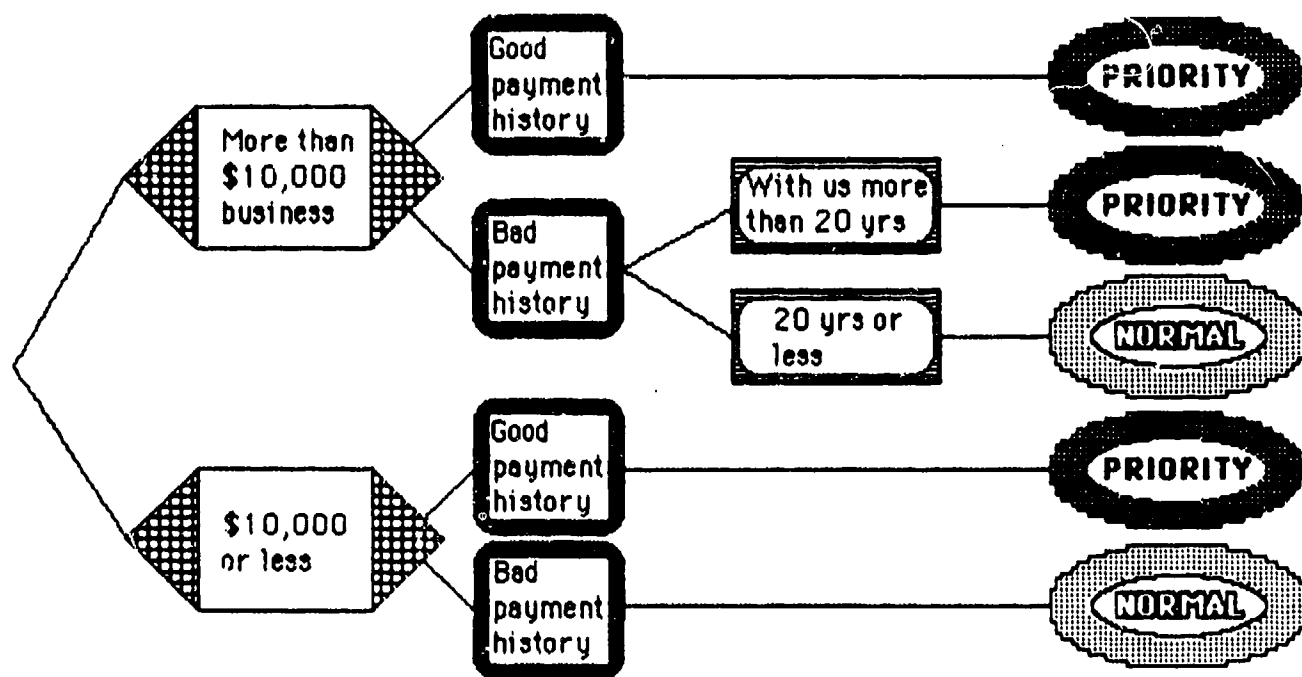
Figure 2-4: Example of a Decision Tree

Data Structure Diagrams

During the construction of the DFD it will become
apparent that groups of data will always be associated.
They will travel together down data flows and they will
reside together in files. This logical association is re-
ferred to as a data structure. For example, a data struc-
ture may be made up of a customer's name, address, phone
number, last order date, and salesman. A group of these
(records) may make up a file of all your customers. Perhaps
you might have another file made up of records whose struc-
ture included salesman, region served, and salary.

In the normal course of business a user will require

access to these files. Access is usually gained by a prede-
fined key, an element(s) that can be used to differentiate
between all the records in the file. In the example above,
customer name might be a key for the first file and salesman
for the second. By using these keys the user will be able
to see all the salesmen who make over a given salary or all
the customers who reside in a certain part of the country.

The Data Structure Diagram (DSD) will show graphically
the logical representation of all the files. Figure 2-5
shows an example DSD, each file represented by a block with
the title of the file at the top and the key(s) listed
immediately under the file name and all other non-key data
elements listed under the key. This graphic display allows
the user to confirm accuracy. By showing each file and the
keys used to gain information from the files, a user will be
able to see where he might combine files to reduce duplica-
tion.

The use of the DSD is also a way to show the user how
files are related. The arrows between files show their
interrelationship via a key access, whether by a predefined
key or by another element of the structure. Through the use
of the DSD you can show the relative importance of access to
a file. If access needs to be immediate, then use of a key
is in order; however, if access can wait for a sort, then
access by a non-key element can be used. This is also
helpful in showing the relative cost of accessing the files.

```
 ┌─────────────────────────┐
 │ CUSTOMER_FILE           │
 ╞═════════════════════════╡
 │   CUST_NAME        ◄─────────────
 ├─────────────────────────┤
 │   Address               │
 ├─────────────────────────┤
 │   Phone_no              │
 ├─────────────────────────┤
 │   Last_order            │
 ├─────────────────────────┤
 │   Salesman          ────────┐
 └─────────────────────────┘   │
                               │
                               ↓
                     ┌─────────────────────────┐
                     │ SALESMAN_FILE           │
                     ╞═════════════════════════╡
                     │   SALESMAN         ◄─────┘
                     ├─────────────────────────┤
                     │   Region                │
                     ├─────────────────────────┤
                     │   Salary                │
                     └─────────────────────────┘
```
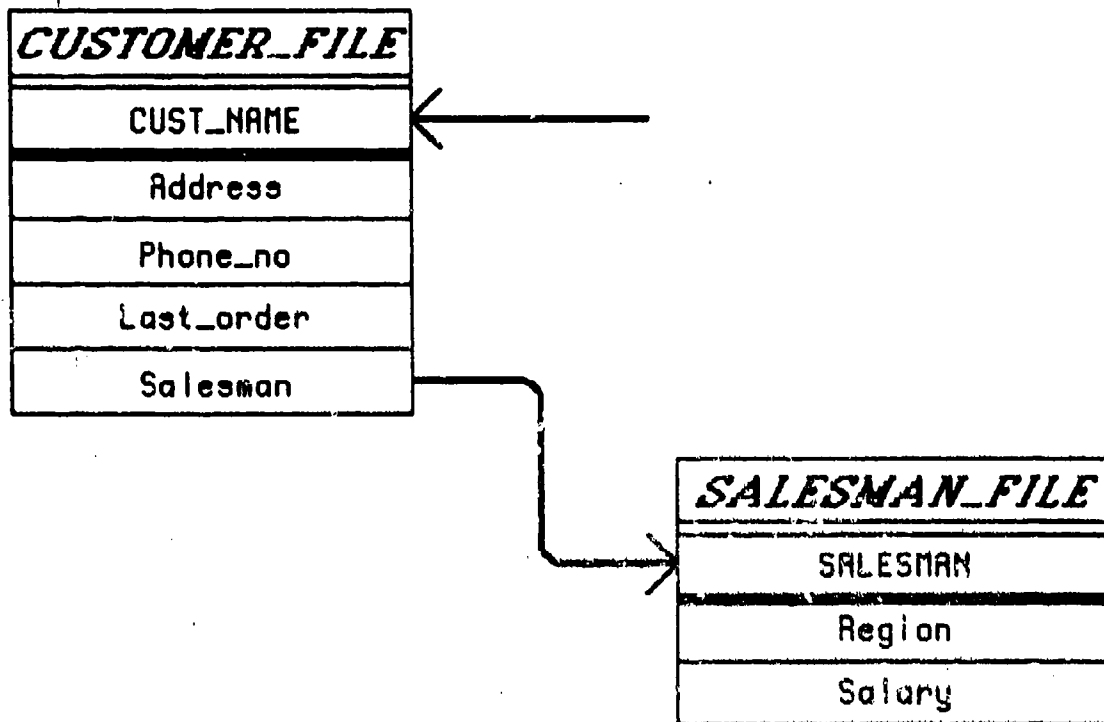
Figure 2-5: Example Data Structure Diagram (DSD)

Generally, immediate accesses to a file will be more costly than those that can wait overnight.

Feasibility Study

Prior to conducting the Systems Analysis, it is common to perform a Feasibility Study. As the name implies, this study is intended to show whether the reason for conducting the Systems Analysis is feasible. Normally five areas of feasibility are addressed:

1) technical feasibility - is there technology available to implement a solution to the problem?

2) economic feasibility - can the business afford the

solution to the problem?

3) legal feasibility - can a solution be implemented that does not conflict with the law or governmental regulations?

4) operational feasibility - can a solution to the problem operate in the company?

5) schedule feasibility - will the solution meet with schedule constraints?[2, pp. 341-342]

These questions must be answered in a feasibility study; however, they are also questions that must continually be answered throughout the development of the system.

In a sense, the feasibility study is a mini-Systems Analysis that determines as inexpensively as possible, the feasibility of developing a system. Consequently, the tools of Structured Analysis described above, would be used during the feasibility study, but only at a very high level with little detail. The process used during a feasibility study is essentially the same as that used during the analysis phase. Based on the data processing plan established for the hospital or an explanation of the problem, the analyst collects information from the users about the present system, trying to formulate at a very high level a possible solution to the problem.

The success of such a study is dependent on many factors but the conflicts created by one deserve mentioning. Naturally, the hospital wants to provide every opportunity

for the analyst or analysts to complete their task. The conflict arises because the hospital also expects business operations to go on as usual. The personnel that will be involved with the system must be available for the analyst and the hospital must expect some loss of productivity. The result of people not being available will be either an untimely completion of the study or inaccurately drawn conclusions by the analyst. The hospital must count the cost before entering into an agreement for systems development.

The result of a feasibility study will be a document generally consisting of three sections; the Project Abstract, a Statement of Goals and Objectives, and Schedule Constraints. This document provides the hospital an opportunity to first ensure the analyst understands what is desired and second decide how to continue on.

Project Abstract

The Project Abstract contains necessary information such as the Project Title, the person responsible for requesting the study, a proposed budget, a description of the method used in conducting the study, and perhaps a list of the sources of information used in the study. Some consideration might be given to including information that supports the study, such as interviews and sources of information, in an appendix available on demand.

Statement of Goals and Objectives

This section of the feasibility study should clearly define the problems or reasons for conducting the analysis and state the goals and objectives of the new system. It is here the hospital will determine if the analyst understands the proposed scope and objectives of the system. This section should contain DFDs showing alternative compositions of the system and a recommendation. Examination of DFDs will show whether the analyst has included too much (exceeded the scope) or not enough (did not meet the objectives of the hospital). If there are problems with the analyst's study the hospital should not hesitate to reevaluate the proposal. Better to do it here than wait till the code is written.

Schedule Constraints

This portion of the feasibility study will show a proposed schedule for the next phase, the Systems Analysis. This might be shown graphically but should reflect any constraints imposed by the hospital. An extended schedule for the complete Systems Development Life Cycle might also be included but the hospital should understand the fact that it is only a rough estimate [see above section, "Project Estimation"].

Deriving a New System

Assuming the project is determined feasible by the hospital, the analyst continues with the Systems Analysis, the first phase of which is documenting the current physical system.

Document the current physical system

Normally, it is advisable to start with this step since the new system will probably resemble the present system and the current system is the only place to start where both the analyst and user can verify an understanding of what is happening.

There are several sources of information which may be used to get a picture of the current system. If the current system is automated, then much information can be gained by looking at its documentation. If the analyst is lucky, and the current system was developed utilizing the structured tools, they will be of great value in analyzing the present system.

"The single most important source of study facts available to the analyst is people."[2, p. 302] This source, which is internal to the hospital, is invaluable in providing information about how the system _really_ works (as opposed to how it's supposed to work) and expressing expectations about what the new system will do for them.

Another source is external to the hospital and includes facts about other similar hospitals with information

systems like the type desired.

Several techniques are available for gathering this information, including interviewing, using questionnaires, observing, and gathering documents used in the current system. Generally the most useful technique is the interview because it leaves less room for misunderstanding of what the user is trying to say. However, some circumstances may warrant the use of a questionnaire. The caution here is to devise a questionnaire that will provide the information desired. Several books are available which have sample questionnaires that might be helpful[2, 9, 13]. Observation can often be useful to verify or clarify previously gathered information. Documents can be very useful in showing precisely what data are used in each situation.

The summary of all this information gathering is the development of a DFD showing the current physical system. "Remember[ing] that you are attempting to build a verifiable model of the current environment,"[26, p. 28] it is not unusual to use very physical names for processes, data flows, files, and sinks or sources. A Data Flow may be called by the name of a form or a process might be the name of the person or machine performing that process. At this stage, the overly physical nature of the DFD is acceptable for the sake of being able to verify its correctness by the user. Characteristics of the logical DFD still remain though, in that this DFD still represents the flow of data

without regard to controls or timing. It is still a picture
of the system from the viewpoint of the flow of data[26, p.
27].

Derive the logical equivalent
of the current system

Once the hospital has verified the correctness of the
physical DFD, the next step is to convert it to a logical
DFD. This involves changing the description of processes
from names of people to the function that is performed. For
example, a Process might have been titled "Mary produces
paychecks". This would need to be converted to the actual
functions that are performed to create the paycheck. An
example for a Data Flow might be converting the physical
title of "Form 212b" to the actual information contained on
Form 212b.

As DeMarco says, we need to "'logicalize' our model of
the current environment"[26, p. 28] to produce the logical
DFD model of the current system. This product is again
checked by the user to determine accuracy.

Define a new logical system

Up to this point the analyst has not concerned himself
with the requirements of the new system as defined in the
feasibility study. The analyst has to make sure the current
system is understood before the changes can be made. This
is emphasized by Youdon when he says,

In most cases, the analyst can expect at least 75
percent overlap between the old system and the new system
- and it's extremely hard to determine where the new
features fit if one doesn't have a good model of the old
system.[22, p. 48]

The process whereby the new logical DFD is derived

will usually involve many iterations of adding, deleting,

and changing processes; conferring with the user; and re-

drawing the DFD. The product of this phase is going to be

the multi-leveled DFDs, the DD, and the DSDs.

Selecting the Right System

Once the DFD for the new system has been produced it

is easy to play the "what if" game and propose many alterna-

tive solutions. The key, though, is the inherent logical

nature of the DFD and the absence of any restricting physi-

cal references. If the analyst has done his job and pres-

ented a DFD that deals only with what the system does with-

out referring to how it is done, then the process of defin-

ing options and selecting the right one is much simplified.

Establishing man-machine boundaries

Basically, establishing these boundaries involves

proposing alternative solutions based on automating all,

none, or part of the proposed system. This can be shown on

the DFD by partitioning it into groups of functions that

might be automated and those that will be done manually.

Although this process involves physical considera-

tions, it does not become overly physical in the sense of

choosing hardware and software, because that does not happen until the design phase has begun. However, it is physical in the sense that this step does provide necessary alternatives for the hospital to decide how automated their system will be.

Perform cost/benefit analysis

For each option defined in the last step, a cost/benefit analysis should be performed by the analyst. Because neither the specific vendors or hardware will be specified, this analysis should be limited to costs for a type of computer (mainframe, mini or micro) that might be used in each option. As many factors as possible need to be considered in deriving a cost or benefit of a particular system, including risk, financial terms, facility modifications, maintenance costs, operating costs, training, personnel, and other set-up costs.

There needs to be an understanding by both parties. The hospital needs to realize that the figures are truely only "ball park" figures, but the analyst needs to realize how important it is to provide the best estimates possible so the hospital can make a decision on the continued direction of the project.

Consider constraints to the system

As much as the analyst tries to develop a totally
logical system, there are usually constraints that will
influence the physical structure of the system. Normally,
these will be specified during the feasibility study, but
they may also be introduced during the evaluation of alter-
natives. The constraints should be included in the Struc-
tured Specification, the document resulting from the Systems
Analysis, for the benefit of the design team but they should
also be considered by the analyst as they develop the alter-
natives and cost/benefit figures.

These constraints could fall into two categories,
physical and organizational.

Physical constraints might include such things as a
user specifying the dimensions or brand of computer that
must be used.

Organizational constraints deal more with hospital
philosophy of management that might dictate whether a cen-
tralized or decentralized system is desired and whether or
not a reduction of personnel is acceptable.

Select an option

This step is important because of what the analyst
doesn't do. Once the previous steps have been accomplished,
it is then the responsibility of the hospital, not the ana-
lyst, to pick an option. Granted, a responsibility of the
analyst is to make a recommendation, but the final decision

for the next step is up to management.

The basis for this decision will be the information presented in the Structured Specification, which is discussed next.

The Structured Specification

Although the Structured Specification (which encompasses all the work performed during the Systems Analysis) entails assembling all the structured deliverables previously mentioned, the presentation of this package should not be taken lightly.

If at all possible, a formal presentation, utilizing many visual aids, should be made in person to the people who have the authority to make the decisions. This face to face meeting limits the possibility of misunderstandings by allowing an opportunity for the hospital representatives to question the analyst about the Structured Specification.

Considering all the factors involved, the hospital will make a decision. Although they may decide to pick one alternative with no modifications, this is unlikely. There are really five alternatives a hospital has in deciding the next step: 1) Stop, 2) Wait, 3) Modify, 4) Conditional Proceed, and 5) Unconditional Proceed[2, pp. 342-343].

The benefits of using Structured Analysis are portrayed again in the Structured Specification. It is a document that is easy for the user to understand and main-

tain. The result for the user is a document that they are
confident depicts the specifications of the new system they
desire. Also, benefits accrue to the designer in that he
has a document that is logical and doesn't limit his design
options. After the transition to the next step in the
Systems Development Life Cycle, Systems Design, the hospital
will see more clearly how use of the structured tools of
Systems Analysis impact favorably in the design process.

CHAPTER III

SYSTEMS DESIGN

In the analysis phase, the primary responsibility of the analyst was to derive a logical representation of an information system that would meet all of the users requirements. The emphasis was in determining "what" the system would do to meet those requirements. Assuming the successful completion of the analysis phase, the next step is to translate that logical picture of the system into a physical design that will provide the answers to "how" the system will meet the information requirements defined during the analysis. Gane and Sarson define design this way,

> . . . the (iterative) process of taking a logical model of a system together with a strongly stated set of objectives for that system and producing the specification of a physical system that will meet those objectives.[21, p. 176]

The output of the Systems Design phase will be documents that are given to the programmer to turn into code which will run on a computer. This documentation includes structure charts, detailed module specifications, input and output definitions, and file designs[30, p. 7]. By necessity, the decision on hardware must also be made during the design phase.

Although a user may end up with a design that fulfills all of the specified requirements, other factors should be considered in deciding whether the design is good or bad. A

fact that surprises many hospitals who have not dealt with automated systems before, is that nearly 90% of all the costs incurred during the life of an automated information system are in the areas of maintenance (debugging in production, changes to fit new hardware/software, and enhancements) and testing and debugging during systems development[21, p. 183]. With this in mind, it is not hard to see that any factors which produce a design that would decrease the time (cost) spent in these areas would be of immense benefit. This is the main reason the concepts of Structured Design were introduced.

Many hospitals decide to take the information found during the Systems Analysis and immediately prepare a Request for Proposal (RFP). This would be the case for a hospital which has decided to acquire a system that is already designed and running. There are advantages and disadvantages to this decision, but, in the context of the discussion here, the hospital must realize there will still need to be maintenance of the system after it is installed. It would be wise for such a hospital to know the methods used to design the system they purchase to determine how easily maintained it will be. Even if maintenance will be provided by the vendor, the easier the system is to maintain, the less time the vendor will spend and the less you will be charged.

With this said, it must be pointed out that Systems

Design will normally be performed for a hospital which is seeking an individualized system of their own, instead of a vendor produced "package". Whether this design is done in-house or contracted out, the hospital should be familiar with the design techniques that produce systems which will be least costly over their lifetime, i.e., are most easily maintained. The two approaches to Systems Design discussed next will have that as their primary distinction.

Two Approaches to Systems Design

Of the concepts used in developing information systems utilizing the structured techniques, Structured Design and Structured Programming have probably been around the longest and are the most widely accepted. These concepts resulted from nearly ten years of study done by Larry L. Constantine and first appeared in print in 1974 in the IBM Systems Journal in an article entitled "Structured Design", co-authored with Wayne P. Stevens and Glen L. Meyers. Even though some of these ideas (primarily documentation conventions) have trickled over into the traditional method for System Design, it is important to contrast the differences.

If it has not already become obvious that this author prefers the use of structured techniques, it will become so in the remaining chapters. But at this point the role of Structured Design in the total context of Systems Design must be explained. Structured Design does not replace all the functions discussed during the section on Traditional

Design. As Stevens explains,

> Structured design is not a comprehensive system design
> technique, since it will not aid in file design, input
> and output lay   choice of access method, operating
> environment, hardware or software, and so forth . . . It
> is done prior to detailed program design, where the
> decisions are made as to how to implement the require-
> ments of the program in code.[30, pp. 6-7]

As can be seen, Structured Design would not be formal-
ly used until the latter stages shown in the section on
Traditional Design. By this time, the General Systems
Design will have been done and many decisions such as which
parts of the system would be automated and which would be
manual would have been made. This is not to say that the
concepts used in Structured Design could not be useful
during the earlier stages of General Systems Design. The
idea of breaking the system down into modules that perform
single specific tasks can be very useful in producing an
early flexible design.

With the context set for Structured Design, let's look
at how it differs from Traditional Design.

Traditional Design

Traditional Design involves the steps of General Sys-
tems Design, Evaluation and Justification, and Detail Sys-
tems Design.

The purpose of General Systems Design is to take the
functional specifications developed during the Systems Anal-
ysis and produce alternative designs that will meet those

specifications while utilizing the present and expected resources. The alternatives must then be evaluated and one picked for Detailed Design. The result of Detailed Design is a document which can be sent to programmers to be turned into code.

One problem has traditionally occurred during this process. Not coincidentally, it is also a problem that has been attempted to be corrected by the techniques of Structured Design. It relates to the thinking that typically accompanied the transition from analysis to design. When the product of the Systems Analysis was a written functional specification's document, the first step a designer had to perform was a translation of the written specifications into a picture of the proposed system. The most well known tool used was the flowchart. However, the problem with using a flowchart is that the designer must think procedurally, step by step through the system. First this procedure must be done then this one . . . The problem with this is that the resultant design is prematurely bound up in the details of the system and the overall design tends to be very inflexible and hard to maintain[26, pp. 303-305]. This is commonly called bottom-up design.

Remembering that one goal for the design is easy maintenance, this method of design should be rejected.

Structured Design

It has been mentioned that Structured Design is a technique that occurs after many of the steps in Traditional Design have already been accomplished. In other words, "Structured Design is specifically a program design technique."[30, p. 6] Most important, though, are the concepts behind Structured Design.

As opposed to the procedural fashion which characterizes the Traditional Design, DeMarco says Structured Design

> . . . should take its shape from the hierarchical view of the application . . . The top level shows the most important division of work; lower levels further subdivide the work allocated to each of their managers. The underlying philosophy of the system appears at the top, and the details at the bottom.[26, p. 305]

The idea is to look at designing a system the way one would an organization, with the boss at the top and various levels of managers in the middle and the workers at the bottom. One would not organize a hospital based on what is done step by step throughout the day, and the same applies for designing a system.

Another concept used in Structured Design is the use, again, of graphic representations of the system. The use of Structure Charts and HIPO (Hierarchy plus Input, Process, and Output) charts will be used to show the hierarchical nature of the system. The virtues of using graphics have been espoused before during the discussion of Systems Analysis and the same applies here. The increased ability for everyone involved, the user, designer, analyst, and program-

mer, to understand what the design of the system looks like and does, cannot help but ensure a system that will end up being just what the user desired.

Another concept that is vital to Structured Design is that of modularity. A system should be "built up from manageably small modules, each of which are as far as possible independent of one another, so that they can be taken out of the system, changed, and put back in without affecting the rest of the system."[21, p. 184] Latter discussions will deal more with modularity and the relationships between modules needed to accomplish the goal of independence.

The goal of Structured Design is to produce a design that is easily maintained, changed, and tested. What does this mean to the hospital? Structured Design will save you money because less time will be spent on the most labor intensive aspects of the system life cycle, maintenance and testing.

Traditional Design

General Systems Design

At this stage of the Systems Development Life Cycle the hospital has decided to continue with the development of their information system. Until now, the hospital has invested relatively little proportionate to the total costs accrued during the entire life cycle. Some of the alternative man-machine boundaries presented during the presenta-

tion of the structured specification have been eliminated
and others have been chosen to be pursued. Decisions wheth-
er to modify the existing system or design a totally new
system should also have been made or will be made during
this stage. With the prospect of much larger expenses for
development and implementation of the system confronting
them, the hospital must now be given more detailed options
to chose from. Whereas, details were not considered rele-
vant during the analysis of what the proposed system would
do, those details must now be included in our design. For
example, this includes determining what kinds of edit rou-
tines need to occur and what should happen with rejected
inputs; timing and control should be considered now as well.
Whereas, the cost/benefit estimates developed during analy-
sis were very much only "ball-park" figures, these must now
be refined. Although this design will not be as detailed as
the one produced during the later stage of Detailed Design,
it must be more detailed than during the analysis phase.

Many ideas introduced during analysis will continue
during the design phase. These include the absolute neces-
sity for participation by the administration and the users
and also the continued questioning about the feasibility of
the project. There will be predetermined meeting times
where the decision to continue will be raised, but if it
becomes obvious before those times that the project is no
longer feasible, then the hospital should have the foresight

to stop, regardless of the money already spent, and pursue another plan.

Specifically, during this early stage of design, the following events will occur: refine the system's goals, develop a conceptual model of the system, apply organizational constraints to the design, define data processing activities, and prepare a Systems Design Proposal Report.

Define the system goals

The system goals will be gleaned from several sources. The functional specifications which define requirements, any constraints identified during the analysis phase, and the hospital's short and long range plans for the information system will all provide input to defining the system goals.

A distinction should be drawn between the hospital's organizational goals, system goals, and system requirements. Normally, the organizational goals that apply to the information system being developed will fall into one or more of the following three areas: 1) Increase Revenue; 2) Avoid Costs; or, 3) Improve Service (IRACIS)[21, p. 156]. For one or more of these three reasons (objectives), a system is being developed. The system goals should be defined to state how the system will help achieve the organizational goals. It is important to realize, though, that just because the designed system achieves its goals does not mean the organizational goals are automatically fulfilled. For

example, if an organizational goal was to avoid costs in the warehouse by implementing a system goal of having more up-to-date information on the inventory, the fact there is more up-to-date information does not necessarily insure that a warehouseman will utilize it to keep the inventory as low as possible. Although there may be cases where the new system will automatically fulfill organizational goals, usually the system will only make it possible to satisfy those goals[21, pp. 162-163].

In the same way, user requirements don't directly translate into system goals. The contrast can be seen in our inventory example by realizing that our broad system goal, to provide more up-to-date information on the inventory, could be fulfilled by many different user requirements. The user may require a daily written report when the system is first installed but later require an on-line query capability to get even more up-to-date information. The broader system goals will not likely change over the life of the system but the user requirements needed to fulfill those system goals might indeed change[4, pp. 375-376].

As important as knowing the context within which system goals are placed is knowing how to write them. A goal that is obscure or ambiguous will pose a great problem for both the designer and the hospital. For, after the system is designed, the conflict will invariably arise where the designer feels the system meets the goals and the hospital

does not. Each goal should be reviewed thoroughly to determine if these defects occur.

Develop a conceptual model

Whether or not all the Structured Design techniques are used, most software development companies have seen the value of instituting some of them. One of these is the use of top-down-design. Yourdon defines top-down-design as

a design strategy that breaks large, complex problems into smaller, less complex problems - and then decomposes each of those smaller problems into even smaller problems, until the original problem has been expressed as some combination of many small, solvable problems.[22, p. 59]

Another technique used is the HIPO chart which documents the inherent hierarchical and conceptual nature of a system. This is used instead of a system flowchart which is procedural in nature and prematurely emphasizes the details of the system. When designing a conceptual model, the procedural fashion of a system flowchart i 'nadequate while the HIPO chart accomplishes the task very well. Although the use of top-down-design and HIPO charts don't comprise the substance of Structured Design, they are compatible with the more esoteric concepts relating to module relationships.

Top-down-design and the use of HIPO charts are linked by the fact that top-down-design implies breaking down the system into manageable parts or functions while HIPO charts are a way to display those functions graphically. When Data Flow Diagrams are used during Systems Analysis, it is a

simple matter to turn them into HIPO charts. This process will be discussed later in the chapter.

The HIPO chart is very similar to the Structure Chart, which will also be discussed later in this chapter; however, the difference is that HIPO charts do not show how the modules are interfaced or what information is passed between the modules. HIPO charts are also used to show the inner workings of each module using the same format of hierarchy plus input, process, and output. An example of an overall HIPO diagram documenting an inventory control application is shown in Figure 3-1[2, p. 380] and the use of the HIPO format describing one module (module number 2.0, "Update Inventory Master") is shown in Figure 3-2[2, p. 381]. Notice the hierarchical (bosshood) nature of the modules with the main function "Maintain inventory control" being very general and successively lower levels being more detailed, with the bottom level of modules performing the work. As can be seen in Figure 3-2, the function of the intermediate levels is to handle timing and control and make the decisions about when to call the lower-level modules. The flow of the overall HIPO chart also follows the characteristic that the input functions appear on the left, the process functions are in the middle, and the output functions are on the right.
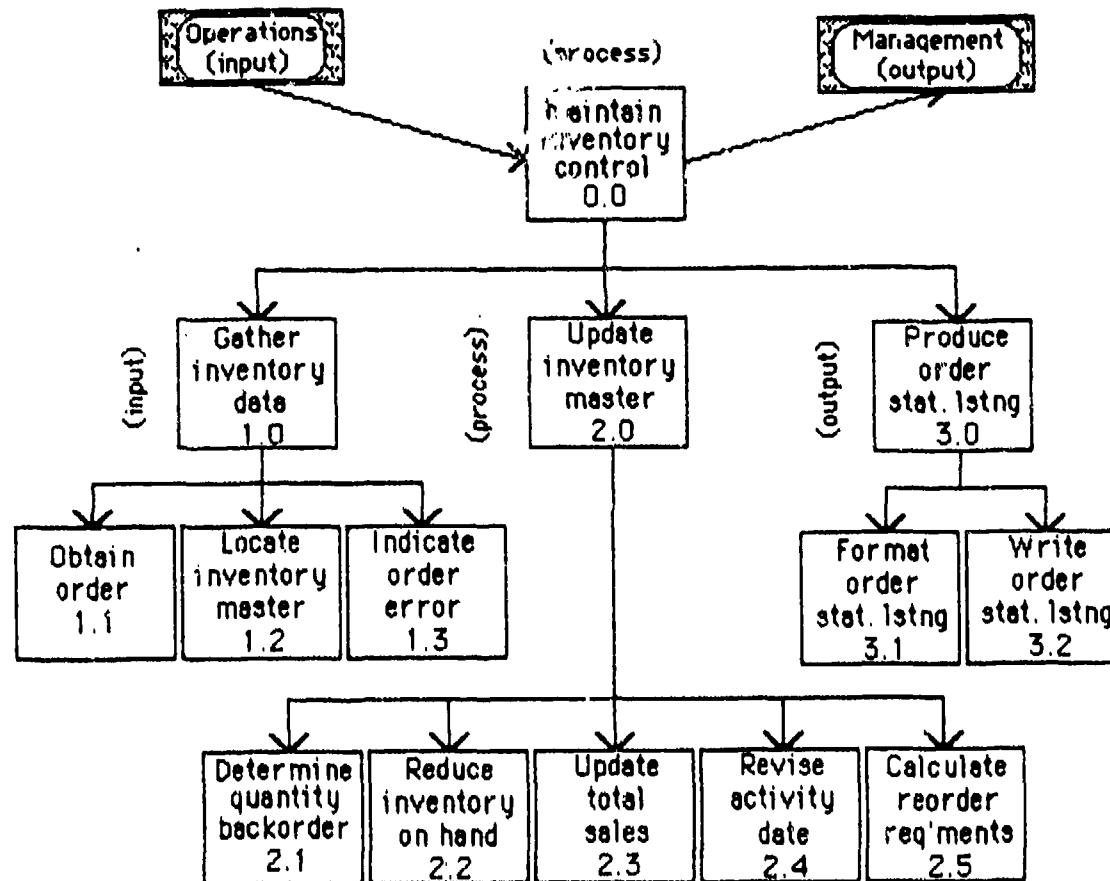
Figure 3-1: Example of a HIPO Chart

Apply organizational constraints

The dream of every designer might be to design a
system in a vacuum where he would not have to be concerned
with the realities of organizational constraints (often
called "limiting the creative potential"). To the contrary,
the true creative genius of a designer comes when they are
required to produce the best design they can, which fulfills
the requirements of the user, while utilizing only the
resources available. By resources we mean people, money,

FROM: Maintain-Inventory-Control (0.0)

Input	Process	Output
ON-HAND QNTY-REQD ON-HAND ON-ORD RECORD-LVL	1. If ON-HAND minus QNTY-REQD is less than zero then a. QNTY-AVAIL = ON-HAND b. Perform "Determine-Quantity-Back- Order (2.1)" else c. QNTY-AVAIL = QNTY-REQD 2. Perform "Reduce-Inventory-On-Hand (2.2)" 3. Perform "Update-Total-Sales (2.3)" 4. Perform "Revise-Activity-Date (2.4)" 5. If ON-HAND plus ON-ORD is less than REORD-LYL then a. Perform "Calculate-Reorder-Require- ments (2.5)"	QNTY-AVAIL

Box No. 2.0

Diagram Title: *Update Inventory Master*

TO: Determine-Quantity-BackOrder (2.1)
     Reduce-Inventory-On-Hand (2.2)
     Update-Total-Sales (2.3)
     Revise-Activity-Date (2.4)
     Calculate-Reorder-Requirements (2.5)

Figure 3-2: Module Description Using HIPO Concept

equipment, materials, methods, and others. Often an organization must make trade-offs which result when two constraints conflict. For example, a hospital may require an extremely reliable system, but is not willing to make the money available to obtain that reliability. As a consequence, the designer and organization must weigh constraints such as reliability, cost, installation schedule, maintainability, flexibility, growth potential, life expectancy, and others, to come up with the proper design[2, pp. 377-378]. It is the meshing of the users requirements with the constrained resources that produces the optimum system.

Define data processing activities

A goal of the designer is to produce several alterna-
tive design solutions from which the hospital will pick one
to continue with into the Detailed Design. As a rule the
designer should try to provide at least three alternatives:

1) A low cost solution that does the job and nothing
   more.

2) An intermediate cost solution that does the job well,
   and is convenient for the user . . .

3) A high cost "Cadillac" system, with everything the
   user could possibly want.[31, p. 12]

An important factor in establishing these alternatives
is the extent to which each is automated. During Systems
Analysis, the analyst segregated different functions or
processes in the Data Flow Diagram (DFD) into man-machine
boundaries. The same process occurs here except that now
the hospital has a better idea of the work that must go into
each part of the system. Many considerations must be made
to come up with these boundaries but the task is much easier
when proper time is spent on the previous step, applying
organizational constraints. To a large degree, the con-
straints on the system will dictate which functions will be
automated.

Gane and Sarson give good examples of generic alterna-
tives that might be considered for design.

1) A batch system where work can be accumulated man-
ually and then presented for entry into the system at one

time.

2) A <u>source data entry system</u> with <u>overnight update</u> where data is input continuously during the day but the computer stores it until the processing is done overnight.

3) An <u>on-line data entry system</u> with <u>immediate update</u> and and <u>on-line inquiry</u> feature. In this alternative system, entries during the day would be immediately processed and users of the system could inquire as to a status at any time without having to wait until a report is generated later.

4) A <u>distributed system</u> where sections of the hospital would have there own limited data processing capabilities but would transfer transactions or updates to a centralized or host computer with results possibly being returned to the section. In previous discussions this configuration has been called an integrated system.

5) A system with <u>dedicated computers</u> where each section would have their own data processing capabilities but would not have the capability to directly pass information between computers. Another term for this is a stand-alone or non-integrated system.

6) An improved <u>manual system</u> without automation. Consideration should always be given to this potential solution.[21, pp. 165-166]

Prepare Systems Design Proposal Report

Another formal opportunity is now provided the hospital to decide how to continue with the project. The designer will provide documentation showing HIPO charts for each alternative and again estimates of costs for each. A restatement of system goals and user requirements should be made with indications of how each alternative will satisfy them. It is also important to show how each alternative will impact the resources of the hospital.

Lastly, the designer must make an effort to clearly identify each assumption made during the design. This critical point, if not developed fully, can lead to many misunderstandings and possible legal ramifications. These points must be brought in the open and clarified or a plan developed to deal with each assumption.

After this package is presented to the hospital a decision must be made on how to continue. Normally, some of the alternatives will be rejected and some retained for further development. A Request for Proposal (RFP) will be prepared for each alternative chosen for continuation. This will allow more detailed analysis on the hardware that might be chosen and the costs and benefits for each alternative.

One option for the hospital that grows harder to choose as each step in the System Development Life Cycle passes, is the option to terminate the project. Even if there is no feasible solution, a hospital will rationalize

continuation based on the money already spent. As appealing
as this logic may appear, it could be a death knell to the
hospital that forgets that about 80% of the costs associated
with the system can be attributed to maintenance. If an
infeasible solution is accepted, the costs involved with
manipulating the system to meet the original requirements
are going to be great. The way to proceed when there are
no feasible solutions is to ignore the sunk costs and stop
the project.

Evaluation and Justification

Assuming a decision is made to continue with certain
alternatives, the designer must more specifically ascertain
the costs and benefits associated with each alternative.
The hardware is one part of the system for which the de-
signer must gather price information in order to do the
cost/benefit analysis. If the designer's company is not
going to produce the software for the system, then proposals
for that work must also be requested. Once all costs have
been determined, the analysis can be done and a Final
General Systems Design Report can be prepared from which one
design will be chosen for Detailed Design and coding.

Request for Proposals (RFP)

"The RFP provides you with an opportunity to gain, in
a systematic and comprehensive manner, vendor information
needed to make sound purchasing decisions."[32, p. 18] As

necessary as the proposal from a vendor is , it will be only as good as the RFP. If the RFP is ambiguous and unclear then the resultant proposal will obviously not provide the accurate information necessary to make a sound decision. If, however, the steps taken during analysis and design were done correctly and completely, the information provided to the vendor should be precise.

A step that is sometimes taken just prior to sending out RFPs is that of making a Request for Information (RFI). This is done primarily when a hospital uses in-house capabilities and is not familiar with the availability of services in the marketplace. It is essentially a preliminary screening to determine which companies should be sent an RFP. The RFI usually is not as detailed as the RFP and may contain only summaries of system goals and user requirements[32, p. 17].

Several alternatives are open to the hospital when sending out an RFP. The hospital may decide to request proposals for a specific configuration. In this case the decision has already been made as to what type of equipment is desired and how it will be utilized. The equipment supplier is asked to provide only costs and other information related to that configuration.

If a hospital has not decided on the specific configuration but has determined specific performance requirements based on the system's requirements, it may request proposals

for only those performance requirements. Bas? . .y ‘ ⸱
hospital is saying, "This is what we want the ⸱ ⸱ ⸱ ⸱ t⸱ ⸱,
what equipment do you (the vendor) propose will sa' ⸱ ⸱ ⸱
these requirements?"

Finally, the option exists for the hospital to request
proposals from only one vendor. Obviously, the impact of
competitive bidding is lost in this situation, but there may
be times when its use is desired. In the case where a brand
of hardware is in use at the hospital and the proposed
system must interface with it, a proposal from one vendor
may be called for. Political influences may also lead to
this approach, especially when the CEO of the requesting
hospital is also a stockholder or board member in a hardware
firm.

Whichever approach is taken, certain information
should be contained on the RFP as well as requested from the
vendor. The RFP will consist primarily of documents already
developed during the previous stages of the project. This
points up another value of the use of structured tools. The
ability to supply the vendor with graphic representations of
the current system as well as the proposed system, goes a
long way toward preventing misunderstandings. As well as
showing the two items (the current and proposed systems) the
hospital should supply as much information about the desired
performance requirements as possible. These are important
because only if the requirements are measurable will there

be any way to determine whether a vendor's equipment is actually meeting the desired system design goals. The RFP should also ask the vendor to detail all other peripheral costs including delivery, set-up, equipment maintenance, financing, and training costs. Another item the hospital might include is a brief profile of itself. This might help the vendor by allowing them to compare the proposed system with systems they've already installed or with installed systems of other vendors.

In addition, the RFP should request the vendor to provide certain information. Costs of all kinds are obvious, but other items such as maintenance agreements, warranties, training requirements, data conversion requirements, financing plans, and a vendor profile are all necessary for the hospital to consider. Another important item to request is information relating to other similar installations the vendor has done. This will enable the hospital to verify some of the claims made by the vendor.

Finally, to aid in evaluating the proposals, the hospital should specify a format for the returned proposals. This will insure comparisons are done fairly and between like items. In other words, the evaluators want to compare apples with apples and telling the vendors how to format their proposals will insure this happens and speed the evaluation process. Note: An excellent guide for preparing an RFP, called <u>Hospital Computer Systems Planning</u>, has been

written by the American Hospital Association. It is included in the References listed at the end of the paper.

Evaluation of proposals

Evaluation of proposals may require several iterations because it is not uncommon for the evaluation team to need clarification on items presented by the vendor. Also, if information presented by the vendor is not comparable with that provided by the other vendors, it must be converted.

Once all the proposals are back and ready for evaluation, the process of elimination begins. Some vendors can be eliminated immediately because they cannot meet the essential system requirements of any alternatives. For the contending vendors, several methods can be employed to rate them and narrow the field.

One method involves creating a matrix with all the system requirements listed on the left side and the vendors listed across the top. How each vendor satisfies the requirements can then be viewed easily and an initial group can be selected.

When the field has been narrowed to a few vendors, the hospital may then want to verify the claims made by the vendors by calling the hospitals listed in the vendor's proposal. Another way to verify equipment claims is to do testing. Two common tests include benchmark testing where the hospital uses a program written to model the anticipated workload of the system and simulation testing. The criter-

ion for benchmark testing is the execution time of the equipment. Because the program will model only the projected workload, the validity of the results are only as good as the test program. Simulation tests use mathematical models to predict the way the equipment will operate based on certain parameters such as file sizes and structures, numbers of transactions, and file accesses.

The final evaluation sometimes requires the hospital to pick one vendor out of several that meet all the system requirements to some degree. At this point the hospital can give a priority to system requirements and assign a weighting factor to each. Then the hospital can rate how well each vendor meets the requirement and multiply the rating by the weighting factor to get a weighted value for how well each vendor satisfies each requirement. When this is done for each vendor, summing all the weighted values will give an overall value that ranks the vendors by how well they meet the most important (most highly weighted) requirements of the system.

Most hospitals have statistical methods similar to these for ranking vendors; however, a hospital usually includes some subjective rating also. The value of this should not be overemphasized nor discounted totally. There is something to be said for "gut feelings" by executives who have been in the business for many years.

Whatever methods are used, the hospital must now

choose a vendor and proceed to the next step of considering acquisition methods.

Acquisition considerations

Basically there are four methods for financing the acquisition of computer equipment: 1) rent from the vendor; 2) purchase from the vendor; 3) lease from a third party; or, 4) a combination of these[2, p. 414]. Many factors must be considered in deciding which method to use and is an area where accountants and lawyers must assess each possibility and decide which is best for the hospital. Figure 3-3 is a chart listing the relative advantages and disadvantages of each[2, p. 417].

Cost/Effectiveness analysis

The purpose of this analysis is to determine "if the proposed system produces benefits which outweigh costs."[2, p. 418] The procedure is simple; determine the life of the equipment and tally all the annual costs incurred by implementing the system, and do the same with the cost savings or increased revenues and see which is greater over the useful life of the system. If there is a net savings, the next step is to determine whether the internal rate of return on the investment is acceptable. If there is a net loss, the system should not be considered favorably. As simple as the procedure sounds on paper, the determination of specific

METHODS	ADVANTAGES	DISADVANTAGES
RENT	1. Helpful to user who is uncertain as to proper equipment application. 2. Normally psychologically more acceptable to management. 3. High flexibility. 4. If an organization does not have past experience with computers, this may be the safest method. 5. Maintenance charges included in rental payments. 6. Allows a favorable working relationship with the vendor. 7. No long-term commitment. 8. Avoids technological obsolescence.	1. Over approximately five years, this is the most expensive method. 2. Rental payments increase by some factor less than one if usage exceeds a specified number of hours per month, assuming prime shift contract.
PURCHASE	1. The more mature users no longer need to depend on the security of renting. 2. Stabilization of computer industry means that changes in technology are not as disruptive as they once were. 3. Lower costs for an organization with a fairly stable growth pattern that will keep the equipment relatively longer than a growth company (i.e., not subject to operational obsolescence.) 4. Investment credit offers certain tax advantages. 5. All other advantages accruing to ownership.	1. Organization has all the responsibilities and risk of ownership. 2. Usually if equipment is purchased separate arrangements must be made for maintenance. 3. In a growth company there is a high probability of being locked into a computer configuration that fails to meet the changing requirements of the system. 4. Must pay taxes and insurance on equipment. 5. If the organization has better alternative investment opportunities, it would be more profitable for it to use the funds for these alternatives. 6. Ties up capital, thereby impinging upon cash flow. 7. Increase risk of technological obsolescence. 8. Low resale value.

Figure 3-3: Advantages and Disadvantages of Acquisition Methods

METHODS	ADVANTAGES	DISADVANTAGES
LEASE	1. In the long run, can save 10-20% over rental method. 2. Tax benefits. 3. Conser, ation of working capital because of low monthly payments. 4. Allows users to select their equipment, have it purchased, and then have it leased to them.	'. Lessee is obligated to pay a contracted charge if lease is terminated before end of lease period. 2. Little support and consulting service. 3. Lessee loses a great deal of negotiating leverage. 4. For maintenance, the lessee must depend upon a service contract from the vendor, not from the leasing co.
COMBINATION	1. Optimizes the best of other methods. 2. Flexible.	1. More recordkeeping. 2. Might have to deal with several vendors in case of breakdown.

Figure 3-3 (con't): Advantages and Disadvantages of Acquisition Methods

costs and benefits is very difficult.

Until 1975, there are not many supporting data to evaluate the cost/effectiveness of a large HIS. In 1975, though, a comprehensive study was done of the HIS implemented at El Camino Hospital in Mountain View, California. The results of the study showed a $3 to $5 per patient day savings from the system. The study also showed that 95% of the savings were labor related but were attained only when management enforced personnel changes that had been predicted earlier[1, p. 231]. In other words, an attempt was made to realize the potential benefits of the system instead of allowing procedures to go on as they had before. Other intangible benefits included reduced errors, improved time-

liness, and enhanced availability of medical information[6, p. 20].

From this and other studies it is seen that benefits can be clas. ʌᵗed as either tangible or intangible. Tangible benefits are those that might be associated with the elimination of a position or a process which results in a quantifiable benefit. More difficult to measure are the intangible benefits which are usually qualitative in nature. It is hard to assign a value to the enhanced availability of medical records, but an effort should be made to estimate where possible. A technique that can be used when someone hedges at an estimation for fear of being held liable, is to estimate three different values and multiply each by the likelihood (odds or probability) of its occurring. Summing the products gives an estimate that is more realistic.

Although it is less difficult to estimate costs, the important aspect of this part of the analysis is to make sure all the costs associated with the implementation of the system are included. The hospital should be sure to include:

1) acquisition costs which are the actual costs of the equipment;

2) environmental costs which include such things as power requirements, air conditioning, furniture and fixtures, and other room modifications;

3) physical installation which includes costs asso-

ciated with the actual setting up of the equipment;

4) training costs for both users and operators;

5) additional project development costs which include
software development;

6) conversion costs incurred when changing from the
present system to the new one; and,

7) operations costs such as staff costs, supplies,
quipment maintenance, systems maintenance, power and light,
and insurance[2, pp. 422-424].

Once the annual costs and benefits have been deter-
mined and if the benefits are greater than the costs, a net
annual savings should be calculated. The net present value
of the flow of annual cost savings should be determined and
compared against the costs required to continue the project
(the investment). If the net present value of the cost
savings is greater than the investment, the project is
favorable and should be continued.

Now that a design alternative has been chosen and the
method of acquisition has been determined and appropriate
cost/benefit analysis has been done, the Final General Sys-
tems Design Report can be prepared. Also included in this
report will be a detailed implementation plan which indi-
cates the schedule of events for areas such as equipment
acquisition, software development and testing, training,
set-up of the system and any conversion activities, plus the
remaining steps in the System Development Life Cycle. Again

the hospital will decide whether the project is still feasible and how to continue. If the hospital decides to continue, the chosen Systems Design will then be further refined during Detailed Systems Design.

Detailed System Design

By this time the hospital and designer have established a design for the proposed information system. The decisions have been made as to which parts of the system will be accomplished manually and which will need to have programs written so they can be automated. A general idea exists about which hardware will be used. The purpose of Detailed Systems Design is to move from the conceptual ideas to detailed plans that a programmer can use to generate programs and detailed equipment specifications that the purchasing department can order from. In general, it has been decided how the system will operate. Now the question must be answered, "How specifically will the system operate?"

To answer this question, details concerning the HIPO charts or program specifications must be refined to the level necessary to develop the software and controls, forms and reports designs, and procedures manuals.

Controls

If a hospital considers information important enough to invest millions of dollars to develop a sophisticated information system, that hospital would also want proper controls placed in the system to insure the veracity and integrity of the information and thus protect its investment.

Controls can be included in many places throughout a system. The extent to which this is done depends on the amount of money a hospital wishes to spend. Like an insurance policy, an organization must establish the value of the information and insure it accordingly.

Generally controls can fall into the following categories:

1) External Controls might include auditors or evaluators from outside the hospital that would evaluate the system.

2) Administrative Controls are those policy and management controls that dictate the overall operation of the system both for normal and contingency operations.

3) Input Controls are used to insure that garbage does not get into the system so garbage will not come out of the system (GIGO). Some tools that control input are transaction codes which provide a check that proper types of transactions are being done, forms designs which can provide an easy to read document for data entry, verification of the

accuracy of source documents by another individual, and control totals. Control totals are especially useful when blocks of transactions are processed several times in different locations. The totals are calculated by summing a code used for each transaction in the block to come up with a total that can then be checked each time the block is processed to insure a transaction was not dropped or an extra one added.

4) _Programming Controls_ can be used to check that a value is within prescribed limits, or arithmetic checks might be done in a case where values can be correlated. Error logs can also be used as a programming control to decide if processing should continue or when errors are occurring too frequently. Another necessary control is a transaction log that provides an audit trail.

5) _Data Base Controls_ are necessary to prevent unauthorized changes to the data base and to insure that proper back-up procedures are carried out if a data base must be recovered.

6) _Output Controls_ are a final check on the accuracy of the information and include visual inspections or screenings. Strict controls should be placed on especially valuable outputs like paychecks or classified information.

7) _Documentation Control_ refers to the need for documentation such as a general systems manual, a user's manual, and technical manuals.

8) <u>Hardware</u> <u>Controls</u> are built in checks which the user normally does not notice, included to catch possible errors caused by the hardware itself. This might include sophisticated algorithms to detect transmission errors or vendor software controls which are related to programming controls but refers specifically to routines an operating system might use to verify the correctness of its operations.

9) <u>Computer</u> <u>Operations</u> <u>Controls</u> include physical controls related to the actual environment of the equipment and procedural controls involving the operations of the equipment.

10) The last area of controls, <u>Security</u> <u>Controls</u>, is becoming more important all the time. There is much that can be said about this area but it is outside the scope of this paper. A brief description is presented to stimulate the reader to consider these controls and seek more information about them.

Many accidents can occur at a computer facility, some from natural causes and others premeditated. The specific goal of security programs should be to deter, detect, minimize impact of the disaster and loss, and then to investigate and recover.

Basically, the techniques used for security fall into two categories, physical and procedural. Some of the many physical techniques that can be used are controlled access,

physical location of the facility, and actual physical protection of the facility. Procedural techniques involve procedures that insure that only approved personnel have access to the appropriate information in the system[2, pp. 449-473].

The value of information and of the investment in systems to produce information dictates that the highest level of accuracy be achieved for that information. The extensive use of controls bears directly on the quality of the product generated by the system.

Forms/Reports design

Another factor that can determine the effectiveness of the system is the quality of the forms and reports that are used. A hospital can have the most accurate and potentially useful information available, but if it is not presented in a way the user can clearly understand, it is wasted.

When evaluating the design of input forms or output reports, several factors should be considered. The function of the document, its distribution, and the required physical characteristics of the document will help a designer decide how the form should look. Ergonomic factors should also be evaluated when designing inputs and outputs.

During this time, consideration should also be given to alternative means of input and output. Automated inputs such as optical character readers and point-of-sale terminals might be used as well as analog sensing devices tied

directly to the computer. Outputs could be generated on
terminals or by voice sythesizers.

Human procedures

Too often, a system is installed and people are
trained but very shortly the user forgets something and
there is nowhere to turn. Perhaps the reason so many people
are intimidated by computers is because they have seen this
happen to their friends. Without properly written proce-
dures and enforcement of those procedures, many anticipated
benefits of the system may never appear. As El Camino
Hospital in California discovered, this was a major factor
in their realized cost savings[1, p. 231].

Two aspects of human procedures must both be present
for procedures to be effective. The content must he valid
and its presentation must be clear and understandable. One
without the other will not do. Good procedures muddled in
an unreadable format are as useless as bad procedures writ-
ten well.

Of course management is responsible for providing
procedures that work but thought should be given to having a
professional write the documents. Included in the document
should be a description of the procedure, how it fits into
the total picture, and the actual details of how and when to
perform the activity. As a feedback to those involved in
the activity, the anticipated results should also be ex-

plained. This concept follows closely to those detailed in
the discussion on HIPO charts; show hierarchically how the
procedure fits, show the inputs and outputs and how the
actual procedure is performed.

Structured Design

"When we have decided on the automation boundary and
carved the computer system up into subsystems, . . . we have
to design the software within each subsystem."[33, p. 137]
In the example of an architect, this stage can be compared
to producing a blueprint from which the structure will be
constructed. Many companies that produce software claim to
have implemented Structured Design. They base this asser-
tion on the fact they use top-down-design and HIPO charts;
however, they have missed the point of Structured Design.

Top-down-design and HIPO charts are tools that may be
found in Structured Design but it is the modular approach
which produces easily changeable and maintainable systems
that define Structured Design. The concepts by which mod-
ules are determined allow Structured Design to accomplish
its goals, not just the fact there are modules. It is
important for a hospital having software designed for it to
understand these semantic differences in order to make cer-
tain it gets the best product.

The Goals of a Structured Design Approach

DeMarco claims that, "As the average life of a system increases slowly toward six years, the average percentage of the lifetime software cost devoted to maintenance approaches 60 percent! (Figures again from Barry Boehm)"[26, p. 298]. This supports earlier claims by Gane and Sarson. If these figures are anywhere near accurate, an obvious desire for a hospital would be to reduce those costs as much as possible. The goals of Structured Design, the production of designs that are easily changed or maintained and tested, have proven to accomplish this reduction in cost.

A maintainable system - modularity

A module can be thought of as a program segment that is characterized by the features listed below. To say that each module should be manageably small leaves much room for interpretation, but a good guide is program code no longer than a page or two[30, p. 94]. Besides being manageably small, modules should be able to be changed without creating a ripple effect throughout the system. Also, functions should be limited to as few modules as possible. Lastly, the results of each module should be predictable. These characteristics of an easily maintainable system are accomplished by incorporating the concepts of coupling, cohesion, morphology, and scope of control/scope of effect during the design process.

Coupling

Coupling is a measure of the dependence of modules on each other. The more dependent they are the more chance there is that when a change is made in one module it will affect others. This rippling effect makes it very difficult to make changes to the system and predict the results without spending a great deal of time tracing through the system to see how a change affects each module. Obviously there is no way to create completely independent modules since they must be called by someone to be useful, but as we limit the connections (coupling) or dependence we eliminate the potential ripple effect.

Cohesion

Cohesion measures the unity of purpose for a module. Each module should perform one and only one function and all the code in that module should be written with that one function in mind.

Morphology

Morphology refers to the overall shape or structure of the Structure Chart or HIPO chart. A well designed system should of course be hierarchical with one module at the top and more and more modules at each level below until a point is reached where commonly used modules (utilities) are called by several modules from above. This would appear similar to a diamond shape structure.

Scope of Control/Scope of Effect

These two related concepts refer to the relationship
between modules. As in a military setting, if a superior
(module) has control (calls) over too many subordinates
(other lower level modules), the superior's ability to con-
trol those subordinates effectively could be questioned. If
one module calls too many other modules, the chances that a
change in that superior module could unexpectedly affect a
subordinate are much greater than when a superior calls on
only a few subordinate modules. In such a case the superior
should be broken up into logical subsystems.

Scope of Effect refers to a situation where the action
of one module affects another module other than an immediate
superior or subordinate. When this happens, changes in the
module could have unexpected rippling effects throughout the
system.

An efficient system

This topic is mentioned not necessarily because it is
a goal of Structured Design, but because so many place so
much emphasis on it. This is perhaps a carry-over from the
days when main memory was limited and it was desirable
(almost to the point of being a status symbol) to perform a
function with the least amount of code. Now that memory is
much less expensive and even microcomputers normally come
with more memory than many mainframes used to, that reason

for condensing code is not nearly so valid.

Despite the fact the use of Structured Design tech-
niques can add as much as 10 percent to the CPU time and
memory requirements of a system [22, p. 101], the cost
factors related to maintenance of the system should over-
whelmingly suggest that the benefits of using Structured
Design far outweigh the costs in efficiency.

If efficiency is still a concern for the hospital,
Yourdon has a very revealing discussion in his book that
will help alleviate those concerns[22, p. 101-102].

Transition From Analysis to Design

The documentation tool most often identified with
Structured Design is the Structure Chart. The Structure
Chart is very similar to a HIPO chart in that it is hierar-
chical in nature and modular. The difference between the
two is primarily in two areas. The Structure Chart shows
interfaces between modules. It shows what data are passed
between modules. The other difference is that the Structure
Chart shows control and to a limited degree, decision
making. These items not included on the HIPO chart are
normally documented elsewhere. The value of the Structure
Chart is that one has not only the hierarchical, graphical
representation of the system, but also all the pertinent
information about the interfaces.

In describing the Structure Chart, it is not unremark-

able that it sounds similar to a DFD. Whereas, the DFD
showed what a system would do, the Structure Chart will show
how it is implemented in design. Because the two concepts,
DFDs and Structure Charts, are linked so closely, the task
of converting between the two is made much easier. The
processes used to do this conversion are called Transform
Analysis and Transaction Analysis.

Transform Analysis

One typical configuration for a DFD is to have an
identifiable input stream of data, a section of the DFD
where the input is transformed to output, and an identifi-
able stream of output data.

To reduce this type of DFD to a Structure Chart,
identify the stream of data that can be considered input and
follow it from its inception to the point at which it can no
longer be considered input. Conversely, determine the out-
put of the system and trace it back to the point where it
can no longer be considered output. The point at which the
two streams meet is called the point of transformation.

The actual Structure Chart would look like a pyramid.
The module that describes the overall process taking place
would be the top module and would call modules on the left
that get the input, then send the input to another module in
the center that would transform it to output, and then would
send it to modules on the right that would deliver the
output. Once the modules are identified, the data flow

between the modules (interfaces) are inserted along with any controls.

Obviously this is a simplified description, but it does represent the essential steps that would be done iteratively to produce a Structure Chart. An example of this is shown in Figure 3-4[21, p. 187].
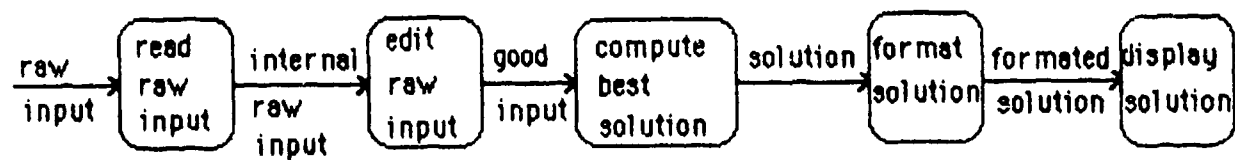
Transaction Analysis

The difference between Transaction Analysis and Transform Analysis is based primarily on the type of system represented by the DFD and more specifically by what happens during the transform phase, between input and output. Whereas in a transform type of DFD there is one identifiable stream of data through the transformation stage, with transaction type systems, the transformation stage appears as many parallel data flows.

An example is when a transaction comes into the transformation phase (depending on the type of transaction it was), it might take one of several paths and then converge with all the other transactions again to produce the output.

The steps for conversion are essentially the same as Transform Analysis and an example showing the DFD segment and the Structure Chart for a transaction type data flow is shown in Figure 3-5[21, p. 189].
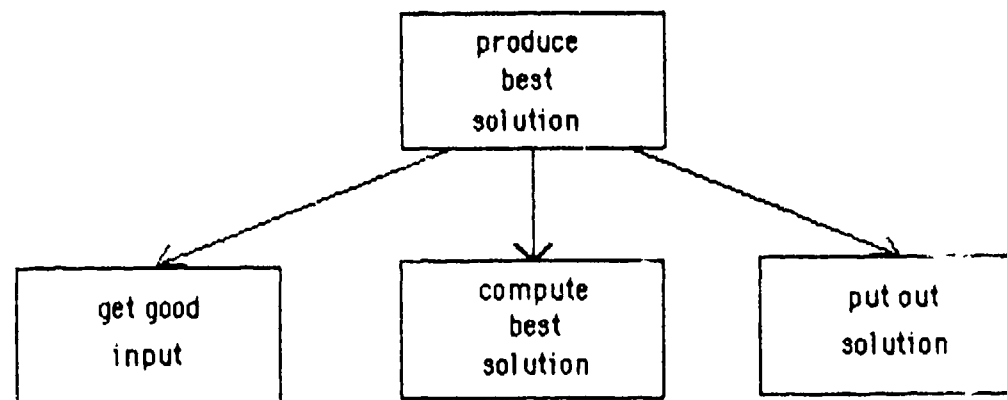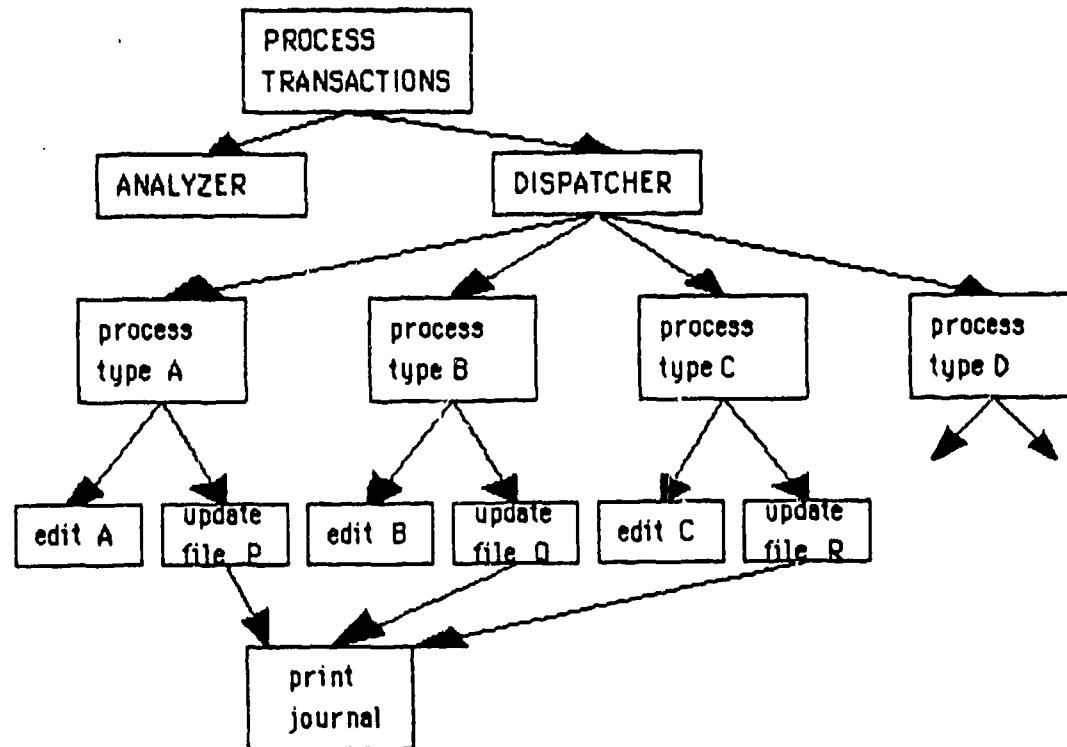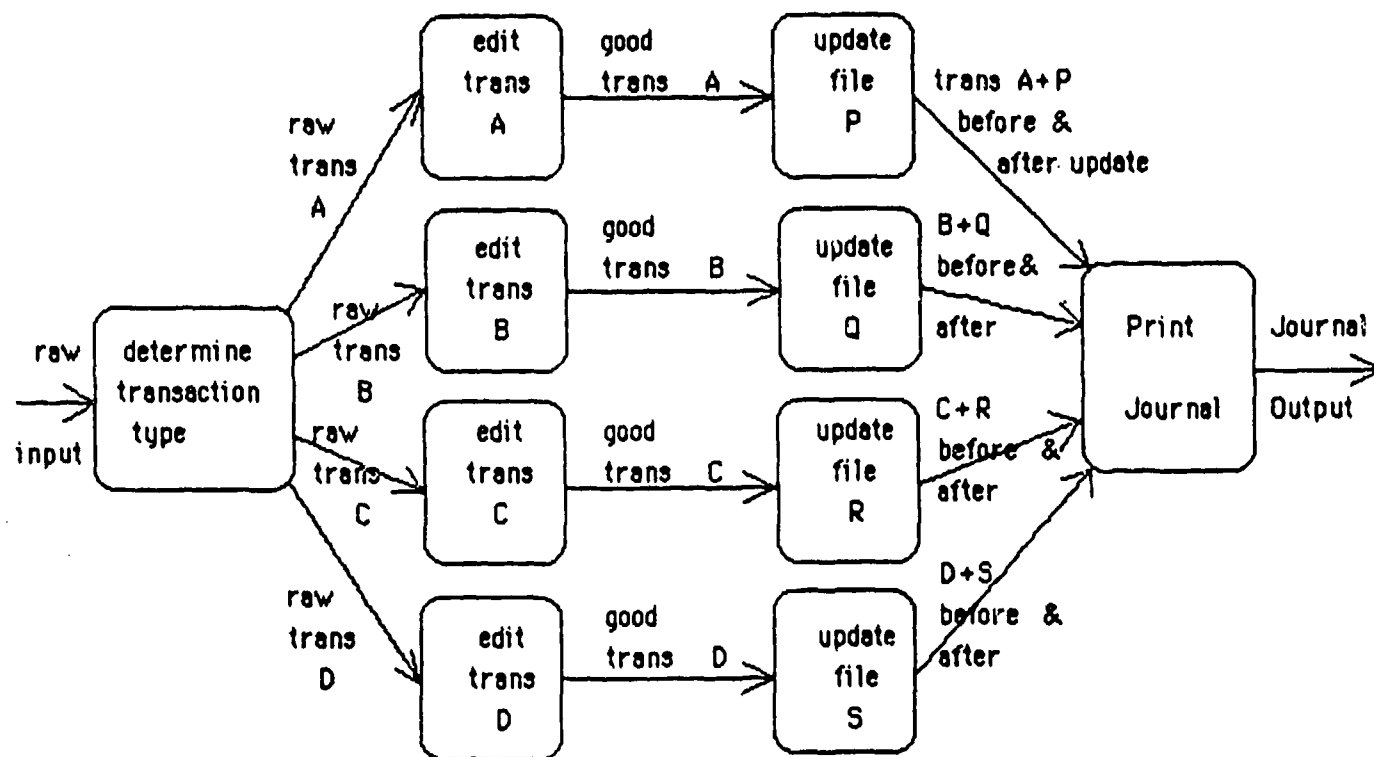
107
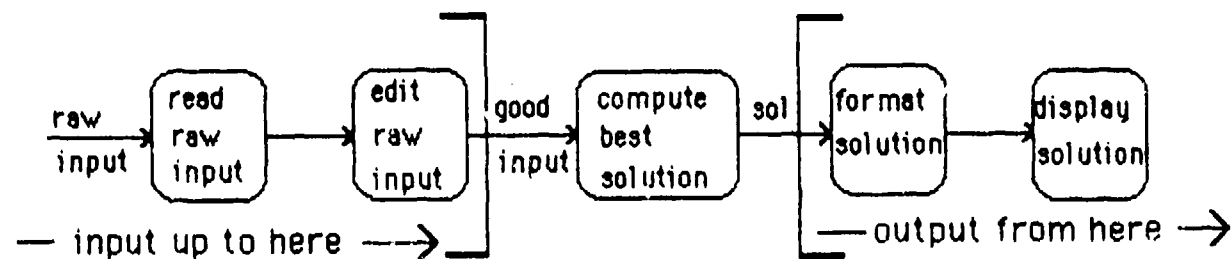
Figure 3-4: Example of
Transform Analysis

Figure 3-5: Example of
Transaction Analysis

Structure Charts

Examples of partial Structure Charts have been shown in the last two figures. A completed Structure Chart is shown in Fi ire 3-6[21, p. 207]. The added symbols are what makes this different from a HIPO chart.

The symbols are described below.
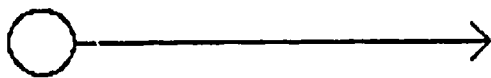
a data structure or element

a control flag or sequence symbol

a decision process

a looping process

These symbols are important tools to show explicitly the programmer what is happening in the design of the system.

Whether the ' ansform or Transaction analysis is done to produce a Structure Chart, the goal for the design is to be easily changeable, maintainable, and testable. This is accomplished by producing modules that are independent, non-complex, and that produce predictable results. If the designer follows the rules established for producing good

Figure 3-6: Example of a Structure Chart

modules that have strong Cohesion, limited Coupling, a good
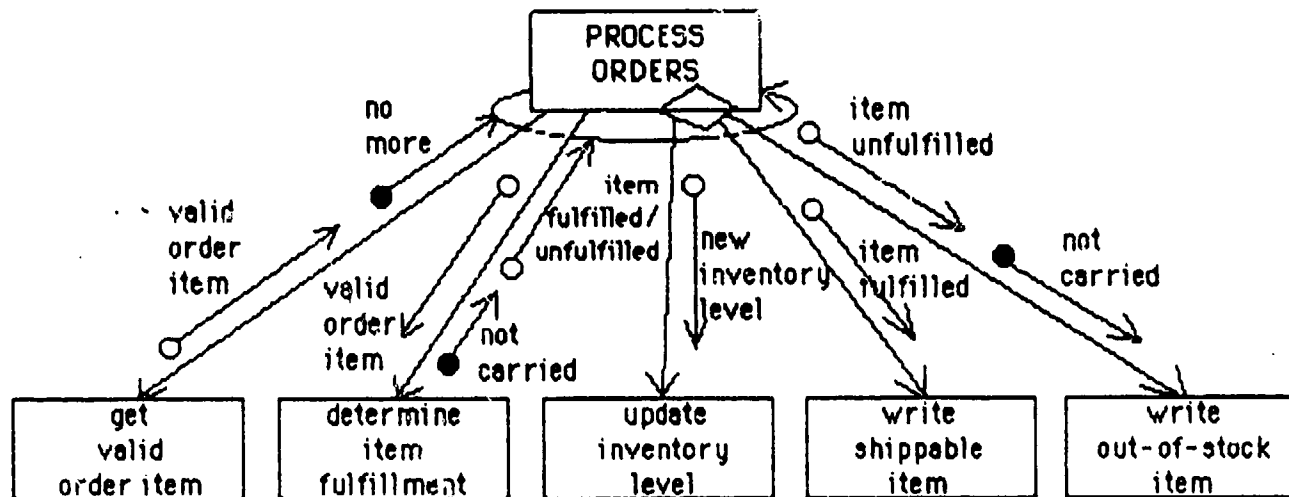Morphology, and the proper Scope of Control/Scope of Effect,
the design will undoubtedly reach its goal and the hospital
will get a good design that will save them money over the
life of the system.

CHAPTER IV

CODING AND TESTING

The decisions have been made. A design has been
selected, schedules approved, budgets agreed upon; the work
is now to begin. For the first time, perhaps, the hospital
believes the   will be an automated information system pro-
duced from all   e effort that has preceded. The hospital
typically responds in two ways. First, they want to be let
alone to resume their business, only to be interrupted again
when the project is complete, and second, they want it done
"yesterday".

The traditional programming techniques, using bottom-
up development, have courted this first desire of the compa-
ny. The only bother the software developer would be to the
hospital was to call a meeting periodically to announce
successful completion of another milestone. The hospital
executives would be ecstatic to hear that 95% of the coding
was done and the developer anticipated on-time completion if
not an early delivery. The scenario was repeated at each
completion milestone, but, suspiciously, the coding was
always 95% done. The final scene, though, was not a cordial
one. Inexplicably, the delivery date would come and the
developer was still 95% complete with just a few more bugs
to correct. The hospital's desire was honored; they had no
real involvement in the program development and testing

phase. However, the unfortunate result was that their sys-
tem was, and probably remained, 95% complete.

Conversely, a goal of the top-down structured program-
ming techniques is to involve the users by delivering pro-
gressively more complete, working systems.

> One of the major objectives of top-down testing [and the
> top-down structured programming techniques] is to involve
> the user in the early skeleton versions of the system.
> If something is wrong with the system (from the user's
> point of view), it is better to discover the problem as
> early as possible.[22, p. 84]

Also, the hospital can put great pressure on the
developer to get the system up and running as quickly as
possible. Often the request is not so subtle and it turns
into a demand that the software be developed as quickly as
is not possible. Perhaps these unreasonable demands are
what lead Brooks to observe that, "More software projects
have gone awry for lack of calendar time than for all other
causes combined."[23, p. 14] The developer needs to commu-
nicate to the users that their "urgency . . . may govern the
scheduled completion of the task, but it cannot govern the
actual completion."[my emphasis][23, p. 21] While the impa-
tience of the hospital can be understood and tolerated to a
degree, hospital management needs to be educated early in
the Software Development Life Cycle, that the analysis and
design phases of the cycle take approximately one third of
the cycle while coding and testing consume the other two
thirds (excluding maintenance)[40, p. 3]. Although use of

the structured techniques reduces the overall time for coding and testing, the vendor still cannot produce a system as fast as some companies desire; however, the ability of top-down structured programming to produce tangible deliverables often satisfies the unrealistic demands of the hospital.

Considering again the scope of this paper, it is not the intention to provide exhaustive information such that a hospital administrator or any other company executive could sit down and program a system. It is one thing to observe the need for continued user involvement and realistic time demands, and quite another to teach the techniques of programming in COBOL. However, it is important that a hospital which has contracted for software development be aware of the techniques and philosophies used by the developer. For, just as how analysis and design are performed affect directly the quality of the product and its subsequent maintenance costs, so too do the techniques used for coding and testing.

As in previous chapters, the discussion will be divided among two different approaches to coding and testing, the Traditional Approach and the Structured Approach. Included in the Traditional Approach are the concepts of bottom-up coding and the various stages of testing; unit/module, integration, system, and acceptance testing. These traditional techniques will be contrasted to the

Structured Approach with its integral concepts of structured programming and incremental testing. While it is true that all the stages of testing in the Traditional Approach are also accomplished while using the Structured Approach, there are obvious philosophical and technical differences which will be noted.

One common aspect, however, is the need for test data and a test plan. "Testing is the process of executing a program with the intent of finding errors."[34, p. 5] This idea is confused by many who believe that testing is done to provide a product that is bug (error) free; however, this is corrected by Dijkstra who claims that, "Program testing can be used to show the presence of bugs, but never to show their absence!"[35, p. 38-39] Still, the inexperienced would say to test all the possible inputs and determine whether the program produces the expected outputs or test every path in the program and then one could be sure of a "correct" program. While it sounds like exhaustive input and path tests would certainly accomplish the task of proving the correctness of a program, its economic feasibility and possibility are questionable[34, p. 10].

So, how can the software developer confidently produce a program and turn it over to the customer without the feeling that it could blow up at any moment? Two factors come to bear on this question. First, testing must certainly be done, but with the proper perspective. Test data must

be developed by the hospital and by the developer to test
the modules to determine whether they meet the functional
specifications; to determine whether they do what they are
supposed to do. The task is greatly simplified by properly
developed documents resulting from the design process. How-
ever, the amount of data used is going to depend on factors
such as the criticality of the program and the economic
constraints imposed by the hospital. So while the program
cannot be tested exhaustively, it can be tested to the
degree that the developer and hospital are confident in the
product.

Secondly, the use of the structured programming tech-
niques can also provide confidence that the product is
correct. The use of the structured programming techniques
has shown that more often than not, a programmer can produce
error free code. While it is logically and mathematically
impossible to prove the correctness of a program, being
confident that the tools and techniques one is using will
normally produce error free code, can perpetuate itself into
more and more error free code[35, p. 121].

Therefore, while it can never be proven by testing
that a perfect product has been produced, the use of clearly
defined tests and structured programming will greatly in-
crease confidence in the product and the likelihood that it
is indeed correct.

The Traditional Approach

Coding - Bottom-up

Traditional bottom-up coding, as the name implies, involves programming the lowest level module first and then successively higher levels, until a..1 modules in a system are completed. The technique seems reasonable. Since the programmer must start somewhere, why not start at the bottom?

One problem has been mentioned already -- lack of user involvement. Throughout this paper one of the watchwords has been "user involvement". If this virtue has been accepted in the previous stages of the life cycle, why is it not here? For one, the method forbids it since a working system cannot be provided until the entire system is complete. For another, no one ever really knows how far along the coding is. One writer observed that, "In the traditional approach, where the coding and testing were done from the bottom up, managers never knew just where the project stood."[36, p. 14] It is no wonder the users are not asked to participate more often.

In addition to the lack of user involvement, the negative effects of the bottom-up approach to coding are seen throughout the different stages of testing.

Unit/Module Testing

The first step in testing involves unit or module testing. In bottom-up module testing, the lowest level modules (ones which call on no other modules) are tested first. Testing at this level centers on two questions: "does the module perform according to specifications and is the logic correct?"

Testing whether the module meets specifications means providing inputs and determining whether the outputs are according to specifications. To provide data to the module, a "driver" routine or "harness" must be written to pass data to the module and then capture and display the results. The value of the test is directly related to the test data, so careful consideration should be given to its selection. As mentioned previously, it is practically impossible to test all data that might be input to the module, so the determination should then be made, "What subset of all possible test cases has the highest probability of detecting the most errors?"[34, p. 36] The functional specification will give the range of inputs and test data should test not only common values within the range but also values that are at the extremes of the range. Exception tests should also be done to test how the module handles data outside the range or in a form it is not anticipating.

In addition, the logic of the program should be tested. Obviously the programmer checks his code as he progres-

ses; however, the motivation for a programmer to find errors
in his own project is suspect. Having another programmer or
an independent test department do the checking of the logic
is of more value.

After modules at the lowest level have been tested and
corrected, the next stage of testing is integration testing.

Integration Testing

This testing involves putting together independently
tested modules and determining if they work together. As
the testing progresses through these later stages, more of
the testing will rely on test data than checking the logic.
Several problems usually arise at this stage when using the
bottom-up method.

Because different programmers have been sent off to
create their program modules without the benefit of clearly
established interfaces, a common problem is that each pro-
grammer will have his own idea about what the interface is
supposed to be. As a consequence, the modules have great
difficulties passing data among themselves and extensive
revisions are often necessary. The problem is compounded by
the fact that the major interfaces (the ones that would
occur at the top of the hierarchy) are the ones hardest to
deal with, but which, because of their location in the
design, are left till the end of the coding. Consequently,
it is not uncommon for the majority of the debugging to be
done immediately prior to the delivery date or later.

Besides the interface problem, there is a problem of trying to decide where a bug is located when more than one module at a time is tested. When several lower level modules are tested together for the first time, the ability to locate a bug is greatly reduced because one has no idea which module is causing the problem. The even greater problem exists when two modules have errors which cancel each other out to produce an overall correct answer. All is well until an unsuspecting programmer corrects the error in one module only to be rewarded by another error "somewhere" in the system.

As in module testing, a separate test department within the developer's company usually carries out the testing.

Systems Testing

At some point (hopefully), all modules will have been combined to pass the integration tests and the result is a completed system. Usually, the last test done under the control of the developer, prior to turning over the system to the users, is the system test. This test combines all aspects to determine if the total system meets the specified system objectives. In a sense, this is the last chance the developer has to refine his product before displaying it to the world. With the pressure of meeting a deadline and an inability to test the system objectively, it is recommended this test be done by an independent organization[34, p.

118].

Myers lists the following categories that should be explored when developing test data for the system:

1) Facility Testing - tests if all specifications have been implemented.

2) Volume Testing - tests if the system can handle extreme numbers of transactions.

3) Stress Testing - tests the ability of the system to handle volumes of transactions over time.

4) Usability Testing - tests how ergonomically designed the system is.

5) Security Testing - is used to evaluate specific security measures.

6) Performance Testing - is done to see if the system meets standards set for such things as response time and throughput.

7) Storage Testing - looks at how the system handles main memory and secondary storage requirements.

8) Configuration Testing - is done to test if the system will work on the hardware specified.

9) Compatibility/Conversion Testing - is needed when the new system must interface with an existing system or convert from an older system.

10) Installability Testing - tests whether the system can be installed correctly and easily.

11) Reliability Testing - tries to establish that

Mean-Time-Between-Failure (MTBF) rates are acceptable.

12) Recovery Testing - test the recovery procedures to determine if they function properly.

13) Serviceability Testing - might test diagnostic tools or utilities provided with the system to aid in service or maintenance of the system.

14) Documentation Testing - is important to determine if the written documents are readable and useful.

15) Procedure Testing - tests if written procedures involving the system produce the desired outcome[34, pp. 112-118].

While all of these tests may not be applicable for each system, they are listed for the reader's consideration.

Acceptance Testing

The system test is completed and the system is turned over to the users for their evaluation. It would seem the developer's work is completed since, to the best of his knowledge, all specifications have been implemented and confirmed during the systems test. Unfortunately, this is not the case. More often than not, because of ambiguities in the specifications, what the developer thought a specification required and what the users actually wanted are two different things. This situation leads to several observations.

First, the use of bottom-up techniques compounds the problem. Regardless of how well system objectives, design

goals, and functional specifications are written, there is still going to be some degree of misunderstanding. The top-down approach tends to minimize the impact of this eventuality by forcing the users to evaluate the system early in the development instead of at the tail-end. Conversely, in the bottom-up approach, the results of misunderstandings can have devastating effects on the software developer.

Also, the frequency with which this scenario is carried out should again point out the need for diligence in establishing specifications. The added time needed to insure clear, measurable specifications will pay for itself during a relatively "painless" acceptance test. An added benefit to the clearly written specification is the ability of the developer to ascertain whether the users are honestly enforcing a specification or just trying to add another feature to the system. It is true the users may realize the need for another function in the system, but they should not be able to gain its implementation without cost, through the use of ambiguous specifications.

The Structured Approach

As is so often necessary, a proper context is needed for the following discussion on structured programming. Yourdon, a well known protagonist of the structured techniques, provides this insight.

> Structured programming, no matter how brilliant, can-
> not compensate for poor design and poor analysis. Proper
> structured analysis and structured design are of para-
> mount importance; they far outweigh the impact of struc-
> tured programming.[22, p. 118]

Said in another way, the proper utilization of each of the

structured techniques is necessary but not sufficient by

themselves to produce a good system. It becomes apparent

when operating with the structured techniques that not only

are they each individually necessary, but they also build on

each other with a synergistic effect. This is no more

apparent than in the programming phase where the top-down,

hierarchical design produced during Structured Design be-

comes the foundation for structured programming. While the

value of structured programming is not to be minimized,

there are other techniques often associated with the struc-

tured approach which should also be mentioned.

"Step-wise refinement" or "top-down development" are

terms used to describe the reduction of modules from broad

functional descriptions into code utilizing the structures

described later. This concept, very similarly described in

top-down design, carries out the common-sense approach that

a problem is best attacked by breaking it down into smaller

more understandable parts; from high levels of abstraction

to lower levels of abstraction.

The programming team is also employed with the struc-

tured approach. This tactic combines three or four people,

a chief programmer, backup programmer, programming secre-

tary, and possibly other junior programmers or necessary personnel, into a single team. The goal of the programming team is to produce programs using the structured tools while at the same time eliminating the degrading effect caused by communications among larger numbers of programmers. In addition, the function of the programming secretary is to assume all the administrative tasks which also tend to stifle the productivity of a programmer. The chief programmer should be a senior programmer and one who is given the responsibility of running the team. The backup programmer is also a senior person and is considered the chief programmer's alter-ego. Depending on the size and type of the project, a few junior programmers or other necessary personnel (other engineers or trainers) might be added to the team.

An added benefit to the team concept (using a programming secretary) is that better documentation is usually produced. With a secretary to take care of the busy-work of maintaining the documentation, programmers are more apt to keep it current and refer to it.

Walkthroughs can also be used to have a peer review of the programs. The successful use of this tool depends on a conscientious walkthrough coordinator and peers who will prepare before the meeting to provide constructive criticism. Not only will walkthroughs provide an environment for detecting errors, they will also provide training time for

programmers to learn new programming styles.

Many benefits accrue to the software developer using
the structured approach. Some other benefits that have been
observed by those at IBM who have used these techniques in a
production environment are:

1) A project [is allowed] to staff up more gradually and
reduce[s] the total manpower required.

2) Computer time requirements [are] . . . spread more
evenly over the development period.

3) The user [can] . . . work major portions of the system
much earlier and identify gross errors before accept-
ance testing.

4) Most of the system [has been] . . . used long enough
by the time it is delivered that both the user and the
developer have confidence in its reliability.[35, p.
200]

The Transition from Structured Design to
Programming - The Implementation Plan

At the beginning of this chapter a scenario was pres-
ented in which the users were excluded from the development
process except for being notified month after month that the
project was 95% complete. Throughout the chapter, the er-
rors of this situation have been brought to light and a
solution, the use of the structured approach, has been
proposed. This proposal is formally carried out through the
development of an implementation plan.

The implementation plan is a schedule showing the
users to what extent each version of the system will be
usable. In other words, "The implementation plan should
specify the deliverables of each version in terms of strong-

ly stated objectives."[21, p. 220] In a large system with many subsystems, the implementation plan will establish a priority of production most beneficial to the users. This will enable the developer to develop portions of the system first which can be used in a limited way by the users.

A key figure in the development of the implementation plan is the analyst, since he should be most familiar with the needs of the users and the realistic capabilities of the software developer. The analyst will act almost as an interpreter for the users, explaining the development of the system and encouraging the users to participate actively.

Structured Programming

When confronted with the concept of structured programming, what most often comes to mind is a programming technique which absolutely forbids the use of the GOTO statement. However, ". . . the essence of structured programming is not the absence of GOTO's in programs, but the presence of rigor in programming."[35, p. 6] The outcome of this rigor is programs which are easily tested, changed, and maintained. While a product with these characteristics is an output goal, another ". . . major objective of structured programming is simplification of the program development process."[35, p. 12] A major element necessary for attaining the goals of structured programming is the use of only three programming structures.

The Structures

The mathematical justification for using only three programming constructs was provided by two Italian computer scientists, Bohm and Jacopini, in the mid-1960s. Their work showed that any flowchartable logic can be represented by these three structures[37, pp. 366-371]. These structures, ". . . also have the desirable property of being black-box in nature. That is they all are characterized by having a single entry point, and a single exit point."[22, p. 107] The result of using these structures is a program which is easy to read, change, test, and maintain. This cannot be said of programs written from standard flowcharts which detail logic that wanders incoherently throughout the program.

The three basic structures are shown in Figure 4-1 and include sequence instructions, decision instructions, and loop instructions. The basic instruction is the sequence instruction (Figure 4-1(a)) which chains together in order, several processes. Through step-wise refinement these processes can be refined further to any of the three structures.

The decision instruction involves a binary decision whose path through the structure is determined by the results of a test. In Figure 4-1(b), if the result of the test is true or yes, "process a" would execute while if the test was false or no, "process b" would execute.
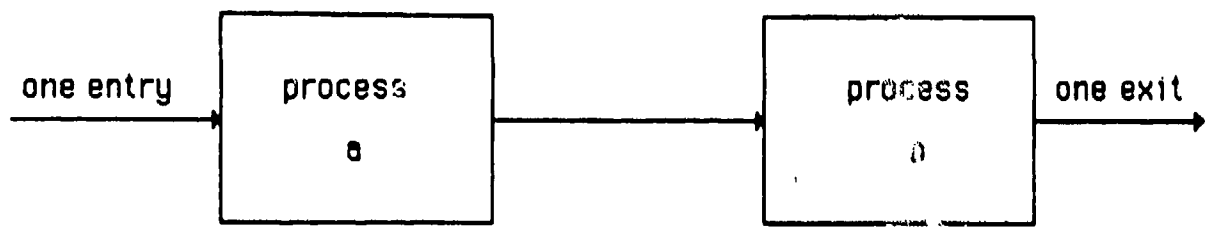
(a)
SEQUENCE INSTRUCTIONS
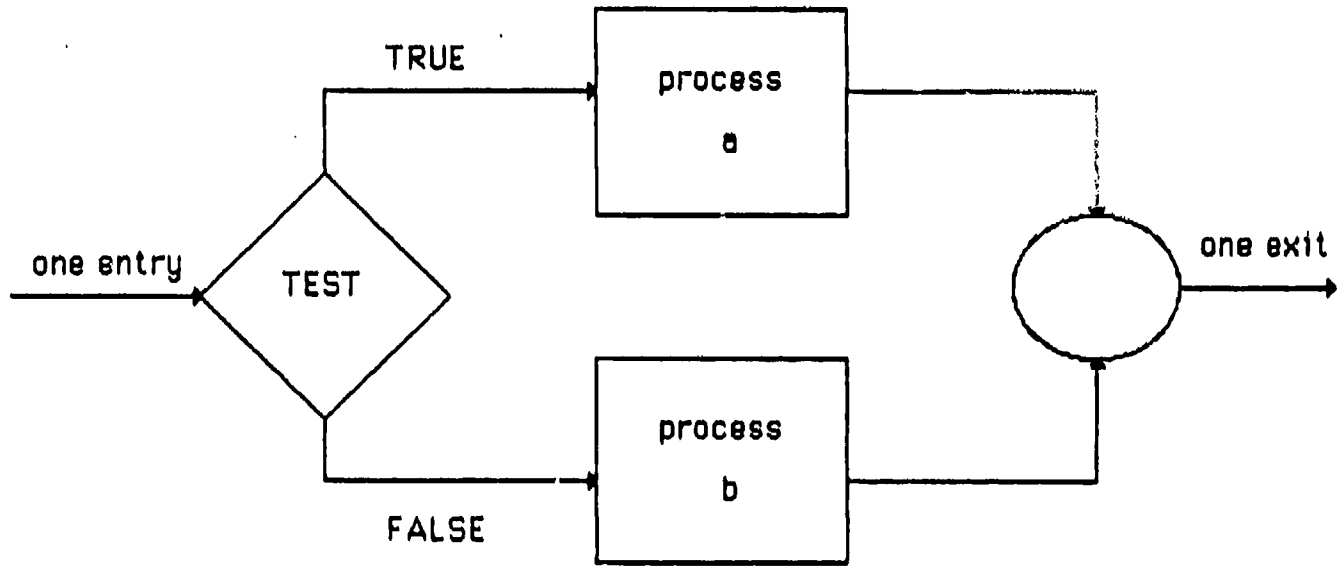
(b)
DECISION INSTRUCTIONS (IFTHENELSE)
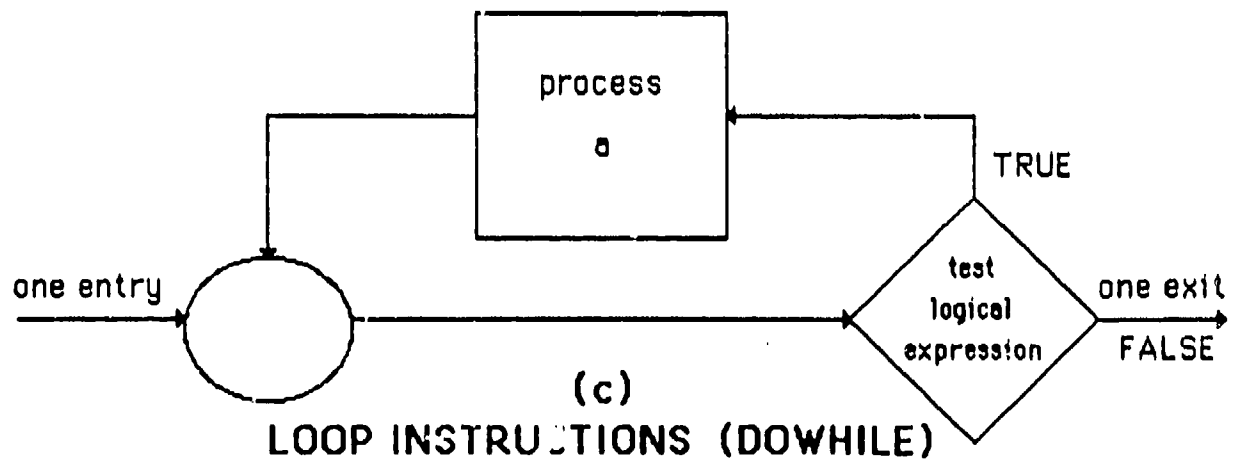
(c)
LOOP INSTRUCTIONS (DOWHILE)

Figure 4-1: Basic Structured Programming Constructs

Lastly, the loop instruction shown in Figure 4-1(c) performs "process a" until the "test logical expression" is false.

Acceptable variations of these structures, supported by some higher-level languages, include the case statement and the repeat-until statement.

The transitions through Structured English to coding

During Systems Analysis, primitive processes had their logic described by one of several methods. Decision tables and decision trees were used to describe logic involving many conditions and combinations of conditions which produced certain actions. However, another way to express the logic of a less complicated process was through the use of Structured English. By using the structures defined above and active English verbs and data elements described in the Data Dictionary, it was possible to express concisely the logic in a way the users could understand and in a way that could be utilized later by the programmer.

Later, during the design phase, when more details concerning the physical structure of the system were developed, these details needed to be included in the description of the functional, primitive process. By including such things as control features, initialization procedures, and file access procedures, the Structured English was converted to a more programming-like expression called Pseudocode.

While still not a program, and while it still did not dic-
tate to the programmer how to program the module or process,
it was much closer to a program than Structured English. In
essence, it was a more precise, rigorous definition of the
function to be performed.

From the Pseudocode, the programmer is then able to
understand more clearly the function of the process and
convert it to a higher-level language using the structured
programming constructs.

Incremental Testing

Much of the discussion above on testing applies to the
incremental testing done during the structured approach.
The goals of module testing, integration testing, and system
testing are all accomplished with incremental testing, al-
though the time divisions between each are not so clearly
defined. The differences, though, are seen in several
areas.

Whereas the traditional approach coded and tested from
the bottom-up, incremental testing follows the top-down
approach. The inherent difficulties of the bottom-up ap-
proach are eliminated. The problem of not being able to
deliver a workable system until the very end of the project
and thus not having the users actively involved in the
development process is solved by top-down, incremental test-
ing. This is important because one must,

> . . . keep in mind that many of the problems will come
> from the user - . . . There is no point in finishing the
> entire design and letting the programmers reassure them-
> selves that all the interfaces are proper, and that the
> design is perfect, only to have half of the entire effort
> thrown out because the user changed his mind."[22, p. 81]

Another major flaw in the bottom-up approach is that

major interfaces, which are most often hardest to deal with

and consume the most debugging time, are left until last.

These interfaces should be defined first and ". . . that is

exactly what the top-down approach is trying to accomplish:

forcing the precise definition of major interfaces, and

forcing those interfaces to be coded and exercised in a

computer to ensure that they work."[22, p. 83]

Incremental testing is accomplished top-down by start-

ing with the top module and testing it by connecting it to

its subordinate modules. By definition, an untested module

is not tested with another untested module, so the untested

subordinate modules are replaced by "stubs".

> A program stub is some very short code . . . [used] to
> serve as a place holder for an uncoded segment. A pro-
> gram stub should at a minimum permit any code that refer-
> ences it to continue executing. A stub must therefore
> meet any interface requirements specified for the uncoded
> segment.[38, p. 97]

As a module is successfully tested, the stubs are replaced

one at a time by actual programs with new stubs written as

needed.

The procedure of testing only one untested module at a

time greatly simplifies the task of tracking down bugs.

Since all other modules would have been determined error

free, any bugs discovered can be attributed to the new module.

When the modules for each version of the system are completed and tested, the "deliverable" (a version of the system) is shown to the users, who then have an opportunity to evaluate it. Consequently, the module tests, integration tests, and system test are finished together as the last module is added to the system and tested. No 11th hour heroics are needed.

While the acceptance test still needs to be accomplished, the ordeal should not be nearly so traumatic. The users have seen, and perhaps used parts of the system for most of the development time. There should be no surprises during that phase of testing as most of the discrepancies have been worked out. When it comes time for installation, the users should be quite familiar with the system, and more than that, will be getting the product they need and one which will save them money over the life of the system.

CHAPTER V

SYSTEMS IMPLEMENTATION

The art of progress is to preserve order amid change
and to preserve change amid order.

Alfred North Whitehead

Perhaps the greatest challenge during this phase of
the Systems Development Life Cycle is expressed in the above
quote. It is quite possible to develop a technically sound
information system and yet have the system fall far short of
its anticipated goals and objectives. In many cases a
hospital will spend much time and money developing the tech-
nical aspects while neglecting the organization which must
deal with this system on a daily basis. This is substan-
tiated by a survey done for the Hospital Financial Manage-
ment Association which identified major problem areas asso-
ciated with Systems Implementation. Of the five most fre-
quently mentioned problems, four dealt with the organization
and staff. They were:

1) integrating the system into the hospital environ-
ment;

2) communicating between data processing and other
departments;

3) planning; and,

4) training[39, p. 111].

People are just one part of an organization and while

adequately preparing staff for the impact a new HIS will
have on them is important, impacts on other aspects of the
organization must be considered also.

When going from a manual system to an automated one,
the structure of the organization will be impacted. Whereas
prior to implementation there might not have been a need for
a separate Electronic Data Processing (EDP) Department, the
growth or emergence of this function may demand it. As
happens in many hospitals, the situation is not one of going
from no system to a fully automated HIS, but rather proceed-
ing from a partially automated to a more automated system.
In these cases, the earlier EDP functions were most likely
under the auspices of the finance department since they were
usually the first to use automated applications for its
financial functions. However, as the automated HIS grows
beyond the boundaries of just financial applications, many
hospitals have seen the need to establish an autonomous
department which reports directly to the administrator[24].

Woodrow Wilson once said, "If you want to make ene-
mies, try to change something." This is especially true
when applied in a situation where a department head (busi-
ness department, perhaps) has grown quite accustomed to the
power derived from being in charge of the EDP functions in
the hospital. To suddenly strip that power away and give it
to someone else (probably to a "computer person") can cause
animosity and possibly account for a system that works far

less effectively than anticipated. Careful consideration, however, of all the political implications and power struggles a change like this and other structural changes will have, can help allay many problems.

Another aspect of the organization that will be affected by the HIS is the way things are done; the procedures. Hopefully, during the earlier stages of the Systems Development Life Cycle all affected procedures were reviewed and decisions made about whether those procedures were valid implementations of current policy or whether they needed to be changed. As a result of this analysis and the intrinsic changes to procedures a computerized system will bring, chances are people will have to start doing things differently. The way a hospital goes about developing and preparing for the implementation of these new procedures can make the difference between a good system and a good system that works well.

When looking at installations of HISs, one can conclude that the success or failure of a system depends on a myriad of factors which can be categorized into two groups. Although there is some overlap between these two categories, basically, problems relate either to the technical development of the system or to the system's impact on the organization. The previous chapters have attempted to provide guidelines for establishing a technically "good" system. This chapter deals with the way the technically "good"

system is introduced to the organization that will work with it. The way the system is introduced (installed) is as important as the way it was developed. One author goes so far as to say that, "It is the organizational impact of computerization that can determine the success or failure of a hospital information system."[39, p. 112]

Installing/Conversions

The actual installation of a system can be a mammoth task, which, without proper planning, could turn into a disaster. As well as the technical problems which could arise from such things as improper site preparation, other problems result when the installation causes the hospital staff to question the systems credibility. Remembering that first impressions often stay with people far longer than the physical effects of the meeting, it is very important to create a good impression of the system for the staff. Both types of problems need to be considered regardless of which approach to systems conversion is applied.

Approaches to Systems Conversions

Basically, there are four approaches to installing or converting to a new system. They are the direct conversion, parallel conversion, modular conversion, and phased conversion.

The direct conversion involves one day stopping activity of the old system and starting the new. An advantage of

this method is the cost savings generated by not having to run both systems at the same time. Normally this approach is taken when it is either not feasible to run both systems together or there is no other system to begin with. While the monetary advantage may be very appealing, the decision should be weighed against the inherent risks of using this approach. The hospital must have great confidence in the reliability of a system installed this way since there are no checks on the data produced by the new system which would be available if both old and new were run together. And, even if the hospital management is confident, a good "sales job" must be done on the staff to instill that same confidence in them. Lastly, much consideration needs to be given to the added stress on the organization incurred during a direct conversion. A little stress over an extended period of time is often tolerated better than a great deal of stress inflicted all at once.

The antithesis of the direct conversion is <u>parallel conversion</u>. When the situation permits, many companies opt for this approach. While it may not be feasible because the old and new system are too dissimilar or the cost of running both systems is prohibitive, when it is possible to use the parallel conversion it can offer several advantages. Whereas, in the direct conversion there are no data to check the operation of the new system, in parallel conversions the outputs of both the old and new systems can be compared and

discrepancies evaluated. Some might consider this an exten-
sion of acceptance testing, and to a degree it is. In the
last chapter it was shown there was no way to prove the
correctness of a program, and it is very likely that even in
a fully tested system the user will discover errors. This
fact does not, however, imply that acceptance testing should
be any less stringent, thinking that the errors might be
found when the system is installed. The consequences of
this kind of thinking can lead to a system with no credibil-
ity because all the user sees are errors continually being
uncovered. While some may consider this a form of testing,
it should only be so considered to the extent that during
the rest of the life of the system, it is inevitable that
some errors will be found. The fundamental difference is
that formal testing is done with the intent of trying to
make the system fail, while parallel conversion is not done
with this intent. Parallel conversion might also be done
because of necessity. If a system, for some reason, is not
complete, parallel conversion may be the only way to proceed
until the new system is done. This approach might also be
used to compensate for inadequate training done prior to
installation. Whenever parallel conversion is chosen, the
decision of when to move completely to the new system should
continually be evaluated.

In situations where the same system is going to be
installed at several locations, a modular conversion ap-

proach might be used. For example, if a company has several warehouses throughout the country an inventory control system might be installed in one location as a pilot case. After successful implementation of the system, it would then be installed at the other locations. The use of this approach does not preclude the use of either a direct or parallel conversion at each location.

The fourth approach to conversion is the _phased approach_. Similar to the modular approach in that the installation is done in stages, the difference is that the phased approach segments the system itself and a different segment is installed at each location. This approach combines well with the structured programming techniques which provide usable segments of the system throughout the development of the system. It seems this approach also lessens the shock of implementing the entire system all at once. The benefit to the user is similar to that desired by Jerry Brown, ex-governor of California, when he said, "I reject the get-it-done, make-it-happen thinking. I want to slow things down so I can understand them better." The phased approach allows the user to deal with manageable parts of the system and better understand its function.

Data Base Considerations During
System Conversion

"... a data base is a repository of interrelated data of interest and value to the users of the system."[2, p. 163] Whether it be a physical file or a computer disk-pack, every hospital accumulates valuable information necessary for the functioning of the business. When a new information system is installed, many of the ways data are stored will be affected. For example, the new system may require information that is now stored on paper in a file be stored instead in the computer. Or, a collection of data which included only name and age may now also require the social security number. Another example of a conversion that might need to be done is if the dates used in certain files were recorded one way and now the new system requires they be done in another. Generally, these changes will fall into one or more of three categories: 1) changes in the format of a file, 2) changes in the content, or 3) changes in the storage medium of the file[2, p. 524].

The oversight of these changes will normally be left to the data base administrator with assistance from the developers of the system. Often times, the systems developers will provide utility programs used during the start-up of the system which are used for these conversions. When conversions involve much entry of data by hand, it is sometimes helpful to hire extra data entry operators to assist.

Also important during data conversion is verification

of the accuracy of old data. A valuable asset of a computer
is its ability to consistently perform accurate calculations
and produce reliable information. This presupposes, though,
that the computer has accurate data to begin with. Many
times the accuracy of information produced by manual or
other means is not so reliable. Data conversion is an
appropriate time to correct any inaccuracies in the old data
to insure the information produced by the computer will
indeed be correct.

The Conversion Plan

The method by which a system is installed is often
dictated by factors other than preference. If a system is
developed using the bottom-up approach where the system is
not available for use until the entire system is complete, a
direct conversion might be required. If the top-down ap-
proach techniques are used in which parts of the system are
available for use throughout the development of the system,
a phased approach might be possible. Whichever of the
approaches is used, its impact on the organization must be
considered.

The organization will understandably be interested in
getting the system installed as quickly as possible. They
should, however, realize there will be an enormous burden on
the users of the system during the conversion. The hospital
must accept this fact and anticipate either a loss in pro-

ductivity or, if this is not acceptable, plan to increase the manpower to sustain the acceptable productivity. The hospital that fails to accept this inevitability will certainly end up with staff who resent the system because of the unrealistic workload it placed on them.

The approach used to install or convert the system will also be affected by or affect the way hardware is procured. The availability of equipment might determine how the system can be installed, while, in a situation where the equipment is readily available, the proposed installation plan will dictate the timing for equipment purchases. The idea here, then, is that the installation plan must match the plan for purchasing components of the system.

In addition, site preparation will affect the installation plan. Such things as power requirements, building modifications or construction, safety preparations, physical security, environmental requirements such as air conditioning and humidity, and the actual delivery of equipment must be considered when determining a conversion plan.

Another item to consider during the conversion plan is the hiring of new personnel. If possible, the timing of these staff additions should allow for proper systems training prior to the arrival of the equipment.

Probably one of the most important parts of the conversion plan deals with training. The proper training, especially of users, can not only result in more capable

users but also in users with a more positive "feeling" about the system.

Training

"Use is indeed the critical element in successful systems, rather than technical perfection in the design concepts."[18, p. 121] If this is true, then it behooves the hospital to determine methods which will stimulate users to use the system. An integral part of getting this done is training.

User Training

> Perhaps the single most prevalent yet underappreciated negative phenomenon accompanying computerization is user resistance. Professionals and clericals have consistently tended to challenge the use of computers in their organizations. Because managers have failed to deal with staff resistance, many technically sound computer applications have been inordinately troubled or have failed altogether.[7, p. 135]

User resistance can be displayed in varying ways. It might be seen as open rebellion or "bad-mouthing" the system. Resistance, whether it be the very insidious aspect of just not using the system to its potential or the sometimes destructive act of sabotage, ultimately results in an inefficient system.

What causes people to act in these ways? Some fear that the computer will replace them. Others don't use the system because they don't understand well enough how it works and are embarrassed to admit it. Some have not been convinced of the value of the system and have not been shown

how it will benefit them. Those who were not consulted during the planning of the system might be resentful. The implementation of new procedures or the break-up of a closely knit work group might also cause some to see the system as an intruder instead of a helper. The list could go on and on, but the tragic part is that with proper training much of this resistance could be stemmed.

Why is user training so often underutilized? One reason could be a lack of awareness on the part of management of the need for this training. For those who have been involved in the development of the system and who are so familiar with it, it is difficult to realize that the users have not been as concerned about the system's development nor do they understand as well how it works. Management assumes more knowledge and motivation on the part of the user than is actually there.

Another reason could be the financial outlay required for a good training program. While it is easy to project the costs of training, it is much more difficult to quantify the benefits. However, a shortsighted decision to trim the training budget will compound itself in limited use of the system.

Training should also not be viewed as a last minute event. Training should begin as soon as system procedures have been established and equipment is available. A vendor many times will provide training on their equipment. This

may require some added expenses, but if a few key people can be trained early they can be a very valuable asset when training the rest of the users later.

Training should be tailored to the needs of the different users. A hospital should evaluate the impact of the system on the physicians, nurses, ancillary staff, patients, administrative staff, and management and should provide the specific training needed to counter those negative impacts and turn resistant users into motivated users.

An aspect of training users that is sometimes overlooked is the need for ongoing training. As staff leaves a hospital or are moved to new sections within the hospital, requirements for training continue. This requirement can best be fulfilled by a training program that produces qualified people, capable of training others, in each function of the system.

Operator Training

Usually, operator training is not as involved as user training. Because it is the operator's job to work with the computer (which they have probably been doing for some time) the problem of resistance does not need to be overcome. There is, however, still a need for training because of the new equipment involved or new software that is installed.

Both the equipment vendor and the software developer should be responsible for training the operators to run the

new system. In addition to initial training, operators should also be trained as changes or additions are made to the system. As in user training, this will require an on-going training program.

Training Methods

Considering the importance of training, it is not unreasonable for a hospital to solicit professionals who know how to train others effectively. Whether the vendor supplies these professionals or they are hired by the hospital, the methods used should be suitable for the particular needs of each user.

Methods that might be used include:

1) Seminars and group instruction, which can be used to train a large number of people who perform the same task.

2) Procedural training, which involves giving the user written procedures for a task and allowing them to ask questions.

3) Tutorial training, which, although very expensive, is very effective for explaining complex tasks.

4) On-the-Job training, which is used extensively and can be very effective if proper time is available in the work environment[2, pp. 511-512].

Regardless of which method is used, proper planning is required for effective training. This requires that each task needing training be established along with the skills needed to perform the task and a plan for teaching those

skills. Surveys of the work done by new trainees should also be evaluated to determine the efficacy of the training. Reviews should be done periodically to determine if training is needed in previously undocumented tasks.

Maintenance and Managing

"A program doesn't stop changing when it is delivered for ustomer use. The changes after delivery are called program maintenance . . ."[23, p. 120] One book claims that 67% of all costs associated with a system can be attributed to maintenance[40, p. 9]. The culmination of all work done during previous steps of the Systems Development Life Cycle will determine whether this is true or not. If techniques have been used which make this inevitable maintenance easier, then the costs will be lowered. Whichever the case, the maintenance must be managed or chaos will reign.

Managing Maintenance

Besides the obvious maintenance required to correct errors in the program, changes to the system can be generated for several reasons.

From a very broad perspective, these requests are going to arise for exactly the same reasons the system was developed in the first place except the originator might be at a lower level within the hospital. Requests to change the system are going to result either because an entity within the hospital wants to take advantage of an opportun-

ity to increase revenues, avoid costs, or increase service, or because it is reacting to a pressure[21, p. 155]. Either of these could result in requests to enhance the system or change it to conform to new policy.

The methods and considerations used to develop the system can and should be used to perform maintenance on it. This involves going through the Systems Development Life Cycle of analysis, design, coding and testing, and implementation. The same considerations need to be given to who performs the maintenance, whether it be in-house or by a contractor. Granted the maintenance may not take nearly the effort required during the original system development, but if these steps are not taken, over time, an unwanted metamorphosis can occur with the resulting system being a hodgepodge of unreadable, illogical code and the hospital would have wasted the money spent on the careful development of the original system.

A situation that often occurs when new systems are first installed is a flood of requests for changes to the system. New users who have had their imaginations stimulated by the use of the system think of new ways the system could be enhanced. Without squelching this very useful process, it is important for the hospital to allow the new system to "settle-in" before extensive changes are made. The formation of a user's committee to prioritze and periodically review requests will allow a forum for new ideas and

yet keep it under control.

Auditing the System

". . . [audits] are performed to ensure the integrity and operational efficiency of the system."[2, p. 545] Whether it be an information system or any other system, audits are necessary to make sure a system is still perform- ing as it should. It has been established that the system will change and if audits are not done the system could certainly change to the point where it no longer functions as it should.

Several types of audits can be done. A post-implemen- tation audit is used to ensure that the system and all vendors have satisfied the contracts. This audit could also finalize all costs involved during the development of the system and a comparison could then be made between projec- tions and actual costs. This information could be very valuable when other systems were contemplated later.

Routine operational audits are necessary to determine if established procedures are still valid and if they are being followed.

Financial audits are not unique to automated informa- tion systems. However, since an HIS usually performs finan- cial applications, a financial audit is necessary to insure proper controls and accounting methods are being applied.

A systems audit is done to evaluate all aspects of the

system, hardware and software, to ensure that current opera-
tions have not degraded the effectiveness of the system.
This will forestall the situation where a hospital suddenly
finds itself overtaxing the system and ends up with costly
emergency changes to keep the system operating. Systems
audits should show trends that will predict these occurences
and allow proper planning to take place.

Audits can be done either by reviewing output informa-
tion to see if it matches the expected results or the compu-
ter can be used to accumulate data used in an audit.
Thought should be given when using the computer for audit,
as the additional overhead needed to perform audit tasks can
sometimes degrade the system and give skewed results.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

Over the years, Hospital Information Systems have been found primarily in the administrative areas such as accounting and patient billing and in the ancillary services like the laboratory and pharmacy. Attempts to develop integrated systems which would allow data to be shared among all areas of the hospital met with limited success. The result, many times, is hospitals with many separate stand-alone systems that make it difficult to consolidate information to make important decisions. Until recently, this has not been a very serious problem.

When the federal government and other third party payers who reimburse hospitals for the cost of treatment rendered its members enacted new procedures for determining the amount to pay hospitals for those costs, serious problems arose. Whereas hospitals had not previously been as concerned about monitoring costs which they knew could be recovered from third party payers, now they found themselves forced to institute new ways of doing business or go bankrupt. For many hospitals, procedures needed to be instituted which could track the types of services provided their patients (customers) and also determine specifically what

151

costs had been incurred. Not only was this needed to determine where to concentrate efforts in attracting customers, it was also necessary to determine if reimbursements were going to cover costs.

For the first time, decisions needed to be made which required detailed information from not only the financial area but also the clinical area. Administrators had to have financial information which would indicate revenues and costs but also needed clinical information which would show the type of health problem being treated. This clinical information was important because the type of health problem would determine the amount of the reimbursement. At once, the folly of investing in stand-alone systems became apparent. The need to share information between the administrative activities and the clinical activities, in most cases, was next to impossible.

As a result of this new impetus, hospitals are searching out information systems which will consolidate information that can be used in decision-making activities. The overwhelming amount of work entailed in doing this manually dictates that automated systems be used when possible. This late start in the field of computerization has caused many problems for hospitals. It is not that the problems are any different than those experienced in other businesses, but that hospital administrators are generally less experienced in dealing with those problems or even recognizing where the

problems might lie.

Perhaps for this reason, many automated HISs fail. Lack of management involvement, lack of user involvement in planning, lack of proper training, unrealistic claims for the systems, not taking advantage of all benefits of new systems, not planning for future expansion of information needs, and not adequately dealing with the impacts of a new system on the organization are other specific reasons why HISs do not perform as anticipated. Sensing the void in managing the implementation of information systems, vendors have stepped in to market their products.

It is much easier to have a vendor come in and tell the administrator what should be done and vendors are ready and willing to accept the resposibility abdicated by the hospital. While many products can be very useful to a hospital, the lack of involvement in managing the process almost always results in less than adequate systems.

This thesis has been written to provide basic information about the development life cycle of an information system so administrators will be more able to manage the process of implementing their Hospital Information System. By no means is this paper meant to be exhaustive, and, for the areas which an administrator feels he needs more comprehensive explanations, the references at the end of this paper are provided.

Recommendations

There are many barriers to the successful implementation of Hospital Information Systems. They are not so great though, that they cannot be successfully dealt with. The first step in managing the problems is knowing what they are and then learning ways to overcome them. This common-sense approach should be ample cause for the profession of Hospital Administration to insist that more academic coursework relating to the development of automated information systems be included in Health Care Administration undergraduate and graduate programs.

Resistance to the use of computers in hospitals could also be more adequately controlled. Almost every member of a hospital staff requires extensive training through colleges and universities. The great increase in the use of computers in hospitals should encourage these programs to include sufficient training in their use so that professionals would feel more comfortable when confronted with them in "real life."

While the most benefit can be derived from an information system developed specifically for an individual hospital, sometimes this is not feasible. The empahsis of this paper has been toward this total development approach, following all the steps of the Systems Development Life Cycle. For those hospitals not able to justify a personalized system, there are many "package" systems on the market.

Much benefit could therefore be gained from research done in analyzing the current canned HISs.

A recommendation to include more training in professional programs is a very broad one and the decisions of exactly what training and how much should not be done without much thought. Research in this area would be very useful.

Finally, the development of automated Hospital Information Systems is still in the infancy stages. While this paper was concerned with the management of the development process, research into the development of specific applications for hospitals is greatly needed.

LIST OF REFERENCES

1. IFIP WORKING CONFERENCE ON HOSPITAL INFORMATION
    SYSTEMS, Roger H. Shannon ed., <u>Hospital Information
    Systems: An International Perspective on Problems and
    Prospects</u>, (New York, N.Y.: North-Holland Publishing
    Company, 1979)

2. JOHN G. BURCH, JR., FELIX R. STRATER, AND GARY
    GRUDNITSKI, <u>Information Systems: Theory and Practice</u>,
    3rd ed. (New York: John Wiley & Sons, 1983)

3. MATS LUNDEBURG, GORAN GOLDKUHL, AND ANDERS NILSSON,
    <u>Information Systems Development: A Systematic Approach</u>
    (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1981)

4. JOHN G. BURCH, JR., FELIX R. STRATER, AND GARY
    GRUDNITSKI, <u>Information Systems: Theory and Practice</u>,
    2nd ed. (New York: John Wiley & Sons, 1979)

5. SHERMAN C. BLUMENTHAL, <u>Management Information Systems</u>
    (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1969)

6. MELVILLE H. HODGE, <u>Medical Information Systems: A
    Resource for Hospitals</u> (Germantown, Maryland: Aspen
    Systems Corporation, 1977)

7. JOHN ABBOTT WORTHLEY, <u>Managing Computers in Health
    Care: A Guide for Professionals</u> (Ann Arbor, Michigan:
    Aupha Press, 1982)

8. MARI MALVEY, <u>Simple Systems, Complex Environments:
    Hospital Financial Information Systems</u> (Managing
    Information: A Series of Books in Organization Studies
    and Decision-Making, Vol. 2) (Beverly Hills, CA: Sage
    Publications Inc., 1981)

9. OWEN DOYLE, CHARLES J. AUSTIN, AND STEPHEN L. TUCKER,
    <u>Analysis Manual for Hospital Information Systems</u> (Ann
    Arbor, Michigan: Aupha Press, 1980)

10. REGINA HERZLINGER, "Why Data Systems in Non-Profit
    Organizations Fail," <u>Harvard Business Review</u>, Jan-Feb
    1977, pp. 81-86 cited by CHARLES J. AUSTIN AND BARRY
    R. GREENE, "Hospital Information Systems: A Current
    Perspective," <u>Inquiry</u>, June 1978, p. 95

11. "The Medicare Squeeze Pushes Hospitals into the Infor-
    mation Age," <u>Business Week</u>, June 18, 1984, p. 87

12. CYNTHIA BARNARD, "Health Care Information System Issues Today," *Hospital Forum*, January/February 1984, p. 27

13. CHARLES J. AUSTIN, *Information Systems for Hospital Administration*, 2nd ed. (Ann Arbor, Michigan: Health Administration Press, 1983)

14. GERALD A. GIEBINK AND LEONARD L. HURST, *Computer Projects in Health Care* (Ann Arbor, Michigan: Health Administration Press, 1975)

15. JOSEPH S. MACIES, "Strategic information planning for the 1990's," *Health Care*, February 1984, p. 22

16. CONNIE ZWEIG, "Work in Progress," *Esquire*, June 1984, p. 291

17. LEE LAMPIRIS, "Developing a Case-Mix Strategy," *Southern Hospitals*, March/April 1984, p. 64

18. DONALD A. B. LINDBERG, *The Growth of Medical Information Sytoms in the United States* (Lexington, Massachusetts: Lexington Books, 1979)

19. JAMES C. WETHERBE, *Systems Analysis and Design: Traditional, Structured, and Advanced Concepts and Techniques*, 2nd ed. (St. Paul, Minnesota: West Publishing Co., 1984)

20. RANDALL W. JENSEN AND CHARLES C. TONIES, *Software Engineering* (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1979)

21. CHRIS GANE AND TRISH SARSON, *Structured Systems Analysis: Tools and Techniques* (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1979)

22. EDWARD YOURDON, *Managing the Structured Techniques*, 2nd ed. (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1979)

23. FREDERICK P. BROOKS, JR., *The Mythical Man-Month: Essays on Software Engineering* (Reading, MA: Addison Wesley Publishing Co., 1975)

24. C. WILLIAM DOTSON AND IRENE R. HINSON, R.N., "Hospital moves from totally manual to computerized information system in 7 years," *Hospitals*, October 16, 1981, p. 131

25. T. M. BOYLE AND DANIEL F. WALSH, "One medical center's

approach to the development of a hospital information system," *Hospitals*, October 16, 1981, p. 143

26. TOM DEMARCO, *Structured Analysis and System Specification* (Englewood Cliffs, N.J.: Prentice-Hall, 1979)

27. JOHN G. WILLIAMS, "The CEO Perspective on Information Systems Planning," *Hospital Forum*, January/February 1984, p. 33

28. EMILY FRIEDMAN, "Information Systems," *Hospitals*, May 1, 1982, p. 70

29. JEFFRY BEELER, "Proprietary Language Snags $3 Million Budget System," *Computerworld*, February 21, 1983, Vol. XVII, No. 8, p. 1

30. WAYNE STEVENS, *Using Structured Design: How to Make Programs Simple, Changeable, Flexible, and Reusable* (New York, N.Y.: John Wiley & Sons, 1981)

31. WILLIAM S. DAVIS, *Systems Analysis and Design: A Structured Approach* (Reading, MA: Addison-Wesley Publishing Co., 1983)

32. AMERICAN HOSPITAL ASSOCIATION, *Hospital Computer Systems Planning: Preparation of Request for Proposal* (Chicago, Illinois: American Hospital Association, 1980)

33. PETER FREEMAN AND ANTHONY I. WASSERMAN, ed. *Tutorial on Software Design Techniques*, 3rd ed. (New York, N.Y.: IEEE, 1980)

34. GLENFORD J. MEYERS, *The Art of Software Testing* (New York, N.Y.: John Wiley & Sons, 1979)

35. VICTOR R. BASILI AND F. TERRY BAKER, ed. *Tutorial on Structured Programming: Integrated Practices* (New York, N.Y.: IEEE, 1981)

36. JOAN K. HUGHES AND JAY I. MICHTOM, *A Structured Approach to Programming* (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1977)

37. C. BOHM AND G. JACOPINA, "Flow Diagrams, Turing Machines, and Languages With Only Two Formation Rules," *Communications of the ACM*, Vol. 9, No. 5 (May 1966), pp. 366-371

38. CLEMENT L. MCGOWAN AND JOHN R. KELLY, *Top-Down Structured Programming Techniques* (New York, N.Y.:

Mason/Charter Publishers, Inc., 1975)

39. DOUGLAS E. HAGER, "Computers Affect Hospital
     Organization, Staff, Patients," Hospital
     Administration Currents, October-December 1977, pp.
     27-30 cited by JOHN ABBOTT WORTHLEY, Managing
     Computers in Health Care: A Guide for Professionals
     (Ann Arbor, Michigan: Aupha Press, 1982)

40. MARVIN V. ZELKOWITZ, ALAN C. SHAW, AND JOHN D. GANNON,
     Principles of Software Engineering and Design (Engle-
     wood Cliffs, N.J.: Prentice-Hall, Inc., 1979)

VITA

Larry L. Cobler was born October 6, 1950, in Richland, Washington, the son of Charles L. and Marilyn Cobler. Prior to entering active duty in the United States Air Force, he attended the University of California at Santa Barbara and San Jose State University. In 1973 he enlisted in the Air Force as a Pharmacy Technician and taught in that career field for approximately five years at the USAF School of Health Care Sciences. During that time he completed the degree of Bachelor of Business Adminstration at Midwestern State University, Wichita Falls, Texas. In 1978 he was commissioned a 2nd Lieutenant in the United States Air Force and until 1983 served in the Medical Service Corps, performing various hospital administration functions at the USAF Hospital, Laughlin AFB, Del Rio, Texas. In May 1983 he began work on a Master of Science degree in Computer Science at Trinity University, San Antonio, Texas. After award of the degree in December 1984, Captain Cobler was assigned as chief of the Microcomputer Branch of the Medical Computer Systems Division at the USAF School of Health Care Sciences at Sheppard AFB.