

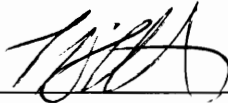
Decision Support Systems Design: A Nursing Scheduling Application

by

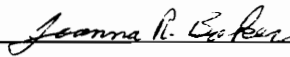
Wendy A. Ceccucci

Submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy  
in  
Management Science

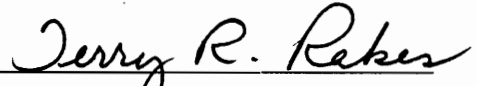
APPROVED:



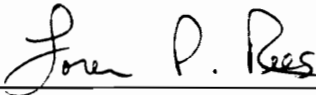
Robert T. Sumichrast, Chairman



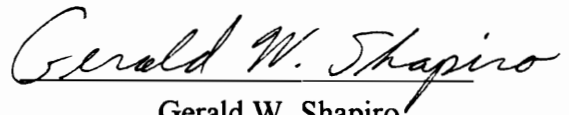
Joanna R. Baker



Terry R. Rakes



Loren P. Rees



Gerald W. Shapiro

January 28, 1994  
Blacksburg, Virginia

# Decision Support Systems Design: A Nursing Scheduling Application

by

Wendy A. Ceccucci

Robert T. Sumichrast, Chairman

Management Science

(ABSTRACT)

The systems development life cycle (SDLC) has been the traditional method of decision support systems design. However, in the last decade several methodologies have been introduced to address the limitations arising in the use of the traditional method. These approaches include Courban's iterative design, Keen's adaptive design, prototyping and a number of mixed methodologies incorporating prototyping into the SDLC.

Each of the previously established design methodologies has a number of differing characteristics that make each of them a more suitable strategy for certain environments. However, in some environments the current methodologies present certain limitations or unnecessary expenditures. These limitations suggest the need for an alternative methodology. This dissertation develops a new methodology, priority design, to meet this need.

To determine what methodology would be most effective in a given situation, an analysis of the operating environment must be performed. Such issues as project complexity, project uncertainty, and limited user involvement must be addressed. This dissertation develops a set of guidelines to assist in this analysis. For clarity, the guidelines are applied to three, well-documented case studies.

As an application of the priority design methodology, a decision support system for nurse scheduling is developed. The development of a useful DSS for nurse scheduling

requires that projected staff requirements and issues of both coverage and differential assignment of personnel be addressed.

## **Dedication**

Dedicated, with love to my parents,  
Rudolph and Stasia Ceccucci, and  
my sister, Cynthia McGrath.

# Acknowledgements

I would like to thank my chairman and advisor, Dr. Robert T. Sumichrast for all of his patience, guidance and support throughout my graduate studies.

I would also like to thank Dr. Joanna Baker, Dr. Terry Rakes, Dr. Loren Rees, Dr. Gerald Shapiro, for their advise and for their support as members on my dissertation committee. I would further like to extend my thanks to Dr. Lance Matheson for standing in on my dissertation defense.

Finally, I would like to thank, with all of my love, both my parents and my sister. Without their loving support, I would have never completed this dissertation.

# Table of Contents

<b>Chapter 1: Introduction</b> .....	<b>1</b>
Statement of the Problem .....	3
Objectives of the Study .....	6
Methodology.....	6
Scope and Limitations .....	7
Plan of Presentation .....	8
<b>Chapter 2: Nurse Scheduling Literature Review</b> .....	<b>10</b>
Introduction .....	10
The Nursing Environment .....	11
Scheduling Criteria.....	12
The Traditional Approach to Nurse Scheduling .....	13
Cyclical Scheduling .....	15
Computer-Based Nurse Scheduling Algorithms .....	17
Warner's Mathematical Approach .....	17
Miller, Pierskalla and Rath's Mathematical Programming Approach .....	20
Arthur and Ravindran's Multiple Objective Scheduling Model .....	23
Musa and Saxena's Goal Programming Approach .....	27
Conclusions.....	29
<b>Chapter 3: Decision Support Systems Design</b> .....	<b>31</b>
Introduction .....	31
Decision Support Systems.....	32

Design Approaches .....	33
Traditional Systems Development Approach.....	33
Iterative Design .....	36
Keen's Adaptive Design .....	39
Prototyping .....	43
Mixed Methodologies.....	48
Guidelines for Constructing a Decision Support System .....	49
The ROMC Approach .....	49
Kroeber's Guidelines.....	50
Keen and Gambino's Guidelines for Constructing a DSS .....	48
Priority Design Methodology .....	54
Introduction.....	54
The Steps to the Priority Design Methodology.....	55
Benefits of the Priority System.....	58
The Selection of a Design Strategy.....	60
Conclusion.....	67
<b>Chapter 4: Case Study Analysis.....</b>	<b>69</b>
Introduction.....	69
Case #1. A DSS for Tuition and Fee Policy Analysis .....	70
Introduction.....	70
The Design Methodology .....	71
Analysis.....	73
The Tuition and Fees Operating Environment.....	73
Priority Design.....	75

Conclusion .....	77
Case #2 The Eastern Manufacturing Company .....	78
Introduction.....	78
The Design Methodology .....	79
Analysis.....	79
Eastern's Operating Environment.....	79
Priority Design.....	81
Conclusion .....	83
Case #3: Decision Support at Conrail.....	84
Introduction.....	84
The Design Methodology .....	85
Analysis.....	87
Conrail's Operating Environment.....	87
Priority Design.....	89
Conclusion .....	92
Summary.....	93
<b>Chapter 5: The DSS for Nurse Scheduling.....</b>	<b>94</b>
Introduction .....	94
Montgomery Regional Hospital.....	95
Nursing Classifications for Montgomery Regional Hospital.....	95
The Baylor Plan .....	96
PRN Staffing.....	96
LPN and RN.....	97
The Intensive Care Unit.....	97



The Scheduling Policies .....	98
The Development Process .....	101
The Initial Meeting .....	101
Montgomery Regional Hospital's Environment.....	104
The Priority Design Methodology .....	107
Stage 1. Determining the Minimum Requirements for the Partial System .....	107
Stage 2. The Simulation .....	110
Stage 3. Construction of the User's Priority List .....	114
Stages 4 and 5. Developer Suggestions and User Reevaluation .....	114
Stage 6. Partial System Development .....	117
Hardware & Software.....	117
A Description of the DSS .....	119
Development Problems and Limitations .....	125
Stage 7. System Implementation.....	125
Analysis of the Development Process .....	129
Conclusion.....	131
<b>Chapter 6: Conclusion.....</b>	<b>132</b>
Introduction.....	132
Summary of Research Results .....	132
Priority Design .....	133
Guidelines for Selecting a Design Methodology .....	134
The Nurse Scheduling DSS.....	135

<b>Bibliography .....</b>	<b>137</b>
<b>Vita.....</b>	<b>141</b>

# List of Illustrations

Figure 2.1 Block Scheduling .....	14
Figure 2.2 Cyclical Scheduling.....	16
Figure 3.1 The Traditional Systems Development Life Cycle.....	34
Figure 3.2 User Involvement in the Traditional Systems Development Life Cycle.....	37
Figure 3.3 Keen's DSS Design Cycle.....	40
Figure 3.4 Keen's Predesign Cycle .....	41
Figure 3.5 Prototyping.....	45
Figure 3.6 A State-Transition Model for Application Development and Prototyping .....	47
Figure 3.7 Priority Design.....	56
Figure 3.8 Guidelines for Design Methodology Selection .....	62
Figure 4.1 Design Steps in the Development of the Tuition and Fees DSS.....	72
Figure 4.2 Design Steps in the Development of Conrail's DSS.....	86
Figure 4.3 Conrail's Interactive Computing Software .....	88
Figure 5.1 Critical Care Organizational Chart.....	99
Figure 5.2 Early List of Requirements.....	108
Figure 5.3 Montgomery Regional Hospital's Work Schedule.....	109
Figure 5.4 The Schedule and Edit Windows.....	113
Figure 5.5 Priority List.....	115
Figure 5.6 The Schedule with the Command Window .....	121
Figure 5.7 The Schedule with the Selected Menu Feature.....	122
Figure 5.8 The Nurse Scheduling DSS Handout.....	127-128

# List of Tables

Table 2.1 Special Requests .....	24
Table 4.1 Project Uncertainty .....	75
Table 4.2 Design Methods .....	93
Table 5.1 Vacation Schedule.....	101
Table 5.2 Environmental Characteristics for Determining an Appropriate Design Methodology .....	106

# Chapter 1: Introduction

Decision support systems have become an integral part of today's businesses. These systems have a wide range of business applications, including finance, marketing, accounting and production. With the growing popularity of these systems, at all levels of the business hierarchy, increased attention has been focused on the design methodologies for decision support systems. In the past, the systems development life cycle (SDLC) has been the conventional method for development. However, in the last decade several approaches have emerged which have increased the level of communication between the user and the developer. The developer is now confronted with the problem of determining the most appropriate design methodology. To determine which design approach would be most appropriate for a given project, its individual characteristics and operating environment need to be compared and contrasted with the benefits and limitations of each of the various design approaches.

This research reviews previously published approaches, proposes an alternative design methodology, called priority design, and develops a set of guidelines for recommending an appropriate design strategy based upon the user's operating environment. To demonstrate the proposed design methodology, a significant, practical semi-structured problem will be addressed: the problem of determining the work schedules for hospital nurses.

The application proposed, as a part of this research, focuses on the issue of scheduling in a system with variable demand and requiring specialized nursing services. This part of the decade had witnessed a drastic decline in the availability of nursing personnel. This research investigates the use of a decision support system (DSS) as a practical aid to scheduling. The process for carrying out this investigation entails developing and implementing a DSS, for a local hospital. The DSS environment provides some unique characteristics from the standpoint of scheduling and DSS development, such as stochastic demand, unreliable availability, variable skill requirements and schedule preferences. The development of such a useful DSS for nurse scheduling requires that the issues of coverage and differential assignment of personnel be addressed, given projected staff requirements.

In determining a schedule, management must resolve the conflicting goals between the hospital and the nursing staff. The hospital requires a schedule that maintains a high level of care while keeping costs at a minimum. The nurses on the other hand, desire a favorable on-off sequence of workdays and a balanced workload. In addition, the nurses have varying shift and case preferences that must be taken into account when developing a schedule. Several approaches have been used, including simulation and mathematical programming, to address the problem of scheduling of health care personnel. However, these approaches have lacked the flexibility needed to accommodate the dynamic nature of

the scheduling environment and to provide immediate feedback for alternative scheduling scenarios; a flexibility that may be supported by a specific DSS.

### ***Statement of the Problem***

To date, there are five primary methods of DSS design: the traditional systems development life cycle, Courban's iterative design, Keen's adaptive design, prototyping, and a mixed methodology incorporating prototyping into the SDLC. The majority of the approaches that have been applied to the development of a DSS have been adapted from common design strategies. These strategies, with the exception of Keen's adaptive design, were not developed to specifically deal with the unique environment of a DSS. As a result, these approaches have varying limitations when applied to the DSS environment. They do not provide the user with a formal method of communicating to the developer the features that are most important to the success of the project and the features that are considered auxiliary items, that the user would eventually like to see in the completion of the DSS. These approaches, also, do not provide a structured method for the developer to communicate to the user the amount of time and cost required for the development of the various features requested by the user. In a complex environment, where the problem is not well-defined, a discussion of these concepts between the user and the developer becomes an important and integral part of the development process. The limitations in the

various design strategies can have a serious affect on the successful development of the DSS. These limitations can result in:

- the development of a DSS that does not meet the needs of the user,
- unnecessary expenditures.
- excessive development times

These limitations suggest the need for an alternative methodology.

Due to the growing popularity of decision support systems, increased importance has been placed on the selection of an appropriate design methodology. Depending on the project's characteristics, a high degree of communication between the user and the developer is not always necessary. Before the development process begins, the developer needs to select a development methodology that complements its operating environment. It is necessary for the developer, when selecting an appropriate design methodology, to evaluate the user's operating environment and the project's characteristics before proceeding. By evaluating the user's environment, an efficient methodology that best complements the user's project characteristics can be determined.

To test the alternative methodology and the proposed selection guidelines the problem of nurse scheduling was selected. The development of a nurse schedule is a semi-structured task that often requires a large amount of management's time and is often met with dissatisfaction by the corresponding nurses. The task entails generating a schedule that meets the hospital's and nurses' requirements, and maintaining concise records and updated summary reports on both the individual employees and on the actual schedule worked. Maintaining accurate records is important to the hospital, not only for payroll purposes, but also for potential liability cases. The hospital must have records showing who was responsible for the patients if a law suit ensues.



Much of the nurse scheduling literature has focused primarily on developing schedules for large hospitals, through the use of mathematical models. However these models have limited usage. The solution to the majority of the models determines the days the nurses are to work and then the specific shifts that the nurse work are determined through a second procedure. Using separate procedures assures suboptimal solutions. The models, when determining the days, do not account for the effect different shifts patterns may play on the day schedules. For example, a nurse would not work a night shift and then the next day shift. The shift effects are not independent factors and should be considered when the schedule is determined.

A DSS can provide the flexibility that is needed to support the nurse manager. It will allow the nurse manager to offer a variety of shift lengths, which is not currently practical with the use of a mathematical programming solution. A DSS will also provide the nurse manager a user friendly atmosphere to develop alternative schedules and maintain statistics on the nurses' work history. By maintaining statistics on past schedules, such as number of weekends, holidays, vacations, etc., the nurse manager will be able to make better informed decisions in regard to future schedules. The DSS will help in ensuring that the vacations, weekends and holidays are distributed in an equitable manner. In addition, a DSS will provide statistics on the number of nurses worked during the various shifts. A DSS will be able to provide the necessary flexibility to accommodate all of these different issues.

## ***Objectives of the Study***

There are three main objectives of this study. The first objective of the study is to address some of the limitations of the current DSS design methodologies by developing an alternative methodology. The proposed alternative methodology would incorporate some of the benefits of the various methodologies and expand on the methodologies, to strengthen their weaknesses.

The most appropriate design methodology is dependent on the characteristics of the project and its operating environment. The second objective of the study is to develop a selection guideline to determine the most suitable design strategy.

The final objective of the study is to employ the proposed design methodology to the problem of nurse scheduling at Montgomery Regional Hospital. The proposed DSS would be developed to assist the nurse manager in the scheduling of nurses in the Intensive Care Unit.

## ***Methodology***

The first step in developing an alternative technique is to evaluate existing design strategies. When studying the strategies the weakest areas of the strategies need to be noted. An alternative strategy is developed so that it compliments the existing strategies i.e. it's strength is in the areas where the previous strategies are weakest.

The development of a DSS can address a wide range of different problems and may occur in many different types of environments. For example the problem may be highly complex, requiring a large amount of new information, or it may be a small system

with very low complexity. The environment may be uncertain, the user unclear as to the requirements of the system or the task may be well-defined. Each design strategy, including the proposed methodology is more suitable for environments with certain characteristics. To determine which methodology is most appropriate for the development of a specific DSS, a set of guidelines is created. These guidelines aid in evaluating the characteristics of the project and its corresponding environment, to determine the most appropriate design methodology.

To demonstrate the proposed design methodology, the strategy is applied to a specific problem. The selected problem was that of developing a DSS for nurse scheduling. The nurse scheduling DSS provides a mechanism to further articulate the steps of the proposed methodology and provides a platform for testing the strategy.

To evaluate the guidelines for selecting an appropriate design methodology, three DSS development cases, with different operating environments are used as test cases. In each case, the operating environment and the steps employed in the development process are studied. Following the study, the proposed guidelines are used to recommend a more suitable strategy based on the case's operating environment and its unique characteristics. An analysis followed, comparing and contrasting the situation had the recommended design approach been employed.

## ***Scope and Limitations***

As previously stated in the objectives, this research is intended to evaluate the issues of designing a decision support system. In evaluating the DSS, a prototype, interactive, scheduling system will be constructed. This system enabled the hospital to

develop an efficient nurse schedule and provided the capability for the manager to generate alternative scenarios. The DSS did not provide a mechanism for recommending and developing schedules. But, the DSS was intended to assist the hospital in this process. This research required data from past records and knowledge on the various nurses, which was acquired from Montgomery Regional Hospital. Since the information gathered was from the local hospital, the system was designed to meet its specific needs and to follow its current scheduling notation.

### ***Plan of Presentation***

Chapter 2, entitled Nurse Scheduling Literature Review, contains a review of prior literature pertinent to the area of nurse scheduling. First, the importance of nurse scheduling is presented. This is followed by a discussion of the problems and the different criterion that must be considered in determining an effective schedule. Finally, a review of the different approaches to nurse scheduling is presented.

Chapter 3, entitled Decision Support Systems Development, contains a discussion of the limitations and assumptions of the various methods of systems development and proposes an alternative methodology. In addition, a guided search is recommended for the selection of a development methodology based on the project characteristics and the operating environment.

Chapter 4, entitled Case Study Analysis, contains a review and analysis of the design approach used in three documented DSS development cases. For each of the three cases, a synopsis of the steps employed in the development process is given. The cases are then evaluated, as proposed in Chapter 3, to determine which methodology would

have been most appropriate, and what advantages might have been gained had a different methodology been employed.

Chapter 5, entitled The DSS for Nurse Scheduling, gives an overview of the development methodology used in the construction of a nurse scheduling DSS for Montgomery Regional Hospital. The chapter opens with a discussion of the dynamic nature of the environment and the constraining factors which influence the hospital's decision making. This is followed by a detailed account of the DSS and the development process.

Chapter 6, entitled Summary and Conclusions, provides a summary of the research results and a discussion of the limitations and possible enhancements to the system.

## **Chapter 2: Nurse Scheduling Literature Review**

### ***Introduction***

The beginning section of this chapter explains the importance of nurse scheduling as a means of better utilizing a valuable resource, namely nurses. This is followed by a discussion of the factors that influence the scheduling task. The second section of this chapter focuses on the different approaches that have been presented in the literature to solve the scheduling problem. A review of each method is followed by an analysis of the limitations and the shortcomings of each of the different approaches.

## *The Nursing Environment*

The increasing cost of health care and the shortage of nurses have become critical issues in the United States today. The shortage of nurses can be attributed to both increased demand for nursing services and diminished supply (Easton, et al., 1992). There are a number of reasons for the increased level of demand for nurses. One reason explained by Dr. Fitzpatrick, Dean of the Bolton School, is that "advances in medical technology enable people to live longer. And because hospital patients are sicker now than ever before, the demand for nurses is greater," (Pross, 1991). This can be supported by the number of nurses that are now required in acute care hospitals. In 1986, acute care hospitals required an average of 91 nurses per 100 inpatients, up from 50 per 100 in 1972 (Moran, 1988). Another reason for the increased demand for nurses is the proliferation of outpatient clinics, along with home care diversification strategies (Easton, 1992).

The supply of nurses entering the health care profession has also declined. Aiken and Millinex expect the number of new nurses graduating annually to decrease from 82,700 in 1985 to 68,700 by 1995 (Aiken and Mullinex, 1987). Another issue that compounds this problem is the approaching retirement age of many of the licensed nurses (Curran, et al., 1989).

In order to more efficiently utilize the available resources, a flexible and effective nurse scheduling system needs to be developed. The potential benefits from an effective scheduling system are innumerable. Currently, the nurse manager and the clerical staff spend a large number of hours determining a proper nursing schedule. This administrative function is one of the most expensive for management in terms of time and money (Eusanio, 1978). In addition to improving the nurses' time management, an effective

schedule has several indirect benefits. Nursing turnover and absenteeism rates are lowered when the nurses are satisfied with the scheduling system (Melbin, 1968). The quality of patient care is also increased by a scheduling system that maintains a continual level of care, rather than a system that just fills in the gaps. In addition, by maintaining a continuous staffing level, nurses become more acquainted with one another and a greater level of communication develops, resulting in an effective team atmosphere (Eusanio, 1978).

### ***Scheduling Criteria***

A high quality schedule must weigh the goals of both the hospital and the nursing staff. In order to evaluate a scheduling system several criteria should be addressed. These criteria include the following:

- Coverage - The schedule must meet the minimum staffing requirements and provide "even" coverage.
- Schedule Quality - The schedule should meet the nurses' desires in terms of various scheduling patterns, including work stretch length, days-off patterns and requests for days-off
- Stability - The policy should be consistent and enable the nurses to be able to predict their future work schedule.
- Flexibility - The system must be able to adapt to changes in the work environment. The schedule should be flexible enough to adjust to changes in the nurses' shift preferences, i.e. full-time vs. part-time, day vs. night.



- Cost - This criterion is two-fold. First, the cost in the development of a schedule including typing time, management's time, and computer time should be at a minimum. Second, the cost in terms of the utilization of the available nurses, taking into account that the various nursing levels have different levels of cost associated with them, must also be kept low (Warner, 1976).
- Fairness - The "bad" shifts should be distributed in some equitable manner.

### ***The Traditional Approach to Nurse Scheduling***

Currently, there are three primary methods of nurse scheduling, they include: (1) the traditional approach or block scheduling (2) cyclical scheduling, and (3) computer scheduling (Stevens, 1985). Each of these represents an improvement on the preceding method. But, all three methods are used today. Although the traditional approach is the most frequently used, it is the most time consuming strategy. It involves developing from scratch a schedule for each planning horizon or block, usually four to eight weeks, thus allowing the nurse manager the flexibility to adapt to any changes in the working environment. The term "block" originates from the fact that the days to be worked were often blocked together to form specific patterns, such as the one given in Figure 2.1 (Stevens, 1985). However, the disadvantages to this system are numerous; not only does it take a considerable amount of time, but it is often very difficult for the manager to develop a schedule that is considered fair and consistent to all the participating nurses. As a result, the relationships between the nurses and the nurse manager often become strained and the manager becomes the "bad guy" if he refuses requests for days-off (Gahan and

Week	RN	M	T	W	Th	F	Sa	Su
1	A			X	X	X	X	X
	B	X	X		X	X	X	X
	C	X	X	X	X			X

Week	RN	M	T	W	Th	F	Sa	Su
2	A	X	X		X	X	X	X
	B	X	X	X	X			X
	C			X	X	X	X	X

X = days worked in the week

### Figure 2.1: Block Scheduling

Source: Stevens, Barbara, *The Nurse as Executive*, 3rd Edition, Aspen Publishers Inc., Rockville, MD, 1985, pg. 128.

Talley, 1975). Another disadvantage is the lack of consistency in the schedule. The system does not provide a mechanism that would allow the nurses to be able to predict what days they would be working in the next planning horizon.

## ***Cyclical Scheduling***

The second approach, cyclical scheduling, is an improvement on block scheduling in that it attempts to alleviate the disadvantage given above by creating repetitive work patterns for the nurses. Since each nurse has a permanent work pattern he can determine in advance when he will be on duty. A cyclical schedule consists of repeated patterns of interwoven schedules. These interwoven schedules are a part of a permanent plan, or a fixed schedule of usually four to six weeks. A nurse may have a different schedule for each of the weeks contained in the cycle, but the individual nurses have their own schedule pattern that repeats without change. An example schedule using the cyclical pattern, adapted from Rowland is given in Figure 2.2 (Rowland, 1985). The advantages of the cyclical schedule, once it has been established include the following:

- A drastic reduction in the amount of time the nurse manager spends developing a schedule. This frees up the manager's time for more personal patient care.
- Favored days-on and off are distributed fairly among all the nurses.
- Individual schedules are known in advance by the nurses, which allows the nurses to plan their personal activities on the given days-off (Eusanio, 1978).

A Three-Week Schedule for Nursing Unit Staff:

Nurse	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
A	X		X	X	X	X			X	X	X			X	X	X			X	X	X
B		X	X	X			X	X	X			X	X	X	X		X	X	X	X	
C	X	X			X	X	X	X		X	X	X	X			X	X	X			X

A Six-Week Schedule for Nursing Unit Staff:

Nurse	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
A	X	X			X	X	X	X		X			X	X	X	X	X			X	X
B	X	X	X	X		X	X	X	X	X	X	X			X	X	X	X	X		
C			X	X	X				X	X	X	X	X	X			X	X	X	X	X

Nurse	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
A	X	X	X	X	X				X	X	X	X	X	X	X		X	X	X		
B	X	X	X			X	X	X	X	X			X	X	X	X	X			X	X
C			X	X	X	X	X	X	X		X	X				X	X	X	X	X	X

Figure 2.2: Cyclical Scheduling

Source: Rowland, Howard S. and Beatrice L., *The Nursing Administration Handbook*, 2nd Edition, Aspen Publishers, Inc., Rockville, MD, 1985, pg168.

However, a crucial disadvantage to this system is that it lacks the flexibility to adapt to changes in the environment. This was shown in a survey done by Warner indicating that out of 13 cyclical scheduling systems, only one survived longer than a year (Warner, 1976).

### ***Computer-Based Nurse Scheduling Algorithms***

The third approach, computer scheduling, has received a lot of attention in the last fifteen years. Many of the new approaches have focused on solving the scheduling problem by the use of a heuristic or a mathematical formulation. This section reviews several of the methodologies that have been presented in the literature. Although, this is not an exhaustive coverage of the literature, it is a representative one (Easton, et al., 1992).

#### **Warner's Mathematical Approach**

One of the earlier scheduling systems, a mathematical programming formulation developed by Warner, consists of a modification of an algorithm proposed by Balintfy and Blackburn to configure a schedule based upon nursing preferences (Warner, 1975). The system can be broken into two phases, a policy definition phase and a mathematical programming phase.

The first phase, policy definition, is used to determine the parameters and variables used in the programming model. These parameters and variables set by the nurse manager include the following:

- coverage - the minimum number of nurses required for each shift.
- weekend - the number of weekends-off per cycle per nurse.
- rotation - each nurse is classified as "permanent shift" or "rotating shift". If the nurse rotates, the manager must also set the shifts onto which he will rotate (day to evening, day to night, or both).
- schedule quality evaluation - Each nurse rates his aversion to certain scheduling patterns. For example, a nurse would indicate that he is four times averse to a "single day on" as he is to an eight-day stretch.
- restrictions - Any special restrictions, such as requiring a nurse to work a specific shift.
- requests for days-off - the requests for days-off or vacations are handled in one of three ways:
  - (1) The nurse manager grants or denies the request.
  - (2) The nurse manager weights each request; the higher the weight the more likely the nurse will be granted the request.
  - (3) The weighting in method (2) is done by the nurses themselves.

Once the first phase is complete, the second phase is presented as a multiple choice programming problem, where each nurse defines a category of variables and each variable within a category is a potential schedule for the nurse. The program then generates the alternative schedules with a planning horizon of 14 days and allowing up to three categories of nurses, e.g. LPN, RN. The model is constructed as follows:

**Variables and Parameters:**

$i = 1, 2, \dots, L$  the subscript indicating the nurse to be scheduled, the first set, from 1 to R are the RNs and the second set, from R+1 to L are the LPNs.

$j = 1, 2, \dots, J_i$  the subscript used to index the potential 14-day schedules for each nurse, with  $i$  representing the nurse.

$A_{ij}$  - a 0-1 vector of length 42 (14 days time 3 shifts) indicating the  $j$ th possible schedule for nurse  $i$ . An element in the vector with a value of a 1 indicates the nurse will work the shift, a 0 indicates that he will not work the shift.

$$X_{ij} = \begin{cases} 1 & \text{if nurse } i \text{ is to work schedule } A_{ij} \\ 0 & \text{if not} \end{cases}$$

$C_{ij}$  - the relative value of schedule  $j$  to nurse  $i$

$k$  - index used to represent the 42 shift combinations in the scheduling period

$E_k$  - value of having an extra RN on shift  $k$

$F_k$  - maximum additional RN coverage for shift  $k$

$s_k$  - slack variable associated with the  $k$ th RN minimum coverage constraint

**RN** - a vector of length 42 representing the minimum coverage of RNs.

**TOT** - a similar vector for total coverage (**RN** + **LPN**)

$$g(s_k) = \begin{cases} 0 & \text{if } s_k = 0 & \text{- Optimal assignment} \\ E_k & \text{if } 0 < s_k \leq F_k & \text{of slack (extra nurses)} \\ E_k - (s_k - F_k) & \text{if } s_k > F_k \end{cases}$$

$t_k$  - slack variable associated with the  $k$ th minimum coverage **TOT** constraint

$g(t_k)$  - the same manner as  $g(s_k)$  using values from the **TOT** coverage

$t$  - a vector of length 42 of which  $t_k$  is the  $k$ th element

$s$  - a vector of length 42 of which  $s_k$  is the  $k$ th element

Objective Function:

$$\text{Maximize } \sum_{i=1}^L \sum_{j=1}^{J_i} C_{ij} X_{ij} + \sum_{k=1}^{42} [g(s_k) + g(t_k)]$$

Constraints:

$$\sum_{i=1}^R \sum_{j=1}^{J_i} X_{ij} \mathbf{A}_{ij} - \mathbf{s} = \mathbf{RN} \quad (\text{minimum coverage of RNs})$$

$$\sum_{i=1}^L \sum_{j=1}^{J_i} X_{ij} \mathbf{A}_{ij} - \mathbf{t} = \mathbf{TOT} \quad (\text{minimum total coverage})$$

$$\sum_{j=1}^{J_i} X_{ij} = 1, \quad i=1,2,\dots,L, \quad (\text{nurse works exactly 1 schedule})$$

$$X_{ij} = (0 \text{ or } 1), \quad i=1,2,\dots,L; \quad j=1,2,\dots,J_i$$

The model was successfully employed at the University of Michigan Hospital and as a result, the average time to make a six-week schedule had been decreased from 18-24 hours to 1 hour (including 40-80 seconds of CPU time on a IBM 386/67).

### **Miller, Pierskalla and Rath's Mathematical Programming Approach**

Miller, Pierskalla and Rath used a mathematical approach, similar to the approach by Warner, to solve the nurse scheduling problem. They divided the constraints into two classes: a feasibility set, which defined the set of feasible nurse schedules, and a nonbinding set, whose violation incurs a variety penalty costs. Although, Miller et al. did not indicate the mathematical formulations of the constraints, they did give a description of each set of constraints. The feasibility set,  $\pi_i$ , was described as follows:



- A nurse works ten days every pay period
- No work stretches are allowed in excess of  $\sigma$  days
- No work stretches of  $\tau$  or fewer days are allowed
- Satisfy as many as possible, of the constraints we would like to see binding, given the nurse's special requests

The indexed set of nonbinding constraints,  $N_i$  included:

- No work stretches longer than  $S_i$  days (where  $S_i \leq \sigma$ )
- No work stretches shorter than  $T_i$  days (where  $T_i \geq \tau$ )
- No day-on, day-off, day-on patterns (101 patterns)
- No more than  $k$  consecutive 101 patterns
- $Q_i$  weekends off every scheduling period
- No more than  $W_i$  consecutive weekends working each scheduling period;
- No patterns containing four consecutive days-off;
- No patterns containing split weekends (i.e., a Saturday-on, Sunday-off pattern)
- the number of nurses scheduled to work on day  $k$  is greater than or equal to  $m_k$ , the minimum staffing level
- the number of nurses scheduled to work on day  $k$  is equal to  $d_k$ , the desired staffing level for day  $k$

The following steps describe the solution algorithm that finds the near optimal solutions:

1. Determine the set of feasible schedules for each employee,  $\pi_i$ . Let  $\|\pi_i\|$  denote the number of schedules in  $\pi_i$ .
2. Calculate the schedule pattern costs for each schedule  $x_i \in \pi_i$ , for  $i=1, \dots, I$ , where the cost is calculated as follows:

Define:

$g_{in}(x^i)$  = the cost of violating nonbinding constraint  $n \in N_i$  of schedule  $x^i$ ;

$\alpha_{in}$  = the "weight" nurse  $i$  gives a violation of nonbinding constraint  $n \in N_i$ .

$A_i$  - the aversion index of nurse  $i$ , i.e., a measure of how good or bad nurse  $i$ 's schedules have been historically in regard to preferences.

The total schedule pattern cost to nurse  $i$  for a schedule pattern  $x^i$  is:

$$A_i \sum_{n \in N_i} \alpha_{in} g_{in}(x^i)$$

3. Choose an initial schedule mix (i.e., a schedule for each nurse  $i=1..I$ ) and let BEST=its cost (e.g., choose the lowest cost schedule from each  $\pi_i$ ).
4. Let  $i = 1$ ,  $K = \|\pi_i\|$ ,  $k = 1$ , and cycle = 0.
5. Try the  $k$ th candidate schedule,  $x^{ik}$ , in the schedule mix by temporarily removing the present schedule for nurse  $i$  from the current schedule mix and inserting  $x^{ik}$ . Let TEST=the cost of this new schedule.
6. If TEST<BEST, go to step 8.
7. Let  $k=k+1$ . If  $k=K+1$  go to step 9. Otherwise, go to Step 5.
8. Let CYCLE=0 and BEST=TEST. Insert  $x^{ik}$  in place of the current schedule for nurse  $i$  in the schedule mix. The schedule mix now contains the best schedules found so far. Go to step 7.
9. If CYCLE=I, stop. Otherwise, let  $i=i+1$  (if  $i>I$ , let  $i=1$ ) and let  $K=\|\pi_i\|$ ,  $k=1$ , and CYCLE= CYCLE+1. Go to step 5.

The schedules generated by the algorithm were compared to actual schedules used by the hospitals. The results were favorable. The algorithm generated 13 more three and two day weekends and far fewer split weekends than the hospital. In addition, the

deviation of the desired staffing level from the actual level was as small or smaller than the hospital's schedule. This approach was employed at a number of hospitals in the United States and Canada with successful results.

### **Arthur and Ravindran's Multiple Objective Scheduling Model**

Arthur and Ravindran viewed scheduling as a multiple objective problem and implemented a two-phase goal programming approach to solve it (Arthur and Ravindran, 1981). Before the first phase begins, the weekend shifts are assigned based upon the nurse's individual preferences and hospital policy. The first phase then assigns the nurses their days-on and off schedules with a goal programming model. The objective function prioritizes the following four goals: (1) minimum staffing requirements, (2) desired staffing requirements (3) nurses' preferences, and (4) nurses' special requests. The results of phase one indicate what days the nurses should work but not which shifts. The second phase uses a heuristic procedure based upon the nurses' shift requirements to determine the overall schedule. To better, illustrate the model, an example of the goal programming model for a unit of four nurses and a planning horizon of one week is given below.

Variables and Parameters:

$$X_{ij} = \begin{cases} 1 & \text{if nurse } i \text{ is to work the } j\text{th day} \\ 0 & \text{if not} \end{cases}$$

$d_k^+$  - the positive deviation corresponding to the kth constraint

$d_k^-$  - the negative deviation corresponding to the kth constraint

$a_i$  - represents the number of points that RN number  $i$  assigned to the day-off/day-on/day-off schedule pattern on the individual preference form

$m_j$  - represents the total minimal RN requirement for all three shifts on day  $j$

$n_j$  - represents the total number of RNs desired for all three shifts on day  $j$

$P_k$  - the weight of the priority given to the  $k$ th constraint

$N$  - represents the total number of nurses

$s_j$  - represents the shortage cost assigned to day  $j$

Assume the following special requests and weekend assignments given in Table 2.1.

Table 2.1 Special Requests:

RN Number	Special Requests		Weekend Requests	
	Day	Request	Sunday	Saturday
1	F	Off	On	Off
2	F	On	Off	On
3	W	Off	Off	On
4			On	Off

Objective Function:

$$\begin{aligned} \text{MIN } Z = & P_1 \left( \sum_{j=1}^5 s_j d_j^- \right) + P_2 (d_6^- + d_7^- + d_8^-) + P_3 (a_1 d_9^- + a_1 d_{10}^- + a_1 d_{11}^- + a_1 d_{12}^-) + \\ & P_3 (a_2 d_{13}^- + a_2 d_{14}^- + a_2 d_{15}^- + a_2 d_{16}^-) + P_3 (a_3 d_{17}^- + a_3 d_{18}^- + a_3 d_{19}^- + a_3 d_{20}^-) + \\ & P_3 (a_4 d_{21}^- + a_4 d_{22}^- + a_4 d_{23}^- + a_4 d_{24}^-) + P_4 (d_{25}^- + d_{26}^- + d_{27}^- + d_{28}^- + d_{29}^- + d_{30}^-) \end{aligned}$$

Constraints:

$$\sum_{i=1}^4 X_{i,j} + d_j^- - d_j^+ = m_j \quad \text{for } j = 1, 2, \dots, 5 \quad (\text{minimum staffing requirements})$$

$$-X_{1,5} + d_6^- - d_6^+ = 0 \quad (\text{request by nurse 1 for day 5 off})$$

$$-X_{4,3} + d_7^- - d_7^+ = 0 \quad (\text{request by nurse 4 for day 3 off})$$

$$X_{2,5} + d_8^- - d_8^+ = 0 \quad (\text{request by nurse 2 to work day 5})$$

$$\sum_j^7 X_{i,j} = 5 \quad \text{for } i=1,2,..N \quad (\text{each nurse must work exactly 5 days})$$

the following constraints are in regard to the nurse 1's preferences to the day off/day on/day/off schedule patterns, assuming he is working Sunday and is off on Saturday:

$$\begin{aligned} X_{1,1} - X_{1,2} & \qquad \qquad \qquad +d_9^- - d_9^+ = 0 \\ -X_{1,1} + X_{1,2} - X_{1,3} & \qquad \qquad \qquad +d_{10}^- - d_{10}^+ = -1 \\ -X_{1,2} + X_{1,3} - X_{1,4} & \qquad \qquad \qquad +d_{11}^- - d_{11}^+ = -1 \\ -X_{1,3} + X_{1,4} - X_{1,5} + d_{12}^- - d_{12}^+ & = -1 \end{aligned}$$

the following constraints are in regard to the nurse 2's preferences to the schedule patterns. Note: nurse 2's constraints vary due to the different weekend assignments.

$$\begin{aligned} -X_{2,1} + X_{2,2} - X_{2,3} & \qquad \qquad \qquad +d_{13}^- - d_{13}^+ = -1 \\ -X_{2,2} + X_{2,3} - X_{2,4} & \qquad \qquad \qquad +d_{14}^- - d_{14}^+ = -1 \\ -X_{2,3} + X_{2,4} - X_{2,5} + d_{15}^- - d_{15}^+ & = -1 \\ -X_{2,4} + X_{2,5} + d_{16}^- - d_{16}^+ & = 0 \end{aligned}$$

The constraints corresponding to nurse 3 and 4 are similar to nurse 1 and 2's preference constraints.

The following constraints apply to the desired staffing levels:

$$\sum_{i=1}^4 X_{i,j} + d_{25+j}^- - d_{25+j}^+ = n_j, \text{ for } j = 1, 2, \dots, 5$$

$$-\sum_{j=1}^5 \sum_{i=1}^4 X_{i,j} + d_{31}^- - d_{31}^+ = -\sum_{j=1}^5 n_j$$

Having determined which day each of the nurses will work, the next phase is to determine the corresponding shifts for the scheduled days. The following steps describe the heuristic used to schedule the shifts.

1. Assign those nurses who work only one shift.
2. Assign shifts to the nurses who rotate between two shifts
  - A. Evaluate the minimum requirements for the two shifts.
  - B. Assign the nurse to the shift with the largest unattained minimal requirement. If the minimal requirements are met the desired levels are then compared.
3. Assign shifts to the nurses who rotate between three shifts.
  - A. Evaluate the minimum requirements for the three shifts.
  - B. Assign the nurse to the shift with the largest unattained minimal requirement. If the minimal requirements are met the desired levels are then compared.

The scheduling algorithm was developed with the assistance of the Wishard Memorial Hospital and the Health and Hospital Corporation of Marion County in Indianapolis, Indiana. However, there is little indication to suggest that the system was employed by the hospital.

## Musa and Saxena's Goal Programming Approach

Musa and Saxena also used a goal programming approach to address the scheduling problem (Musa and Saxena, 1984). Their method used the following single phase model:

Variables and Parameters:

$$X_{ij} = \begin{cases} 1 & \text{if nurse } i \text{ is to work the } j\text{th day} \\ 0 & \text{if not} \end{cases}$$

$d_i^+$  - the positive deviation corresponding to the  $k$ th constraint

$d_i^-$  - the negative deviation corresponding to the  $k$ th constraint

$D_i$  - the contracted number of days the  $i$ th nurse works

$DL_j$  - the desired number of LPNs required for day  $j$

$DR_j$  - the desired number of RNs required for day  $j$

$k$  - index the constraint

$L_j$  - the minimum number of LPNs required for day  $j$

$m$  - the total number of constraints:

$M$  - number of days in the planning horizon

$N$  - number of nurses

$P_k$  - the weight of the priority given to constraint  $k$

$R_j$  - the minimum number of RNs required for day  $j$

Objective Function:

$$\text{MIN } Z = \sum_{k=1}^m P_k (d_k^- + d_k^+)$$

Constraints:

$$\sum_{j=1}^M X_{i,j} + d_k^- - d_k^+ = D_i \quad \text{for } i = 1, 2, \dots, N \quad (\text{contracted number of hours for nurse } i)$$

$$\sum_{i=1}^N X_{i,j} + d_k^- - d_k^+ = R_j \text{ for } j = 1,2..M \text{ (minimum number of nurses for day j)}$$

$$\sum_{i=1}^N X_{i,j} + d_k^- - d_k^+ = L_j \text{ for } j = 1,2..M \text{ (minimum number of nurses for day j)}$$

$$\sum_{i=1}^N X_{i,j} + d_k^- - d_k^+ = DR_j \text{ for } j = 1,2..M \text{ (desired number of nurses for day j)}$$

$$\sum_{i=1}^N X_{i,j} + d_k^- - d_k^+ = DL_j \text{ for } j = 1,2..M \text{ (desired number of nurses for day j)}$$

for each full time nurse (l)

$$\sum_{j=1}^7 X_{l,j} + d_k^- - d_k^+ = 5 \text{ (each full time nurse has two days-off in the 1st week)}$$

$$\sum_{j=8}^{14} X_{l,j} + d_k^- - d_k^+ = 5 \text{ (each full time nurse has two days of in the 2nd week)}$$

for each full time nurse, who prefers the first weekend off (l)

$$X_{l,6} + X_{l,7} + d_k^- - d_k^+ = 0 \text{ (attempts to assign the 1st weekend off)}$$

for each full time nurse, who prefers the second weekend off (l)

$$X_{l,13} + X_{l,14} + d_k^- - d_k^+ = 0 \text{ (attempts to assign the 2nd weekend off)}$$

for each part time nurse, who prefers the first weekend off (l)

$$X_{l,6} + X_{l,7} + d_k^- - d_k^+ = \text{ (attempts to assign the 1st weekend off)}$$

for each part time nurse, who prefers the second weekend off (l)

$$X_{l,13} + X_{l,14} + d_k^- - d_k^+ = 0 \text{ (attempts to assign the 2nd weekend off)}$$

Musa's model allows the nurse manager to set priorities on the following goals: (1) for each nurse to work the contracted number of hours (2) to meet the desired staffing requirements (3) to satisfy the weekend preferences for the full-time nurses, and (4) to satisfy the weekend preferences for the part-time nurses. Musa et al. took a different approach from Warner's in considering nurses' preferences. Previously, the nurses' preferences were based on their aversion to different scheduling sequences, Musa, on the other hand regards their preferences in terms of which weekend out of the two available



they would prefer off. An example problem, with 154 decision variables and 120 constraints was run on a UNIVAC 1100 with a resulting CPU time of 28.3 seconds. This methodology is limited, for it only determines the day the nurse's should work and does not determine the corresponding shifts.

## *Conclusions*

The traditional and cyclical approach to scheduling do not provide a mechanism, other than by hand, of calculating statistics on the nurses. This includes such statistics as the number of nurses working each shift, the number of holidays taken, and the number of unexcused absences. A decision support system would provide a variety of capabilities including a mechanism to calculate these statistics. The other capabilities that a DSS could provide include, a feature to better save the schedules for the future, rather than simply saving the schedule on a sheet of paper, and a database to quickly access the nurse's records.

All of the mathematical models presented above limit the nurses to eight-hour shifts and do not provide the versatility to enable the nurse manager to schedule a variety of shift lengths. Although it is possible to modify the formulations to accommodate for four, eight or twelve hours shifts, the computational effort for the current formulations is already significant and further subdividing the shifts will increase it significantly. The eight-hour shift has been the most common scheduling pattern, yet many hospitals, in order to improve job satisfaction, have developed twelve-hour shifts (Atwood and Hinshaw, 1977). Studies on the effect of twelve-hour shift have indicated nurse satisfaction by, the general preference for the twelve-hour shift, the increased morale, the additional leisure

time and the feeling of providing more efficient patient care (Vik, 1982). Thus, it is important to have a scheduling system that is flexible to easily incorporate a variety of shift lengths.

All the mathematical models, except Warner's provide limited schedule details. Either the specific shifts are not assigned or they are assigned through a second procedure. Using separate procedures, independent of the day assignments, to assign shifts to the set days assures suboptimal solutions. The models, when computing the scheduled work days do not account for the effect the different time patterns may play on the overall schedule. For example, a nurse would not work a night shift followed immediately by a day shift. The shift effects are not independent factors and should be considered when the schedule is determined.

A DSS can provide the flexibility that is needed to support the nurse manager. It could allow the nurse manager to offer a variety of shift lengths. A DSS could also provide the nurse manager with a user friendly atmosphere to develop alternative schedules and maintain statistics on the nurses' work history. By maintaining statistics on past schedules, such as number of weekends, holidays, vacations, etc., the nurse manager would be able to make better informed decisions on future schedules. The DSS could help to ensure that vacations, weekends and holidays are distributed in an equitable manner. In addition, a DSS could provide statistics on the number of nurses worked during the various shifts. A DSS could provide the necessary flexibility to accommodate all of these issues.

## **Chapter 3: Decision Support Systems Design**

### ***Introduction***

In the past, the systems development life cycle (SDLC) has been the traditional method of decision support system design. However, in the last decade several design approaches have been introduced which focus on an increased level of user involvement. These approaches include Courban's iterative design, Keen's adaptive design, prototyping and a number of mixed methodologies integrating prototyping into the SDLC.

This chapter addresses several areas related to decision support development. First, the main decision support development strategies are summarized and contrasted. Second, an alternative methodology, titled priority design, is proposed. This is then followed by a review of a variety of guidelines for the development of a DSS. Lastly, a guided search is proposed for the selection of a development methodology based on the project's characteristics and the operating environment.

## *Decision Support Systems*

To assist management in its decision making process, companies have invested large amounts of time and money in decision support systems (DSS) (Pracht and Courtney, 1988). The increasing interest in DSS is due to the inability of traditional management information systems to aid in solving semi-structured and ill-structured problems (Sprague and Carlson, 1982). Unlike other operational information systems that collect, manipulate, and distribute information, DSS are designed to work with managers as they develop their decision strategy and arrive at a solution.

The goal of a decision support system is to assist managers in solving semi-structured problems by generating a range of alternative solutions so that managers can better understand the options that are available to them. Although there is no universally accepted definition of a DSS, one definition given by Keen and Scott-Morton is as follows: "Decision support systems couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions. It is a computer-based support system for management decision makers who deal with semi-structured problems." (Keen and Scott-Morton, 1978) Three major characteristics that better describe the features of a DSS include:

1. DSS incorporate both computational models and data,
2. They assist managers in solving semi-structured problems,
3. The user interface is interactive and user friendly.

These characteristics aid the DSS in providing the capability to provide a valid representation of a real world system and to evolve as the decision maker learns more

about the problem. This enables the user to explore a range of decision-making alternatives and their consequences.

## ***Design Approaches***

The design and implementation strategy are crucial to the success of a decision support system. Several methods have been proposed to develop information systems over the years. The earliest formal method, called the traditional systems development life cycle (SDLC) approach is still widely used. Other, more recently proposed methods, include iterative design, prototyping, evolutionary development and heuristic development. These methods may partially or completely replace traditional systems development in many applications. This section provides a brief overview of each of these methods.

### **Traditional Systems Development Approach**

The traditional framework for information systems development during the past thirty years has been the system development life cycle, consisting of a series of sequential phases illustrated in Figure 3.1. The primary purpose of this life cycle is to provide a highly structured and controlled method of system development. In the first phase, analysis, the developer works with the user to study and define the system's requirements. In the second phase, design, the system is constructed on paper. Approval, from the user, for the proposed system is required in order to proceed to the next phase, development.

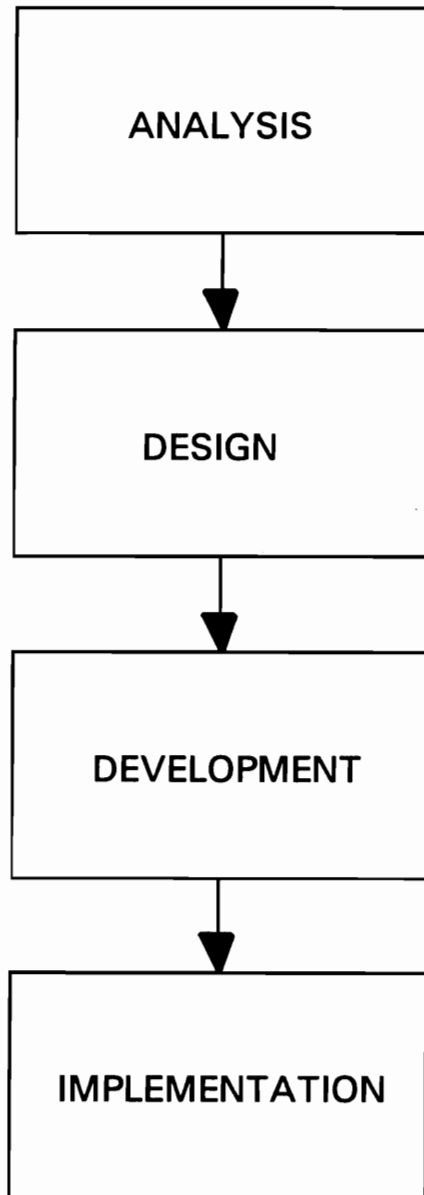


Figure 3.1: The Traditional Systems Development Life Cycle

Generally, once the system has been approved and development has begun, no changes to the system are permitted until the final phase, implementation, has been completed (Burns and Dennis, 1985).

The traditional systems development approach has been the subject of much debate by both academicians and practitioners (Cervený, et al., 1986). Three primary criticisms of this approach include:

1. the large amount of time and money the development cycle requires,
2. the limited amount of communication between the user and designer, and
3. the inflexibility of the design phase.

To reduce the errors in system design, the traditional approach recommends a high degree of formalization. Formal sign-offs, or agreements between the user and the developer are required to mark the completion of each stage and a formal division of labor between the user and the developers is also emphasized. This high level of formalization results in an increased level of bureaucracy. Generating the paperwork and the large amount of specifications, and the sign-off documents are very time consuming and costly and may delay the installation of a system for several years (Laudon and Laudon, 1993).

SDLC has several additional limitations and makes assumptions about the user that are not always valid. The methodology assumes that the user understands at the onset of the design process, what the problem is and what the desired system characteristics and capabilities are. However, this is not always the case. It has been noted that 60-80% of the failures can be traced to an inadequate understanding of user requirements, by either the designer or by the user, when the user approved the initial system design (Boar, 1984).

In addition, SDLC does not encourage user/designer interaction that would promote the concurrent learning process between the user and designer. (Cervený, et al., 1986) Communication between the developer and the user is primarily limited to the

Analysis Phase. During the Development stage, the designer consults with the user only if some specifications are unclear. Rarely are there system performance reviews, which would enable a user to see the progress to date. In addition, the user is not allowed to see the system or make suggestions for improvements until the entire system is complete. Figure 3.2 illustrates the phases of the development life cycle indicating where the user is involved in the design process. From the figure, it can be noted that the user is involved in only a small portion of the life cycle and only at the end of the cycle is the user allowed to make changes. This makes enhancements inefficient and costly. The traditional method is best suited to applications that are well-defined. In this type of application, the user knows, to a large extent, the requirements of the system and can successfully convey these ideas to the developer.

## **Iterative Design**

Courban and Bourgeois suggested an iterative or evolutionary strategy in which the designer first builds a small system or module that addresses a small portion of the overall problem, and then expand it in cycles (Courban and Bourgeois, 1980) This enables the user to be kept abreast on the status of the system, and provides the user with the capability to make changes throughout development. The iterative development process consists of the following four primary steps:

1. Identify an important module of the system, which is of high interest to the user.
2. Develop a small system that is appropriate for end-user development. This enables the user to contribute simple ideas and facts to individual modules and not be



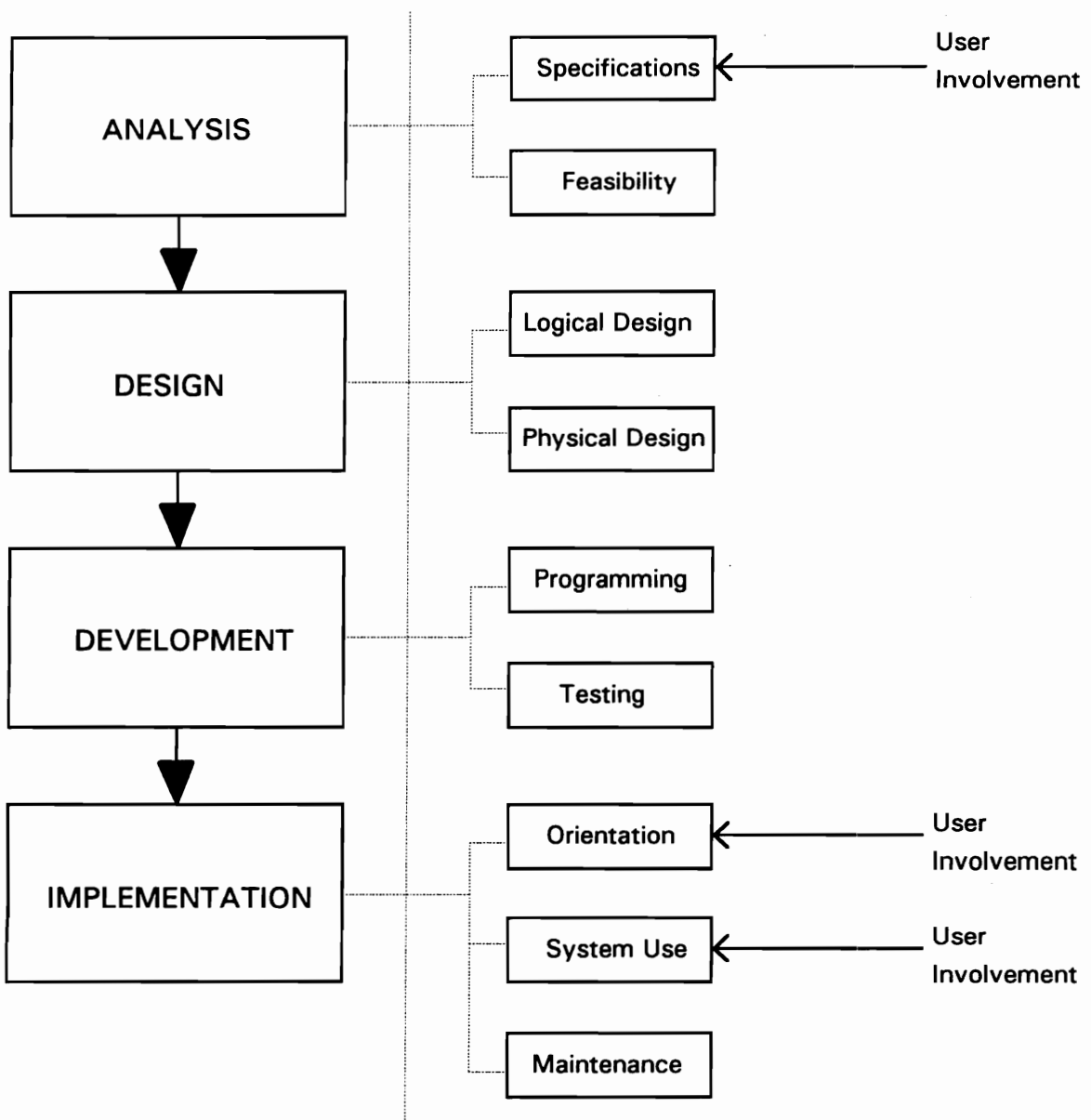


Figure 3.2: User Involvement in the Traditional Systems Development Life Cycle

concerned with the overall organization of the modules or their relationships in the system.

3. Expand and modify the system in cycles. Each cycle goes through the traditional life cycle phases including analysis, design, development, and implementation. At this step there is a large degree of communication between the user and analyst to determine the characteristics of the desired system.

4. Evaluate the system constantly. The user's continuous evaluation of the process normally results in a satisfactory system (Courban and Bourgeois, 1980).

An important benefit of this approach is the increased amount of user involvement early in the design phase of development. Continuous user involvement is assured as additional modules are added, each requiring analysis, design, development, and testing. However there are several drawbacks to this strategy. While the iterative approach has improved the development and usability of the system (Alavi and Mullinex, 1984) it is not always easy to implement. For example, if the time frame is short the designer may not be able to respond quickly enough, especially if the variety of information is large. Also, if the time frame is short it is important that the system have some incremental value to the user. In other words, the completion of the initial module should be of some use and importance to the user. In addition, if the goals and perceptions between the user and the developer or between multiple users are not similar, it may take several iterations before a satisfactory system can be reached (Sauter and Schofer, 1988).

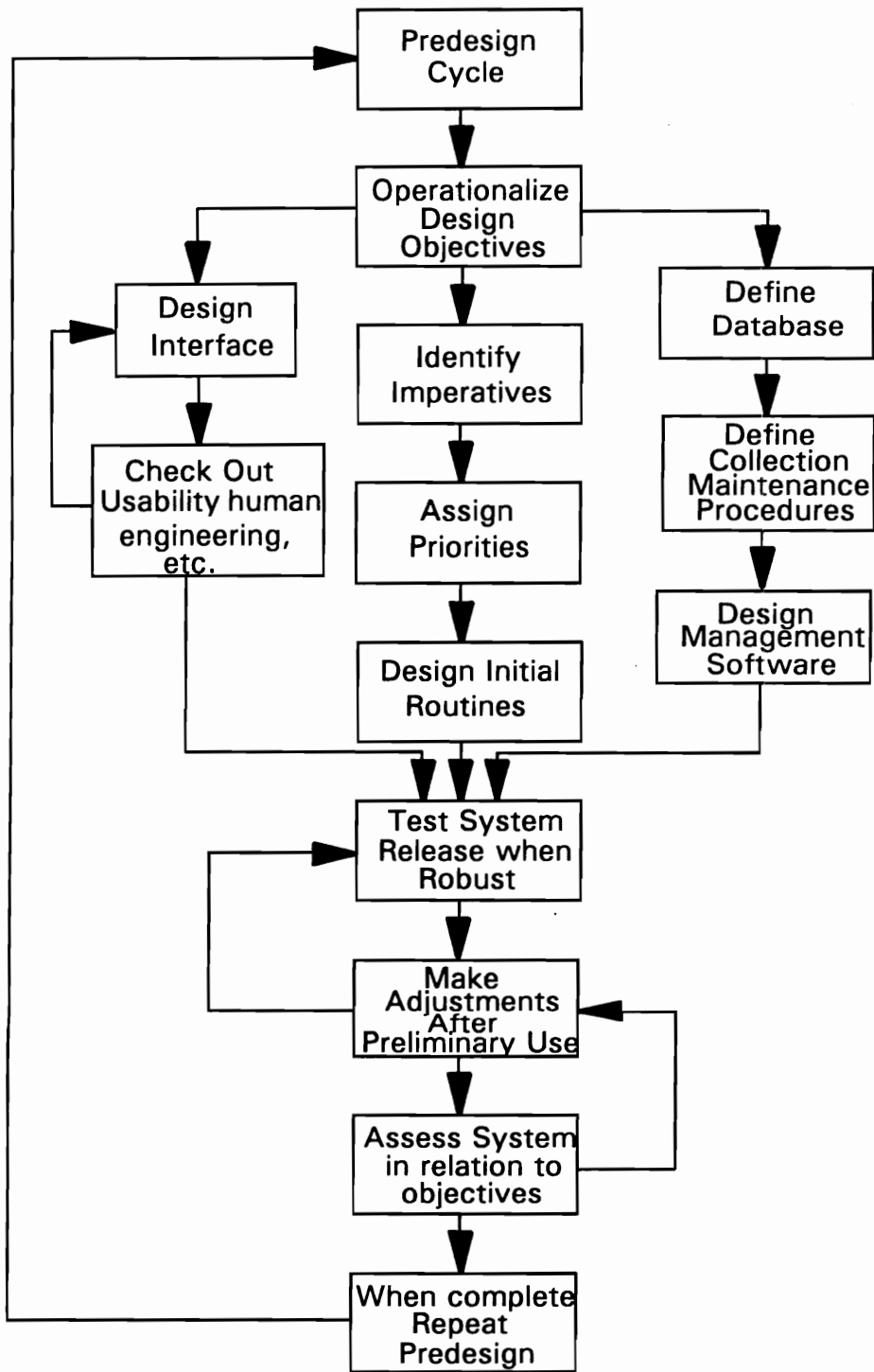
Another drawback to this approach is that the development of the system stems from one significant module. This requires that important decisions be made, based on the current module, which may not be efficient or applicable to the system later in the

development life cycle. As a result, large portions of the system may need to be reconstructed and the resulting system may not be efficient.

## **Keen's Adaptive Design**

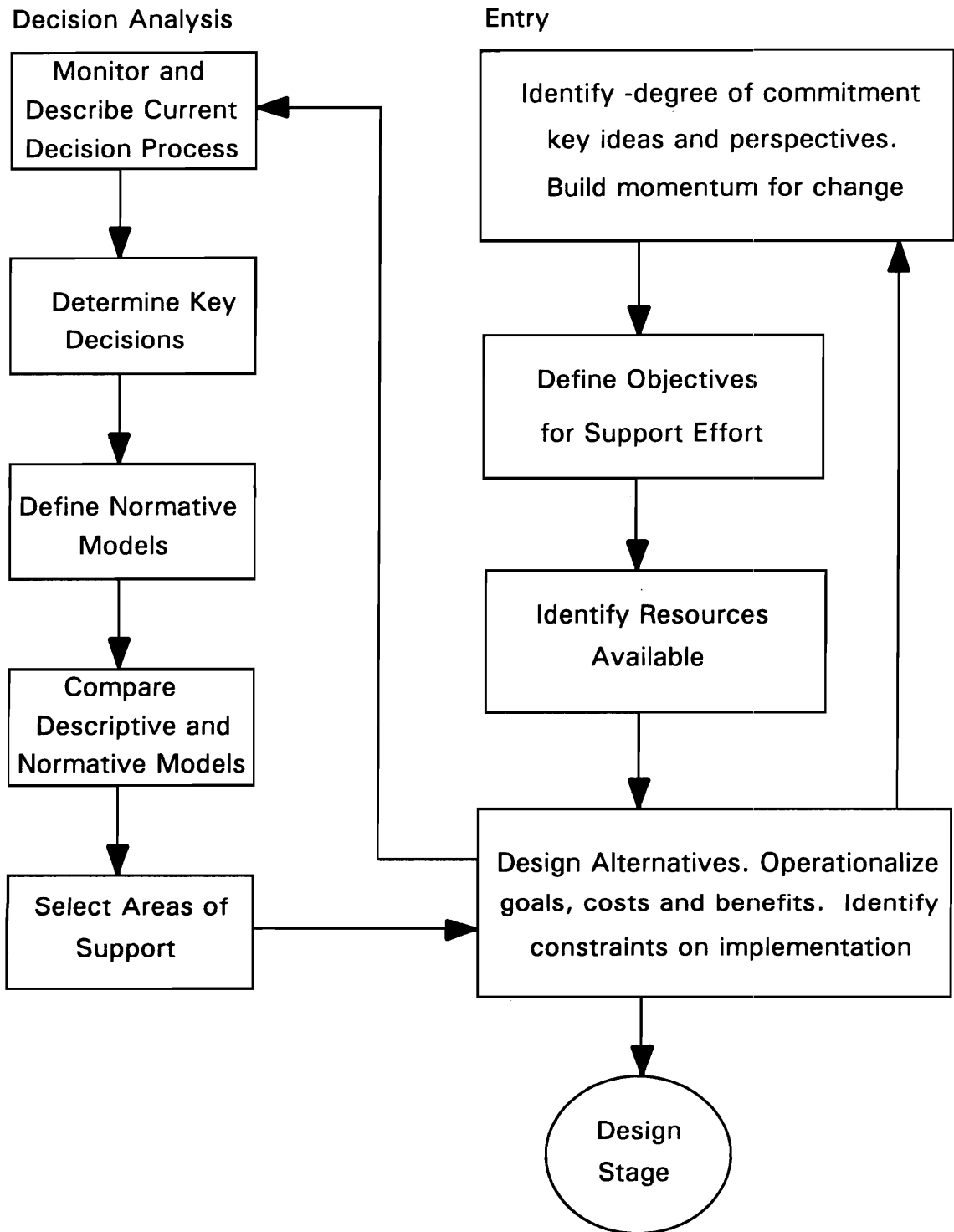
The design of a DSS is not a one-step process, but is an iterative process. At each iteration the designer and user must interact at various phases of the design process. Keen summarized the design phase in terms of a cycle in which the analyst provides "quick delivery of an initial system to which users can respond and thus clarify what they really want," (Keen, 1979). Keen's adaptive design approach is based on the assumption that a final system can most effectively be developed through an adaptive process of learning and evolution. An adaptive design means that the system can adapt to the user's changing requirements.

The first phase in Keen's design framework, illustrated in Figure 3.3, is defined as the predesign cycle, and its major objective is to ensure that the correct problem is addressed. The right hand side of the predesign cycle defined as entry, shown in Figure 3.4, is what Keen defines as the steps of the implementation process. This process focuses on: "building momentum for change and developing a 'contract' for action that involves realistic, mutual expectations and commitment among the parties involved" (Keen, 1978). The first stage in the entry portion of the predesign stage is to identify the degree of commitment to the project determine the key ideas and perspectives. Once these have been established, the objectives for support need to be defined and resources identified. At this point the developer should be able to create a number of alternatives and operationalize the goals, costs and benefits.



**Figure 3.3: Keen's DSS Design Cycle**

Keen, P.G.W. and M.S. Scott-Morton, *Decision Support Systems: An Organizational Perspective*. Reading, MA: Addison-Wesley, 1978.



**Figure 3.4 Keen's Predesign Cycle**

Keen, P.G.W. and M.S. Scott-Morton, *Decision Support Systems: An Organizational Perspective*. Reading, MA: Addison-Wesley, 1978.

The next process of the predesign phase is described as decision analysis. The first phase in this process is to monitor and describe the current decision process. The key decisions must then be determined. The next phase is to define the normative models that are the proposals for change, that indicate the potential range of designs. Once the normative models are defined, they need to be compared to the descriptive models. This is in essence comparing the measure of payoff with the difficulty of implementation. The final phases in the predesign cycle focus on first defining the areas of support and then redefining the objectives and goals. Once the goals are defined and there still exists a commitment to the project then the design can be implemented.

Once the predesign cycle has been completed, the next phase of the design framework or cycle is to operationalize the design objectives. In the predesign phase the objectives were fairly broad, now they must be more precise in terms of identifying the main criteria. Once this has been completed, the imperatives or commands, must be identified and priorities set for the various routines. Many of the needs of the user can be described by a set of verbs, which is often common in many other systems. Imperatives such as Plot, Display, Table, etc., can easily be identified. The design of the imperatives is only one portion of the design process of a typical DSS. Keen suggests that the design process involves three separate areas: the imperatives, the user interface and the data base. Once these items have been established, the initial design routines can be constructed and the system tested for robustness. When the system is found robust, it then can be implemented and preliminary adjustments can be made. Following some preliminary use, the system is then assessed in relation to the objectives and adjustments made accordingly. In order to maintain an evolutionary design structure the predesign cycle should be repeated to keep up with changes in management's decision process.

Keen's approach was employed in the development of a DSS for the conversion of a job shop system to a cellular manufacturing system (Durmusoglu, 1993). Durmusoglu found that Keen's approach of developing the user interface, the database and the models independently made the system easy to modify and the enhance, which allowed the system to evolve without difficulty.

Keen's adaptive design was also used in the development of an online retail banking DSS (Omar, 1991). It was found that the adaptive process "stimulated the learning of the users and their understanding to what the system might offer. This resulted in active participation and constructive feedback that helped expand the system's capabilities and the range of its uses" (Omar, 1991).

## **Prototyping**

One of the shortcomings in the traditional systems development life cycle method has been the assumption that the user understands and can describe the systems requirements. Prototyping attempts to alleviate this problem by first developing and presenting to the user, a working model that encompasses the essential elements of the desired system. After development, the prototype is evaluated and revisions and enhancements are recommended for the system. The prototype is then redeveloped and presented again to the user. This process of analysis, design, coding and implementation is repeated several times until a satisfactory system is developed. This repetitive approach increases the amount of communication between the user and the developer and enables the user to visualize the entire system at each iteration.

Janson et al. categorized three types of prototypes: real life, simulated and real life/simulated (Janson and Smith, 1985). The real life prototype is a full-scale representation of the basic design using the technology that is intended to be used for the

final system. The primary purpose of this type of prototype is to verify the soundness of the design and to ensure that the design specifications are met. The simulated prototype is constructed using technology that is not going to be used for the final design. This prototype is not intended to show how the model will work in real life, but to provide an understanding of the proposed design concepts. The real life/simulated prototype is constructed partially with real life or final design components and partially with simulated components. Thus, the prototype shows how parts of the final design work in real life and also provide insight into why the design works (Janson and Smith, 1985)

Naumann et al. described prototyping as a revolutionary process rather than an evolutionary process consisting of four principle stages, illustrated in Figure 3.5 (Naumann and Jenkins, 1982). The first stage is to identify the user's basic information requirements, which can be accomplished by implementing one of the following two distinct approaches, the data abstracting approach and the process simulating approach. The data abstracting approach suggests that emphasis should be placed on the essential characteristics of the data and the data relationships. Konsynski describes the data abstracting approach as a new system design paradigm, in which determining the requirements means constructing a model of the relevant data (Konsynski, 1981). The process simulating approach, on the other hand, assumes that both data and model processing must be emphasized. However, it is important to note that both of these approaches place little emphasis on completeness at this stage (Naumann and Jenkins, 1982).

The second stage is to develop a simple working prototype, which should be implemented in a very short time. McCracken suggests that a running prototype should not take more than a day or two (McCracken, 1980). The initial prototype should not be considered a completed system, but is generally a simulation that represents the essential elements of the system desired by the user. The third phase and fourth phase include the



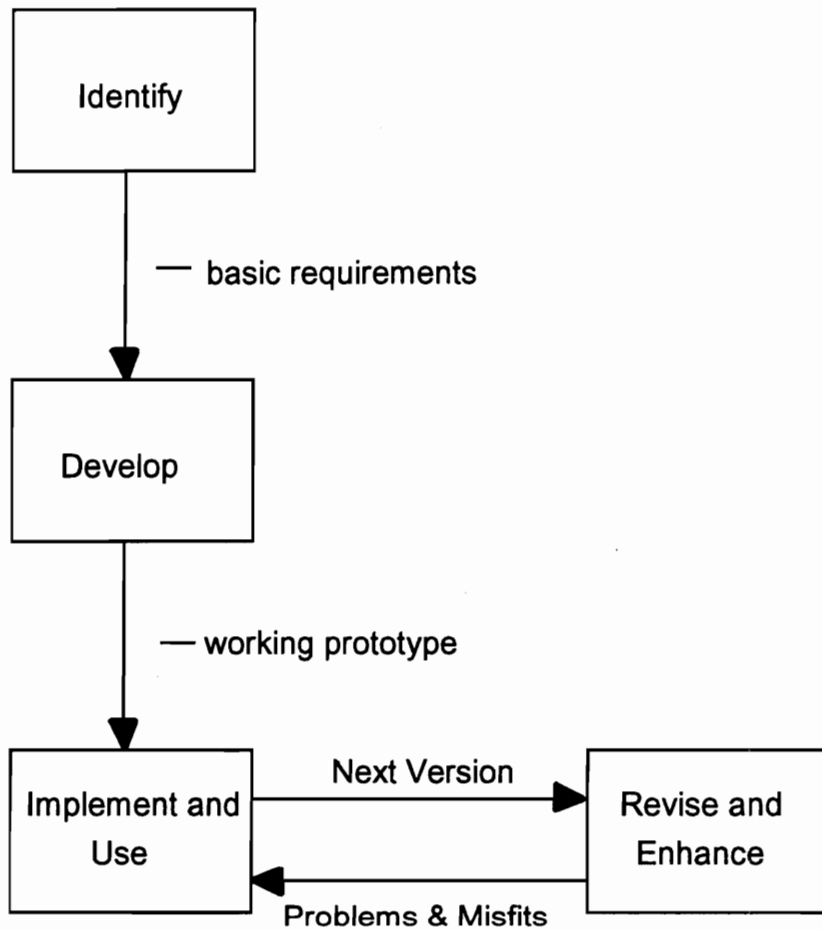


Figure 3.5: Prototyping

implementation and use of the prototype system and then the revision and enhancement of the system. Both of these phases should be repeated until a satisfactory system is developed. One problem with the prototyping approach is that it often produces less efficient systems than the traditional method (Alavi and Mullinex, 1984). This decrease in efficiency is partially due to the lack of initial analysis and design and to the less efficient nature of some of the software tools used (Burns and Dennis, 1985). Kraushaar presents another criticism of the prototyping process indicating that the approach does not clearly show the potential relationship between prototyping and alternative development approaches within the same project (Kraushaar and Shirland, 1985). Kraushaar addresses this problem by considering the application development and prototyping as a state-transition process shown in Figure 3.6 (Kraushaar and Shirland, 1985). The state-transition approach indicates that each transition requires its own development process, and that the application development methods such as prototyping can be viewed as transitions between system states. This type of prototyping approach is appropriate for application projects that are generally developed by information specialists and/or end users and is not recommended for well-defined problems, or when development experience on similar projects is extensive.

Another problem with the prototyping methodology is that to be successful, prototyping requires short turn around times between the user request and the prototype implementation (Ress and Currie, 1993). In addition, there is a tendency for the user to accept the initial version of the system as the final product (Dennis, et al., 1987).

Ress and Currie applied the prototyping strategy in the development of an expert scheduling decision support tool. They found that although prototyping appears simple, in practice it could become complex if the steps are not followed in sequence (Ress and

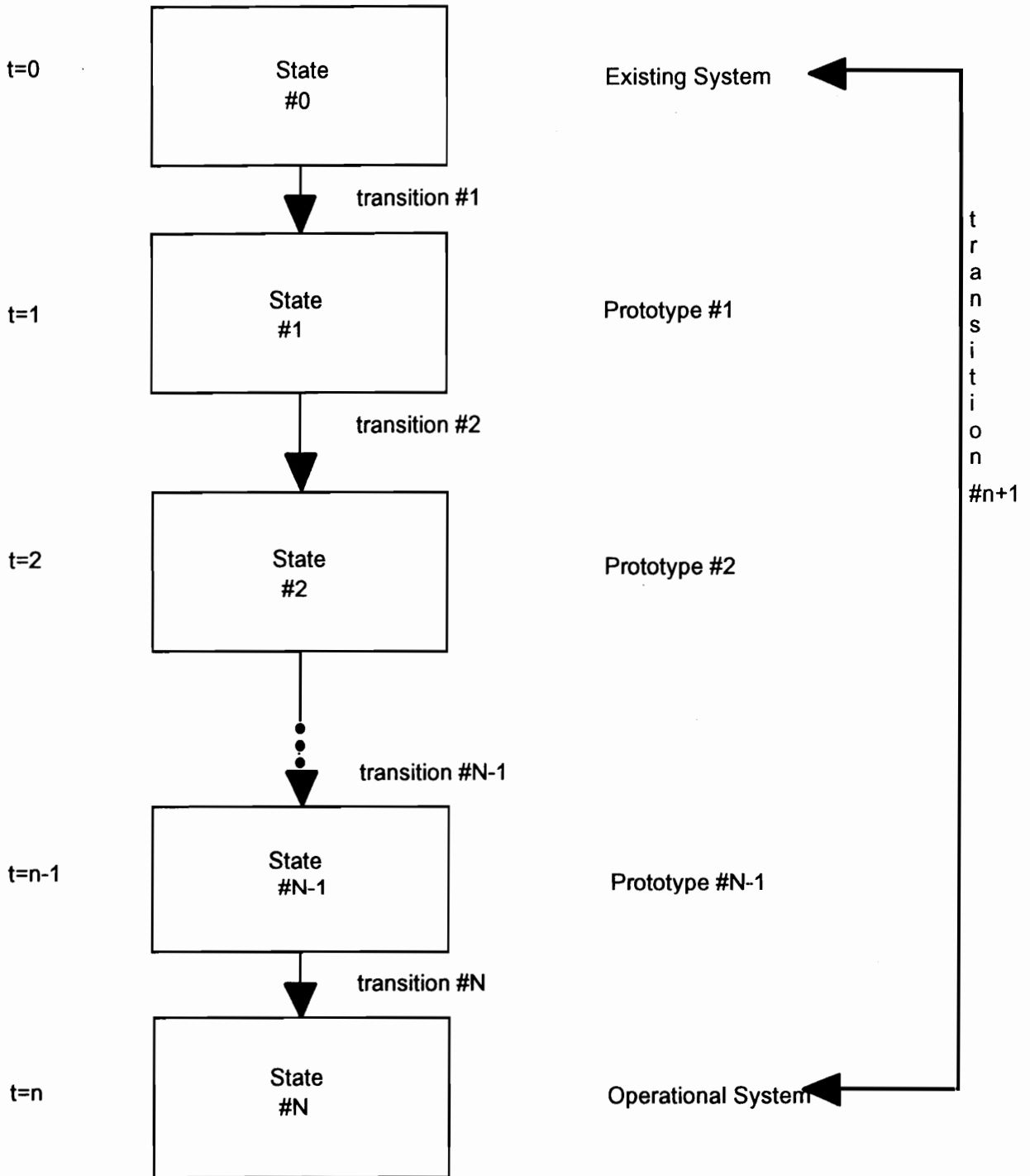


Figure 3.6: A State-Transition Model for Application Development and Prototyping

Currie, 1993). They also noted that it was important when constructing the prototype to remember that prototyping is a repetitive task and not everything has to be accomplished in the first loop.

## **Mixed Methodologies**

Several methodologies that incorporate prototyping into the traditional development life cycle have been discussed in the literature. One method suggested by Berrisford et al., is to incorporate a design/develop output step into SDLC. Berrisford found a major flaw exists in the design phase of SDLC. The design phase assumes that management knows what information is required (Berrisford and Wetherbe, 1979). Prototyping attempts to alleviate this problem, by enabling the developers to heuristically define the user's information requirements while developing an output system. Berrisford recommends that a "Heuristically Design/Develop Output System" phase be integrated between the Analyze and Design Phases of the SDLC. Thus the design and development of an input system will be accomplished after the development of an output system.

Another mixed methodology, phase design, has been proposed by Dennis et al. (Dennis, et al., 1987). The first step in this approach is the sequential development of each major subsystem. The entire system is designed at a coarse level of detail. The next step is to sequentially refine each subsystem. The refinement requires each subsystem be presented to the user, refined, and approved before the next subsystem is designed in detail (Burns and Dennis, 1985).

## ***Guidelines for Constructing a Decision Support System***

Once an appropriate design strategy has been selected, the steps involved in the construction of the decision support system become an important, integral part of the development process. To clarify the information requirements, Sprague and Carlson have developed a framework for DSS analysis, called the ROMC approach (Sprague and Carlson, 1982). This approach identifies four components that every DSS contains. Similarly, Kroeber suggests answering a number of preliminary questions before the design process begins, to ensure the successful completion of the DSS. In addition, to simplify the construction process, Keen and Gambino have assembled a list of twelve rules of thumb for building a DSS (Bennett, 1983).

### **The ROMC Approach**

The Representations, Operations, Memory Aids, and Control Mechanisms Approach, or simply the ROMC Approach, developed by Sprague and Carlson, is a systematic development approach to assist in the construction of a DSS. These four components, ROMC, which are described below, define and constitute a specific DSS.

- **Representations** - The representations are the schemata used to display the data and operations to the user. Every DSS requires representations to aid the user in conceptualizing and interpreting the problem. Examples of representations include: lists, graphs, icons, windows, spreadsheets, and organizational charts (Carlson, 1979).
- **Operations** - The operations are the procedures required to create representations and process data. Examples of operations include: plot data, analyze, generate reports, generate statistics on alternatives, and simulate results of alternatives.

- **Memory Aids** - Memory Aids assist the user in linking the representations and operations. The following are some examples: database, function key labels, rolodex card files, and note pads.
- **Control Mechanisms** - The mechanisms used to control and operate the entire system, thereby, integrating the previous three components into one system. Some examples of control mechanisms are: menus, function keys for operation selections, windows, help comments and tutoring.

The ROMC approach is not a step-by-step procedure, but describes the characteristics and capabilities that a DSS needs to have. It can be incorporated into many of the design methodologies, including Keen's adaptive design. The ROMC approach conceptualizes the requirements of the user into components, that should be integrated into the DSS.

### **Kroeber's Guidelines**

To determine if a Decision Support System is the appropriate solution to the problem and to help ensure that the DSS is successfully implemented, Kroeber recommends addressing the following questions before proceeding to design the DSS.

- Is there a need for a DSS and will it enhance the decision maker's effectiveness?
- Do the benefits of the DSS outweigh the cost of the development?
- Is there time available for approval of the system?
- Does a champion exist?

Before a decision support system is constructed, the developers and the organization must first establish the need for a DSS. The purpose of a DSS is to support the organization's decision making tasks. The DSS that is ultimately constructed should support the

organization in this area. Otherwise, it would be pointless to construct the DSS. Even if it were to be developed, management would not employ the tool because it did not help them in the decision making process. The next issue that should be addressed is the cost trade-off. The cost of the DSS needs to be justified by the increased efficiency or effectiveness of management's decision making process. If the organization cannot justify the cost of the system, the organization may want to look into other avenues to address the problem.

For the DSS to be successful there must be time for the organization to approve the system and to make any necessary adjustments to it. If there is not time for management to approve the system, management may find that the system does not meet its needs and resent the fact that it was thrust upon them without their approval. In addition, management is less likely to adopt the system. The final requirement is the need for a champion. The DSS needs to have at least one person, who believes in the value of the system and is willing to champion the project. This person must be available to meet periodically, with the developer, to assist in any questions that may arise during the development process. This champion would also motivate the other users to employ the system. The DSS will not be successful unless someone is willing and able to support and promote the project.

### **Keen and Gambino's Guidelines for Constructing a DSS**

Keen and Gambino have developed a list of rules of thumb to aid in the design process. The list is composed of the following items:

1. Design the dialog first.
  - a. Define what the user says and sees.
  - b. Define the representation of data.
  - c. Adopt a system model which matches the user's conceptual model.
2. Identify the user's special-purpose verbs.
3. Identify generic verbs relevant to this DSS.
4. Translate the verbs into commands, and vice-versa.
5. Check public libraries for off-the-shelf routines.
6. Set priorities for implementing commands for Version 0.
7. Support first, extend later.
8. Deliver Version 0 quickly and cheaply
  - a. Evolve a complex DSS out of a simple Version 0.
  - b. Version 0 is intended to establish value and to sell itself.
9. Pick a good user who:
  - a. Has substantial knowledge of the task,
  - b. Has intellectual drive and curiosity,
  - c. Will take the initiative in testing and evolving Version 0, and
  - d. Enjoys being an innovator.
10. Recognize data management, rather than commands, as a main constraint.
11. Remember that Brooks is right - programming is 10% of the effort.
12. Know your user at all times. (Bennett, 1983)

The guidelines can be divided into three phases, with the first phase focusing on the development of a preliminary, working system or prototype, which Keen refers to as Version 0. Keen recommends that the first step in the construction of the prototype is to



determine the dialogue or user interface, and the representation of the data. The next stage in the development of the prototype is to identify the verbs, the actions or commands that the user would like to see. For example, list, rank, and histogram are generic verbs that are commonly found in decision support systems.

The second phase of the guidelines centers on soliciting the user reaction to the prototype. At this stage the prototype is not bug-free, but, should be able to give the user an overview of the system and the representation of the data. The qualities that Keen recommends a developer consider when choosing a user to promote the system are listed in Rule 9.

The third phase of the guidelines focuses on converting the system into a product. The developer must now deal with the construction of the system as a product, including such issues as technical support, documentation, training, and cost. Rule 10 in the above list refers to the area of data management, an area which Keen found significantly adds to the system overhead. In the development of the prototype, the data is gathered in a loose or ad-hoc manner, so that the value of the DSS can be established. Once the system is justified, the required investment can be made into the development of the DSS/Data Base link (Bennett, 1983).

The first phase of these guidelines is more applicable to the prototyping and mixed design methodologies. The rapid development of a working system, supports the prototyping strategy, as opposed to the iterative or traditional strategies. However, the remaining two phases of the guidelines can be applied to all of the design methodologies. The guidelines for the selection of a supporting user and other guidelines dealing with the construction of the DSS can also be employed in each of the methodologies.

There is some overlap between Kroeber's guidelines and Keen's predesign cycle and Keen and Gambino's development guidelines. The issues of need and cost benefit

analysis, presented by Kroeber are further developed in the entry phase of Keen's predesign cycle. Furthermore, the need of a champion also presented by Kroeber is given greater detail in Keen and Gambino's development guidelines. Kroeber's guidelines and many of the phases of Keen's predesign cycle should be employed before the development of the DSS begins regardless of the design methodology that is to be used. However, it should be noted, that all of the steps of the predesign cycle are not appropriate for all types of DSS. For a small scale system, with low complexity the large amount of time and detail given to the analysis of the project is often unnecessary.

## ***Priority Design Methodology***

### **Introduction**

All of the development methodologies have a number of shortcomings that arise when used in different operating environments. For example, when there is significant user involvement and the environment is not uncertain, all of the methods presented have some drawbacks. Both prototyping and adaptive design require that a prototype be constructed. In an environment that is not uncertain this is unnecessary. An alternative methodology, called priority design is proposed. Although, similar to Keen's adaptive design it addresses some of these shortcomings that may be found when used in different operating environments.

## **The Steps to the Priority Design Methodology**

The Priority Design methodology focuses on determining, early on in the development process, a priority list of items and requirements for the system. This list will enable the developer to have a better idea of what aspects of the system the user considers most important to the success of the DSS and will allow the developer to focus on the development issues that are most important to the user. In addition, the design methodology, illustrated in Figure 3.7 contains a feedback loop that allows the developer to communicate to the user the cost and development time required for each of the items on the priority list. The steps of the priority system methodology are as follows:

### **Step 1. Define the minimum requirements.**

The first stage of the development is to establish what the user defines as the items of highest priority or the minimum requirements for the partial system. Keen's predesign cycle is recommended as one alternative approach for determining these requirements (Turban, 1992). The requirements should be technologically feasible. Although the user may be unclear of the capabilities of the system and all of the requirements, he should be able to identify the essential components. These components define the necessary boundaries for the system.

### **Step 2. Develop a Simulation Prototype**

To assist the user to fully define and conceptualize the problem, before development of the actual system begins, a simulation prototype should be built. The prototype need not employ the technology that the actual system will be constructed with, but should aid the user in visualizing the actual system. In fact, the simulation prototype may not even perform actual operations, such as data retrieval or model solution. It

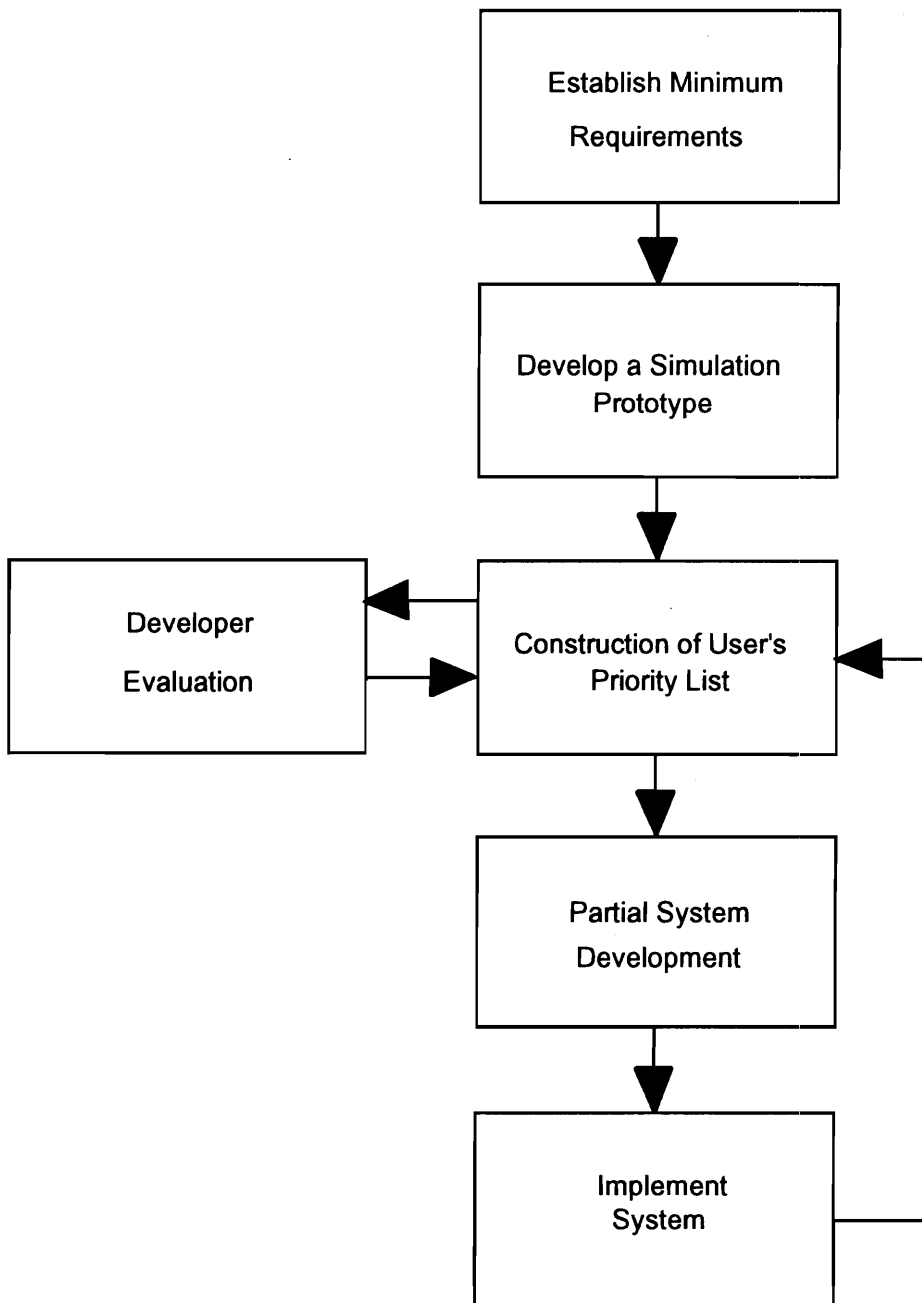


Figure 3.7: Priority Design

primarily displays the user interface and the data that is available. This step will aid the user in clarifying the true requirements for the system. This simulation prototype should be created quickly, one to two days if possible.

### Step 3. Develop a Priority List

Having been able to better visualize the system characteristics and its capabilities, the user should now be able to construct a priority list of suggested requirements and enhancements for the system. The requirements that management considers most important and beneficial to the overall system should be given the highest priority and the minor "extras" should be given lower priorities.

### Step 4. Developer Evaluation

The developer should then go over the user's "wish list", evaluating the time and expense it would take to meet each of the user's requests and adding suggestions that may enhance the system. With the growing advancement of technology, the user is not always knowledgeable of the capabilities of the DSS tool. The developer, having more knowledge in this area may be able to suggest possible refinements and additions to the system. These refinements may or may not be of significant additional cost to the user, but may greatly enhance the overall quality of the system. Thus, this revised list will both enable the user to get a better idea of the time and cost involved in each of the items on the list and will provide the developer an opportunity to make suggested refinements.

### Step 5. Reevaluation of the Priority List by the User

The user should then reevaluate the priority list, considering the expense and time required for each of the enhancements. At this point, the user will have a better idea of

the resources required for the project and can present this to upper management, for review as required. The resulting list will essentially be divided into two parts. The first section will include the items to first be developed for the partial system. The second section will be the items that will be constructed later as the system develops.

#### Step 6. Develop the System Based on the Priority List.

Construct the system based solely on the highest priority requirements for the partial system. A workable system that meets the user's minimum requirement should then be available.

#### Step 7. Repeat Steps 3-6

Steps 3-6 should be repeated with several iterations, until a satisfactory system is developed. By presenting a partial system for use, it will teach the user and the builder more about the problem and the process required to solve it. This new information and experience will alter the list perviously established in Step 3.

### **Benefits of the Priority System**

By constructing the system based solely on the requirements for a partial system, a usable system can be developed in a short time frame. This method differs from Courban's iterative design in that the initial design strategy of the priority system stems from the minimum requirements, rather than an important module. This is an important feature, because one of the short falls of the iterative design is that important design decisions are made from the viewpoint of a specific module and may not complement the overall

system. By maintaining a priority list, the development of an initial system is not hindered by unnecessary enhancements. This allows the development to focus on the most critical aspects of the DSS, as defined by the user. This ultimately keeps the time for the initial system to a minimum. In addition, by keeping a list of the lower items the developer is aware of where and how the system is going to expand. This will prevent the developer from creating "dead ends", where it would be impossible or very costly to develop the other requests.

The simulated prototype provides the user with the capability to view the system before the construction process begins. The simulated prototype also allows the user the opportunity to see the user interface and the capabilities of the system. By viewing the prototype, the user can flesh out any characteristics that were not originally foreseen during the requirements phase, before any expenditures have been made. On the other hand, if a working prototype is developed, the user must invest in the necessary hardware and software, before seeing the system. By developing a simulation, the user does not have to make any investments until he has a clear understanding of the system's capabilities and its user interface. In addition, the simulation prototype provides the user with a mechanism to make suggestions, which will help reduce the number of problems that may arise during the construction and implementation phases.

The feedback loop between the user and the developer is a step that is not clearly defined in the other methodologies. However, it is an important step in the design process. It allows the user to have an idea of the cost of the various items in the project. If the budget needs to be revised and items cut from the list, the user will have a more educated idea of which parts of the system are most expensive to the project. The user unknowingly, may have requested a specific capability, that was not of high importance, but, was very expensive to construct. It is important to remember that the user may have

little knowledge of the cost of the various requests. With the addition of the feedback loop, the user will be more educated to make better decisions regarding the overall capabilities of the system. In addition, this feedback loop provides the developer an opportunity for make suggestions to the system, that the user may not have considered.

In Keen's design the components of the DSS including the user interface, database and the model base are developed independently. The first release of the system may have parts of each of these components developed. In the Priority design the user interface or the representations of the ROMC is developed first and presented as a simulation prototype. In addition, Keen's design does not have an integrated feedback loop between the user and the developer.

### ***The Selection of a Design Strategy***

Several design approaches have been presented. To determine which type of design methodology should be implemented, the characteristics of the user's operating environment need to be evaluated. Each of the design strategies previously presented makes assumptions about the users and their environments, and each is more beneficial when certain operating characteristics exist. Therefore, to select a design methodology the developer must address a number of issues to compare and contrast the user's environment with the benefits and shortcomings of each of the design methodologies. Some of the issues that should be addresses include:

- Is the user's time limited? Is the user able or willing to spend time throughout the development process?



- Does a subproblem or module have value? If a portion of the system is developed is it considered usable, or does the system need to be near completion before it is of any use?
- Is the project uncertain? Does the user know the requirements for the system and is he able to convey these ideas to the developer?
- Is the project complex? Is the project size large or is a vast amount of new information continuously needed?

Depending on the characteristics of the user's environment, some or all of these questions must be answered before a development method is selected. A framework is presented in Figure 3.8 as a selection guideline. The first question that should be addressed is the extent that the user will be available to work with the developer. In some cases, the user's time is extremely limited and the user is either unwilling or unable to spend time periodically evaluating the system. In this situation, before development begins, the user should evaluate the level of uncertainty involved in the project.

The project uncertainty can be determined by evaluating three project characteristics: degree of stability, user task comprehension, and developer task proficiency. The degree of stability refers to how well defined the project is and how likely the system definition would change during the development of the system. A highly structured system, that has a low level of uncertainty, is typically clearly defined and has a stable system definition throughout the development life cycle. The second characteristic of uncertainty, user task comprehension, measures the degree to which the user understands the project characteristics and comprehends the manner in which the DSS will be applied. The last component of uncertainty, developer task proficiency is the degree of training and experience the developer brings to the project. This includes both the developer's experience in the user's operating environment and his familiarity with the

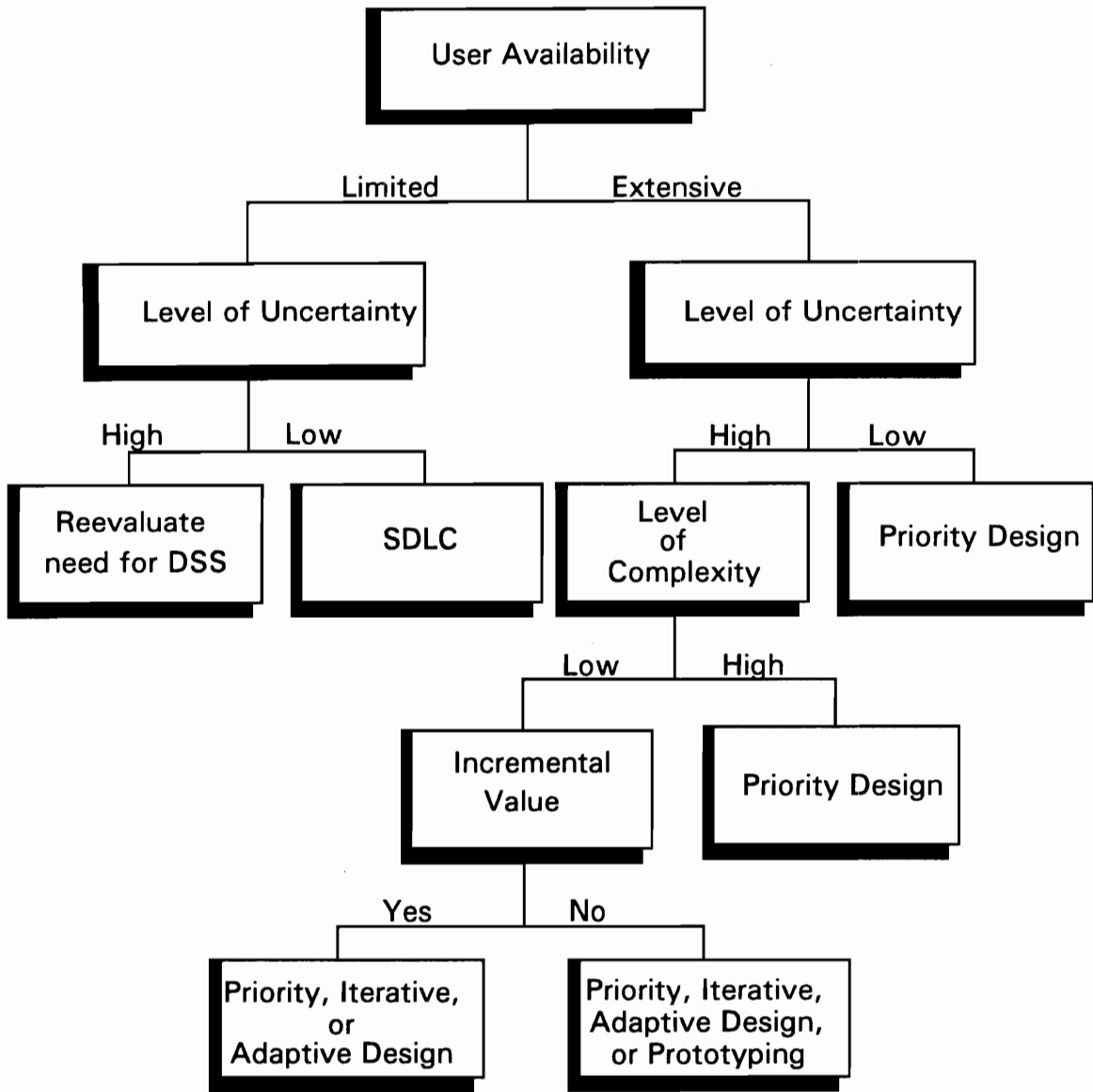


Figure 3.8 Guidelines for Design Methodology Selection

tools that will be used in the construction process (Davis and Olson, 1985) (McFarlan, 1981). Each of these three components is an important factor in determining the level of uncertainty involved in any DSS project.

If the project has a high degree of uncertainty the user should reconsider whether the construction of a DSS is an appropriate mechanism for addressing the problem. If the project is highly uncertain, the probability of constructing and implementing a decision support system successfully is very low. Decision support systems are inherently interactive systems. If the project is uncertain and the user is not available, there is no mechanism available to the developer to clarify the desired system's goals and to guide the form the interaction between the user and the system. In this case, the developer may want to reevaluate the availability of the user's time or consider other avenues to deal with the problem.

When the user's time is limited and the project has little uncertainty, the traditional systems development life cycle would be the more appropriate method of system development. The traditional life cycle only requires the user to be involved with development during the initial analysis phase. In contrast, both prototyping and the iterative approach require a high level of user involvement for the initial design and for periodic evaluations.

If the user is willing to devote extensive time to the project, other issues must be addressed to determine an appropriate methodology. It should be noted that the term "extensive" does not imply that the user has an infinite amount of time to devote to the project. However, it reflects that the user's time is valuable, and that he does have sufficient time available to periodically meet with the developer.

Assuming that the user's time is not limited, the first issue that should be considered is the uncertainty of the project. Given a well-defined project, in this

environment the appropriate methodology would be the priority design. This methodology is better suited to the environment than the other commonly used design methodologies. The alternative approaches to priority design for high user involvement and low uncertainty would include the SDLC, iterative design, adaptive design and prototyping. Although, the SDLC does not require a large amount of interaction between the user and the developer, which is often unnecessary in the situation where the project is well-defined, it however, it is not the most effective methodology. A significant drawback of the SDLC is that it requires a large degree of formalization that generally results in higher project costs and longer development times. Both prototyping and the iterative design rely on detailed steps to reduce the amount of uncertainty. Both the user and the developer are required to perform many of these steps. Since in this situation the project is not uncertain, the extended developer/user participation is unnecessary. In addition, prototyping focuses on developing a working prototype quickly. For a well-defined project the development of a prototype is frequently superfluous (Burns and Dennis, 1985). Keen's rules for development also include a prototype step and therefore introduce unnecessary effort.

Priority design methodology does not suffer the same limitations. Priority design includes a simulation step. The priority design suggests developing a simulation as opposed to a prototype. When developing a prototype there is always some initial investment in both time and money. While both the simulation and the prototype can be considered unnecessary in an uncertain environment, it does help establish that the user has successfully conveyed his needs and the characteristics of the desired system to the developer. A simulation of the system is a cheaper and quicker mechanism for establishing that both the user and the developer are clear on the requirements for the system. After evaluating all of the alternative methodologies, the priority design methodology should be

employed when user involvement is significant and the project has a low degree of uncertainty.

When the user participation is ongoing and there is considerable uncertainty, SDLC would not be an appropriate methodology. To reduce the uncertainty of the project, there needs to be an increased amount of user involvement. However, the SDLC provides little interaction between the user and the developer (Cerveny, et al., 1986) and would not be an effective methodology for alleviating this uncertainty. Assuming user participation and a project with a high degree of uncertainty, the complexity of the systems should be addressed.

The project complexity is defined by four components: project size, number of users, volume of new information, and the complexity of new information. Complex projects are typically large, requiring a considerable number of man-hours. The increasing number of users implementing the system and the increasing volume of information also adds to the complexity of the project. The increased number of users, requires that the developer contend with the conflicting desires that may arise between the different users. The users have their own goals and ideas for the system. The developer must merge these different and perhaps opposing ideas to develop a system that will meet the approval of the users as a whole. The last factor that affects the project complexity is the complexity of new information. Is there a large amount of formulation or complex programming required to create the new information? The need for new information can manifest itself in three different ways. The user may have the need for data that is not in a convenient form. The DSS must be required to present this information in a suitable manner that is easily understood by the user. The DSS may also be required to generate data that does not already exist and has not been previously stored. Finally, the DSS may need to develop models and create or customize a variety of solution techniques. Projects that

require a great deal of effort to produce the new information are commonly more complex.

If the complexity of the project is high, the approach that would be most appropriate in this environment is the priority design. This method focuses on increasing the amount of communication between the user and the developer to reduce the uncertainty. Although, prototyping, adaptive design and Courban's iterative design also concentrate on reducing the uncertainty, they are not as applicable for large, complex systems. Prototyping and adaptive design suggest that a prototype be built cheaply and quickly. However, the larger and more complex the project, the more difficult it is to prototype the entire system (Burns and Dennis, 1985). The iterative design methodology starts with an important module and then expands upon in cycles. In a complex environment, when starting from one specific module, problems can arise, as other modules that may be incompatible with the initial module are developed. As a result large portions of the system may need to be reconstructed and the resulting system may not be efficient. The priority design methodologies establish the initial format of the system by evaluating the important characteristics of the overall system as defined by the user. This enables the development to stem from the more important characteristics and not one specific module.

When user's time is not a restriction, the environment is uncertain, and the complexity is low, the next issue that needs to be addressed is the importance of developing a partial working system quickly. In other words, if a module or an important subproblem is developed, does it have any value to the user or does the entire system have to be constructed before it can be of any use? For example, if the system has a large model that is of prime importance and the user interface and other components are considered secondary, the development of the model would have incremental value to the

user. For the class of projects that have incremental value, the appropriate design strategy would be either priority, iterative or adaptive design. All three of these methodologies advocate constructing a partial system based upon an important component of the system. One of the benefits of the iterative design is that it enables the user to have, in a short time span, an important module of the system developed. Priority design and adaptive design, the alternative methodologies that could be employed, focuses on first developing the more important items. By focusing on the higher priority items, a system that can be of some incremental value to the user is developed promptly. Prototyping should not be used, because it does not guarantee that the prototype will be of any value to the user.

Returning back to the incremental value issue, if the user does not need a module or portion of the system working quickly, any of the four methodologies, disregarding SDLC could be employed. In a relatively simple environment any of the methods would be effective, with no one methodology being more beneficial. It should be left to the discretion of the developer to decide which methodology he would prefer to employ.

## ***Conclusion***

The operating environment and the nature of the task at hand play a vital role in determining which design methodology is most appropriate. The previously established design methodologies have a number of differing characteristics that make each of them a more suitable strategy for certain environments. However, in some environments the current methodologies present certain limitations or unnecessary expenditures. These limitations suggest the need for an alternative methodology. A new methodology, Priority Design, was proposed to meet this need.

To determine what methodology would be most effective in a given situation an analysis of the operating environment must be performed. Such issues as project complexity, project uncertainty, and limited user involvement must be addressed. A guided search to assist in the analysis was presented.



## **Chapter 4: Case Study Analysis**

### ***Introduction***

This chapter contains a review and analysis of the design approach used in three DSS development cases, each with a different operating environment and employing different methodologies for developing the decision support system. The three cases that are addressed include:

"A Decision Support System for Tuition and Fee Policy Analysis," (Greenwood, 1984)

"The Eastern Manufacturing Company," (Bennett, 1983).

"Decision Support at Conrail," (Hoover, 1983)

For each of the above cases, a synopsis is first given of the operating environment and the steps that were employed in the development of the DSS. Following the synopsis, the environment and the characteristics of the project are analyzed, according to the guidelines recommended in Chapter 3, to determine if another development methodology may have

been more appropriate. It should be noted that the analysis is based on limited available information and several assumptions are drawn from the limited information. In addition, the analysis benefits from information that was determined after the system was implemented.

## ***Case #1. A DSS for Tuition and Fee Policy Analysis***

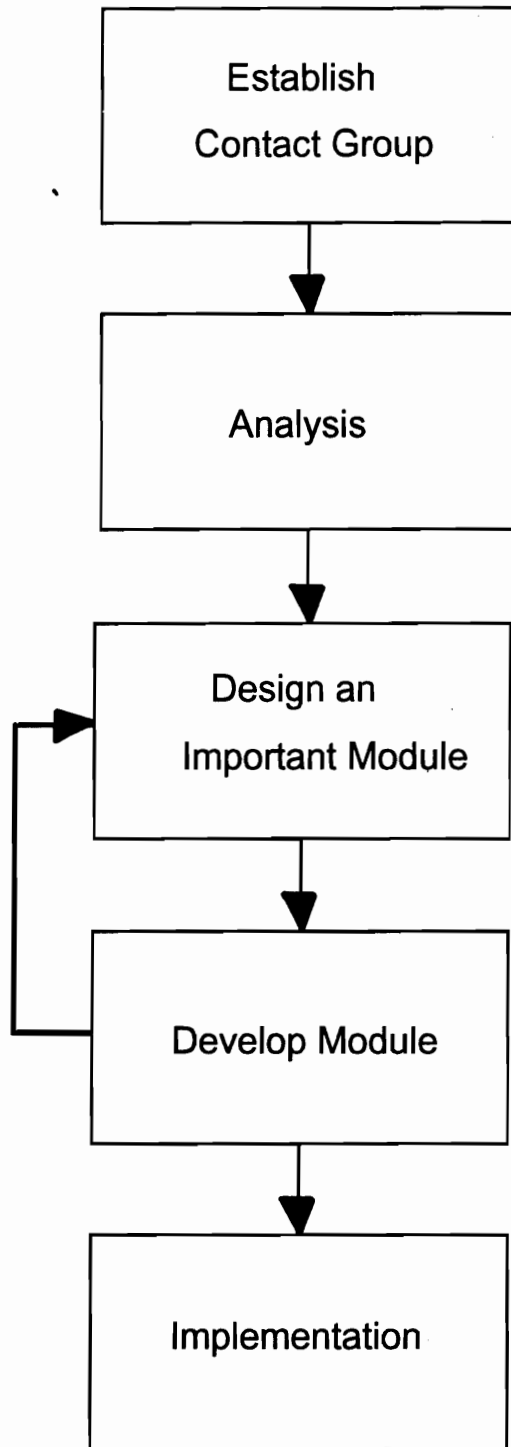
### **Introduction**

The DSS for tuition and fee policy was designed, developed and implemented at Virginia Polytechnic Institute and State University, a large university with an enrollment of approximately 22,000 students. The DSS was designed to (1) aid the university's administration in effectively setting their tuition policies, (2) generate and evaluate alternative plans, (3) perform sensitivity analysis, and (4) determine the impact and analyze the effect of policy changes (Turban, 1990). Prior to the construction of the DSS, the Office of Financial Planning and Analysis had developed a series of financial planning models employing the spreadsheet-oriented language IFPS (Interactive Financial Planning System), to aid them in their decision making process. After using the models, it became evident that the process could be categorized as a multicriteria resource allocation problem. Subsequently, it was decided that a goal programming (GP) model would be used to address the problem.

## **The Design Methodology**

The design methodology employed in the development of the DSS for tuition and fee analysis was an iterative approach similar to the methodology proposed by Courban. The DSS did not begin with a set of preconceived detailed plans for its development, but began with the automation of the existing models and evolved into a large, comprehensive DSS (Turban, 1990). The developer determined that the modular structure of the DSS indicated that iterative approach would be most appropriate. The steps used in the development cycle of the DSS are given in Figure 4.1. The first step employed in the development cycle was to determine a contact group, consisting primarily of members from the administration who would support the project and aid in the development process.

After an initial meeting with the contact group, the next step in the development process was to establish the modeling aspect of the DSS. A GP model was used to assimilate the university's tuition philosophy and policy into a mathematical form. In order to facilitate accurately establishing the model goals, a number of meetings with the administration was conducted. During the development of the model, it became apparent that "a comprehensive interactive on-line decision support system would be a necessity if the GP was to be used" (Greenwood, 1984). Once the goals were established, the design and development of the DSS process were initiated. The construction of the DSS, took on a modular format, in which small manageable portions of the DSS were constructed and tested before being integrated into the overall system.



**Figure 4.1: Design Steps in the Development of the Tuition and Fees DSS**

"The philosophy used in the development process was to have the large complex system evolve from a series of small operable and proven components. The system's development was iterative in that quite often the components had to be modified after they were integrated into the system. Modification was necessary because all of the requirements of the system could not have been foreseen initially and improvements were identified as the development process progressed." (Greenwood, 1984)

Once construction of the DSS was completed, the system was then demonstrated, with a hypothetical example, to the contact group. During the demonstration, several important modifications were suggested by the group. After the changes had been incorporated into the system, another demonstration was given, except this time, an application of a real problem was used during the presentation. One problem became apparent, the system needed to have the capability of handling two sets of enrollment data. Upon completion of the changes suggested, the DSS was then ready to be implemented.

## **Analysis**

### **The Tuition and Fees Operating Environment**

Although the construction of the DSS was successful, in reviewing the iterative design methodology used, another method may have been more appropriate and would have reduced the number of modifications that occurred after the DSS was constructed. In determining which methodology would have been most appropriate, the operating environment must first be evaluated as described in Figure 3.8. The university administration was willing to be involved in the decision making process and was able to commit time needed to periodically evaluate the system. Therefore, user involvement could not be considered as a limiting factor in the development process. The next issue that needs to be addressed is the uncertainty of the project. When determining the

uncertainty of the project three contingencies need to be evaluated. These three contingencies proposed by Davis and Olson (Davis and Olson, 1985), and McFarlan (McFarlan, 1981) include:

- (1) Degree of stability - The degree of stability associated with the task.
- (2) User task comprehension - The degree to which the user understands the characteristics of the project and comprehends the manner in which the DSS will be applied.
- (3) Developer task proficiency - This is the degree of training and experience the developer brings to the project. This includes both the developer's experience in the user's operating environment and his familiarity with the tools that will be used in the construction process.

The development of the DSS for tuition and fees appears to have operated in an environment with a relatively low degree of uncertainty. Although the project did not begin with a predefined set of plans, the task itself was stable. It appears that from the onset of the project, both the developer and users had significant knowledge of the project's goals. Many of the tools were already in place, and what needed to be accomplished was the development of a goal programming model to describe the university's tuition policies and to automate the existing models. At the start of the project, the developer had some degree of proficiency in all the languages used in the DSS. He was quite familiar with the budgeting process and the tuition and fee issues, had a good working knowledge of the university's computer system (Turban, 1990). Drawing from these project characteristics, summarized in Table 4.1, one can conclude that the project maintained a low level of uncertainty. Following the guidelines in Chapter 3 the suggested design methodology would be the priority design.

Table 4.1 Project Uncertainty

Characteristics for Determining the Uncertainty of the Environment	Tuition & Fees DSS	Eastern's DSS	Conrail's DSS
Degree of Stability	High	Low	High
User Task Comprehension	High	Low	High
Developer Task Proficiency	High	High	High

**Priority Design**

After analysis of the environment it was determined that the appropriate methodology was priority design. The following section describes the steps that would have been taken had the priority design methodology been employed.

Regardless of the methodology employed, the contact group would have been determined first. However, if the priority design methodology had been employed, the next step that would have been to establish the minimum requirements for the partial system. In this case the developer, having been familiar with the tuition process, probably had a general idea of the requirements for the system before the construction process began. However, these requirements should have been formalized in writing before the development process was implemented.

Once the requirements of the system were completed, a simulation of the DSS would have been developed. The simulation of the DSS would have forced the developer to clarify the requirements for the system, the method of presenting the interface to the user, and the capabilities to be provided by the DSS. In addition, the simulation would have allowed the user to have an idea of the overall appearance of the system and to make contributions, early in the development process. For example, not until the working DSS

was first demonstrated to the contact group was it revealed that the procedure for handling the enrollment data was very confusing. If a simulation had been presented, this problem would have become apparent before the construction had begun. This problem may have only required minor changes in the programming, but other modifications may have arisen that would have required major changes in the overall system. This is an important aspect of the design process, the simulation enables both the user and the developer to flesh out the characteristics of the project and aids in reducing the problems that will arise once the construction process has begun.

The next step in the development process would have involved the construction of the user's priority list of system requirements. This would have enabled the users to clarify what capabilities they consider most important and allowed the developer to focus on the key aspects of the system. In the development of the tuition and fees DSS, instead of the system just "evolving into a large, comprehensive, yet user friendly DSS" a detailed plan of the requirements should have been constructed. Time spent on constructing the overall system may have been reduced by focusing the construction process on the key aspects and giving the developer a clear plan of the entire desired system.

The subsequent stage of the priority design methodology is for the developer to evaluate the user's priority list. The developer would present a critique and evaluation of the list from a designing viewpoint. The developer should convey an idea of the cost in terms of both time and money required to complete the items of the list. He should also make recommendations for enhancements to the system and clarify if certain aspects of the project must be placed higher on the list. The user may be unaware that certain aspects of the DSS may be dependent on the completion of another part of the system, that was unknowingly placed lower on the list. For example, if the user had placed a specific model high on the list, and the generation of the data employed in the model low in the list, the



developer should clarify the importance of the data generation and give it a higher priority. Although in this case, the cost was in terms of time only, the developer needs to convey all of these aspects to the user, so that the user can make an informed decision on the final priority list.

The next stage is the partial development of the system. The modularity of the system makes this case ideal for developing a partial system. In this case, the builder may have developed the system using a modular format focused on his own preconceived priority list. However, the development should have followed a modular format based on the user's formal priority list. The developer's idea of what is most important to the success of the system does not always coincide with the user's perspective of the important system characteristics. The user's priority list has taken into account both the developers' and the users' viewpoints. Thus, the key aspects of the system are constructed first and can be presented for evaluation, and if applicable can begin to be implemented. If the partial system is working and meets the preliminary needs of the user, the partial system may be implemented while the developer continues to complete the remaining items on the list that were not originally considered of high importance.

## **Conclusion**

In analyzing the environment of the decision support for tuition and fees, it can be argued that the priority design methodology would have been more appropriate than the methodology actually used. By employing the priority design methodology, additional time would have been spent up front, determining the user's requirements, in order to reduce the time spent on making alterations to the system later during the construction

phase. The added time spent on determining the administration's requirements would have outweighed the time saved during the construction process. Although the decision support system was ultimately successful, the overall development time might have been reduced by employing the recommended priority design methodology.

## ***Case #2 The Eastern Manufacturing Company***

### **Introduction**

Eastern Manufacturing Company, with \$4 billion in sales, produces heavy machinery for other manufacturers in production and transportation. The company needed a system to aid them in their procurement planning activities. Management requested a procurement decision support system that would support them in two key issues: (1) maintenance of the decision making power at the local or plant level, and (2) sharing information and experience among the buyers. Eastern's top management approved of the system as a two-year project, with targeted savings of at least one-tenth of one percent on one fourth of the purchased items. Once approval was obtained, a supervisor and two program analysts were hired to develop the system (Bennett, 1983).

## **The Design Methodology**

The development process followed a modified version of the traditional development life cycle. The first steps that the development team employed were to assess the user's information requirements and to design potential screen layouts (Bennett, 1983). The team then proceeded, in isolation, to write the programs and test the resulting system. The design approach diverged from the traditional method in that the system was actually developed and released to users in increments. Although it might appear from the steps listed above, that the process was an iterative one, it did not take on the form of Courban's iterative design. In the development of Eastern's DSS, little attention was given to obtaining the necessary feedback and clarification of requirements when the different increments were delivered. "This incremental approach actually failed in practice, since no notification was ever given to user that new screens were available or that existing ones had been updated to correct bugs or provide new capabilities," (Bennett, 1983).

## **Analysis**

### **Eastern's Operating Environment**

According to Moore, who later analyzed the resulting system, it appears that the system evolved into a structured MIS, rather than the DSS that was originally intended (Bennett, 1983). A DSS focuses on supporting decisions with ad-hoc analysis and modeling capabilities, rather than on efficient information retrieval. Eastern's system was an on-line information retrieval system, not a personalized DSS. The reason the system

had migrated from its original intent was due to the traditional life cycle approach that was implemented (Bennett, 1983).

To determine which design methodology would have been more appropriate the operating environment must first be evaluated as illustrated in Figure 3.8. User involvement did not appear to be a limitation on the project. Since the users were using the incremental system that had been developed during the development process, it probably would have been feasible for the developer to receive periodic evaluations and suggestions for the system. In addition, management was very enthusiastic about the project, which further supports the hypothesis that user involvement was not a limiting factor.

Another issue that must be addressed is uncertainty of the project. As described earlier, the project uncertainty is based on three factors, degree of stability, user task comprehension, and the developer task proficiency. While it was clear that Eastern's management desired a strategic system to assist them in their procurement planning process, it was not clear at the onset of the project, which tasks were to be part of this project. Both the degree of stability and the user task comprehension was low. The programmers were knowledgeable of the tools required for the project. However, the experience of the supervisor on projects of this magnitude was limited. Moore states that the "designer adequately treated the technical system details but failed to address the problem of system introduction - identifying key interests and objectives and obtaining management commitment to the specific design - and was subsequently transferred to another project after the design work was completed," (Bennett, 1983). This failure may have been due to the inexperience or ignorance on the developer's part. Taking all of these factors into account, one can conclude that the project was highly uncertain.

Assuming that there was some uncertainty, the next factor that must be evaluated is the complexity of the project. The project complexity is defined by four components: project size, number of users, volume of new information, and the complexity of new information. The project was large, employing a supervisor and two developers for an estimated project time of two years. It is not clear if the volume of information was large or if it was complex to develop. However, it appears that there was a large amount of new information being shared between the many users including at the plant and local level and amongst the buyers. In this case it would appear that the system was complex. Having completed the evaluation of Eastern's environment, the suggested design methodology would be priority design.

### **Priority Design**

The first step in the priority design methodology is to establish the minimum requirements of the partial system. The developer determined the users' information requirements, but failed to discover what requirements were most important. This is an important component because the resulting system became an on-line retrieval system, with no mechanism for answering the "what if" questions for which the system was originally intended. It is important for the developer to get an overall picture of the required system, not just the information that is required.

The second step of the priority design methodology is to develop a simulation of the system. Although, the developers designed the potential screen layouts, this was not enough. The simulation should indicate most of the features that would be available to the user, including the capability of addressing what if questions. The actual mathematical

models to do this should not have been constructed, at this point. But, a simulation of the results should have been presented to the users. The simulation would have enabled both the users and the developers to get a more complete understanding of the desired system.

Having seen a simulation of the system, the users would now have been prepared to develop a priority list of their requirements. This list would enable the user to clarify what capabilities they consider most important and allow the developer to focus on the key aspects of the system. This step plays an important role in the outcome of the overall system. The developers, in this case, focused their development energies on computer efficiency issues, which is normally required by large data bases. However, the system was originally intended to be a strategic one, development should have shifted "from the purchase of specific parts to the procurement of commodities or classes of parts, such as molded rubber parts or fasteners" (Bennett, 1983). If the developers had received a detailed list of the requirements with the different priorities, they might have seen the necessity of focusing their energies on the overall planning activities and not on providing access to the large volumes of data. The developers had designed the system based on their own priorities and not the users'.

The next stage is for the developers to evaluate the users' priority list. At this stage the developers should have conveyed to the user, that in order to meet the desire for rapid response times, some flexibility in the design may have to be given up. In addition, the developers should have given the users any recommendations that might enhance the system or clarify some of the problems that may arise, when implementing their suggestions. Having obtained the advise from the developers, the users might want to reevaluate their list. At this point, the users might have been able to further convey that the processing of the data was not the highest priority, and might have given the

developers further details on their requirements. The users might have been able to clarify their need for a strategic system.

The subsequent stage is the partial development of the system. The development should focus on the important aspects as defined by the user, with the developers' suggestions taken into account earlier during the revisions of the priority list. The system was initially developed in increments. However, now the developer should deliver a partial system based on the user's list. Development should focus on the higher priority items. In addition, the developers should obtain feedback and clarification after the partial system had been implemented. The developers clearly failed in this area. The evolution process should be user driven, not developer driven. The partial development should continue until a system that meets all of the users requirements and requests is complete.

## **Conclusion**

The developers, in not correctly evaluating management's requirements and plans for the system, constructed a system that was more of an MIS than the DSS that was desired. The traditional development strategy that was employed was too rigid and did not provide a suitable platform for the developers to fully determine management's requirements. By evaluating the environment, it was determined that the priority design would have been a more appropriate design strategy. The priority design methodology would have provided the necessary mechanism for the developers to accurately determine management's needs. The priority design methodology forces the developer to spend more time in analyzing the user's requirements before the construction process begins, rather than making alterations during the construction phase when it may be too late.

Thus, by employing the priority methodology the desired DSS would have been developed and not the limited MIS system that was ultimately constructed.

### ***Case #3: Decision Support at Conrail***

#### **Introduction**

Consolidated Rail Corporation (Conrail), was created by Congress as a for profit corporation to organize and revitalize most of the railroad freight service of six bankrupt railroads in the northeastern United States. Conrail, in operation for less than four years, was faced with a new competitive market; Congress was preparing to reform the regulatory structure of the railroad industry through the Staggers Act of 1980. To prepare for that environment, Conrail required a marketing decision support system that would help place the company in the forefront of the changes in the transportation market.

The marketing analysts needed a system that would enable them to make a hypothesis about a marketing opportunity and then measure its potential effect by using historical data on profitability and volume of business. The marketing department was looking for a system that would drastically reduce the typical turnaround time for historical extractions. In short, the department was requesting an interactive, efficient, user-friendly, large-scale decision support system that was compatible with existing software.



## **The Design Methodology**

The design methodology employed by Conrail in the development of their marketing DSS does not accurately match any of the well-known design approaches, but many of the steps closely resemble those involved in prototyping. However, the development steps illustrated in Figure 4.2, followed more of an ad-hoc procedure than a formalized methodology. The first step involved analyzing the requirements for the system. The marketing department wanted a system that could model the effect of a particular pricing strategy on the company's business and felt that a DSS would accomplish this. In order to accomplish this the department determined the environmental factors related to data management that were needed. The factors included the following:

- one year of detailed historical revenue data contained 4 million to 5 million records
- each record contained 100 fields of data
- more than 30 fields were commonly used for selection criteria
- approximately 10 concurrent users of these data would be expected during normal business hours
- resulting selection volumes would be known in advance of queries.

The primary data manipulation features requested were as follows:

- user-defined computations
- automatic data reduction (sorting and summarizing)
- data combinations from multiple sources
- full range of data presentation formats (tables, graphs, labeled statements, etc.)
- data exchange interfaces to other analytical software tools
- ease of use by personnel without data processing experience.

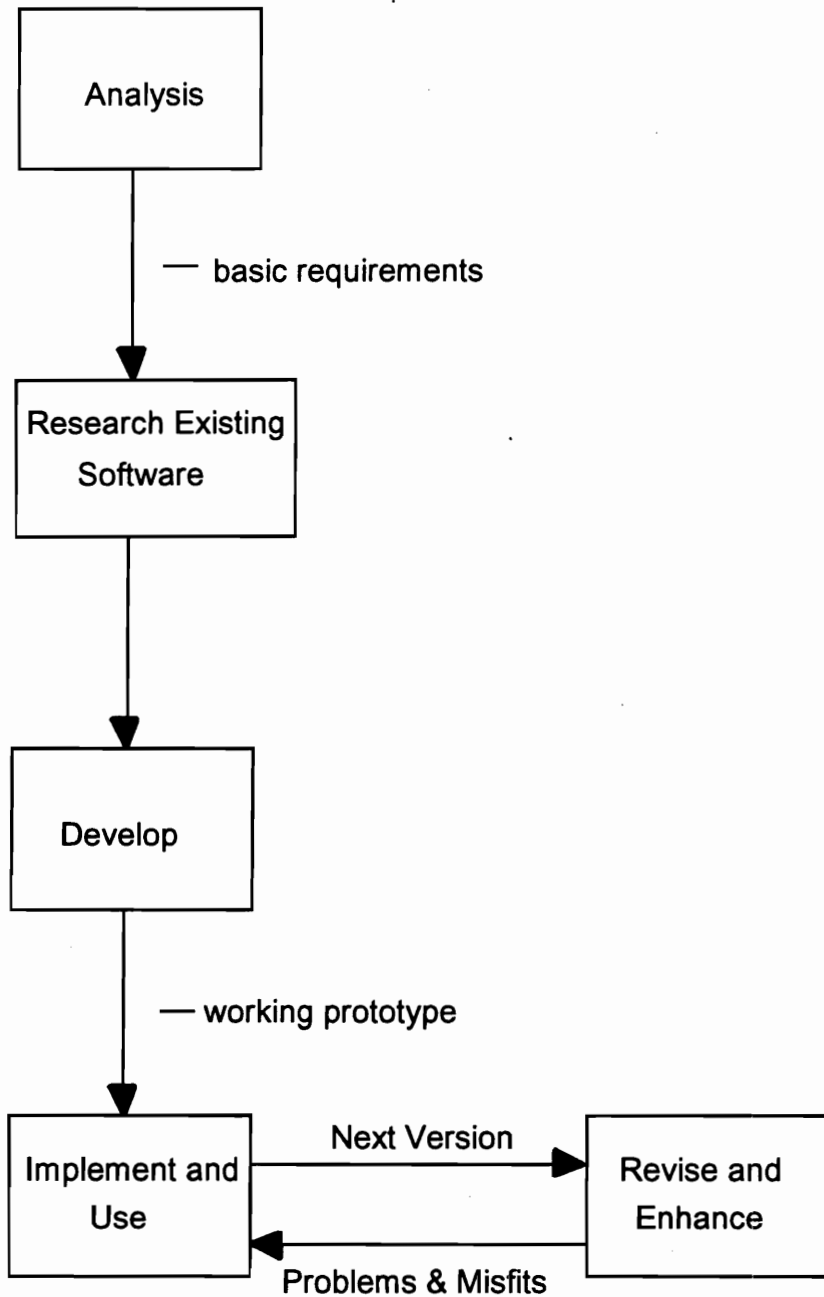


Figure 4.2: Design Steps in the Development of Conrail's DSS

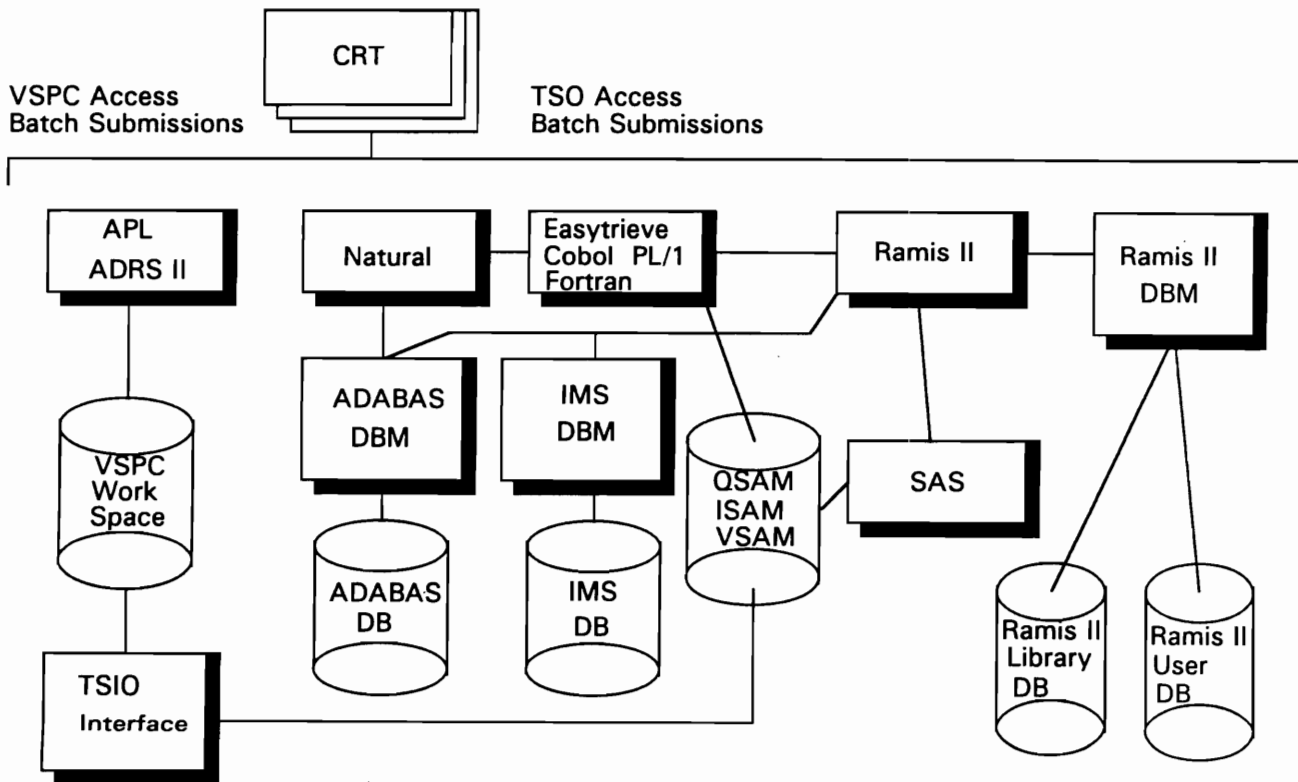
The second step was to research the current software packages that were on the market for the desired functions. After finding that many of the desired functions were distributed amongst several packages, the marketing and systems personnel decided to try interfacing several software products. Figure 4.3 illustrates the network of software Conrail ultimately laced together. Where available, standard interfaces between products were purchased from the software vendors and any additional interfaces were developed at Conrail using simplistic techniques.

Between December 1980 and April 1981, Conrail implemented the third step in the design process by developing a prototype decision support system with the software products. The prototype featured optional prompting menus, utilities, help screens, and interfaces to other products. In May 1981 the new hardware ordered for deregulation studies arrived and a full, 5 million record database was loaded.

## **Analysis**

### **Conrail's Operating Environment**

Although, the development of Conrail's DSS was successful, some of the obstacles that were encountered during the development process may have been avoided if a more formal design methodology had been employed. In determining which methodology would have been appropriate for Conrail's DSS, the project's characteristics and environment must be evaluated as described in Chapter 3. The marketing department



**Figure 4.3: Conrail's Interactive Computing Software**

Source: Hoover, Thomas B., "Decision Support at Conrail", *Datamation*, June 1983, pg. 222.

realized the importance of the DSS and appeared willing to devote its time to the development process. Therefore, user involvement was not a limiting factor in the development process.

The next issue that must be addressed is the uncertainty of the project. The uncertainty is measured by the degree of structure, user task comprehension and developer task proficiency. It appeared that the project maintained a low degree of uncertainty. From the manner in which the different software packages were evaluated, it appeared that the developer had a clear understanding of the problem from both the user's perspective and the development perspective. It seems he was highly knowledgeable of the tools to be used and was clear on the steps involved in the development of the system. It also appeared that the user had a good understanding of the characteristics of the desired system.

The next issue that needs to be addressed is the complexity of the project. Although, the time spent on developing the project was not large, approximately six months, the size of the system and the complexity of producing new information described earlier, categorizes this project as complex. Given the nature of the problem environment the most suitable design methodology would be priority design.

### **Priority Design**

The first stage in the priority design methodology is to determine the minimum requirements of the desired partial system. It appears that Conrail had successfully implemented this stage. Although the methodology employed in determining the requirements was not a formalized process, the developers were knowledgeable of the

user's requirements. This is evidenced by the rigorous search that was employed to find suitable software and linking several products to develop the desired interface.

Once the developers had a clear idea of the marketing department's requirements, the next phase would have been the development of a simulation prototype. Conrail had developed a prototype. It was not a simulation and employed the software that was purchased for the system. A simulation prototype has several advantages over the construction of a working prototype. One of the advantages of the simulation is that it would have allowed the developers to demonstrate the user interface to the marketing department before making any of the necessary software purchases. If the simulation was successful and met the approval of the department, then the necessary software could have been purchased. Another advantage of the simulation prototype is that it could have been constructed in a much quicker time frame than the actual prototype. Although as a general rule prototypes should be constructed quickly, Conrail spent five months developing the prototype. If the prototype did not meet the needs of the users, five months of work would have been invested needlessly. The simulation could have been developed in a week with very little cost and would have enabled both the developer and the marketing department to clarify the requirements for the system.

The next phase in the development process would involve construction of the user's priority list of the system requirements. The developers had generated a list of the database requirements and the data manipulation features requested. However, the desired system was more than an MIS or a large database management program. A DSS was desired. The marketing department required a system that would enable them to analyze the potential effect of a variety of pricing strategies. The list of requirements did not reflect this goal. In addition, the requirements should have been given priority levels in order for the developer to have a clear understanding of what the marketing department

felt was most important to the success of the DSS and what items were considered secondary.

The subsequent phase of the priority design methodology is for the developers to evaluate the user's priority list. At this stage the developers should critique the list for any shortcomings and evaluate it from a development viewpoint. The developers should also provide a cost estimate for the various items on the list and an estimate of the cost of the overall project. The system Conrail developed required over \$4 million of computer expansion in order to meet the needs of the marketing department. The department should have an idea of the cost of their requests and an idea of where their money was being spent. With this knowledge the marketing department may have considered that the cost involved did not justify some of their requests. This stage also provides the developers an opportunity to make suggestions, the marketing department, may not have considered.

The next phase is the partial development of the system. The development should focus on the key aspects of the system as defined by the user. It appears that in some aspects that the developer did construct the system focusing on the some of the more important components. It is unclear if the developers decided which items to focus the construction effort on or if the users were able to define what they needed most. The developers first focused their construction effort on the ad-hoc database. Then, while the system was employed, the developers constructed the production systems. However, when they started developing the production systems in the same database as the ad-hoc queries response time began to slow. The system could not handle both the ad-hoc files and the production files in the same database. In order to remedy the problem, the files were divided into two databases. Another problem that arose during implementation resulted from some of the users unknowingly making requests that would require the

selection and processing of millions of records. This resulted in the terminal being tied up for hours and it seriously degraded the response time for other users. To resolve the problem, the software vendor modified the selection process to pause after processing the index and inform the user how many records would be retrieved if he were to continue. Both of these problems probably would have still arisen had development clearly focused on the marketing department's highest priority items. These problems would only arise when the system was employed and could not have been foreseen by a simulation or by the partial development of the system based on the higher priority items.

## **Conclusion**

In analyzing the Conrail's environment, priority design it would have been a more appropriate development methodology. Conrail's development strategy did not follow any formalized methodology. Although, it appeared that the developers' were well organized and had a clear idea of the development steps that needed to be employed; by following a formal methodology the development process would have clarified both the developer's and management's responsibilities. The priority design methodology would have provided management, once they had been informed of the cost of the system, with the opportunity to make alterations to the system requirements before the construction process began and before any software or hardware was purchased. Fortunately, for Conrail the software did match the needs of the user and was not purchased needlessly. The problems that arose during the development were encountered during the implementation phase and may not have been alleviated if the priority design methodology had been employed. However,



Conrail took some unnecessary risks in using an informal methodology, that would not have occurred had the priority design methodology been implemented.

### *Summary*

Three cases, with differing operating environments were analyzed to determine the most appropriate design methodology. Table 4.2 summarizes the three methodologies and the design methods suggested after analysis. Included in the table is a list of the benefits or changes that would have occurred had the recommended methodology been employed. From the table, it is evident that the design methodology plays an important role in the successful completion of a DSS. Significant problems may have been alleviated had a different, more appropriate strategy been employed. By first analyzing the environment, an appropriate strategy can quickly be determined.

**Table 4.2 Design Methods**

<b>DSS</b>	<b>Methodology Employed</b>	<b>Recommended Methodology</b>	<b>Benefits</b>
DSS for Tuition & Fees	Iterative	Priority Design	Reduction in number of final alterations
Eastern Manufacturing's DSS	SDLC	Priority Design	clearer understanding of the user's needs
Conrail's DSS	Ad-hoc / Prototyping	Priority Design	clarification of user's and developers responsibilities

## **Chapter 5: The DSS for Nurse Scheduling**

### ***Introduction***

This chapter presents an overview of the development methodology used in the construction of a nurse scheduling decision support system for Montgomery Regional Hospital. The chapter opens with a discussion of the Montgomery Regional Hospital's scheduling policies and its operating environment. This is followed by an evaluation of the hospital's environment, as proposed in Chapter 3, to determine the most appropriate methodology for developing a decision support system. The last portion of this chapter contains a detailed account of the steps followed in implementing the Priority Design Methodology for the nurse scheduling DSS.

## ***Montgomery Regional Hospital***

Built in 1971, Montgomery Regional Hospital (MRH), located just outside Blacksburg, maintains approximately 150 beds. Its size is small enough to enable the hospital to provide personal attention and care, while still providing the specialty services of a major medical center. With over 80 physicians on the staff, MRH provides a large range of specialties, ranging from obstetrics to ophthalmology, from neurology to urology. MRH is a modern, well-kept facility with state of the art equipment. The facility includes an emergency department with physicians on duty 24 hours-a-day, an intensive care unit for the seriously ill patients; a progressive care unit for patients who have "graduated" from the intensive care/coronary unit; and many other traditional units.

### **Nursing Classifications for Montgomery Regional Hospital**

Nurses have become a scarce resource in the United States today, and in order to entice more people into the nursing field, MRH has developed a number of different job classifications to better fit the diverse lifestyles of its employees. Another benefit to maintaining a variety of job classifications is that the hospital is able to fill in those positions that are often considered undesirable to the rest of the nursing work force. This section reviews the particular classifications used at MRH to demonstrate its need for creativity and better scheduling practices.

## **The Baylor Plan**

The Baylor Plan, is one job classification employed by MRH which appeals to those people who would prefer to work weekends. The nurses on the Baylor plan are committed to working a total of 50 weekends a year and are required to work two twelve-hour weekend shifts each two-week pay period plus an additional eight-hour shift during each week. The greatest benefit to the nurse on the Baylor plan is that the employee is paid as if he was working 80 hours per pay period rather than the actual 64 hours worked. Although this plan is more costly, it enables the hospital to increase its weekend staffing without requiring additional weekend coverage by the existing staff.

## **PRN Staffing**

The PRN personnel are those nurses who want to work occasionally and work only when the hospital has a shortage of nurses. Their position is very similar to that of substitute teachers in the school system. The purpose of PRN staffing is to provide the hospital with additional staffing during high patient care activity periods and to provide coverage for absent nursing personnel. The PRN staff are required to work at least once every two months to be maintained on the roster. The personnel may work completely flexible hours that can be scheduled in advance or "on call." The base salary for those on the PRN plan is five percent above the established scale for in-house personnel. However, benefits are not available to the PRN personnel.

## **LPN and RN**

The difference between a registered nurse, RN, and a licensed practical nurse, LPN, lies in the amount of formal education he has obtained. An LPN undergoes 13 months of training at a state certified school. This includes approximately 2,000 hours of specialized training, primarily dealing with bedside patient care. Upon finishing his education, an LPN must pass state exams to be licensed, and must continue to pass yearly exams to renew his license. On the other hand, a RN has two years of intensive training, as opposed to the 13 months required of the LPN. These two years, include over 3,840 hours of training, dealing with bedside care and management orientation. RNs have additional training from a management perspective and, depending on the school, they will have an associates or even a bachelors degree.

## **The Intensive Care Unit**

At Montgomery Regional Hospital, scheduling is done at the floor or unit level. Each unit's nurse manager is in charge of determining the work schedule for the corresponding nurses. Since the procedure for determining the schedule for each unit is similar and a level of autonomy exists between the floors, it was decided to focus on one particular unit of the hospital, namely the Intensive Care Unit (ICU).

The ICU is an eight bed, multi-service unit for adult and pediatric patients. It is comprised of five rooms, with ICU A designated as an isolation room. The purpose of the ICU is to provide care to adult and pediatric patients who are critically ill or injured and in varying stages of recuperation from diagnostic and therapeutic interventions.

The overall management of the ICU is the responsibility of the nurse manager. Figure 5.1 depicts the organizational structure of the ICU. The unit employs approximately twenty nurses and requires three nurses, two RNs and one LPN for each shift. An RN may be substituted for an LPN. But, this results in the inefficient utilization of a costly resource and is generally not recommended. The ICU currently develops the schedule by the traditional approach, in other words, by hand, each planning horizon or every four weeks. The nurse manager develops the schedule based upon the requested days off and the shifts that are preferred by the nurses. This task often takes many hours and is met with dissatisfaction by many of the nurses.

The ICU has not been able to establish a cyclical schedule due to the dynamic nature of the environment. Nurses are frequently changing their job characteristics making a consistent cyclical schedule impossible to follow. Thus, the nurses are not able to predict what their future work schedule will be with any level of accuracy. The nurses work primarily eight- or twelve-hour shifts, with the weekends primarily consisting of twelve-hour shifts. The shifts are divided into three eight-hour shifts, day, evening or night, or two twelve-hour shifts.

## **The Scheduling Policies**

The following scheduling policies are taken from Montgomery Regional Hospital Healthtrust policy number 603-17 and have been in effect since 1984.

- Personnel will be hired for straight evening, straight night shift or a rotating shift. No one will be hired for straight day shifts.

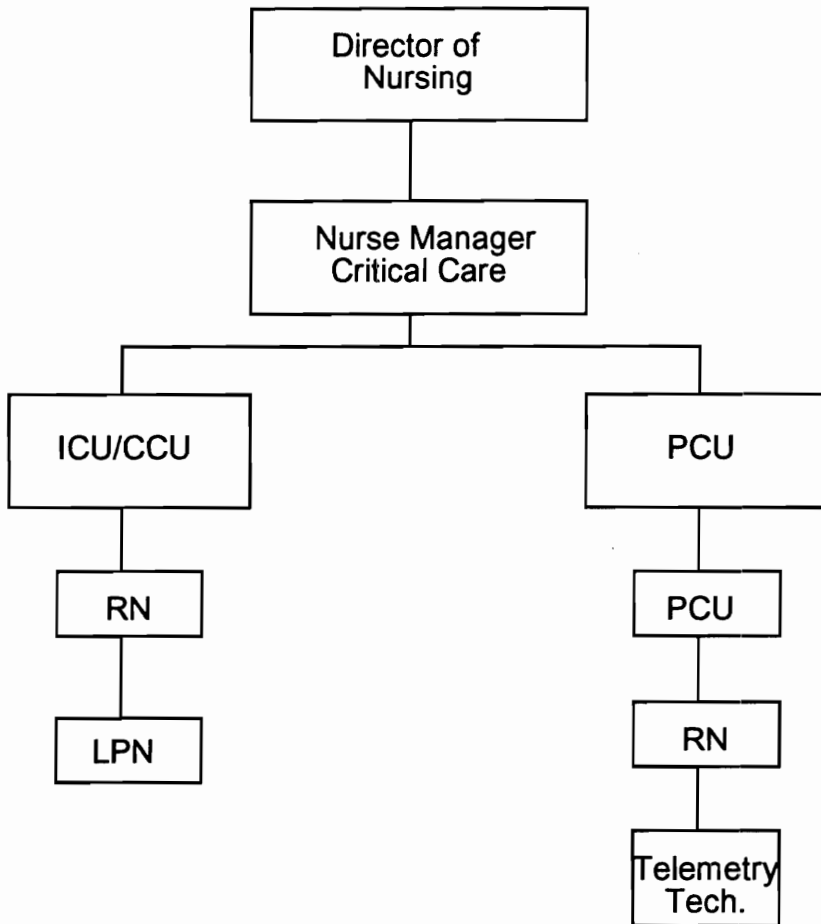


Figure 5.1 Critical Care Organizational Chart

- Personnel hired for permanent 11/7 shift will have "days off" when possible scheduled together unless requested otherwise by the employee.
- Personnel rotating shifts will not be scheduled more than two shifts within a pay period.
- Personnel will not be scheduled for more than seven consecutive work "days" without day(s) off, unless requested by the employee.
- To the extent possible personnel hired for permanent evenings or night shifts will not be expected to rotate shifts.
- Time schedules will be posted 2 weeks prior to the end of the current schedule.
- Weekends: (defined as Friday, Saturday, or Sunday)
  - a. Permanent nursing staff hired for evening or night shift may be scheduled every weekend off if staffing permits.
  - b. Personnel rotating between day shift and a second shift will be scheduled off at least every third weekend.
  - c. Part-time personnel will be expected to work at least two weekends out of every four.
  - d. Those personnel who fail to work their scheduled weekend due to unexcused absence will be required to work an additional weekend on the next schedule.
- Requests for days off:
  - a. Will be submitted in writing by the Monday of the second week of the current schedule.
  - b. Special requests will be granted according to the specific staffing needs of the unit, on an equitable basis.
  - c. Employees will receive a response from their Nurse Manager within two weeks of submitting the request.
- Holidays:
  - a. Must be taken within 30 days before or after the holiday.
  - b. Major holidays (Thanksgiving, Christmas, and New Years) will be covered on a rotating basis by unit personnel.



- Vacation Accumulation and Usage:
  - a. Full- time employees accrue vacation on the basis of 5/6 day or 6.7 hours of vacation per month worked commencing on the date of employment. After six years of continuous employment one additional day of vacation is accumulated for each additional year of continuous employment through the tenth year. Vacation is accrued according to Table 5.1.

Table 5.1 Vacation Schedule:

Employment Period	Vacation Accrual
During first 5 years	5/6 days per month or 6.7 hours/month
During 6th year of employment	11/12 days per month or 7.3 hours/month
During 7th year of employment	12/12 days per month or 8.0 hours/month
During 8th year of employment	13/12 days per month or 8.7 hours/month
During 9th year of employment	14/12 days per month or 9.3 hours/month
During 10th year of employment	15/12 days per month or 10.0 hours/month

- b. Vacation may only be taken once the employee has worked six months.
- c. Weekends requested off for vacation are determined by length of employment
  - During first 5 years of employment - 2 weekends/year
  - During sixth year of employment - 3 weekends/year
  - During tenth year of employment - 4 weekends/year

## ***The Development Process***

### **The Initial Meeting**

The Montgomery Regional Hospital was contacted to investigate the possibility of a cooperative project between their organization and the author. A project was proposed

which would lead to an improved personnel scheduling system for the hospital and which would advance the state of research in the area of personnel scheduling. Possibly this project would be the creation of a customized decision support system. The hospital administrators agreed to support a plan which would result in building a nurse scheduling system for one hospital unit with the intention of reviewing and expanding the system after successful implementation. Subsequently, the hospital administration arranged for a meeting between Carol Mackes, the nurse manager of the Intensive Care Unit (ICU), and the author.

At the initial meeting between the ICU nurse manager and the author, background for the project, goals and possible solution methods were discussed. The nurse manager explained the present scheduling system in the ICU, which is entirely her responsibility. Schedules for the twenty ICU nurses are prepared manually each four-week block. This process is performed on paper; there is no mechanism for quickly and accurately determining the number of vacation days, sick days, holidays, etc. each nurse had taken.

The nurse manager explained the features and characteristics which would be useful if a computer based system was created to help schedule the ICU's nurses. Her requirements relied on the successful, although limited, manual scheduling system already in operation as a base model. Several features expanded on those available with the current manual system. The author discussed how the requirements might be supported through a computer based decision support system.

The initial meeting also highlighted important factors to the successful design and implementation of a computer-based scheduling system. Most significant was the lack of computer support within the unit. The nurse manager did not have a personal computer available to assist her in the scheduling process. The nurse manager did foresee obtaining a computer, but was unsure of the hardware that would be made available to her. Since

the nurse manager may have limited computer capabilities, it was decided that the DSS would be constructed to have the capability of working on any of the machines which might become available. The second factor was the time constraints of the project. Although the hospital did not have a critical deadline for the system, there was a desire on the parts of both the hospital and the developer to have a useful DSS in a short time period. A DSS that assisted the nurse manager with the scheduling process would be nearly as useful as one that included a scheduling algorithm, but could be deployed faster and would be easier to port to a variety of equipment. In general, the system should reflect the needs of the user, in this case the nurse manager, and not the desires of the developer. It is, however, appropriate for the developer to help determine the necessary boundaries of the system.

To determine if a DSS is the appropriate tool for addressing the hospital's scheduling problem, Kroeber recommends answering four questions, before starting the design process. These questions, which are described in further detail in Chapter 3, include:

1. Is there a need for a DSS and will it enhance the decision maker's effectiveness?
2. Do the benefits of the DSS outweigh the cost of the development?
3. Is there time available for approval of the system?
4. Does a champion exist?

These questions were applied to the scheduling problem facing Montgomery Regional Hospital. In response to the first question, a DSS would enhance the effectiveness of the nurse manager by aiding her in developing a schedule that meets the needs of the hospital and is fair to the nurses. The DSS would allow the nurse manager to quickly verify that the nursing requirements are being met and provide up to date statistics on the number of weekends, and number of excused and unexcused absences, taken by each of the nurses.

This will enable the nurse manager to develop a schedule that fairly distributes the weekend shifts, holidays, vacations, etc. over the given year.

In response to the second question, the cost of developing the DSS to the hospital is minimal. The only real cost is time, with the hospital investing a little amount of time and the developer investing a significant amount. The time saved in maintaining the statistics and developing a schedule, will far outweigh the cost of the system. In response to the third question, there was ample time for the nurse manager to approve of the system. The nurse manager was willing to champion the project and devote her time in the development process. Carol Mackes was very enthusiastic about the DSS and was happy to periodically meet with the developer, during the development process, to discuss the different aspects of the system.

Once it had been established that a DSS was the correct tool for this application. The next step was to determine the appropriate design methodology that should be implemented. In order to accomplish this, a number of questions regarding the operating environment had to be answered first. These questions follow the heuristic search developed in Chapter 3.

### **Montgomery Regional Hospital's Environment**

In evaluating the environment, the first question that needed to be answered was the willingness of the user to devote time to the project. In the hospital's scheduling environment the user and the champion were one in the same. This being the case, it was found earlier, when determining if there was a champion, that the user/champion was willing to spend time with the developer. The next question that was addressed, was the

level of uncertainty in the development process. To measure the degree of uncertainty, three characteristics proposed by Davis and Olson and McFarlan, were identified. These characteristics include:

1. Degree of stability - the level in which the task is defined. A system that is considered well defined has a system definition that is fixed and is not subject to change during the development process.
2. User task comprehension - the degree to which the user understands the characteristics of the project and comprehends the manner in which the DSS will be applied.
3. Developer task proficiency - The degree of training and experience the developer brings to the project. This includes both the developer's experience in the user's operating environment and his familiarity with the tools that will be used in the construction process (McFarlan, 1981).

The DSS for nurse scheduling was moderately structured. Although the task of determining a schedule is a semi-structured decision, the goals and the structure of the desired DSS were fairly well-defined. The nurse manager had complete control over the development decisions. The nurses in the unit did not have to be conferred with to determine the different issues that were involved in constructing the DSS. In terms of the user task comprehension, the user, Carol Mackes had a large degree of knowledge on the scheduling process and after the initial meeting, had a clear idea of how the DSS would support the scheduling process. The final criterion is the level of the developer task proficiency. Although the developer had little to no experience in the hospital environment, the developer engaged in a large degree of research on the process of nurse scheduling, before entering into the development of the DSS. In addition, the developer was able to select the tools that would to be employed in the development of the system.

Thus, the developer was able to select a tool in which she was highly proficient. In evaluating each of these characteristics, it is safe to conclude that the development of the nurse scheduling DSS maintained a low level of uncertainty. Following the guidelines proposed in Chapter 3, and given the environmental characteristics of MRH, as summarized in Table 5.2, the recommended development methodology for the nurse scheduling DSS was the priority design methodology. Some of the benefits of employing the priority design methodology, which are stated in greater detail in Chapter 3 include:

- provides the user with the capability to view a simulation of the system before the design process begins;
- clarifies the goals of the project, early in the development process;
- enables the developer to focus on the most critical aspects of the DSS, as defined by the user.

These benefits made it advantageous for the DSS for nurse scheduling to be developed using the priority design methodology.

**Table 5.2 Environmental Characteristics for Determining an Appropriate Design Methodology**

<b>Environmental Characteristics</b>	<b>Montgomery Hospital's Operating Environment</b>
User InvolvementTime	Unlimited
Level of Uncertainty	Low

## ***The Priority Design Methodology***

### **Stage 1. Determining the Minimum Requirements for the Partial System**

The first stage in the development of the decision support system was to determine the minimum requirements of the partial system. A list of the requirements, determined by Carol Mackes, is given in Figure 5.2. The first requirement was for the system to provide an interface that was identical in appearance to the current scheduling sheet, shown in Figure 5.3. The sheet lists when the nurses work each of the shifts, in the planning horizon. A three character code was used to represent the time and length of a given shift.

The notation was as follows:

E-08

The first character represents the time of the shift:

D-Day, E-Evening, N-Night, or X if the worker cannot work the given shift.

The digits represent the length of the work shift in hours.

An empty cell indicated that the nurse was willing , but was not scheduled to work the corresponding shift.

The second requirement for the system was a database containing the nurses' personal data, including material such as address, phone numbers, etc. The database should also maintain updated records on the number of holidays, weekends, vacation days, etc., the nurse had taken off, during the year. Due to the dynamic nature of the environment, in which the nursing staff is constantly changing, the fourth requirement was a facility to easily add and delete nurses from the database and the schedule.

The hospital is required to maintain historic data on the schedules worked, indicating who actually worked each given shift. Thus, the final requirement was that the

**List of Requirements:**

1. An interface identical to current time sheet
2. A database containing information on the nurse's personal and work history including:
  - a. Full name
  - b. Address
  - c. Phone number
  - d. Number of vacation days
  - e. Number of weekends worked
  - f. Number of sick days
  - g. Number of holidays
  - h. Number of excused absences
  - i. Number of unexcused absences
3. A mechanism to save the worked schedules for insurance purposes
4. A facility to add and delete nurses from the database and the schedule

**Figure 5.2: Early List of Requirements**



	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	
RN-																													
Roberson	D	D		X	X		X																						
Metzer	N	N	N		D	D	E/	N	N	N	N	N	E/	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Morrison				N	N	N	N	N	N	N	N	E	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	X
Poole			D	D	D							D	D	D											D	D	D	D	D
Livasy	4- 8	E	E	E				D	4- 8	E	E				E	E	E	E	E										
LPN-																													
Jones	D	D	D		D								D	D															
Echols	D	D	D	D				D	D	D	D				D	D	D	D	D									X	X
Lucas	N	N	N	N				N	N	N	N				N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Radcliffe	N	N	N					N	N	N	N				N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Earnest				D		D						D	D	D															
Neeley				N	N	N						N	N	N												N	N	N	N
Yearout, Ron				N	N	N		D	D	D	D	D	D	D															
Yearout, Don			N		D	D					N	N	N	N															
D	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
E	3	2/ 3	3	3	3	3	3	3	3	3	3	3	3	3	2/ 2	2/ 3	3	3	3	3	3	3	3	3	3	3	2/ 3	3	2
N	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Figure 5.3: Montgomery Regional Hospital's Work Schedule

system would provide a mechanism to store the worked schedules for security reasons. The stored records would then be retrievable, in order to be reviewed at a future date.

## **Stage 2. The Simulation**

After the meeting with the nurse manager, a simulation of the user interface for the DSS was constructed. The simulation would allow the hospital to experience the user interface first hand and be aware of the information that was conveyed in the DSS. The hospital could then request any changes they might deem necessary, before the actual construction process began. Although, the ultimate system had limitations on the memory requirements, the simulation was not as restricted. Since it was not necessary for the simulation to be constructed on the same software as the construction of the DSS and the simulation only needed to be presented to the user, it was not required that the simulation be developed on tools that would be ultimately available to the hospital.

There were several different platforms that could have been employed to develop the simulation. Several different platforms were evaluated, including the use of a third generation language, user interface software, or a spreadsheet package. For the construction of the simulation, two aspects were of prime importance, the look of the interface and the time to develop the interface. The interface needed to be able to convey what information was going to be presented to the user and provide a realistic simulation of the actual DSS. Another aspect of moderate importance, that would affect the development process was the time involved in learning new software. Other issues that were not important in the selection of the simulation platform, but would play an important role in the platform used for the actual DSS were such items as: the calculation speed, size of the on-line data and security.

After reviewing several software programs from each of the different platforms including Turbo Pascal, Object Vision, Lotus 1-2-3, and Symphony; Lotus Symphony was found to be most appropriate. There were several benefits of using Symphony as opposed to Lotus 1-2-3. Symphony provides a unique windows environment, which enables the developer to construct work areas or windows that constrict the movement of the user to the selected area. Thus, preventing unauthorized users from moving into areas without the proper identification. Lotus 1-2-3 does provide a windows environment, but it lacks the capability of restricting the user's movement in the given working area. Symphony provides a unique windows environment that would allow a simulation to be constructed quickly, without a large amount of programming. In addition, the builder had previous consulting experience using Symphony and could quickly write the necessary code to develop the simulation.

If the simulation prototype had been developed in a third generation language, for example Turbo Pascal, a large amount of programming would have been required to construct the appropriate windows. One user interface that was evaluated, Object Vision, was rejected because it was not designed to support tables as a primary interface and would not be efficient to create a screen which would present a realistic view of the DSS. Another detriment of using Object Vision was the amount of time it would take for the developer to be familiar with the software package. Although this was not of prime importance it needed to be considered when selecting an appropriate platform.

As a result of employing Symphony, little time was spent on actually writing the code for the simulation. The majority of the time was devoted to determining the material that was to be presented and the corresponding user interface. Symphony also provided a mechanism to create an attractive interface with buttons and menus. Another benefit that

resulted by the use of Symphony was that the interface developed during this phase was able to be salvaged as part of the interface for the actual DSS.

The construction of the simulation took approximately forty-five man-hours to complete. It was necessary to consult with the nurse manager midterm in the production of the simulation to further clarify the user's needs. The resulting simulation displayed the various windows that would be available to the nurse manager, including the schedule, edit and data windows. The schedule window and edit window, shown in Figure 5.4 were very similar in format, except in the edit window the schedule was not write protected, in other words, changes were allowed in certain areas. The data window displayed information about the nurses. The formulas and the data for each of the nurse's work history were not developed as part of the simulation. But, the nurse manager was able to see what information would be available. A number of the programs that allowed the user to switch from window to window were available. The security feature of password requirement was also used to demonstrate which windows would require password access. In addition, the update program to store the previously worked schedule and replace it with the next schedule was available. The program for the simulation was written to indicate the questions the user would need to address and how the interface would be constructed, but the actual procedures for updating were not available at this point.

The simulation required writing approximately 60 lines of code. The majority of the development time was spent designing the varying windows that would be provided. The memory and space required for the development for the simulation was negligible.

**The Schedule Window:**

SCHEDULE													
01-31-93 TO 2-27-93													
NURSE	LEVEL	S-31	M-1	T-2	W-3	T-4	F-5	S-6	S-7	M-8	T-9	W-10	T-11
AKERS	RN	N-12			D-12	D-12			D-12	D-12		D-12	
BONNER	RN		N-8	N-8	N-8	N-8	N-8			N-8	N-8	N-8	N-8
CLARK	RN	E-8	D-8	E-8			D-8	D-8	E-8		D-8		D-8
DENNIS	RN	D-12			D-12	D-12		D-12	D-12			D-12	D-12
ELGIN	RN		E-8	D-8			E-8	N-12		E-8		E-8	E-8
FLASK	RN	D-8	E-8	E-8			D-8	E-8	D-8	D-8	E-8		
GRANT	RN	E-8	N-8				N-8	N-8	N-8	N-8	N-8		N-8
HENLEY	RN		D-8	D-8	D-8	D-8	D-8			E-8	E-8		
KLEIN	LPN	N-8	N-8			E-8	N-8	N-8				N-12	N-12
ADAMS	LPN		E-8	E-8	N-8			E-8	N-12	N-12	N-8		
LINKS	LPN	D-8	D-8	D-8			E-8	D-8		D-8	D-8		
JONES	LPN			N-8	N-12	N-12	E-8				E-8	N-12	E-8
SMITH	LPN	N-8		N-8	N-12	N-12			N-12		D-8	D-8	D-8
DAY		3	3	3	3	3	3	3	3	3	3	3	3
EVE1		3	3	3	3	3	3	3	3	3	3	3	3
EVE2		3	3	3	3	3	3	3	3	3	3	3	3
NIGHT		3	3	3	3	3	3	3	3	3	3	3	3

**The Edit Window:**

SCHEDULE													
01-31-93 TO 2-27-93													
NURSE	LEVEL	S-31	M-1	T-2	W-3	T-4	F-5	S-6	S-7	M-8	T-9	W-10	T-11
AKERS	RN	N-12			D-12	D-12			D-12	D-12		D-12	
BONNER	RN		N-8	N-8	N-8	N-8	N-8			N-8	N-8	N-8	N-8
CLARK	RN	E-8	D-8	E-8			D-8	D-8	E-8		D-8		D-8
DENNIS	RN	D-12			D-12	D-12		D-12	D-12			D-12	D-12
ELGIN	RN		E-8	D-8			E-8	N-12		E-8		E-8	E-8
FLASK	RN	D-8	E-8	E-8			D-8	E-8	D-8	D-8	E-8		
GRANT	RN	E-8	N-8				N-8	N-8	N-8	N-8	N-8		N-8
HENLEY	RN		D-8	D-8	D-8	D-8	D-8			E-8	E-8		
KLEIN	LPN	N-8	N-8			E-8	N-8	N-8				N-12	N-12
ADAMS	LPN		E-8	E-8	N-8			E-8	N-12	N-12	N-8		
LINKS	LPN	D-8	D-8	D-8			E-8	D-8		D-8	D-8		
JONES	LPN			N-8	N-12	N-12	E-8				E-8	N-12	E-8
SMITH	LPN	N-8		N-8	N-12	N-12			N-12		D-8	D-8	D-8
DAY		3	3	3	3	3	3	3	3	3	3	3	3
EVE1		3	3	3	3	3	3	3	3	3	3	3	3
EVE2		3	3	3	3	3	3	3	3	3	3	3	3
NIGHT		3	3	3	3	3	3	3	3	3	3	3	3

- The bolded area is not write protected

**Figure 5.4: The Schedule and Edit Windows**

### **Stage 3. Construction of the User's Priority List**

After seeing the simulation of the system, the user constructed a priority list, shown in Figure 5.5, which was essentially an extension of the minimum requirements that were previously established in the development process. One added feature that was high on the priority list, was for the system to provide an area for developing the schedule for the next planning horizon. The system would then be able to maintain two schedules, the current working schedule and the schedule for the next planning horizon.

Another feature that was overlooked in the initial simulation, that the nurse manager requested was a printing option. The nurse manager requested that the system provide a function to quickly print the nurse database, and the current and future schedules.

The initial simulation was constructed with the capability of allowing the schedule to start on any given day of the week, i.e. Monday, Tuesday, etc. The nurse manager, would indicate the starting date. However, during the simulation, the nurse manager assured the developer that the schedule always started on Monday and ended four weeks later on a Sunday. Therefore, the program did not have to calculate the corresponding days, the days could simply be hard coded into the system.

### **Stages 4 and 5. Developer Suggestions and User Reevaluation**

At this stage the developer evaluated user's priority list, Figure 5.5 and determined that all of the items were feasible and considered reasonable in terms of the required development time. If the system was to be a large DSS, with time and cost being important aspects of the development process, the developer would have given the

**List of Requirements:**

1. An interface identical to current time sheet
2. A database containing information on the nurse's personal and work history including:
  - a. Full name
  - b. Address
  - c. Phone number
  - d. Number of vacation days
  - e. Number of weekends worked
  - f. Number of sick days
  - g. Number of holidays
  - h. Number of excused absences
  - i. Number of unexcused absences
3. The maintenance of two schedules, the current and the future schedules
4. A mechanism to save the worked schedules for insurance purposes
5. A printing feature.
6. A facility to add and delete nurses from the database and the schedule

**Figure 5.5: Priority List**

hospital an itemized estimate of the time and cost of many of the items on the list. However, in this situation, there was no direct cost to the hospital for the DSS.

Also at this time the developer made recommendations to the user for other system enhancements. One suggestion of the developer was the establishment of a password feature that would prevent others, without proper authorization, from changing the database or the schedule. Another feature that was suggested is a notebook, or memo area, in which the nurse manager can write notes regarding the nurses and the shifts.

The developer also recommended including a nurses' preference ratio. This ratio could be used to measure how the nurses regarded their work pattern. However, the nurse manager rejected this concept. She explained that the unit was small enough for her to personally know the nurses and have a good idea on their individual preferences.

The final suggestion by the developer was a feature to keep track of the cost of the various schedules in terms of the type of nurses employed. For example, the use of nurses on the Baylor plan or PRN staffing is more expensive to the hospital than the full time LPNs and RNs. However, this feature was also rejected as unnecessary. The nurse manager, when developing the schedule, was aware of the additional cost of the different staffing levels, and took this into consideration.

If the DSS had the capability of developing a schedule for the nurses, the developer's suggestions on improving the system would have had more merit and played an important role in that scheduling process. However, since the DSS was focused on aiding the nurse manager in the scheduling process, these extra features that were recommended by the developer did not appeal to the nurse manager and were deemed unnecessary.



## **Stage 6. Partial System Development**

### **Hardware & Software**

At this stage, it was still unclear as to what type of computer support the ICU would obtain. Therefore, the DSS had to be constructed so that it could be run on any IBM compatible machine, with limited memory capacity. Early in the development process, this decision did not seem to play an important role. However later, as the system was expanded, it had a large effect.

There were many applicable software packages available for PC-based systems that could have been used to support the DSS. However, software that required a large amount of memory or a Microsoft Windows environment would not be applicable for the hospital environment. Several software programs, which were compatible with this limited environment were evaluated including, Turbo Pascal, Lotus 1-2-3, and Lotus Symphony. While contrasting the different options, it was determined that if the DSS was to be developed using Turbo Pascal the system would operate very fast, but would have a longer development time and be more difficult to modify than a spreadsheet environment. The top item on the user's priority list, required that the user interface resemble the time sheet. When evaluating the layout of the time sheet it appeared that it would be advantageous for the time sheet to be constructed in a spreadsheet environment, with the rows representing the nurses and the columns representing the individual days, during the planning horizon. The advantage of a spreadsheet is that when you change a value, all other related values are automatically recomputed. The nurse manager frequently makes several changes in the schedule, and these changes can be quickly recalculated in a spreadsheet environment. When comparing the spreadsheet environment

to a third generation language, it was decided that the development time and flexibility were more important qualities to the success of the DSS than the operational speed.

Once it had been determined that the DSS would be most effectively constructed using the a spreadsheet environment, the choices of software packages were narrowed to Lotus Symphony or Lotus 1-2-3. Symphony performs a number of tasks, including word processing, database management, and spreadsheet calculations, that were not available from some of the other software packages. As stated earlier in the simulation discussion, Symphony provides a unique windows feature not available in Lotus 1-2-3. Another benefit to Symphony, that was not important for the simulation but is relevant for the actual system is word processing. Symphony provides a word processor environment that will enable the nurses to write and print memos or notes. Word processing is not available with Lotus 1-2-3.

Symphony is a PC-based system that requires a minimum of 512K of random access memory (RAM). Symphony can access both conventional memory and expanded memory. While constructing the simulation, memory was not a problem. There was very little programming involved and the simulation was basically a shell of the DSS that would be constructed later. However, during the construction process of the DSS, it was found that the system was large enough to require at least 2MB of expanded memory. Symphony requires very few other additional capabilities. Symphony is capable of running on any type of monitor, black and white or color, does not require a mouse, and the required hard disk space is minimal.

## A Description of the DSS

When the nurse user runs the program, the initial screen is the current schedule in a spreadsheet environment as shown in Figure 5.4. The current schedule was selected to first appear, rather than a menu displaying the various options, due to the frequency with which the current schedule is examined and altered. The current schedule is evaluated on a daily basis and is the window that is most often needed. Once in this area, no alterations can be made to the current schedule. This allows the nurses in ICU to use the system to examine the schedule, but prevents them from making any unauthorized changes.

The display matches the ICU's current scheduling time sheet. The rows represent each nurse in the unit and the columns indicate the dates in the schedule. The notation in the cells follows the format previously described in the minimum requirements phase. The available starting times are as follows:

- Day shift starting at 8 am.
- Evening shift starting at 4 pm.
- Night shift starting at 8 pm or 12 am, depending in the length of the shift.

The length of the shifts can vary from 1 hour to 12 hours. Although, four, eight and twelve-hour shifts are most common, the system was designed to handle shifts of any length from 1 to 12 hours. The bottom rows of the schedule, indicate the number of nurses scheduled to work on each of the selected shifts. The shifts are divided into the following time frame:

- Day: 8 am - 4 pm
- Evening1: 4 pm - 8 pm
- Evening2: 8 pm - 12 am
- Night: 12 am - 8 am

The evening shift is divided into two components, because during the evening shift the eight-hour evening nurses are on duty the entire period but the twelve-hour day or night nurses are only on duty for half the evening shift. Therefore, in order to distinguish the number of nurses on duty throughout the shift it was necessary for the shift to be broken into two components.

In Symphony, there are two calculating modes, automatic recalculating and manual recalculating. In automatic recalculating, every time a change is made in the worksheet, the entire sheet is recalculated. This mode was found to be inefficient and time consuming, since the worksheet was large and changes were frequent. Therefore, the manual method was selected. Every time the user needs to calculate the correct number of nurses on the given shifts, the recalculate key must be pressed.

The system was constructed so that the various features can be run from a spreadsheet menu environment or with the use of the alt key and the established letters. If the user is unable to recall the letter that corresponds to the desired feature, a help menu with a list of the possible commands, shown in Figure 5.6, is available. The menu, displayed in Figure 5.7, appears by typing Alt-M, for menu. The second line in the menu indicates the various features available from the menu interface. The first line displays the corresponding submenu or a brief description of the selected or highlighted feature. The features available to the user include:

- **Add a Nurse to the Schedule and Database (Alt-A)** - This enables the nurse manager to add a nurse to the database and to the current and next schedules. When this option is selected, the user is prompted to insert the nurse's first and last name and his level, RN, LPN, etc. The nurse is then placed in the correct position in the schedule and database, based on the level and then alphabetically.

				01											
NURSE	LEVEL	S-31	M-1												
AKERS	RN	N-12													
BONNER	RN		N-8												
CLARK	RN	E-8	D-8												
DENNIS	RN	D-12													
ELGIN	RN		E-8												
FLASK	RN	D-8	E-8												
GRANT	RN	E-8	N-8												
HENLEY	RN		D-8												
KLEIN	LPN	N-8	N-8												
ADAMS	LPN		E-8												
LINKS	LPN	D-8	D-8												
JONES	LPN														
SMITH	LPN	N-8													
DAY		3	3	3	3	3	3	3	3	3	3	3	3	3	3
EVE1		3	3												
EVE2		3	3												
NIGHT		3	3												

Commands:

Alt Key

- A - Add a Nurse
- E - Edit Current Schedule
- G - Generate Next Schedule
- H - Home (Returns to Current Schedule)
- I - Nurse Database
- M - Menu
- N - Notebook
- P - Print Menu
- Q -Quit
- S - Save Changes
- U - Update (Replace Current Schedule)

F8 - Calculate

Figure 5.6: The Schedule with the Command Window

Edit the Schedules or the Database

**Edit** Home Notes Save Print Quit Update

SCHEDULE													
01-31-93 TO 2-27-93													
NURSE	LEVEL	S-31	M-1	T-2	W-3	T-4	F-5	S-6	S-7	M-8	T-9	W-10	T-11
AKERS	RN	N-12			D-12	D-12			D-12	D-12		D-12	
BONNER	RN		N-8	N-8	N-8	N-8	N-8			N-8	N-8	N-8	N-8
CLARK	RN	E-8	D-8	E-8			D-8	D-8	E-8		D-8		D-8
DENNIS	RN	D-12			D-12	D-12		D-12	D-12			D-12	D-12
ELGIN	RN		E-8	D-8			E-8	N-12		E-8		E-8	E-8
FLASK	RN	D-8	E-8	E-8			D-8	E-8	D-8	D-8	E-8		
GRANT	RN	E-8	N-8				N-8	N-8	N-8	N-8	N-8		N-8
HENLEY	RN		D-8	D-8	D-8	D-8	D-8			E-8	E-8		
KLEIN	LPN	N-8	N-8			E-8	N-8	N-8				N-12	N-12
ADAMS	LPN		E-8	E-8	N-8			E-8	N-12	N-12	N-8		
LINKS	LPN	D-8	D-8	D-8			E-8	D-8		D-8	D-8		
JONES	LPN			N-8	N-12	N-12	E-8				E-8	N-12	E-8
SMITH	LPN	N-8		N-8	N-12	N-12			N-12		D-8	D-8	D-8
DAY		3	3	3	3	3	3	3	3	3	3	3	3
EVE1		3	3	3	3	3	3	3	3	3	3	3	3
EVE2		3	3	3	3	3	3	3	3	3	3	3	3
NIGHT		3	3	3	3	3	3	3	3	3	3	3	3

Figure 5.7: The Schedule with the Selected Menu Feature

- (Alt-E) - This feature allows the nurse manager to edit the current schedule. The user is first prompted for the correct password before allowing any changes to be made. This prevents any nurses without proper authorization from altering the schedule. The areas that can be altered are shown in green. Areas that can not be altered, such as the days of the schedule, are indicated in white. The changes made in this area must be saved before they replace the previous schedule. This allows the manager, once having made changes the option of returning to the previous schedule without losing the original schedule.
- Generate (Alt-G) - This feature enables the nurse manager to edit or create the schedule for the next planning horizon. When this option is selected the user is prompted for the password. If the next schedule has not yet been created, the user is prompted for the starting date of the next schedule. At this point, if an incorrect date is entered, the system beeps to indicate that a wrong date was entered and prompts the user again for the correct date. A blank schedule then appears with the nurses' last names and levels on the first and second columns and the corresponding dates along the first row. If the schedule for the next planning horizon has been previously created, that schedule appears and allows the user to make necessary alterations.
- Home (Alt-H) - This feature allows the user to return automatically to the current schedule.
- Information (Alt-I) - The information feature moves the user to the database window where the data on the individual nurses are stored.

- Notebook (Alt-N) - The notebook command moves the user to the word processor area, where they can write notes regarding the nurses and their desired shifts. This feature does not require a password and any of the nurses who had access to the computer could enter and relevant information to the notebook area.
- Print (Alt-P) - The print feature enables the user to print either the current and next schedules or the nurse database. When the user selects the print option a menu comes up asking the user what he would like printed. The options include (1) the current schedule, (2) the schedule for the next planning horizon, (3) personal data on the nurses, which includes material such as their address, phone number etc., and (4) information on the work history of the nurses, including data on the years worked at the hospital, number of holidays taken, vacations, etc.
- Quit (Alt-Q) - Exits the program and returns the user to the DOS prompt.
- Save (Alt-S) - The save feature saves the changes made to the current work area. Depending on the location of the user, the corresponding material is saved, this includes the nurse database, the current schedule or the next schedule.
- Update (Alt-U) - The update feature files the current schedule and replaces it with the next schedule that was previously created. When this feature is selected, the user is first prompted for the name of the file under which the current schedule should be saved. It is recommended that the name should represent the dates of the schedule, so that at a future time, the schedule can be easily retrieved. At this point the nurse's work history is updated, including number of vacation days, holidays, weekend, sick



days, etc. The user is then asked if he would like to delete a nurse from the schedule and database. If the user responds with a yes, he is asked to input the first and last name of the nurse to be deleted. If the name does not occur in the database he is notified and asked if he would like to enter in the name again.

### **Development Problems and Limitations**

The time spent developing the system would have been greatly reduced had the problems of memory with using a 286 IBM compatible computer had not been encountered. Since it was still unclear as to what type of hardware the hospital would maintain, the partial DSS was developed on a 286 machine. A lot of time was spent determining the formulas in the cells to compute the number of nurses on duty. Once it was complete and expanded for the entire scheduling period it was discovered that there was not enough memory. The formulas needed to be rewritten in the form of macros. The size of the program, 4 weeks with 15 nurses, and the slow calculation speed of the 286 computer, increased the computation time of the macros to determine the number of nurses to approximately 25 minutes. The same problem arose with the update feature. The limited amount of memory and the slow speed of the computer resulted in the update feature having a run time of approximately 40 minutes.

### **Stage 7. System Implementation**

At the time of installation, the hospital was able to obtain a 486 Gateway computer, with Microsoft Windows. This had a large affect on the design of the system. As a result, the features that were originally written for a system with a limited amount of memory could then be redesigned to take advantage of the additional memory. These

alterations to the DSS took approximately two days to accomplish. Both the update feature and the calculation of the number of nurses on duty were rewritten with a resulting run time of only 10-25 seconds.

At the meeting to install the system, the nurse manager informed the developer of two changes that had been made since the last meeting. One change, made by budgeting, concerned the starting day of the schedule. The schedule now started on a Sunday, rather than on Monday as originally established. Fortunately the DSS was versatile enough for the developer to quickly incorporate this change when installing the system. Another item, not indicated earlier, was a new shift length. Originally the system was constructed to handle shifts from one to twelve hours in length. It was now necessary to include the possibility of a sixteen-hour shift. Again the DSS was versatile enough for both of the changes to be made in approximately 40 minutes at the time of installation. The system was then initialized to include the current schedule and the schedule for the next planning horizon. At the meeting the nurse manager was given a two page handout, shown in Figure 5.8 describing the features and the DSS. It was also recommended that each schedule and the program be backed up and stored in a safe location. These records are important for security reasons and the hospital should always have additional copies in case the original was damaged.

The development of the DSS was a part-time endeavor covering an eight months time span. The overall development time for constructing the DSS took approximately two hundred man hours to complete, resulting in a system consisting of 300 lines of code and over a dozen macros.

## **A Decision Support System for Nurse Scheduling**

### **To Load:**

**Once in DOS move into the symphony directory by typing:**

**cd\symp**

**To run symphony type:**

**symphony**

**To load the file:**

**Press the F9 key and then type F R Sched Enter**

**For a list of the commands: Press Alt C**

**When typing in the zip code, address, and telephone you must type a ' before typing in a number. Any cell that begins with a number, if it is not a formula must have a ' before it.**

**Whenever you are editing you must remember to save afterwards, otherwise your changes will not be saved. In addition, when exiting the program will ask you if you want to save the changes you have made during the session. If you want to save the work that has been completed, even though you have already saved the edited material, you must indicate a yes.**

**In order to recalculate the number of nurses on each of the given shifts you must press F8 to recalculate the screen.**

**At the end of the year you must go through the update procedure available in the menu feature. At the menu (Alt M) select update and at the next menu select year.**

**The program should be run with the Cap Lock on at all times**

**In the schedule the following characters denote:**

<b>V</b>	<b>a vacation day</b>
<b>H</b>	<b>holiday</b>
<b>S</b>	<b>sick day</b>
<b>U</b>	<b>unexcused absence</b>
<b>O</b>	<b>excused absence</b>

**Figure 5.8 The Nurse Scheduling DSS Handout**

## **A Decision Support System for Nurse Scheduling**

### **The Alt keys**

- A**    **Add a nurse to the database**
- C**    **A list of the commands will appear ( Hit any key to exit)**
- E**    **Edit the current schedule**
- G**    **Generate the next schedule (Note: this will not replace the current schedule)**
- H**    **Home (return to the current schedule)**
- I**    **Move to the database**
- M**    **Menu**
- N**    **Notebook**
- P**    **Print Menu**
- Q**    **Quit**
- S**    **Save**
- U**    **Update (File the current schedule and replace the current schedule with the next schedule)**

**If you have any problems or would like some changes made please let me know. My phone number and address is on the attached card.**

**Figure 5.8 The Nurse Scheduling DSS Handout**

## *Analysis of the Development Process*

When analyzing the development process of the DSS one question that must be asked is if priority design was the appropriate design methodology for the hospital environment. One method of addressing this question, is to address the alternative. What would have been lost or gained had a different methodology been employed? If the traditional life cycle had been used the nurse manager and the developer would have met initially and the requirements would have been determined without the benefit of the prototype simulation. Although the environment had a low level of uncertainty, the simulation was needed to allow the nurse manager to picture the system and its capabilities. Once the nurse manager had a clear idea of the system only then could she make educated decisions concerning the list of requirements for the system. The entire system would have been designed, with the developer having little knowledge on what requirements were most important and what was of little importance to the user. If the traditional methodology was used, the most important feature might have been unclear. From the developer's viewpoint, it might have been the generation of data rather than the schedule itself. The resulting system may have not met the needs of the nurse manager.

Similarly, if Courban's iterative design was implemented, the nurse manager would not have had the benefit of the simulation prototype before the developer and the user met to determine the characteristics of the system and before the construction process began. For example, during the simulation it was determined that the schedule would always start on a Monday. The simulation did not have any real formulas, it only simulated the option of starting on any given day. If Courban's method had been employed, all of the formulas would have been constructed to allow the schedule to start on any day.

In addition, development would have stemmed from one important module of the system. In this case, the development would have started with the development of the user interface matching the time sheet. The actual system, developed using priority design did start with the time sheet interface, but some of the other functions, that were to be developed, later were taken into account during the construction process. Although, in this case it is not clear that the time sheet interface would have differed in any way from the interface that would have been developed using Courban's process it is important to note that it may have had an affect on the development process.

Another methodology that could have been employed is prototyping, which is generally recommended for an uncertain environment. This did not match the hospital's environment. However, if the prototyping methodology had been employed, a working model of the system would have been first developed. The prototype would have been examined by the user and another prototype constructed. There would be no opportunity to quantify the development time and the cost of the user's requests, and no mechanism for determining the priority of the items.

Another question that should be addressed when analyzing the system, is if *Symphony* was an appropriate tool for the development of the DSS. Although during the system development, there appeared to be a problem with limited memory which forced the system to be slow, *Symphony* was an appropriate tool. The speed and memory limitations were resolved when it was determined that the hospital would be getting a computer with expanded memory. *Symphony* was a good choice for a number of other reasons. It did not require a large amount of programming and there is a great deal of flexibility in the system. This is illustrated by the speed in which the changes were made during the implementation stage. If a third generation language had been used these changes could not have been incorporated as quickly.

## ***Conclusion***

The development of the nurse scheduling DSS accomplished two goals. First, it provided a mechanism to further articulate the proposed priority design and provided a platform for testing the proposed priority design methodology. The development process followed the steps proposed in the design methodology. Each step was carefully noted and further analyzed. Second, the development of the DSS assisted the nurse manager in her scheduling task.

## **Chapter 6: Conclusion**

### ***Introduction***

This chapter summarizes the results of the research. This includes a discussion of: (1) the distinguishing features and benefits of the priority design, (2) the guidelines for the selection of a design methodology and (3) the limitations and possible enhancements for the nurse scheduling DSS.

### ***Summary of Research Results***

When decision support systems were first conceptualized, the traditional method of design was the systems development life cycle. However, it was soon recognized that decision support systems have distinctive characteristics that suggest the need for alternative development strategies. In the last two decades, several different methodologies have been applied to the development of decision support systems. These



approaches include Courban's iterative design, Keen's adaptive design, prototyping and a mixed methodology, integrating prototyping into the systems development life cycle.

## **Priority Design**

Each of these existing methodologies were carefully studied and the strengths and weaknesses were evaluated. An alternative methodology called priority design was developed to augment the existing methodologies. The two most distinguishing features of the priority design methodology include the development of a simulation prototype and the idea that the design of the DSS should stem from a priority list of items, which is evaluated by the developer.

The simulation prototype in priority design incorporates the advantages of the prototype methodology. Prototyping is usually recommended when the problem and the system requirements are unclear. By constructing a prototype, the user is able to view the system before the construction process begins. After seeing the system and interacting with the interface, the user can then further clarify the system requirements. A simulation prototype, as opposed to a working prototype, was recommended to prevent the user from investing in the system before seeing it. The user will then have a clearer understanding of the desired system's capabilities and limitations before a large investment is made.

In priority design, the recommended priority list is created by the user and then evaluated by the developer. The partial construction of the DSS then stems from the higher priority items. The development of the priority list has a number of benefits. By allowing the user to create his own priority list, the developer receives a clear

understanding of the items that are considered most important to the success of the DSS and which items are considered secondary. Thus, the construction process is focused on the most important capabilities. This ultimately keeps the development time of the partial system to a minimum.

One of the advantages of Courban's iterative design, is that it provides the user with a partial system to use. The priority system incorporates this idea. However development does not start with one important module, but from the most important items on the priority list. In addition, by maintaining the list of lower priority items the developer can foresee how the system will need to expand in the future.

The evaluation of the list by the developer provides an opportunity for the developer to suggest enhancements and additions to the system, which the user may have not considered. The evaluation also provides the user with estimates on the development time and cost required to meet each of the user's requests. If the budget or the time schedule for the partial system development needs to be adjusted, the user can then make a better decision on what items to cut or postpone.

In conclusion, the priority design was not developed to replace any of the current design methodologies. The characteristics of the priority design are advantageous in certain environments. Priority design was developed to complement the existing ones.

## **Guidelines for Selecting a Design Methodology**

Most decision support systems share some general characteristics. They are designed to aid in solving semi-structured problems. They are generally user friendly and are adaptive over time. Although DSS share some of these common characteristics, they

are applied to solve a number of different problems in a variety of environments. The different characteristics of the problems and their corresponding environments suggest the need for a design methodology that best complements these characteristics. Characteristics such as the complexity, uncertainty, and user involvement are some of the issues that need to be evaluated in order to determine an appropriate design methodology. In this research, guidelines were proposed to evaluate the environment and characteristics of a project and to recommend an appropriate design strategy.

To illustrate the use of the selection guidelines, they were applied to three case studies. The environment of each of the cases were examined and a design strategy was recommended. An analysis followed which discussed how the development process might have differed if the recommended process had been employed. In all three cases, a number of different advantages were foreseen if the recommended methodology had been employed. Some of the advantages include a shorter development time, a clarification of the user's requirements of the system and financial savings.

### **The Nurse Scheduling DSS**

To apply the proposed priority design, a DSS for nurse scheduling was developed. The DSS was constructed to assist the nurse manager, of the ICU at Montgomery Regional Hospital in the nurse scheduling process. The development of the DSS followed the steps of the priority design methodology. The DSS was successfully completed and employed by the hospital.

After the nurse manager had used the system for a period of time, a follow up visit was conducted, to determine the nurse manager's opinion of the system and if any changes would be necessary. Generally the nurse manager was pleased with the system. She

reported that the data was very easy to enter and that the system met all of her requests. When asked if the manager would like any additional data or statistics to be provided, her reply was no and further added that some of the statistics that were originally requested were now unnecessary and were being provided by payroll.

The nurse manager also commented that she would like the system to create the schedule. For the future, this is would be an enhancement that would greatly add to the value of the DSS. This process would require a large amount of time and some major changes in the DSS. It would probably require that the computations be carried out using a third generation language and then be imported into the current spreadsheet environment of the DSS.

The nurse manager's final comment was that she was happy with the overall system, but would have liked to have used it in a Microsoft Windows environment. The developer responded that when the system was initially constructed it was not clear if they would have the computing power to operate in a Windows environment. However, if the hospital was willing to invest in the necessary software the system could easily be converted to a Windows application.

As a final comment, the developer, foresees that the long-run continued value of the system is questionable. The DSS was constructed to be as flexible and adaptive as possible and to be able to withstand any foreseeable changes in the scheduling process. However, if in the future, there are changes that were unforeseeable it would be difficult for the hospital to implement them. There is no one currently available at the hospital with the technical expertise to incorporate changes into the system. The availability of someone to maintain the system and make necessary changes should be considered before the development process begins. In this case, the hospital has been left with information on where to contact the developer for any problems in the near future.

## Bibliography

- Aiken, L. and C. Mullinex, "Special Report: the Nursing Shortage: Myth or Reality?" *New England Journal of Medicine*, 317, 1987, pp. 641-646.
- Alavi, Maryam, "The Evolution of Information Systems Development Approach: Some Field Observations," *Data Base*, Spring, 1984, pp. 19-24.
- Arthur, J.L. and A. Ravindran, "A Multiple Objective Nurse Scheduling Model," *AIEE Transactions*, Vol. 13, 1981, No. 1, pp. 55-60.
- Atwood, J.R. and A.S. Hinshaw, "Multiple Indicators of Nurse and Patient Outcomes as a Method for Evaluating a Change in Staffing Patterns," *Communicating Nurse Res.*, 10: 235, 1977, pg 255.
- Balintfy, J. L. and C.R. Blackburn, "General Purpose Multiple Choice Programming by Truncated Block Enumeration," presented at the 36th National Meeting of ORSA, Miami Beach, Florida, November, 1969.
- Bennett, John L., *Building Decision Support Systems*, Reading, Massachusetts: Addison-Wesley Publishing Co., 1983.
- Berrisford, Thomas, and James Wetherbe, "'Heuristic Development: A Redesign of Systems Design", *MIS Quarterly*, March 1979, pp. 11-19.
- Boar, B.H. *Application Prototyping*, New York, New York: John Wiley and Sons, 1984.
- Burns, R.N., and A.R. Dennis, "Selecting the Appropriate Application Development Methodology," *Data Base*, Fall, 1985, pp. 19-23.
- Carlson, E.D., "An Approach for Designing Decision Support Systems," *Data Base*, Winter, 1979.
- Cervený, R., E. Garrity, and G. Sanders, "The Application of Prototyping to Systems Development: A Rationale and Model", *Journal of Management Information Systems*, Vol III, 1986, No. 2, pp. 52-62.
- Courban, J.C. and M. Bourgeois, The Information Systems Designer as a Nurturing Agent of a Socio-Technical Process. In Lucas, H., et al.(eds.). *Information Systems Environment*, New York:North Holland, 1980.

- Curran, C., A. Minnick, and J. Moss, "Who Needs Nurses?" *American Journal of Nursing*, 1981, 81, pp. 2162-2164.
- Davis, G.B. and M.H. Olson, *Management Information Systems: Conceptual Foundations, Structure, and Development*, New York: McGraw-Hill Book Company, 1985.
- Dennis, A.R. and R.N. Burns, and R.B. Gallupe, "Phased Design: a Mixed Methodology for Application System Development," *Data Base*, Summer, 1987, pp. 31-37.
- Durmusoglu, M. Bulent, "Analysis of the Conversion From a Job Shop System to a Cellular Manufacturing System," *International Journal of Production Economics*, Vol. 30-31, 1993, pp. 427-436.
- Easton, Fred F, Donald F. Rossin, and William S. Borders, "Analysis of Alternative Scheduling Policies for Hospital Nurses," *Production and Operations Management*, Vol. 1, No. 2, Spring 1992, pp.159-174.
- Eusanio, Patricia L., "Effective Scheduling - The Foundation for Quality Care," *Journal of Nursing Administration*, Vol. 8 1978, No. 1, pp. 12-17.
- Felton, Geraldene, "Body Rhythm Effects on Rotating Work Shifts," *Journal of Nursing Administration*, March-April, 1975, pp. 16-19.
- Gahan, Karen, and Rosanne Talley, "A Block Scheduling System," *Journal of Nursing Administration*, Vol. 5 1975, No. 9, pp. 39-41.
- Greenwood, Allen G., *Dissertation: A Decision Support System for Tuition and Fee Policy Analysis*, Department of Management Science, Virginia Tech, Blacksburg, VA 1984.
- Hoover, Thomas B., "Decision Support at Conrail", *Datamation*, June 1983, pp. 220-230.
- Janson, M.A., and L.D. Smith, "Prototyping for Systems Development: A Critical Appraisal," *MIS Quarterly*, Vol. 9 1985, No. 4, pp. 305-316.
- Keen, P.G.W., "Adaptive Design for Decision Support Systems," *Database*, Vol. 12, 1 and 2, Winter 1979, pp. 15-25.
- Keen, P.G.W., and M.S. Scott-Morton, *Decision Support Systems, An Organizational Perspective*. Reading, MA: Addison-Wesley, 1978.

Konsynski, B.R. "Data Base Driven System Design," Proceedings of the 1st Conference on Systems Analysis and Design, Elsevier-North Holland, Atlanta, Georgia, 1981.

Kraushaar, James, and Larry Shirland, "A Prototyping Method for Applications Development by End Users and Information Systems Specialists", *MIS Quarterly*, September, 1985, pp.189-197.

Kroeber, R. "Designing Decision Support Systems: Critical Issues", *Database*, 1989, pp.152-160.

Laudon, Kenneth and Jane Price, *Business Information Systems, A Problem-Solving Approach*, Philadelphia, PA, The Dryden Press, 1991, pp.389-390.

McCracken, D.D. "A Maverick Approach to Systems Analysis and Design," Proceedings from Naumann's article.

McKibbin, R. and C. Boston, "An Overview: Characterisitcs, Impact, and Solutions," Monograph 1 in *The Nursing Shortage: Opportunities and Solutions*, American Organization of Nurse Executives and American Nurses' Association, Chicago, IL, 1990.

McFarlan, F.W., "Portfolio Approach to Information Systems", *Harvard Business Review*, Sept.-Oct. 1981, pp. 142-150.

Melbin, M. "Curbing Absenteeism and Turnover Among Nursing Personnel," Massachusetts General Hospital, Boston, Massachusetts, 1968.

Miller, H.E., W.P. Pierskalla, and J.R. Gustave, "Nurse Scheduling Using Mathematical Programming," *Operations Research*, Vol. 24, 1976, No. 5, pp. 856-870.

Moran, "At Work; Pushing Nurses to a Breaking Point," *The New York Times*, The New York Times Company, January 10, 1993.

Musa, A.A., and U. Saxena, "Scheduling Nurses Using Goal-Programming Techniques," *IIE Transactions*, Vol. 16, 1984, No. 3, pp. 216-221.

Naumann, Justus, and Milton Jenkins, "Prototyping: The New Paradigm for Systems Development," *MIS Quarterly*, September, 1982, pp. 29-43.

Omar, Mohammed H., "A DSS Approach for Implementing an Online Retail Banking System," *Information & Management*, Vol 21, 1991, pp. 89-98.

Pracht, W.E. and J.F. Courtney, "The Effects of an Interactive Graphics-Based DSS to Support Problem Structuring," *Decision Sciences*, Vol. 19, 1988, pp. 598-620.

- Pross, "ANA Says Staffing Concerns Not New to Nurses; Inadequacies Seen As Part of Overriding Concern for Quality Patient Care", *U.S. Newswire*, January 7, 1993.
- Ress, David A. and Kenneth R. Currie, "Development of an Expert System for Scheduling Work Content in a Job Shop Environment," *Computers and Industrial Engineering*, Vol. 25, No. 1-4, 1993, pp. 131-134.
- Rowland, Howard S., and Beatrice L. Rowland, *The Nursing Administration Handbook*, Second Edition, Aspen Publishers, Inc., Rockville, MD., 1985, pp. 166-182.
- Sauter, Vicki L., and Joseph L. Schofer, "Evolutionary Development of Decision Support Systems: Important Issues for Early Phases of Design," , Vol. 4, 1988, *Journal of Management Information Systems*, No. 4, pp. 77-92.
- Sprague, R. H., Jr., and E. D. Carlson, *Building Effective Decision Support Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- Stevens, Barbara, *The Nurse as Executive*, Third Edition, Aspen Publishers, Inc., Rockville, MD., 1985, pp. 127-130.
- Turban, Efraim, *Decision Support and Expert Systems: Management Support Systems*, Second Edition, Macmillan Publishing Co., New York, 1990.
- Vik, G. Astrid, and Ruth MacKay, "How Does the 12-Hour Shift Affect Patient Care?" *The Journal of Nurse Administration*, January 1982, pp. 11-14.
- Warner, D.M., "Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach", *Operations Research*, Vol. 24, No. 5, pp. 843-855.
- Warner, D.M., "Nurse Staffing, Scheduling, and Reallocation in the Hospital," *Hospital and Health Services Administration*, Summer, 1976, pp. 77-90.



## Vita

Wendy Ann Ceccucci was born in Cohoes, New York on December 30, 1965. She attended elementary school in Nashua, New Hampshire and graduated from Newport High School in Bellevue, Washington, in 1983. In September of that year she entered Union College in Schenectady, New York. In August 1987 she received her Bachelor of Science degree in Math-Computer Science. In September she entered the Masters of Business Administration program at Virginia Polytechnic and State University. She completed the degree in August 1989. Immediately following the Masters, she entered the doctoral program in Management Science. While attending her studies at Virginia Polytechnic Institute and State University she worked as a graduate assistant and as a part-time instructor. In September 1992 while working on her dissertation, she took a position as assistant professor at Central Connecticut State University in New Britain, Connecticut.

*Wendy Ceccucci*