

A SEMI-AUTONOMOUS ON-LINE CHEMOTHERAPY
PRESCRIPTION SYSTEM

SYED NAQVI

A Semi-Autonomous On-Line Chemotherapy Prescription System

by
© Syed Naqvi

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Master of Science

Department of Computer Science
Memorial University of Newfoundland

October 24, 2007

St. John's

Newfoundland





Abstract

To advance the causes of reducing medical errors and improving the quality of patient care, healthcare is currently going through a period of vast reforms. CPOE (Computerized Physician Order Entry) is one such reform. An important example of CPOE is drug prescription, both of individual drugs and complex regimens, *e.g.*, chemotherapy. By reducing medication errors, CPOE can both increase the quality of patient care and decrease cost. Several types of problems may arise during the design and implementation of CPOE systems. The most obvious of these problems are due to insufficient attention being paid to the nature of existing manual clinical workflows. Much more damaging are subtle problems related to insufficient attention being paid to the nature of the clinical workplace – in particular, the frequent lack of resources, *e.g.*, system administrators and programmers, for ongoing system maintenance and evolution after the system is deployed.

In this thesis, we will describe the development of an on-line chemotherapy prescription system in which many aspects of system maintenance and evolution can be performed by the system's users. The user-guided operational model underlying this system overcomes many of the problems arising from limited system maintenance and evolution resources in the target workplace. This thesis will also include several discussions of various lessons learned during the development of this system that are applicable to the development of medical informatics systems in general.

Contents

Abstract	ii
Acknowledgements	iii
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Contributions	3
1.4 Organization of Thesis	5
2 Background	7
2.1 Medical Software	7
2.1.1 Data Storage Systems	8
2.1.2 Analytical and Decision Support Systems	10
2.1.3 Computerized Physician Order Entry Systems (CPOE)	11

2.2	Developing Medical Systems: Problems and Solutions	12
2.2.1	System Development	13
2.2.1.1	Importing Legacy Knowledge	13
2.2.1.2	Interoperability	14
2.2.1.3	Legal Issues	15
2.2.1.4	Social and Organizational Issues	17
2.2.2	System Operation	20
2.3	Software Development	20
2.3.1	What is a Software Process Model?	21
2.3.2	Types of Software Process Models	22
2.3.2.1	Waterfall Model	22
2.3.2.2	Incremental Model	23
2.3.2.3	Prototyping Model	24
2.3.2.4	Spiral Model	24
2.3.3	What is a System Operational Model?	25
2.3.4	Types of System Operational Model	26
2.3.5	Which System Process and Operational Models are Appropriate for Medical Informatics?	31
3	Chemotherapy Prescription	33
3.1	CPOE	34
3.1.1	What is CPOE?	34
3.1.2	Benefits of CPOE	35
3.1.2.1	Reduction of Medical Errors	35

3.1.2.2	Time and Cost	36
3.1.2.3	Research Purposes	37
3.1.2.4	Administrative Purposes	37
3.1.3	Problems with CPOE	37
3.1.4	Factors in Successful CPOE Implementations	39
3.2	Chemotherapy Prescription	41
3.2.1	What is Chemotherapy?	41
3.2.2	Manual Chemotherapy Workflow	42
3.2.3	Computerized Chemotherapy Workflow	44
3.2.4	Benefits of Computerized Workflow	45
3.2.5	Consideration Concerning Computerized Workflow	46
3.3	Previous Work	46
4	System Design	58
4.1	System Objectives	59
4.2	Key Features	61
4.2.1	Democratic Model of System Operation	61
4.2.2	User Interface Design	63
4.3	System Objects and Classes	64
4.4	Single User Design	65
4.4.1	Single-Session Activities	66
4.4.1.1	Login/Logout	68
4.4.1.2	Search Patient	71
4.4.1.3	Prescribe Regimen	73

4.4.2	Short-Term Activities	80
4.4.2.1	Add Patient	80
4.4.2.2	Create New Regimen	83
4.4.2.3	Alter Existing Regimen	89
4.4.2.4	Manage Regimens	94
4.4.3	Mid-Term Activities	94
4.4.3.1	Add New Chemotherapy drug	96
4.4.3.2	Password Retrieval	99
4.5	Multi User Design	102
4.5.1	Indirect Interaction	102
4.5.1.1	View Patient History	104
4.5.2	Limited Direct Interaction	107
4.5.2.1	Import Regimen	107
4.5.3	Direct Interaction	110
4.5.3.1	Add Proposal	111
4.5.3.2	Vote Proposal	116
4.5.3.3	View Status of Proposals	118
4.6	Database Design	120
4.7	Summary: Implications for Medical Informatics Development	124
5	System Development	127
5.1	Development Process	128
5.2	Development Model	128
5.3	Technology Decisions	130

5.3.1	Development Framework	131
5.3.1.1	Jakarta Struts	131
5.3.1.2	Spring MVC Framework	133
5.3.1.3	Evaluation of Struts and Spring frameworks	134
5.3.2	Development IDE	134
5.3.3	Database	135
5.3.4	WebServer	135
5.4	Summary: Implications for Medical Informatics Development	136
6	System Acceptance and Implementation	138
6.1	System Acceptance Model	139
6.2	Implementing Medical Informatics Software in the Workplace	141
6.2.1	Software Adoption	141
6.2.2	Software Implementation	142
6.3	Summary: Implications for Medical Informatics Development	143
7	Conclusion and Future Directions	145

List of Tables

2.1	Types of System Operational Models	32
4.1	Single User Design Table	67
4.2	Multi User Design Table	103

List of Figures

3.1	Workflow Diagram: Manual Chemotherapy Prescription Process . . .	43
3.2	Workflow Diagram: Computerized Chemotherapy Prescription Process	47
3.3	Screenshots: Meditech - (a) Patient Profile (b) Finding Patient. . . .	49
3.4	Farrell Version 1 - Sample Prescription Page	52
3.5	Farrell Version 2 - Sample Prescription	53
3.6	Farrell Version 2 - Sample Prescription (Cont'd)	54
3.7	Farrell Version 3 - Sample Prescription Input Page	56
3.8	Farrell Version 3 - Sample Prescription Output Page	57
4.1	Workflow Diagram: Login/Logout	69
4.2	Screenshot: Login	70
4.3	Workflow Diagram: Patient Search by MCP Number	72
4.4	Screenshot: Patient Search Page	73
4.5	Workflow Diagram: Patient Search by Demographic Information . . .	74
4.6	Workflow Diagram: Prescribe Regimen	76
4.7	Screenshot: Sample Prescription Input Page	77
4.8	Screenshot: Sample Prescription Output Page	78
4.9	Workflow Diagram: Add Patient to Database	81

4.10 Screenshot: Add Patient to Database	82
4.11 Workflow Diagram: Create New Regimen	85
4.12 Screenshot: Create Regimen - Regimen Name	86
4.13 Screenshot: Create Regimen - Chemo Drugs Details	87
4.14 Screenshot: Create Regimen - Antiemetic Drugs Details	88
4.15 Workflow Diagram: Alter Regimen	91
4.16 Screenshot: Alter Regimen - Regimen Name	92
4.17 Screenshot: Alter Regimen - Chemo Drugs Details	93
4.18 Workflow Diagram: Manage Regimens	95
4.19 Screenshot: Manage Regimens	95
4.20 Workflow Diagram: Add New Chemotherapy Drug	97
4.21 Screenshot: Add New Chemotherapy Drug	98
4.22 Workflow Diagram: Password Retrieval	100
4.23 Screenshot: Password Retrieval	101
4.24 Workflow Diagram: View Patient History	105
4.25 Screenshot: View Patient History	106
4.26 Workflow Diagram: Import Regimen	108
4.27 Screenshot: Import Regimen	109
4.28 Workflow Diagram: Add Proposal	113
4.29 Screenshot: Add Proposal - Change Emetogenicity Potential	114
4.30 Workflow Diagram: Vote Proposal	117
4.31 Workflow Diagram: View Proposals	118
4.32 Screenshot: View Proposals	119
4.33 Database Entity-Relationship Diagram	121

5.1	Spiral Development Model	129
-----	------------------------------------	-----

Chapter 1

Introduction

1.1 Motivation

Almost all activities involved in the process of improving a patient's health are initiated via a written order by a physician. This order is usually a prescription, which includes instructions for administering medication, or different tests for diagnostic purposes. This initial step or entry point for the process of patient care is crucial, as the entire care process is dependent on it. In particular, a minor mistake in prescribing, *e.g.*, mistakes in dosage or frequency of medication, can cause severe problems, or in the worst case this could result in the death of the patient. Hence, physicians are required to be very careful while writing such prescriptions.

In 1999, a report by the Institute of Medicine, "*To Err Is Human*" [36], cited much evidence to indicate that in the United States alone, preventable medication errors are involved in 44,000 to 98,000 deaths per annum (a greater death toll than deaths due to such feared threats as breast cancer, motor-vehicle wrecks, and AIDS combined)

and cost \$17 to \$29 billion per year. This report and others have renewed efforts to apply information technology to clinical workflow in order to decrease medication errors. Studies have proved that the introduction of a computer in clinical workflow can significantly reduce medication errors [61, 43]. The successful introduction of a computer system, *e.g.*, CPOE (Computerized Physician Order Entry), can cause not only a decrease in medication errors, but also enhance work through many aspects, *e.g.*, reduced cost, standardization of care and improved efficiency of care delivery [37, 59]. CPOE systems are remarkably helpful in reducing prescription related errors, where prescription writing involves complex calculations of dosage. This motivates the development of prescription systems that can be helpful in making the prescription process more accurate and improving the quality of patient care.

Such computerization is desirable; however, there are known problems that occur during the development of these systems, such as inattention to the clinical workflow. These problems can be severe enough that the resulting systems are often rejected by physicians. If the promises of computerization given above are to be realized, these problems must be dealt with during system development.

1.2 Objectives

Given the prescription problem described in motivation, as well as known problems with implementing computerized solutions to medical informatics in general, this thesis has two main objectives:

1. To develop a restricted CPOE system for the chemotherapy prescription process.

2. To present a number of lessons learned during the development of such a system that are also relevant to medical informatics in general.

In the process of developing the required system, we have discovered what we think is a gap in current software engineering theory and practice. This gap is related to the assumptions that are made in software engineering practices and the actual operation of the system both in terms of maintenance and evolution after deployment. Given this gap, we also have a third objective of this thesis, to describe this gap and propose a solution.

The reader should note that though software engineering techniques are used extensively in the design and implementation of this system (and the description of this design and implementation), this thesis is not intended as a software engineering thesis. Rather, it is intended as a contribution to medical informatics, as both a demonstration of how a particular kind of system can be developed and the general lessons that should be learned from this development process.

1.3 Contributions

The major contributions to this thesis are as follows:

- *A detailed workflow analysis of the chemotherapy prescription process:* This workflow analysis was gathered by repeated weekly conversation with Dr. Farrell, under the Spiral software process development model (see Section 2.3.2.4). These conversations allowed us to gather workflow not only because Dr. Farrell was a chemotherapy prescriber himself, but also because he was the developer of

many of several previous chemotherapy prescription systems (see Section 3.3); hence, these conversations access the experience of not only Dr. Farrell, but all users with which Dr. Farrell himself interacted in developing his previous systems. Similarly the workflow was also validated through these conversations with Dr. Farrell (and hence also indirectly validated with the other system users that Dr. Farrell talked to while developing his previous systems).

- *A set of software development guidelines for medical informatics:* These guidelines are encoded as various lessons learned during this project and are listed at the end of Chapters 4, 5, and 6. We are aware that many of these lessons are common sense in the software engineering community; however, it is still of interest to list such guidelines here because these have particular importance in medical informatics.
- *Addressing problems of post-development system operation in medical informatics:* This involved developing what we believe to be an original classification of post-development system operational models (see Section 2.3.3), and explaining why certain medical informatics systems such as chemotherapy prescription operate under a model that is not currently addressed in an adequate fashion by software development practice. We also propose a solution to this problem in the form of a democratic model of system operation (see Section 4.2.1).

1.4 Organization of Thesis

In Chapters 2 and 3, we describe the general problems with medical informatics software and look in detail at the prescription process, particularly as it relates to chemotherapy.

- In Chapter 2, we will describe different types of medical software/tools (Section 2.1), problems that arise while developing them, and possible solutions to those problems (Section 2.2). We will see how software techniques are helpful in developing medical software (Section 2.3). We will discuss how the operation of software after its development is important to consider while developing any software. We will discuss in detail the different software operational models defined in terms of software evolution and maintenance (Section 2.3.3). We will discuss which software process model and system operational model is appropriate for the development of medical systems (Section 2.3.5).
- In Chapter 3, we will discuss in detail both CPOE (Section 3.1) and chemotherapy prescription (Section 3.2) as a restricted form of CPOE system. We will also examine different problems for the development of such a system that are introduced due to the limited resources available for the maintenance and evolution of a medical system during its operation. Finally, we will look at the previous work that has been done to implement chemotherapy prescription systems (Section 3.3).

In Chapters 4, 5, and 6, we will look at the design, development, and implementation of a chemotherapy prescription system. Note that at the end of each of these chapters,

as this thesis is on medical informatics, we will give in-depth discussions of the specific lessons we draw for developing medical informatics software in general.

- In chapter 4, we will discuss the design of a chemotherapy prescription system. We will describe the objectives of the system (Section 4.1), and its two key features (Section 4.2), related to user interface design and a newly proposed model of system operation, which we call the democratic model of system operation. We identify system objects and classes (Section 4.3), and break the design of the system into single-user (Section 4.4) and multi-user (Section 4.5) features. Finally, we will describe the database schema for the system (Section 4.6).
- In Chapter 5, we will look at different technology choices, *i.e.*, development framework, IDE, database and webserver, that were made for the development of our chemotherapy prescription system. We will see how the choice of technologies for development of software is affected in the case of medical software.
- In Chapter 6, we will discuss the different system implementation models describing different ways to get software accepted in a target workplace. We will discuss the different factors that affect the acceptance of medical software/tools by healthcare.

Finally, in Chapter 7, we will provide conclusions and a list of directions for future research.

Chapter 2

Background

In this chapter, we will review different background topics for this thesis. These include general descriptions of the types of medical software (Section 2.1), different problems involved in the development of medical software and their solutions (Section 2.2), and a description of software engineering techniques (Section 2.3). The latter includes not only standard techniques such as the software process model (Section 2.3.1), but also includes the software operational model (Section 2.3.3), which is of particular concern to the system developed in this thesis.

2.1 Medical Software

Modern healthcare organizations are designed and structured to enhance the efficiency of healthcare. The introduction of information technology into healthcare can greatly enhance the process of quality patient care. Along with the basic effects of introducing computers to any workflow, *i.e.*, cost reduction and increased time efficiency, computerized workflow can reduce many flaws attached to manual healthcare

workflow, which are explained later in this section with reference to different types of medical systems.

However, healthcare organizations have given little attention to the introduction of computers to clinical workflow [14, 45]. If we explore the causes of such inattention to the clinical workflow further, we will discover many important and interesting facts about healthcare organizations. The clinical environment is different from any other work environment, because of the uniqueness and complexity which arise from the types of services, complexity of services, and motives behind the services it provides. These factors are further elaborated later in this section.

There are many types of medical systems, which are designed for use by different user communities. These communities include single doctors, small groups of doctors, and large groups of doctors. To examine past efforts for the details of successes and failures in an attempt to implement different types of electronic medical systems, we have divided systems into three levels depending on system functionality and type of user community. When we discuss each type of system, we will describe what that system does, the potential advantages it has, and whether or not it has been a success in the past in the USA and Canada.

2.1.1 Data Storage Systems

These systems include Electronic Health Records (EHR) and Electronic Medical Records (EMR). According to the HIMSS Electronic Health Record Committee, EHR is defined as a secure, real time, point-of-care, patient-centric information resource for clinicians [30]. An EMR is conceptually different from an EHR, as an EMR is the ac-

tual computerized clinical record in different hospitals and physician's offices, whereas an EHR is designed to share data among different EMRs [27]. EHRs are reliant on EMRs being in place and EMRs can never reach their full potential without EHRs. In both kinds of systems, patient data from different sources is stored in databases to make it available to clinicians. This data can be scanned images of paper documents, diagnostic images, or other type of medical data related to patients. EHRs and EMRs are designed for many uses, from individual clinics which have single-user EMRs to hospital level EMRs, which may have dozens or hundreds of users. Hence, EMRs and EHRs cover a broad spectrum of user communities.

A report by HIMSS on EMRs and EHRs establishes their ability to improve patient care in the following ways [30]:

1. Helping to reduce medical errors.
2. Making access to patient data faster.
3. Providing remote access to data.
4. Allowing the sharing of data among different clinicians.

Despite the potential of EMRs and EHRs to make contributions to the improvement of healthcare quality, significant initiatives and strategies to implement them in the past have caused only a relatively small number of sites in North America to use them. In 1991, the Institute of Medicine (IOM) [50] proposed to have comprehensive implementation of EHR in the US over a period of 10 years. However, after 12 years, no more than 17-25 percent of the medical user community employs EHR [32]. Like the US, Canada is also far behind in introducing computers in the healthcare. In

Canada, major steps to introduce nation-wide EHR were taken through a multi-million dollar investment by Infoway, funded by the federal government. Infoway [15] is a not for profit organization which aims to put an interoperable EHR into place across 50 percent of Canada (by population) by the end of the year 2009. However, a report by Infoway states that about 91 percent of physicians are still using paper records and/or prescribing using paper [14].

2.1.2 Analytical and Decision Support Systems

Analytical and decision support tools are designed to help doctors in making critical decisions on patient health. Analytical tools are used to access large transactional datasets by various data mining techniques. The results extracted using these tools can then be visualized using different knowledge representation methods. These tools can be used broadly to find the useful patterns in health data which can help in improving patient care, *e.g.*, analytical tools to find adverse drug events. Analytical tools may also be used for visual data analysis or image processing [53]. A clinical decision support system is any system that helps in making diagnostic decisions for patient care [67]. Both analytical and decision support systems are typically focused systems, aimed at single users or small groups of users.

Work on clinical decision support systems originated in the 1970s, when systems such as the de Dombals Leeds abdominal pain system [23] and the MYCIN system [80] for selecting antibiotics, were developed. In 1999, Perreault and Metzger outlined the key benefits of clinical decision support systems [56]:

1. Supporting clinical diagnosis and treatment plan processes.

2. Promoting the use of best practices in patient care.
3. Helping with cost reduction by reducing medication errors, and helping to avoid duplicate and unnecessary tests.

Unfortunately, decision support systems are underused in a manner similar to that seen with data storage systems. For example, a report by the Joint Clinical Decision Support Workgroup concluded that clinical decision systems are still used only by minority of physician in their practice [77].

2.1.3 Computerized Physician Order Entry Systems (CPOE)

Order entry systems, such as Computerized Physician Order Entry (CPOE) are one level above EHR/EMR. Such systems are designed on top of physician workflow and are used to assist physicians in accomplishing patient-related tasks such as ordering tests and prescribing medications. The purpose of such systems is mainly to prevent medication errors that are related primarily to prescription writing. Such systems may have additional features such as decision support, patient safety features such as real-time patient identification, and billing. CPOE is primarily designed for use by single doctors, though a CPOE system may serve a group of doctors.

Successful order entry systems are guaranteed to improve healthcare in the following ways [9, 20, 38, 60]:

1. Reducing the prescription errors that are inherent in writing prescriptions which involve complex calculations, such as chemotherapy prescriptions.
2. Giving direct access to patient data and history.

3. Removing interpretation of illegible hand-written orders from the workload for nurses and pharmacists.
4. Providing quicker turnover time for medications.
5. Making data more readily available for other uses such as research.

Though the concept of order entry systems is very old, and research has proven its effectiveness in improving clinical processes [9, 20, 38, 60], it remains underused in healthcare. Indeed, in the US and Canada, the use of CPOE is even lower than the use of EHR/EMR described earlier in the section. In the US, out of 17-25 percent of users of EHR, only about 9.6 percent use some kind of electronic ordering system, which may include electronic prescribing or electronic lab test ordering [21]. Similarly, in Canada, less than 5 percent of physicians use electronic ordering systems [14]. As order entry systems, and specifically Computerized Physician Order Entry systems, are the subject of this thesis, they are discussed in more detail in Section 3.1.

2.2 Developing Medical Systems: Problems and Solutions

In the previous sections, we have looked into different medical systems shown to be helpful in improving the process of patient care and noted that despite the benefits of such systems, their usage is low. If such systems are guaranteed to improve the quality of healthcare, but are not widely used, then there must be factors involved to prevent their adoption. In this section, we will explore the problems underlying such minimal use of medical systems and that must, therefore, be taken into consideration

when designing medical systems. We have divided these problems into two categories: system development problems (Section 2.2.1), which affect the development of a medical system, and system operation problems (Section 2.2.2), which arise during the use of a system after development.

2.2.1 System Development

In this section we will identify and briefly explain the different problems that arise while changing from a manual medical system to an electronic medical system. We will also discuss how these problems have affected the introduction of medical systems in healthcare and how (if at all) they have been addressed.

2.2.1.1 Importing Legacy Knowledge

In order to make legacy data readily available to physicians in new electronic systems older patient data must be added to electronic patient records. This addition of data can be expensive and difficult, as it may include scanning all paper records into an electronic form. This transfer of data from paper to a scanned digital format needs careful attention to ensure that all necessary details are transferred successfully. Unfortunately, research has shown that many physicians find patient data hard to understand in a scanned digital form, which causes them to revert back to manual systems [39]. Importing patient data is also difficult as it involves the assimilation of medical knowledge from different sources into the new system. These systems are often further complicated due to the use of specific medical terminology, as systems which use non-standard medical terms are likely to be rejected by the users.

Scanning old patient records and adding them to new systems should be considered only as an intermediate step towards developing a fully electronic medical record [39]. This step not only eliminates the problem of slow processing and rigid structure in scanned documents, but also makes the electronic data more readily available for sharing, administrative, and research purposes. However, in order to prevent the possibilities of errors involved in the manual addition of data to a database, necessary validity checks are required to ensure the correctness of the data. Moreover, to deal with the problem of usage of medical terminology, it is important to work in close collaboration with the actual users of the system, *i.e.*, physicians, who will be obliged to work with the medical terminologies employed by the system.

2.2.1.2 Interoperability

Interoperability is an important issue if we are to merge data from different sources to make it available for sharing among the users of those sources. Most stand-alone medical systems are developed and targeted for a particular department or hospital and lack any shared conventions of data with other sources. This restricts large amounts of medical data to only one medical system at a time and renders that data useless to other systems that could benefit by sharing it [24, 42].

In order to make data available for sharing from different sources it is necessary to standardize medical data. There are many standards, such as Health Level 7 (HL7) [31], that can be introduced as guidelines to enable the exchange and interoperability of electronic health records. HL7 introduces a set of "rules of conversation" that enable different systems to communicate patient-centric data. In Canada, Infoway, which has invested millions of dollars provided by the Canadian government, has

established a set of guiding principles for information standards. Infoway is also leveraging HL7 messaging, which is standard throughout North America [24].

2.2.1.3 Legal Issues

There are many legal issues involved in use of EHR and electronic prescribing systems, which make the development process more complex. Privacy laws make it difficult to share data among the users of electronic health data, *i.e.*, the personal data of a patient or the history of his medical record is not allowed to be open for general access like research work or administrative use without the consent of the patient. For example, under the Food and Drug Regulations in Canada a pharmacy is allowed to fill only those prescriptions that are ordered by physician in a written or verbal form. Food and Drug regulation states: "*C.01.041 (1.1) Subject to C.01.043 and C.01.046, no person shall sell a substance containing a Schedule F drug unless (a) the sale is made pursuant to a verbal or written prescription received by the seller;*" [41]. Moreover, there are many hurdles related to the creation of and maintenance of medical records which make the development and maintenance process more complex.

To overcome the privacy problem, we need to deal with the privacy of patients and medical professionals very carefully. Within the healthcare information system, there is an emerging need to ensure the security and integrity of healthcare data while maintaining patient privacy. A relational database is a good solution to problems related to privacy and security issues, where data can be stored in tables and limited access can be granted to each person according to his/her privileges. Moreover, to enhance privacy, field protection can be applied to tables to restrict access to individual columns. To make data available for research, we can use various techniques,

such as data anonymisation, to preserve privacy [75].

In addition to such technical solutions, we can also deal with legal issues by developing legislation that takes into account the specific concerns of health information. For example, a possible solution to the legal issues involved in electronic prescribing has been proposed by the National Association of Pharmacy Regulatory Authorities (NAPRA), which has developed general recommendations for the safe and effective transfer of prescriptions between prescribers and pharmacists. They identify five principles that should be met for safe transfer of electronic prescriptions, which are also supported by the Health Canadas Therapeutic Products Programme (TPP) [51]:

1. The process must maintain patient confidentiality.
2. The process must be able to verify the authenticity of the prescription, *i.e.*, that the prescriber is initiating the prescription.
3. The system must be capable of validating the accuracy of the prescription, and the process must include a mechanism to prevent forgeries.
4. Patient choice must be protected; that is the patient must determine the practitioner to receive the prescription authority.

Legal issues within many other areas of health informatics could be resolved by similarly structured legislation. In any case, such issues are not currently handled by software engineering methods and probably will not be handled in future; hence, they will not be addressed in this thesis.

2.2.1.4 Social and Organizational Issues

Modern healthcare organizations are confronted with new clinical e-health technologies as never before. Early evidence suggests that the adoption of new medical software/tools depends in large part on their acceptance by both hospital bureaucracies and by doctors. This acceptance must take into account various social and organizational issues related to the behavior of doctors and bureaucrats within the hospital environment. These issues and behaviors are discussed extensively in [49]. The following condenses some of the main points in that discussion, and divides these issues into two categories: bureaucratic and physician.

Hospital bureaucracies show the following behavior towards the adoption of new software/tool:

1. **The approval process:** It is very difficult for a medical tool to be approved by hospital bureaucrats. This is primarily due to caution about adoption of new tools because a minor error in those tools could harm patients health and even cause deaths. Moreover, the bureaucracy also has to consider the complex security and privacy issues associated with the adoption of new software.
2. **Limited IT resources:** Many development problems arise because of limited hospital IT resources. For example, the development of some database-enabled tools may require database administration, but a hospital's IT department may not have any database administration personnel to spare.
3. **Preference for existing tools:** As hospital bureaucracies trust only existing tools, they ask for new tools/software based on, and thus similar to, those tools.

This similarity falls in two categories:

- **Look and feel:** Hospital bureaucracies insist that new tools use the same interface as existing tools or, in the case of a paper-based system, they require new tools to have the same input and display format as that system.
- **Underlying technologies:** Bureaucrats are more comfortable using new tools which are based on technologies that they are already familiar with.

Doctors show the following behavior towards the adoption of new tools/software:

1. **Need for familiar look:** Even if they agree to use new tools, doctors need these tools to have same interface and look as tools with which they are already familiar, *i.e.*, the interface of the old system.
2. **Difficulty in extracting workflow descriptions:** Due to the busy schedules of doctors, it is sometimes very hard to get useful descriptions of clinical workflow from them. This may result in the development of a system with insufficient information about the target workplace and users, leading to a failure of adoption.
3. **Cannot be compelled to change:** Doctors have a responsibility to patient care which they always maintain as priority. Hence, no one can compel them to adopt a new technology which they do not feel serves that priority.

This last behavior is particularly important, because it highlights a crucial difference between the relationship of doctors and the administration in hospital and the relationship between employees and the administration in companies. In a company,

the workers are employees of the company and can be compelled by the administration to make any change related to work. However, the relationship between doctors and hospital is not so much that of employees but rather that of private contractor; therefore it is much more difficult to compel doctors to make any changes requested by the administration. This difference in relationship has implications for software adoption. In an ordinary company workers can be told by the administration or IT department to adopt a new technology, while the successful development of medical tools/software requires that doctors and administrators both see the benefits and hence agree to adopting a new technology.

The problems that occur during the development and acceptance of medical systems due to social and organizational issues that are specific to the medical field have been given very little attention in the past [40, 59]. This inattention was a cause for the failure of many medical informatics systems, where either physicians refused to accept new systems or stopped using medical systems to protest against them, *e.g.*, the Cedars-Sinai hospital uninstalled a multimillion dollar system as physicians stopped using it [40]. There is not really an effective solution available in software literature to overcome these problems, but it is recommended that designers involve physicians in the development process as much as possible. This is related to the issue of *physician champions* in medical software development, which will be discussed in more detail in later sections of this thesis.

2.2.2 System Operation

We have discussed many development problems and their solutions in the previous section. Let us now consider the problems that are associated with system operation. System operation is concerned with the issues related to short, mid, and long term system evolution and maintenance, *i.e.*, what a system does after it is developed. Here, issues related to maintenance are how system changes that arise from the day to day usage of the system, *e.g.*, adding a new users to system, are handled. System evolution is related to the need for change in the basic design of a system that arises after using that system for a long period of time. For example, in the case of a chemotherapy prescription system, if the basic method of prescribing chemotherapy drugs is changed, it will cause a need for a corresponding change in the design of the system. These issues are especially important for systems which are developed for a longer period, making it more likely for these issues to come forth.

Though system operation issues occasionally come up in software development, they are not satisfactorily addressed in software engineering literature at present. This is particularly unfortunate for us, as many medical informatics systems are long lived and face these problems. This will be discussed in more detail in Section 2.3.3.

2.3 Software Development

From Section 2.1 and Section 2.2, we see that medical software presents some interesting developmental problems. Here we will analyze these problems in terms of two components of the software development process, the software process model and the system operational model, and we will see how appropriate software engineering

techniques can help us to solve many of the problems arising during the development of medical informatics systems. In this section, we will briefly describe what each model is (Section 2.3.1 and 2.3.3), and the types of each model (Sections 2.3.2 and 2.3.4). The discussion of types of system operational models (Section 2.3.4) is particularly detailed, as this is not, to our knowledge, adequately addressed in the software literature. Finally, in Section 2.3.5, we discuss which software process and system operational models are most relevant to medical informatics.

2.3.1 What is a Software Process Model?

A software process model may be defined as a simplified description of a software process which is presented from a particular perspective [69]. In other words, we can say that a process model is a plan of action for a large and complex software development, which includes a clear statement of what is required and the different tools, steps and series of steps, required to successfully implement a software product [10, pp. 21-22].

In the early days of computers, software development was still evolving. Because projects were small, the programmers' own defined ways of development were sufficient to produce successful software. The most common way to develop software was to write code and then test it, and came to be called the build and fix model. With the passage of time, software projects became more complicated and difficult to develop with traditional programming techniques due to lack of knowledge about the software implementation. To overcome this problem new development models, also called software development paradigms, were introduced to cover the whole software

development life cycle. These models include a comprehensive guidance towards the development of software.

A model, also called the life cycle of a project development, consists of different phases, which can range from three (for a simple model, including Design, Development, and Maintenance) to more than twenty phases. However, most of the models include the phases of Requirement, Design, Implementation, Testing, Deployment, and Maintenance. Different process models include different iterations and orders of these phases, which make them suitable for particular circumstances. Hence, the selection of an appropriate model depends on the nature of the project and its constraints.

2.3.2 Types of Software Process Models

Today, we have many software process models, all of which are actually variations of four traditional software models.

2.3.2.1 Waterfall Model

The waterfall model was introduced by Royce in 1970 and is also called the linear sequential model [63]. In this model the following phases are completed in order.

1. Requirement Specification
2. Design
3. Implementation
4. Testing

5. Deployment

6. Maintenance

In the Waterfall model one should only move to the next phase on the completion of previous phase. There is no jumping back and forth, and there is no overlap between the phases. In spite of the fact that the system being developed is always well documented from the very beginning of the process, one of the main disadvantages of the Waterfall model is that not every part of the product is available until late in the development process. Thus, if mistakes or deficiencies exist in the documentation or earlier phases, they may not be discovered until the deployment of the software. Hence, correction must often be done in the maintenance phase. Because of its sequential nature, the Waterfall model is not applicable when the requirements are not clear and well understood at the start of the project. Moreover, in the Waterfall model the actual participation of the end user is negligible, and only the final version of the product can be delivered to users, which makes it unavailable for comments by the client in earlier stages.

2.3.2.2 Incremental Model

The Incremental model was introduced in 1975 by Basili [8]. The Incremental model is a series of waterfall cycles, where the requirements are known at the beginning and are divided into groups, and the initial group of requirements is fulfilled at the end of a series of several waterfall builds. Hence, this model can get evaluation from the client by showing a working part of software after each cycle, which can allow both alterations during development and the addition of new requirements during

the implementation of the system. In the Incremental model a usable product is available after the first release, and each iteration results in additional functionalities for the product. However, in this model, users are required to learn the new system developed in each cycle. Thus, the Incremental model is beneficial for projects where feedback is necessary from customers at early stages of project development.

2.3.2.3 Prototyping Model

This model, also called the Evolutionary model, was introduced by Floyd [26] and simply refines the prototype system in each iteration. Working software is built in the first iteration and then refined in later subsequent iterations. The specification, development, and testing phases are carried out concurrently. Rapid feedback is made possible by adding customer evaluation in each cycle. The final system will accurately fulfill the user needs; however the project is often started without full knowledge of requirements and thus needs greater coordination with the user.

2.3.2.4 Spiral Model

The Spiral model was defined by Boehm in 1988 [12], and is recommended for high-risk projects where the requirements must be refined and the user's needs must be met. The Spiral development model involves incremental builds which identify areas of risk and decide how to overcome and eliminate chances of risk via the validation and verification of the project in each iteration or build of the product. As such, it is obvious that the Spiral model combines features of the Waterfall, Incremental, and Prototyping models, and hence provides a great deal of user involvement and risk management. If we compare the Spiral model in more detail with other models,

its distinctive feature is that it deals with the risks and uncertainties involved in software development. The Spiral model explicitly recognizes the following risks and uncertainties [10]:

1. During long developments, the users are neglected in the process of gathering their requirements.
2. The users requirements are misunderstood.
3. The users change requirements.
4. The target hardware configuration changes.

The disadvantage of using the Spiral model is its complexity and greater cost. However, given the high risk associated with medical software, this is the preferred model for medical system development (see Section 2.3.5 for more discussion).

2.3.3 What is a System Operational Model?

A system operational model is a description of the short to midterm maintenance and evolution requirements for a given system. Here, system maintenance is support that is available for the operation of a system, which could either be support from the developer of the system or support by the system administrator, who is available onsite or contracted to perform different activities necessary for such system to keep them up and running. System evolution is how system requirements evolve in the short to long term during a system's operation. Both maintenance and evolution are changes to the system that require resources; hence, system operational model is the

description that includes not only the nature of maintenance and evolution but also take into account the resources that are available to make those changes.

How are system maintenance and system evolution typically handled in software engineering? The usual approach is to assume that there will always be sufficient resources for unlimited evolution and maintenance, *i.e.*, one or more system administrators and one or more programmers will always be there to bring about the changes required to handle evolution and maintenance. Under the assumption of unlimited change resources, there is no need to explicitly describe system evolution and maintenance. Instead, to deal with any future change in systems, proper software engineering practices are adopted when the initial system is developed, *e.g.*, structured programming is used and all necessary documentation is done, such that it is easy to make any changes which are necessary in the future. Unfortunately, as we will see in the next section, the assumption of unlimited change resources is not true for certain system operational environments.

2.3.4 Types of System Operational Model

Though there is no explicit mention of system operational models in the software engineering literature, there appear to be three types. Unlike system process models, which depend more on the complexity of the system, system operational models depend on the user community and to a lesser degree on the application being developed. These three types are as follows:

1. **Large Organizational Model:** This includes organizations that have effectively unlimited resources available for system maintenance and evolution.

These organizations can use their resources for system administration, and can handle any changes which they may want in their system due to long term system evolution.

2. **Personal Model:** This includes standalone systems developed for personal use. Users using such systems possess very limited resources for the maintenance and evolution of their systems.
3. **Small Organizational Model:** This includes small organizations or focused groups in large organizations, such as a cancer clinic in a large healthcare organization. These organizations have limited resources available, and cannot usually afford system maintenance and evolution.

This is to our knowledge an original classification of system operation models. We believe that this classification is useful not only for showing how existing systems operate, but also for showing types of systems not adequately addressed in current software development practice. This ordering of models may look somewhat strange, but this is the historical order in which these models emerged. To fully understand the differences between these three models, it is useful to look at this historical order in more detail.

This historical order can be nicely visualized in terms of a 3×2 table (see Table 2.1), whose dimensions are types of available support and size of user communities.

The types of support are:

1. **No Support:** A workplace where no system administrator is available and no or very little support is available from the developer of the system.

2. **Partial Support:** A workplace where support is available but is available either part-time onsite or full-time off-site, *e.g.*, support available via phone from the developer of the system.
3. **Full Support:** A workplace where support is available both, full-time onsite, *i.e.*, system administrators and programmers hired by organization, and/or full-time off-site.

The types of user communities are:

1. **Single-user systems:** Single-user systems were first introduced in the 1970s. These systems are developed for a standalone computer and are less complex and less expensive than multi-user systems.
2. **Multi-user Systems:** Multi-user systems were first introduced in the 1950s, when it was common for big organizations to use such systems. Multi-user systems are more complex and expensive than single-user systems. Due to their complexity and different functionalities such systems require full support and dedicated technical support on site.

Note that two possible combinations of support and user communities are very rare: Multi-user systems with no support, and single-user systems with full support. Hence, they are not addressed below.

Let us now look at the historical order in which the models emerged. When computers were first introduced they came only in a multi-user environment. Initial computer systems in 1950s were targeted to hundreds or thousands of users, and those multi-user systems were, by definition, complex systems. Organizations that

were using such multi-user systems had their own IT divisions, or had the resources to contract different companies for the full system support. These systems were developed under the Large Organizational Model. In the late 1970s and early 1980s stand alone systems were introduced for the use of single users. These systems were simple, but had no support from the developers. Users using these systems were expected to be smart enough to be able to fix any problems that might arise while using them. Therefore, these systems were effectively under the Personal Model.

At this point we have pure versions of the first two models. However, the era of the mid 1980's saw the introduction of partial support, which did not affect big businesses, but did change the Personal Model by moving it from the upper left corner down the column. In this model, users who pay several thousand dollars for a system receive help from designated staff belonging to the vendor companies who created these systems. Because of the large number of users entitled to such services, providing support to them was not an issue for big venter companies, due to the economic factors involved. Hence, single-user systems were the first type of systems for which partial support was provided. As more and more people got into the market one of the ways for system developers to retain market share became the provision of user support. This now meant that by the addition of partial support, single-user systems could become more complex, and in turn could be adapted to the needs of small organizations. This led to the emergence of the third operational model, the Small Organizational Model.

Now that we have seen how these three models emerged, let us examine how each model deals with the issues of software maintenance and software evolution:

1. **Large Organizational Model:** In the case of the Large Organizational Model, as such organizations have effectively unlimited resources, they can afford to allocate these resources as necessary for both system maintenance and evolution.
2. **Personal Model:** In the case of the Personal Model, for maintenance, either the person or the developer company is responsible for system maintenance and evolution. For future system evolution under this model, software companies are always trying to establish the needs of their users. As the number of personal-system users is large, big software companies can invest a lot of money to make a good general package for them. Hence, personal computer evolution is handled very well. In the case of system maintenance, as companies sell these systems to many users, they have the resources to maintain them and to provide support, such as 24/7 help lines or technical support available over the web to correct problems. As systems under the personal model are relatively simple compared to multi-user systems, partial support is sufficient for their maintenance and evolution.
3. **Small Organizational Model:** In the case of the Small Organizational Model, organizations lack the resources to handle the maintenance and evolution. Moreover, these organizations do not have the numbers to encourage the emergence of vendor companies that will provide them with even partial support.

As we can see from the above, there is a gap in how to deal with a system operating under the Small Organizational Model. One way to deal with maintenance for these organizations is that we can minimize the need for full time system administrator support by building systems in such a way that the system users can handle adminis-

tration themselves. To handle evolution, in case of organizations whose activities are small and very circumscribed, with sufficient design during the development of the system; we can also put the evolution of such systems in the hands of their users.

To handle system operation in the Small Organizational Model, we have introduced a democratic model of system operation (see Section 4.2.1). Though this model is described in detail in Section 4.2.1, the following points should be noted here; first, we have built this model so that system changes that are required based on day to day activities are in the user's hands. Second, although the system administrator cannot be fully removed from the loop, minor day to day changes can be made by the users using this democratic approach, so that even partial support is sufficient.

2.3.5 Which System Process and Operational Models are Appropriate for Medical Informatics?

By looking at the complexity of medical systems and taking into consideration the risk involved in the process of developing a medical tool, the Spiral model (see Section 2.3.2.4) seems to be the most appropriate one. As this model needs strong user involvement in the process of development, there is a need to find computer literate physicians who can act as physician champions and can help throughout the process of developing a successful medical tool.

The system operational model depends on the type of medical system being developed. For EMRs/EHRs, and general drug orders CPOE will fall under the large organizational model. Decision support systems, if they are done frequently enough, will fall under the personal model. Infrequently used decision support systems and

targeted CPOE systems will fall under the Small Organizational Model. Hence medical informatics tools fall into all three categories of system operational model. In Chapter 3, we will see that chemotherapy prescription systems fall under the Small Organizational Model and we will show in detail how limited resources can cause problems for the operation of such systems.

Table 2.1: Types of System Operational Models

	Single-User Environment	Multi-User Environment
No Support	Personal ✓ (1970's)	×
Partial Support	Personal ✓ (1980's+)	Organizational [Small] ? (1990's+)
Full Support	×	Organizational ✓ [Large] (1950's+)

Chapter 3

Chemotherapy Prescription

In this chapter we will look at a particular type of order entry system associated with chemotherapy prescription. Such a system is an example of a CPOE system (see Section 2.1). CPOE systems can be of a generalized nature, covering all types of medical prescription writing, or can be targeted systems, where a system is designed to computerize a specific type of medical treatment such as chemotherapy. Targeted CPOE systems inherit all the problems of the generalized CPOE, but also introduce some new problems due to their targeted nature. Hence, in this chapter we will discuss CPOE in general (Section 3.1), and we will look at an example of a targeted CPOE, a chemotherapy prescription (Section 3.2). Finally, we will review previous efforts to develop chemotherapy prescription systems and we will sketch the form of an idealized system that builds on this work (Section 3.3).

3.1 CPOE

In Section 2.2, we looked at the problems associated with medical software in general. In this section, we will look at these problems in the context of the particular system of interest to us in this thesis, namely CPOE. In this section, we more closely examine the definition of CPOE (Section 3.1.1), and we review the benefits and problems associated with CPOE implementation (Section 3.1.2), as well as the problems (Section 3.1.3). Finally, we will summarize the factors associated with successful CPOE implementation (Section 3.1.4).

3.1.1 What is CPOE?

CPOE (Computerized Physician Order Entry) is the generic name for a computer application which was introduced as an effort to eliminate the chance of errors, such as errors in ordering medication by a physician (see Section 3.1.2). To eliminate the chance of errors, instead of writing orders on a prescription pad, the prescriber enters an order directly into the computer. The functionalities of CPOE systems vary in different CPOE tools. Kaushal and Bates describe the basic functionalities of CPOE tools as follows: “...Basic CPOE ensures standardized, legible, complete orders by only accepting typed orders in a standard and complete format... Basic clinical decision support may include suggestions or default values for drug doses, routes, and frequencies...” [35]. Advanced CPOE systems may include, but are not limited to, decision support, drug dose recommendation, drug interaction notification, and billing [13].

3.1.2 Benefits of CPOE

Studies have shown that CPOE systems are effective in reducing the medication errors that frequently occur in the paper based system currently in use by the majority of medical service providers [9, 20, 38, 60]. Note that even a basic CPOE system can be very helpful in reducing calculation errors in prescriptions which involve complex calculations (such as protocols for cancer chemotherapy or HIV therapy). These and other benefits are described in more detail below.

3.1.2.1 Reduction of Medical Errors

Every year, thousands of patients world-wide die as a result of medication errors [36]. Mistakes such as errors in dosage, improper routes of administration, and wrong dosage schedules, among others, are typically responsible for such events. Such mistakes have at least two causes: The first cause is Doctors' poor handwriting, which sometimes leaves pharmacists squinting to read drug names or dosages, and causes them to provide improper medication or dosages by misunderstanding the prescription. The second cause of medication errors arises when prescription writing involves complex calculations. Given the actions of some drugs, especially the cell toxic drugs involved in chemotherapy, any mistake in calculation of dosage has potential to cause patient's injury or even death. This problem is compounded when prescriptions involve multiple drugs; for example, in a chemotherapy prescription, a selection of drugs is given and the dosage of each drug involves calculations (see Section 3.2.1 for details). Even if neither of the above problems is present, the prescription is written perfectly, and everything is calculated correctly, adverse drug events (in which several

drugs combine to have unexpected side effects) can cause another category of errors where wrong combinations of drugs can cause harm to the patient¹.

CPOE systems are regarded as the solution to medication errors which arise from prescription writing. CPOE solves the problems of poor handwriting, as all prescriptions are computer generated and all calculations are done by computer, eliminating the chance of calculation errors. Moreover, CPOE can check the possibility of any known adverse drug events automatically. The results of CPOE can be impressive; published studies have shown that CPOE can prevent 81 percent of errors related to prescription writing [9, 20, 38, 60].

3.1.2.2 Time and Cost

Most studies on CPOE benefits are related to CPOE's ability to reduce medication errors, as this is the primary purpose of CPOE systems. However, CPOE systems also have the ability to reduce overall hospital costs. Evans *et al* [25], have shown that CPOE systems help to reduce the cost of medication and the length of hospital stays by reducing the number of orders for antibiotics when anti-infective-management programs were used. Moreover, the ability of CPOE to reduce adverse drug events also causes a decrease in the costs associated with such events, which are, according to a report by the Institute of Medicine, about 2 billion dollars a year [36].

¹There is another type of error related to deficiency of medication knowledge, which may occur because of less experienced physicians entering into the field, or when physicians occasionally find themselves outside their normal area of expertise [11]. Though this could potentially be mitigated by computerization, *i.e.*, an appropriately designed expert system, this is beyond the scope of this thesis and will not be considered further herein.

3.1.2.3 Research Purposes

Electronic data is always attractive for research purposes. CPOE data is stored electronically in itemized form in databases, *i.e.*, not in scanned document form; hence, it provides electronic data which is more readily available for research, which in turn can improve the quality of healthcare. For example, in the case of chemotherapy treatment, where different combinations of drugs are used to treat cancer, medical data can help researchers to find the best combinations of drugs for the treatment of particular types of cancer.

3.1.2.4 Administrative Purposes

A computerized medical system like CPOE can be useful for administrative purposes, such as evaluation of clinical staff performance. For instance, useful information can be retrieved about how often a particular drug is prescribed by the physician and how much a particular drug costs per year. In this way, administrators can maximize the effectiveness of their resource allocation.

3.1.3 Problems with CPOE

CPOE has been under discussion for over 35 years, which has led to an understanding of many problems with CPOE systems. In 1970, Collen [19] introduced the concept of CPOE by listing the general objectives of Medical Information Management Systems, and stated that, "*Physicians should enter medical orders directly into the computer*". Following this, there were various efforts to implement CPOE systems. Sittig and Stead [68], in 1993, summarized the previous work done related to CPOE. According

to their paper most of the efforts made in the 1970s and 1980s met with failure. Technology constraints at that time, costs, and lack of computer literacy within the medical profession were among the major causes of these systems' failures.

Oddly, recent implementations of CPOE at different sites do not show significant improvements, despite advancements in technology and sufficient computer literacy in the medical profession. As a result, many of these efforts have failed. For example, Cedars-Sinai Hospital in Los Angeles implemented a multimillion-dollar CPOE in late 2002 and three months after implementation, the tool was uninstalled as physicians complained about the poor design of the system [40]. If technology has improved, the cost of technology has decreased, software design and implementation methodologies have improved, and computer literacy in the medical profession has improved, then other reasons must be responsible for the failure of these systems.

The main reason for these failures seems to be a lack of respect by system developers for the clinical workflow, in particular, where insufficient attentions has been paid to the details of going from manual to computerized workflow [40, 59]. This was acknowledged by Michael L. Langberg, M.D., Chief Medical Officer at Cedars-Sinai Medical Center, who stated that, *"One of the most important lessons learned to date is that the complexity of human change management may be easily underestimated"* [40]. We will discuss this in much greater detail when we look at the specifics of chemotherapy prescription in Section 3.3. In the next section, we will discuss the factors that make for a successful CPOE system.

3.1.4 Factors in Successful CPOE Implementations

As seen in previous subsection, many CPOE systems have failed because physicians stopped using them, and one of the main causes seems to be inattention to clinical workflow by system developers. Physicians need a system that is guaranteed to help them provide quality care to their patients and, obviously, they cannot make compromises in patient care. If a tool is designed in keeping with established physician and healthcare workflows (which are already known to work well), then there are fewer chances for that tool to fail. Thus, to make CPOE acceptable for physicians to adopt, it must have at least the following properties [6, 78]:

- It should be accurate and reliable so as to positively affect patient care.
 - While implementing CPOE systems, legacy systems currently in use by the health care providers should be taken into consideration.
 - Physicians should be given full authority to make any decisions about patient health. Imposing something against a physician's final decision should be avoided.
 - It should be fast enough that it improves the speed of workflow or at least it should not be slower than the existing system.
 - It should be easy to use and should require minimal training for effective use.
- A striking feature of medical work is that it is fast paced. Physicians have little time to spare to learn new technologies. Hence, the new CPOE system should be simple enough that it takes little time for physicians to learn it.

- Interface issues should be taken into consideration while designing such systems and the interface should be consistent throughout the system.
- It should have standardization with respect to medical procedures and terminology and a workflow that can be effectively implemented in healthcare.
- During system implementation, physicians should receive any help they need to change their workflow strategies and habits.
- After the system has been implemented, online help should be available.

These are the minimal requirements for a successful CPOE implementation. It is strongly recommended that, to fulfill and to understand the above characteristics, physicians should be an active part of the implementation. Thus there is a need to search for computer literate physicians, referred to as “physician champions” throughout the literature [59, 65], if the process of implementation is to be successful.

When compared to “ordinary” software, the development of medical software/tools is much more complex and risky, as if it is done incorrectly, people can die and it will have excessive costs (see Section 3.1.2). Careful consideration should be given to make sure that the appropriate software process model is selected to deal with this complexity. Moreover, we also need to choose the appropriate system operational model. This is necessary because CPOE, as noted earlier, comes in several sizes. The general CPOE falls under Large Organizational Model. However, as we will see in next section, chemotherapy prescription seems to fall under the Small Organizational Model. It means that chemotherapy prescription has fewer available resources, and its implications will be discussed and dealt with in the remainder of this thesis.

3.2 Chemotherapy Prescription

In this section, we will now examine a specific type of CPOE in detail, *i.e.*, chemotherapy prescription. We will structure this system in parallel with that for CPOE, by first giving a description of chemotherapy (Section 3.2.1). Recall that one of the main problems with CPOE is not paying attention to the workflow of healthcare system. Therefore, we give an overview of chemotherapy prescription workflow (Section 3.2.2), and we will see how this manual workflow can be improved by replacing it with computerized workflow (Section 3.2.3). We will look at the benefits of the computerized workflow (Section 3.2.4), and considerations concerning the development of computerized workflow (Section 3.2.5).

3.2.1 What is Chemotherapy?

Chemotherapy refers to the use of drugs to treat an illness (chemotherapy = chemical-therapy) [3]. Where cancer is concerned, chemotherapy is used to either destroy cancer cells completely or, when this is not possible, to control the growth of these cells [2]. Most cancer chemotherapeutic drugs have been chosen because they act as poisons that attack dividing cells. The typical side effects of cell-toxic drugs result from their effects on other rapidly dividing cells such as hair follicles and bone marrow, which produces new red and white blood cells. Short-term effects of chemotherapy include hair loss, bone marrow suppression, anemia, and susceptibility to infection.

There is a large number of different drugs that are used to treat cancer. Each type of cancer is only sensitive to particular drugs, and there are many different types of drugs. In most cases, a combination of drugs, also called a **drug regimen**,

is given in order to maximize the efficiency of the treatment. This combination of drugs is also called **combinational chemotherapy**. Each chemotherapy drug has an associated dosage, route of delivery, frequency of application, and associated additional instructions as well as a collection of emetogenic levels which vary from 0 to 5 based on the dosage of that drug. The dosages of the drugs depend on both the type of cancer, and on patient characteristics such as height and weight (see Section 4.4.1.3 for details). Antiemetic drugs are also part of each drug regimen, to minimize the side effects of chemotherapy drugs (see [49] and references). Antiemetic drugs are selected based on the emetogenicity level of the chemotherapy drugs which are part of regimen (see Section 4.4.2.2 for algorithm to select antiemetic drugs). Each antiemetic drug has an associated dosage, route of delivery, and frequency of application.

3.2.2 Manual Chemotherapy Workflow

The manual chemotherapy workflow is shown in Figure 3.1. The physician makes decisions based on the patient's history and medical conditions. These decisions could involve writing a new prescription for a new patient or altering or continuing an old prescription. The prescription is then sent to the pharmacy.

This manual workflow has the following categories of problems:

1. **Immediate problems:** The process of chemotherapy involves calculations based on the patient's height and weight. A minor mistake in these calculations, mistakes in the schedule of dosages, or improper methods of administration can have severe effects on the patient's health. Moreover, physicians' notoriously poor handwriting often leaves pharmacists squinting to decipher a dosage *e.g.*,

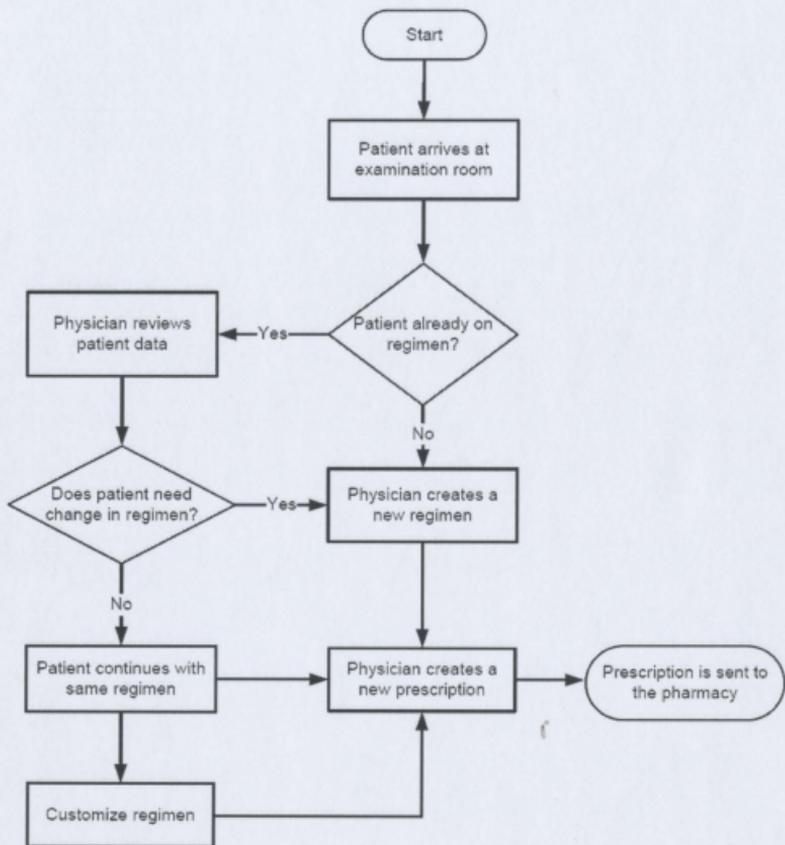


Figure 3.1: Workflow Diagram: Manual Chemotherapy Prescription Process

10 milligrams versus 10 micrograms. These mistakes can even happen with the name of prescribed drugs (see Section 3.1.2).

2. **Secondary problems:** Patient data in files are not readily available for secondary uses such as research work or generating reports. It is almost impossible for a physician to get a report based on large groups of patients, as it would cost too much time and money to compile a report based on patient data stored in paper files. Similarly, patient data can only be used for research purposes if it is stripped of names and other identifying information, which is difficult to do in the manual chemotherapy workflow.

Notice that many of these problems can be resolved by the computerized workflow described in the next subsection.

3.2.3 Computerized Chemotherapy Workflow

From Section 3.1 on CPOE and Section 3.2.1 on the chemotherapy prescription process, we can see that the chemotherapy prescription process is an ideal case for the implementation of CPOE. Such implementation can greatly reduce the inherent problems attached to the manual chemotherapy prescription workflow mentioned in previous subsection. One possible computerized workflow is shown in Figure 3.2. Notice that this computerized workflow is designed to have as few differences from the manual workflow as possible. However, one big difference in this computerized workflow is that patient and regimen data is now stored in electronic form in databases.

3.2.4 Benefits of Computerized Workflow

In addition to the benefits attached to the implementation of a CPOE system that are described in Section 3.1.2, the implementation of a targeted chemotherapy CPOE system offers solutions to the problems related to manual workflow:

1. **Fixes to immediate problems:** With many chemotherapy regimens comprised of cell-toxic chemicals that must be individualized to patients' height, weight and diagnosis, delivering chemotherapy effectively and safely requires specialized knowledge and strict attention to detail. Computerized entry of such prescriptions can help to overcome potential problems of errors in dosages due to complex calculations. A computerized chemotherapy system can also help in fixing the problems that arise from the physicians' poor hand writing.
2. **Fixes to secondary problems:** A database can also help doctors in generating reports and looking beyond an individual patient to see how similar patients with similar medical histories and cultural backgrounds have been diagnosed and treated for similar problems. Similarly, a database can be made available for research work provided privacy is guaranteed to the involved parties. This can be made possible by stripping off names and other personally identifiable information or by anonymizing datasets by entry modification [75].

Along with the fixes to primary and secondary problems mentioned above, we have an additional benefit. Researchers are continuously trying new combinations to improve the quality of chemotherapy. These new combinations can be stored in the database to make them available to physicians in the prescribing process. The computerized system can now not only inform the physician about updates in chemotherapy drugs,

but will also let the physician add new regimens, which they have found effective against a particular cancer, to the database for others to use. These new regimens can be added into a private or public pool based on the consent of the physician who is adding them (see Section 4.4.2.2 for more details).

3.2.5 Consideration Concerning Computerized Workflow

Along with the long list of CPOE implementation considerations given in Section 3.1.4, the big thing which we need to consider for the development of chemotherapy prescriptions is the system operational model. CPOE is a mid-sized medical informatics application where we are dealing with anywhere between 3–10 doctors, who are interacting with 1–5 pharmacists. Chemotherapy and chemotherapy equipment are typically associated with cancer clinic. Regardless of whether or not this clinic is affiliated with a hospital, the cancer clinic will probably not be able to get full support with respect to system administrators and programmers to deal with system evolution and maintenance. In Section 4.2.1 we will see how this problem can be solved by the introduction of a democratic system operation model.

3.3 Previous Work

In the previous sections of this chapter, we noted that there are two issues that need to be addressed while developing a medical system: respect for existing clinical workflow and the system operational model. One aspect of workflow is deciding how to handle errors. If one of the big benefits of CPOE is the reduction of errors then one can handle errors in one of two ways: One can make it as difficult as possible

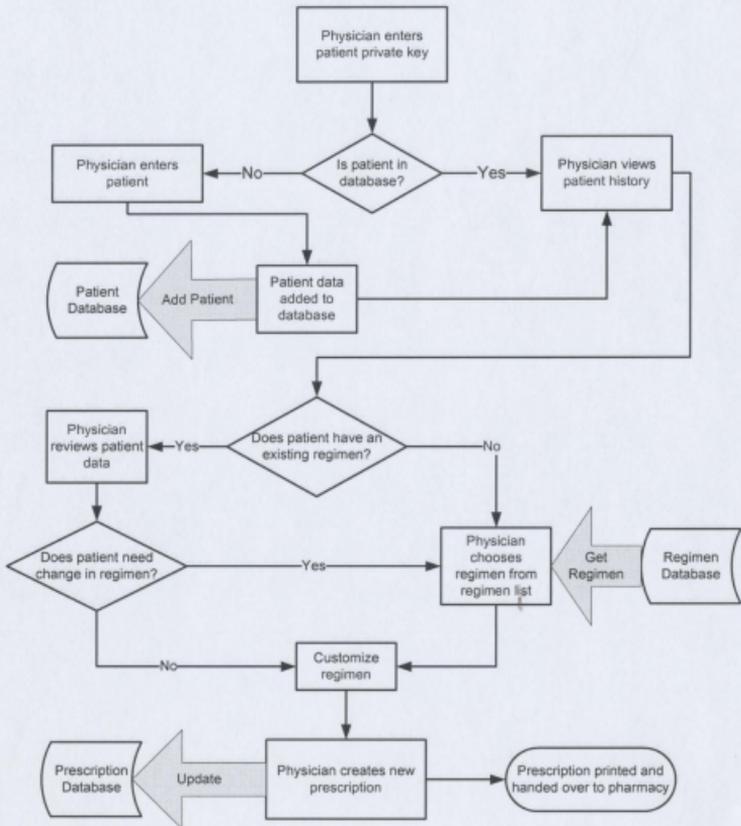


Figure 3.2: Workflow Diagram: Computerized Chemotherapy Prescription Process

to make errors, or one can make it as easy as possible to do the right things. In this section, we will discuss different attempts that have been made in the past to change the manual chemotherapy workflow shown in Section 3.2.2 into the idealized chemotherapy workflow detailed in Section 3.2.3. Past work that has been done can be divided into two categories based on the type of error-handling strategy followed while designing systems.

Systems using the first strategy make it hard for users to make errors. This is typically done by implementing rigorous error checking and by making sure that there is no deviation from the design workflow. In the case of medical systems, which are by definition complex systems, development using this approach is likely to compound the complexity of the system by making it difficult for the users to do what they need to do. For example, if the diagnostic workflow is implemented in a rigid fashion, such that a doctor cannot view a patient history unless that doctor is already treating that patient, *i.e.*, the doctor has already ordered the lab tests for that patient, a doctor in emergency board will not be able to write prescription for a patient, that has just come under their care. Such problems have resulted in the failure of physician to fully adopt such systems. To our knowledge, the major available chemotherapy prescription systems have all used this strategy. They are general CPOE systems, such as *Opis 2000* [54] and *Meditech* [44], that have been customized to work for chemotherapy. Sample screenshots of Meditech are shown in Figures 3.3. Note that while these systems are built under the philosophy of making things in such a way that it make hard for people to create errors, they are incredibly difficult to use. Moreover, as these systems use the Large Organizational model, any customization for use by small organizations such as cancer clinics is very difficult to do.

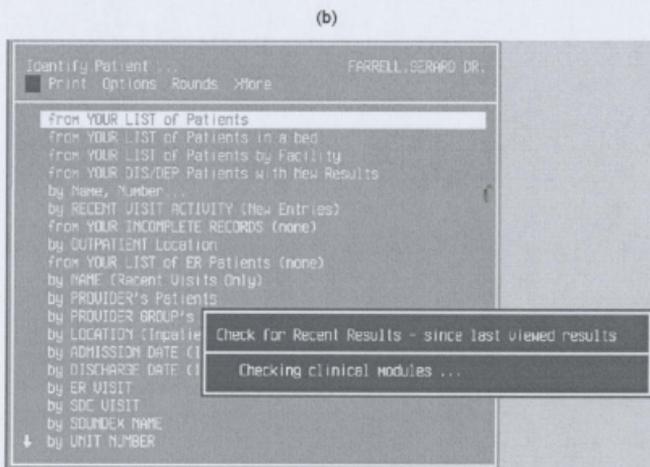
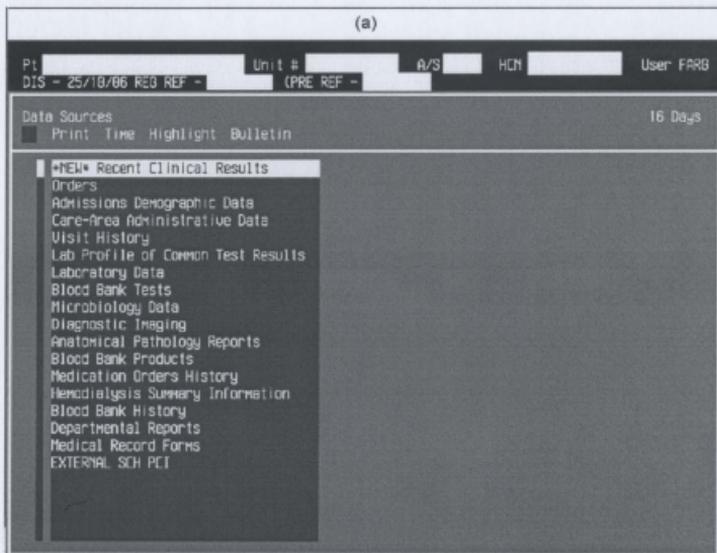


Figure 3.3: Screenshots: Meditech - (a) Patient Profile (b) Finding Patient.

Systems using the second strategy make it easier for users to do the right thing. This is typically done by a more flexible implementation of the workflow and a less rigorous approach for error checking, in which errors are flagged but allowed if the doctor considers it necessary. For example, during patient data entry there may be some normal range check on data; if a doctor enters data outside the range, this potential error will be noted but allowed if physician finds it necessary. Note that part of this certainly involves the reengineering of the user interface, but this type of flexibility also has to be designed in to the system from the start of development.

To our knowledge, there are no commercially available systems that follow this strategy. However, Dr. Gerard Farrell, of MUN Health Sciences, over the last decade, has built a series of three systems on this philosophy:

- Version 1 was developed by Dr. Farrell in 1995. It was a set of hem-oncology chemotherapy prescription pages developed for the Newfoundland Cancer Treatment and Research Foundation (NCTRF) using Microsoft Excel. A sample prescription page is shown in Figure 3.4. This version computerized the paper prescription forms, and did automatic calculation of chemotherapy drug dosages, hence reducing the chances of calculation errors. However, this version had many problems because each oncologist had his own copy of prescription pages, *i.e.*, Excel sheets; hence, these pages were essentially open source and copies could be modified independently. Moreover, if someone altered the formula behind a dosage cell by mistake, it could lead to severe dosage errors. Minor changes in any drug regimen needed changes in all physical copies of that regimen, which was difficult to coordinate and enforce. Similarly, the addi-

tion of a new regimen involved the distribution of new copies to all physicians. Finally, prescription pages were technology dependent, and as the prescription pages were designed using Microsoft Excel, users had to have Microsoft Excel installed on their computers to view and use these pages.

- Version 2 was built by Dr. Farrell in 1997. It was a set of web-enabled prescription pages using Java Script. Sample pages of a prescription are shown in Figures 3.5 and 3.6. This system had hard coded web prescription pages, which solved the problems of technology dependence and as the system was implemented on web, there was only one copy of each regimen. Unfortunately, storage of prescription and patient data was not possible in this version.
- Version 3 was a collaborative effort of Dr. Farrell with the MUN Computer Science Medical Informatics Group; the result of this collaboration appeared as a student project [49]. This version was developed using Java Server Pages (JSP) [72], (a Java technology that dynamically generates HTML and XML), and prescription pages were attached to a database for storage of prescription and patient data. A sample prescription input page is shown in Figure 3.7, and the corresponding output page is shown in Figure 3.8.

Note that the third version overcame the problems associated with Versions 1 and 2, but it still lacked abilities of the idealized computerized workflow described in Section 3.2.3. All three versions were judged to function well with respect to the second workflow and error handling philosophy stated above. With respect to system operational models, though all were adequate to the extent that resources required for installation were minimal, there was no mechanism for modifying these systems.



Name of Patient

Hosp. & Ward

MCP Number

Chart Number

Medicine Program

Physician's Chemotherapeutic Medications

General Hospital Site

Height =	cms	Weight =	kgs
BSA =		0.00 m ²	
Total Bil =	ALP =	Cr =	

Diagnosis -

Drug Allergies -

Date - #####

Round off total dosages where appropriate

CHOP

CYCLOPHOSPHAMIDE

750 mg/m² =

0 mg in 100 mls Normal Saline IV
over 1 hour

DOXORUBICIN

50 mg/m² =

0 mg (in a syringe) IV push

VINCRIPTINE

1.4 mg/m² =

0.0 mg (MAXIMUM 2 mg)
in 20 mls Normal Saline
(in a syringe) IV push

PREDNISONE

100 mg PO od for 6 days

Anti-Emetics:

Dexamethasone 10 mg IV pre-chemo

Ondansetron 8 mg PO pre-chemo & 8 hrs POst chemo

Maxeran 10 mg PO/IV q4h prn

Physician's Signature

Figure 3.4: Farrell Version 1 - Sample Prescription Page

Antineoplastic Drugs Prescription Form

Dr. H. Bliss
Murphy
Cancer
Centre

Supporting
Cancer Treatment and
Research Foundation



300 Prince
Philip Drive,
St. John's,
NF, A1B 3V6

Enabling Hope
Despite

Telephone:
737-6480
Facsimile:
737-6795

Name: _____

MCP #: _____

Weight in Kilograms: _____

Height in Centimetres: _____

Calculate BSA and Full Dose Chemo

Body Surface Area: _____

Date: _____

AC(60/600), Cycle # _____

Percent of Protocol Dose Intended by Physician: 100% ▾

Drug Name	Dose (mg/m ²)	Dose (mg)	Route	Frequency
Maxeran (Metoclopramide) ▾		10 ▾	po ▾	pre-chemo
Anzemet (Dolasetron) ▾		100	po ▾	pre-chemo

Figure 3.5: Farrell Version 2 - Sample Prescription

<input type="text"/>	<input type="text"/>	<input type="text"/>	po	pre-chemo
<input type="text"/>	<input type="text"/>	<input type="text"/>		pre-chemo
Adriamycin (Doxorubicin)	(60)	<input type="text"/>	iv over 30 mins	Day 1 Only
Cyclophosphamide	(600)	<input type="text"/>	iv over 1 hr	Day 1 Only

Total hydration (in mls) should equal the dose of Cyclophosphamide (in mgs).
This prescription represents % of the full dose as per the protocol

This prescription has been rounded off by the signing physician if this box is checked:

Signature : _____

Figure 3.6: Farrell Version 2 - Sample Prescription (Cont'd)

There was no mechanism for creating a new regimen or altering an existing regimen which required a programmer to add new pages or modify existing ones to make any change or introduce a new regimen; indeed, as the implemented version 2 system aged, physicians were forced to modify system pages by manually overwriting the fields on the paper outputs. Moreover, note that none of those versions was explicitly multi-user or had even basic security features such as password-authorized access to the system.

In the next chapter, we will look at the design of a chemotherapy prescription system along the lines of the Farrell systems which solves all problems with Versions 1 through 3 mentioned above. Moreover, to support true multi-user functionality, our new system has abilities above and beyond the idealized computerized workflow described earlier.



Patient Name:

Hosp. & Ward:

MCP Number:

Chart Number:

Medicine Program

Physician's Chemotherapeutic Medications

General Hospital Site

Height = <input type="text" value="180"/> cms	Weight = <input type="text" value="80"/> kgs
Calculated BSA = <input type="text" value="2.01"/> m ²	BSA used = <input type="text" value="2.01"/> m ²
Total Bili = <input type="text" value="4"/>	ALP = <input type="text" value="5"/> CR = <input type="text" value="2.5"/>

Diagnosis: Drug Allergies:

Date:

Round off total dosages where appropriate

CHOP

Percent of Protocol Dose Intended by Physician:

Protocol Dose	Calculated Dose	Prescribed Dose
CYCLOPHOSPHAMIDE 750 mg/m ²	<input type="text" value="1510"/> mg	<input type="text" value="1500"/> mg in 100 ml Normal Saline IV over 1 hour
DOXORUBICIN 50 mg/m ²	<input type="text" value="101"/> mg	<input type="text" value="100"/> mg (in a syringe) IV push
VINCRIStINE 1.4 mg/m ²	<input type="text" value="2"/> mg	<input type="text" value="2"/> mg (MAXIMUM 2 mg) in 20 ml Normal Saline (in a syringe) IV push
PREDNISONE		100 mg PO od for 5 days

This prescription represents % of the full dose as per the protocol

Drug Name	Dose (mg/m ²)	Dose (mg)	Route	Frequency
<input type="text" value="Maxeran (Metoclopramide)"/>		<input type="text" value="10"/>	<input type="text" value="po"/>	pre-chemo
<input type="text" value="Anzemet (Dolasetron)"/>		<input type="text" value="100"/>	<input type="text" value="po"/>	pre-chemo
<input type="text" value="Decadron (Dexamethasone)"/>		<input type="text" value="8"/>	<input type="text" value="po"/>	pre-chemo
<input type="text" value="Ativan (Lorazepam)"/>		<input type="text" value="0.5"/>	<input type="text" value="po"/>	pre-chemo

Figure 3.7: Farrell Version 3 - Sample Prescription Input Page



Medicine Program

Physician's Chemotherapeutic medications

General Hospital Site

Patient Name: John Hopkins

Hosp. & Ward: Hematology

MCP Number: 234-543-234

Chart Number: 23

Height = 180 cms	Weight = 80 kgs
Calculated BSA = 2.01 m ²	BSA used = 2.01 m ²
Total Bst = 4	ALP = 5
	CR = 2.5

Diagnosis- nil Drug Allergies- nil

Date: Saturday, August 11, 2007 6:29:53 PM

CHOP

Protocol Dose	Calculated Dose	Prescribed Dose		
CYCLOPHOSPHAMIDE 750 mg/m ²	1507mg	1500mg in 100 mlis Normal Saline IV over 1 hour		
DOXORUBICIN 50 mg/m ²	100mg	100mg (in a syringe) IV push		
VINCRIPTINE 1.4 mg/m ²	2mg	2mg (MAXIMUM 2 mg) in 20 mlis Normal Saline (in a syringe) IV push		
PREDNISONE		100 mg PO od for 5days		
Drug Name	Dose (mg/m²)	Dose (mg)	Route	Frequency
Maxeran (Metoclopramide)		10	po	pre-chemo
Anzemet (Dolasetron)		100	po	pre-chemo
Decadron (Dexamethasone)		8	po	pre-chemo
Ativan (Lorazepam)		0.5	po	pre-chemo

Physician's Signature

Figure 3.8: Farrell Version 3 - Sample Prescription Output Page

Chapter 4

System Design

In this chapter and the succeeding two, we will design and implement a chemotherapy system along the lines sketched at the end of Chapter 3. This system will be developed under the Spiral model, which we have discussed in Chapter 2 as the best for developing medical informatics software (see Section 2.3.2.4). As we are building this new system to operate under the Small Organizational Model (see Section 2.3.4), we need to design the system in such a way that its operation fits the limited resources of the target workplace. In terms of user interface, we will give our system a similar look to the previous systems pioneered by Dr. Farrell, and we will use the same error handling philosophy used by these systems (see Section 3.3).

This chapter is organized as follows: We will first describe the objectives of our system (Section 4.1); as part of this we will describe the two key features of this system (Section 4.2), the democratic model of system operation, and our user interface philosophy. This is followed by a brief description of the objects in this system (Section 4.3). We will then describe the features associated with single-user functions

and multi-user functions of the system (Sections 4.4 and 4.5). The description of each function includes a description of workflow with a workflow diagram¹, a discussion of design decisions made related to that function, and one or more screenshots associated with the implementation of that function in the chemotherapy prescription system. This is followed by the detailed description of the database schema implementing the objects described in Section 4.3 (Section 4.6). Finally we will summarize the lessons learned while designing our chemotherapy prescription system (Section 4.7).

4.1 System Objectives

In this section, we will give a general overview of system objectives and functionalities; the details of specific tasks are included in specific subsections later in this chapter. The system objectives can be broken down into single-user and multi-user capabilities. Given the description of idealized computerized workflow in Section 3.2.3, we require the following single-user capabilities:

- The system should allow physicians to add new patients and to search for existing patients in the database. Moreover, the system should be able to keep patient profiles in order to allow physicians to look at the histories of patients.

¹It may seem contradictory that we are using flowcharts (which assume a rigorously defined workflow) to specify workflow when in Section 4.7, we claim that requirements in medical informatics software development are more often stated as flexible guidelines. Readers should therefore interpret the workflows flowcharts given in this thesis as the normative processes *i.e.*, the processes that occur 90 percent of the time, and exceptional situations (which occur in the remaining 10% of the time) are then superimposed on these flowcharts.

- The system should allow a physician to prescribe a regimen to any patient. Due to ongoing research, regimens change frequently over time; hence, the system should offer different functions to create new regimens and alter existing ones. Moreover, the system should have a convenient regimen management system, where regimens should be organized in to different lists, *i.e.*, primary and secondary, based on frequency of their usage. This is to provide quick access to regimens at the time of prescribing.
- The system should have a drug database and physicians should be able to add drugs to the database.

Given the requirements for multi-user operations implicit in Sections 3.2.3 and 3.3, we also need the following capabilities:

- The system should allow only authorized access to it.
- Physicians should be able to view the history of patients, which may include prescriptions issued by other physicians.
- Regimen data should be divided into public and private lists, such that a physician can choose to make a regimen public at the time of creation, and any physician can import a regimen from that public list to his/her private list at any time (see Section 4.5.2.1).
- Prescription data derived from regimens must be public. Any physician looking at the patient history must be able to look at the prescription history of the patient, where he can cancel any past prescription but can make changes to

only those prescriptions that were prescribed by him/her (see Section 4.4.1.3 for details).

There are several other multi-user capabilities, *i.e.*, changing the name or emetogenicity potential of chemotherapy drugs and adding or deleting users, which, while suggested by the individual users, should actually be decided by the whole user community. One solution for this would be to provide a mechanism for making, voting on, and resolving proposals within the user community (see Section 4.2.1 for details).

4.2 Key Features

Given the system described in Section 4.1, two key design features are a democratic model of system operation to deal with short to mid-term system evolution and maintenance, and the user interface. In this section, we will discuss both of these key features in more detail.

4.2.1 Democratic Model of System Operation

In Chapter 2, we discussed the operational problems, *i.e.*, system evolution and system maintenance, attached to small organizations. In Chapter 3, we realized that the chemotherapy prescription system we are going to develop falls under the Small Organizational Model. Thus, we have to deal with short, mid, and long term system evolution and maintenance under the assumption that the target workplace does not have enough resources to deal with such operational problems.

To handle system maintenance and evolution, we are going to put as many aspects of system maintenance and evolution as possible into the hands of the users

themselves. We have named this the **Democratic Model of System Operation**. To handle changes in both single-user and multi-user environments, we have built the system to be flexible enough that individual users can make only those changes that do not affect other users. When a change affects more than one user, the democratic process is invoked; this applies for both evolution and maintenance. Under the democratic model, in order to fulfill a requested change which affects multiple users, we have implemented a voting mechanism. To request a change in the system, users can issue new proposals and other users may vote on them. The details of this voting mechanism can be found in Section 4.5.

We have split the later design sections into single-user and multi-user. Under the democratic model, the aspects of maintenance and evolution are scattered in both single-user and multi-user functions. In single-user cases we have features to create or alter regimens and to import a regimen to a user's private list. Meanwhile, most of the multi-user functions fall under the democratic model of system administration.

Note that the democratic model is not a perfect solution for a systems operating under the Small Organizational Model. The democratic model can successfully handle short to mid-term changes. However, long-term changes such as changes in the algorithm for the calculation of emetogenicity potential (see Section 4.4.2.2) or fundamental changes to the way in which chemotherapy is prescribed cannot be handled by our proposed democratic model and will require the intervention of computer professionals. That being said, there is no suggested solution to such problems in the software engineering literature, and at least our proposed model can handle short to mid term changes well.

4.2.2 User Interface Design

The user interface design of our chemotherapy prescription system has the following characteristics:

- To make it easier for the users to adopt the new system, our chemotherapy prescription system mimics the format of the older paper-based versions, such that the prescription pages look exactly the same as pages of the old system and different features of the system mimic the workflow of the physician. This is in accordance with physician and bureaucratic behavior towards software tools as described in Section 2.2.1.4.
- To follow the error handling approach described in Section 3.3, *i.e.*, to make it easier for users to do the right thing, our chemotherapy prescription system has minimal interference in the workflow. For example, the system does not use popup windows to distract users when the system finds any potential error; instead, to notify the user of these potential errors, different color schemes are used, such that the color of field is changed to red for the field with the potential error and a message is displayed on the top of page.

Each of the tasks and subtasks described later in this chapter are illustrated with screenshots from the implemented user interface. From these screenshots, it can be seen that the implemented user interface has the characteristics described above.

4.3 System Objects and Classes

Before we describe in detail the tasks and subtasks making up the system in Sections 4.4 and 4.5, let us first describe the objects in the system. There are many objects implicit in the system as sketched in Chapter 3. However these objects can be divided into two groups: Entity objects, and Control objects. From an object-oriented perspective, an Entity object represents an object in the real-world problem domain while Control objects are responsible for workflow, implementation and task sequencing, as well as user navigation through the system [76].

The system has the following Entity objects: *Physician*, *Patient*, *Regimen*, *Prescription*, *Chemodrug*, *Antiemetic*, and *Administrator*. Note that *Physician* objects correspond to users of the system. The system has the following Control objects, which represent the individual components/tasks of the system: *Login*, *Logout*, *PasswordRetrieval*, *AddUser*, *DeleteUser*, *AddPatient*, *PatientSearch*, *AddDrug*, *CreateRegimen*, *AlterRegimen*, *ImportRegimen*, *RegimenManagement*, *PrescribeRegimen*, *CancelPrescription*, *EditPrescription*, *ViewRegimenList*, *AddComments*, *ViewComments*, *AddProposal*, and *VoteProposal*. The implementation of the Entity objects in terms of database schema is described in Section 4.6. Though Control objects are invoked to implement the tasks and subtasks described in Section 4.4 and 4.5, it is not necessary to examine the details here and they will not be referred to in the remainder of this thesis.

4.4 Single User Design

Workflow associated with single-user is subdivided into smaller subtasks. These subtasks are grouped into three categories based on activity duration: single-session (Section 4.4.1), short-term (Section 4.4.2), and mid-term (Section 4.4.3). This breakdown of tasks into smaller subtasks is shown in Table 4.1.

Before we look into the actual description of each task, we would like to give an overview of the level structure and class naming conventions used in Table 4.1. As this system is designed relative to the MVC model (see Section 5.2), it is split into the following three layers:

1. **Presentation layer:** The presentation layer includes all form beans², which also act as the form data validation layer. In Table 4.1, all classes that end with `Form` are part of the presentation layer. Notice that a few tasks do not have `Form` classes involved in them. This is because those particular tasks do not include any form data. Hence, for any task X, if it includes form data there is a class `XForm`, which acts as a form bean and also validates the form data.
2. **Business logic layer:** The business logic layer lies in between the presentation layer and the data layer and includes classes that are used to handle the information exchange between a database and a user interface. In Table 4.1 all classes that end with `Action` are part of the business layer. Notice that, for all

²A Java Bean is a Java class that follows a specific set of interface specifications. It is a reusable software component that can be manipulated in an application builder tool. A form bean is a type of Java bean. A form bean stores HTML form data from a submitted client request and the data bean provides a simple representation of an entity bean [74].

tasks in the table, there is a corresponding `Action` class attached to it, which performs the necessary functions to fulfill that task. Hence, for any Task X, there is a class `XAction` to perform that task.

- 3. Data layer:** The data layer or data object layer is responsible for the creation of data beans to be transferred to the business layer. Hence, for every database table there is a corresponding data bean class with the same name. Notice that for all data bean class X there is a corresponding `XData` class that is responsible for connection to the database, and for making all types of queries to the database table encoding data of type X. Here, also notice that a number of the entity objects that we have seen in the previous section also map to the database tables. For each entity object there is one or more corresponding table in the database; see Section 4.6 for details.

In Table 4.1, for any task X, `standard` in the class column means standard classes attached to the task that are `XForm` and `XAction`. Similarly every data bean³ class Y has a `YData` class which is responsible for connecting to the database and for all queries to the database table encoding data of type Y. As each data bean class has a corresponding `Data` class, it is not shown in the table.

4.4.1 Single-Session Activities

Single session activities include all activities that a physician does during one session of system usage. This includes the functions Login/Logout (Section 4.4.1.1), Search Patient (Section 4.4.1.2), and Prescribe Regimen (Section 4.4.1.3).

³See Footnote 2

Table 4.1: Single User Design Table

Task Duration	Task (Section #)	Sub-Task (Diagram #)	Associated Classes
Single Session	Login/Logout (4.4.1.1)	Login (4.1)	Standard + Physician
		Logout (4.1)	LogoutAction
	Search Patient (4.4.1.2)	By MCP (4.3)	Standard + Patient
		By Demographic (4.5)	Standard + Patient
Prescribe Regimen (4.4.1.3)	Prescribe Regimen (4.6)	Standard + Prescription + Regimen	
Short Term	Add Patient (4.4.2.1)	Add Patient (4.9)	Standard + Patient
	Create Regimen (4.4.2.2)	CreateRegimen (4.11)	Standard + Regimen
	Alter Regimen (4.4.2.3)	Alter Regimen (4.15)	Standard + Regimen
	Manage Regimen (4.4.2.4)	Manage Regimen (4.18)	Standard + Regimen
Mid Term	Add Drug (4.4.3.1)	Add Drug (4.20)	Standard + Drug
	Password Retrieval (4.4.3.2)	Password Retrieval (4.22)	Standard + Physician

4.4.1.1 Login/Logout

In order to log in to the system, the user enters his/her username and password on the login screen. The system validates the username and password against the information in the database. In the case of a match, the system creates the session variables on the server, and users can use the chemotherapy prescription system. If the login information entered by a user is incorrect that user is sent back to the login screen and error messages are displayed. When a user selects the logout option all session variables are removed from the server and the user is brought back to the login screen. A workflow diagram for this function is shown in Figure 4.1, and a screenshot of the implemented login page is shown in Figure 4.2.

The following decisions were made during the design of the different features involved in the *Login/Logout* function:

1. **Length of username and password:** The security of a password is usually measured by its length, but research has shown that if very lengthy passwords are enforced to increase security it makes it more difficult for users to remember them, which causes them to write them somewhere or to forget their password too regularly [1]. Hence, the password should be lengthy enough that it cannot be broken and simultaneously should be small enough that the user can easily remember it. We have followed the best practices proposed by Gartner [1], and we have required that passwords have the following properties:

- They must be at least 8 characters long
- They may contain alpha-numeric characters
- They are case sensitive

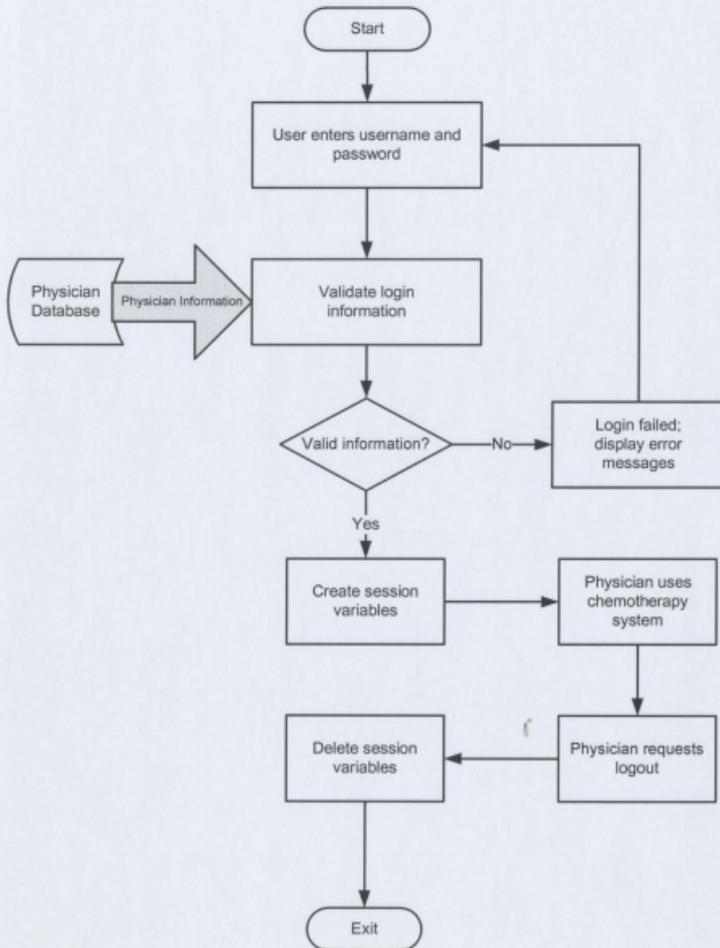


Figure 4.1: Workflow Diagram: Login/Logout

Please
Logon

Please enter your username and password

username:

password:

[Forgot your password?](#)

[Don't have a username?](#)

[Request an account now](#)

Figure 4.2: Screenshot: Login

- They cannot be the same as the username

Similarly, the username is restricted to a minimum length of 6 characters.

2. **Invalid login attempts:** There is no limit on the number of login attempts by the user. This may seem non-standard from the viewpoint of security, as a common practice is to lock the user account after a fixed number of invalid login attempts. However, given the clinical environment, where physicians need to have constant access to the system locking a physician's account under such circumstances is not feasible.

4.4.1.2 Search Patient

To find a patient from the database, the user enters the MCP⁴ number of the patient. In certain cases, where a patient does not have an MCP number available, the user can use patient demographic information to search the patient. The chemotherapy system searches the patient information database and returns results based on the type of search it performs. Workflow diagrams for searching by MCP and searching by demographic information are shown in Figures 4.3 and 4.5 respectively, and a screenshot of the implemented patient search page is shown in Figure 4.4.

The following decisions were made during the design of different features involved in the *Patient Search* functions:

⁴In order to uniquely identify patients in the patient database, there is a need for a unique key. As the system is being implemented in Newfoundland and Labrador, we are using the MCP number that the provincial government issues uniquely to every patient [52]. Note that any implementation of this system relative to another healthcare jurisdiction would have to find a corresponding uniquely identifying number for patients

1. **Search by MCP number:** As the MCP number is unique for every patient, the primary search is based on MCP number.

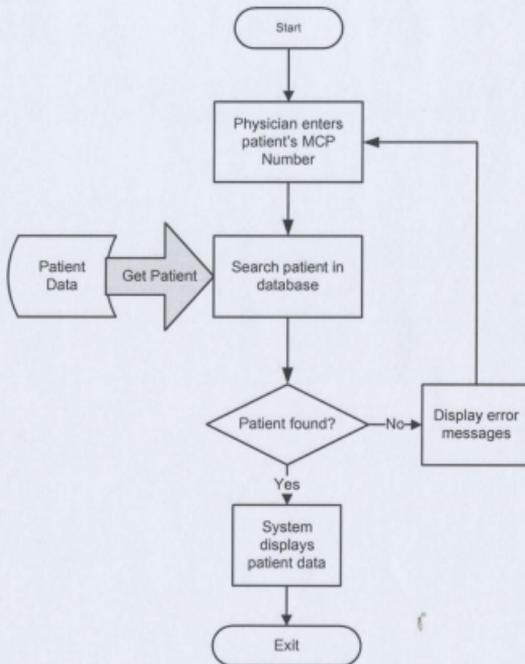


Figure 4.3: Workflow Diagram: Patient Search by MCP Number

2. **Search for patient's demographic information:** In search by demographic information, users can enter the available information on a patient to search for a patient or list of patients from the database. This tool is extremely helpful in

emergencies or other cases where the patients MCP number is not available.

HealthCare
Corporation of St. John's

Patient Search

Quick Links

- ▶ Patient Search
- ▶ Prescribe Regimen
- ▶ Add Patient
- ▶ Create Regimen
- ▶ Alter Regimen
- ▶ Import Regimen (0 New)
- ▶ View Deleted Regimens
- ▶ View Regimens
- ▶ Admin
- ▶ Logout

Search by MCP

Enter MCP Number:

Search by Patient Personal Information

First Name :

Last Name :

Gender :

Date of Birth :

Postal Code :

Page last updated: 09/13/2007 01:25:10

Figure 4.4: Screenshot: Patient Search Page

4.4.1.3 Prescribe Regimen

In order to prescribe a regimen, the user must first search for the patient and then look at the patient history (see Section 4.4.1.2 for details of patient search). To prescribe a regimen, the user selects a regimen from his private list of regimens (see

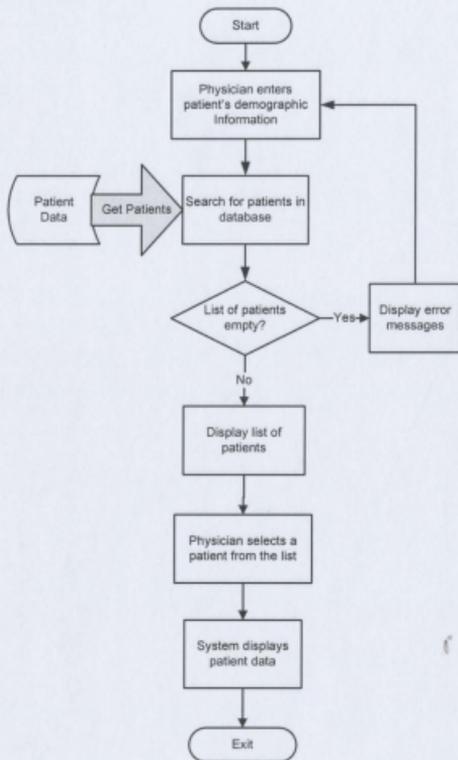


Figure 4.5: Workflow Diagram: Patient Search by Demographic Information

Section 4.4.2.4). The system then generates the prescription with patient information on it, the user enters the patient height and weight, and the system calculates dosages of different drugs (see below for details). The user can make changes in dosages if necessary. Once the system generates a printable version of the prescription, the user prints/saves the prescription. The workflow diagram for this process is shown in Figure 4.6, and screenshot of the implemented prescription input and output pages are shown in Figure 4.7, and 4.8, respectively.

The system calculates dosages of different drugs in a given regimen based on the Body Surface Area (BSA) of the patient. This BSA is calculated using the height and weight of the patient entered by the physician, using following formula:

$$BSA(m^2) = Weight(kg)^{0.427} \times Height(cm)^{0.718} \times 0.007449$$

Drug dosage is then typically a function of BSA. For example, the dosage for Cyclophosphamide is calculated as:

$$Cyclophosphamide(mg) = 750mg \times BSA$$

Note that, the user can modify automatically calculated dosages in two ways: Either as a group, by applying a common factor to each, or individually by changing individual dosage fields.

The following decisions were made during the design of different features involved in the *Prescribe Regimen* function:

1. **Selection of prescription:** In order to show physicians the available regimens to prescribe, the regimens are divided into two lists. Physicians can find more commonly used regimens in the primary list and less commonly used ones in

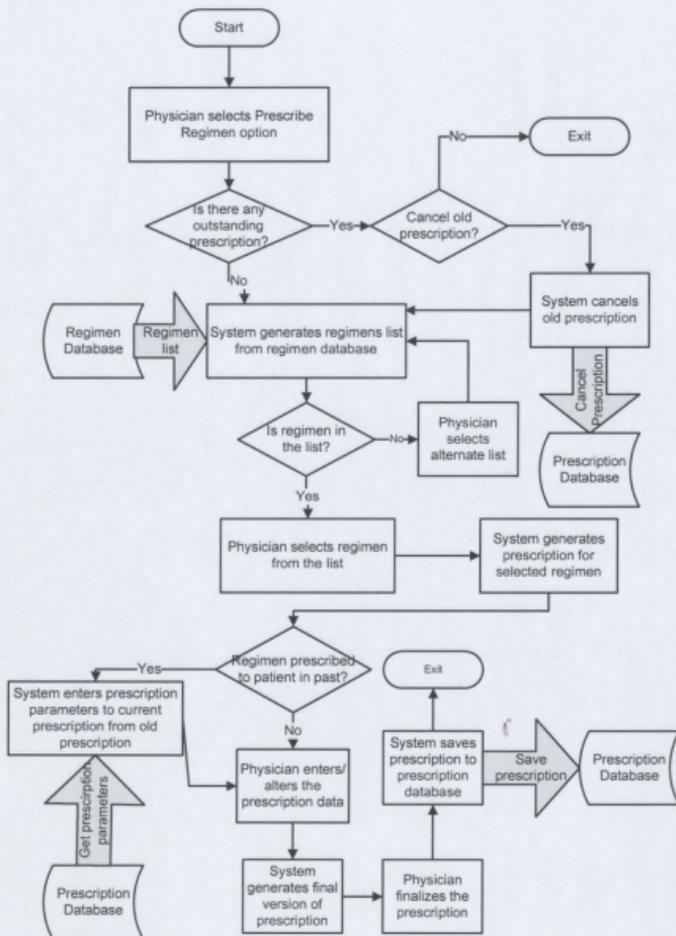


Figure 4.6: Workflow Diagram: Prescribe Regimen



Medicine Program

Physician's Chemotherapeutic Medications

General Hospital Site

Patient Name: John Hopkins

Hosp. & Ward: Hematology

MCP Number: 234-543-234

Chart Number: 23

Height = 180 cms

Weight = 80 kgs

Calculated BSA = 2.01 m²

BSA used = 2.01 m²

Total Bil = 4

ALP = 5

CR = 2.5

Diagnosis- nil

Drug Allergies- nil

Date: 08/12/2007

(MM/DD/YYYY)

Round off total dosages where appropriate

CHOP

Percent of Protocol Dose Intended by Physician: 100% ▾

Protocol Dose	Calculated Dose	Prescribed Dose
CYCLOPHOSPHAMIDE		
750 mg/m ²	1510 mg	1500 mg IV over 1 hour
DOXORUBICIN		
50 mg/m ²	101 mg	100 mg (in a syringe) push IV
VINCRIPTINE		
1.4 mg/m ²	2 mg	2 mg (MAXIMUM 2 mg) in 20 mls Normal Saline (in a syringe) push IV
PREDNISONE		
100 mg	100 mg	100 mg PO od for 5 days

CYCLOPHOSPHAMIDE

750 mg/m² 1510 mg 1500 mg IV over 1 hour

DOXORUBICIN

50 mg/m² 101 mg 100 mg (in a syringe) push IV

VINCRIPTINE

1.4 mg/m² 2 mg 2 mg (MAXIMUM 2 mg) in 20 mls Normal Saline (in a syringe) push IV

PREDNISONE

100 mg 100 mg 100 mg PO od for 5 days

This prescription represents 100% of the full dose as per the protocol

Drug Name	Dose (mg)	Route	Frequency
Stemetil (Prochlorperazine) ▾	10 ▾	po ▾	pre-chemo
Anzemet (Dolasetron) ▾	100 ▾	po ▾	pre-chemo
Decadron (Dexamethasone) ▾	8 ▾	po ▾	pre-chemo
Ativan (Lorazepam) ▾	0.5 ▾	po ▾	pre-chemo

Preview

Figure 4.7: Screenshot: Sample Prescription Input Page



Medicine Program

Physician's Chemotherapeutic Medications

General Hospital Site

Height = 180 cms	Weight = 80 kgs
Calculated BSA = 2.01 m ²	BSA used = 2.01 m ²
Total Bil _i = 4.0	ALP = 5.0
	CR = 2.5

Diagnosis - nil Drug Allergies - nil

Date: 08/12/2007 04:04:26

Round off total dosages where appropriate

CHOP

Percent of Protocol Dose Intended by Physician: 100 %

Protocol Dose	Calculated Dose	Prescribed Dose
CYCLOPHOSPHAMIDE		
750 mg/m ²	1510 mg	1500 mg IV over 1 hour
DOXORUBICIN		
50 mg/m ²	101 mg	100 mg (in a syringe) push IV
VINCRIStINE		
1.4 mg/m ²	2 mg	2 mg (MAXIMUM 2 mg) in 20 mls Normal Saline (in a syringe) push IV
PREDNISONE		
100 mg	100 mg	100 mg PO od for 5 days

This prescription represents 100 % of the full dose as per the protocol

Drug Name	Dose (mg)	Route	Frequency
Stemetil(Prochlorperazine)	10	po	pre-chemo
Anzemet(Dolasetron)	100	po	pre-chemo
Decadron (Dexamethasone)	8	po	pre-chemo
Ativan(Lorazepam)	0.5	po	pre-chemo

Figure 4.8: Screenshot: Sample Prescription Output Page

the secondary list. For details of the division of regimens into primary and secondary lists see Section 4.4.2.4.

2. **Automatic entry of data from previous prescriptions:** To make the prescription writing process faster, if the regimen is already prescribed to the patient then data from the last prescription is automatically entered to the new prescription by the system. However, the physician can make changes in the height and weight of the patient if they have changed since the last visit and the system will calculate dosages based on the patient's new body surface area. Moreover, the physician can also make changes to the dosages of different drugs.
3. **Automatic error checking:** Major calculations involved in the process of prescribing are done by the system to eliminate the possibilities of manual calculation errors. Along with automatically calculating dosages, the system also looks for any possibility of error while entering the height or the weight of patients; if height or weight are beyond the normal range it will inform the ordering physician about the possibility of errors. Here, a normal range for height is 0–225 cm, and for weight is 0–225 pounds. The physician is notified for any possibility of error by the field color turning to red. Note that this is consistent with the error handling philosophy adopted in this system, in which potential errors are flagged but allowed if the physician considers it necessary.
4. **Future prescriptions:** A physician can prescribe a regimen with some future date on it. The system allows prescribing a regimen in the future because some regimens have multiple courses of treatment. However, to eliminate the chances of multiple future prescriptions, the system does not allow more than one future

prescription per patient. If a future prescription is already in database for a particular patient then the system will allow physicians to prescribe only if the previous prescription is cancelled for that patient. The system will notify the physician of the existing future prescription and, if the physician elects to cancel the prescription, the prescription is cancelled automatically by the system.

4.4.2 Short-Term Activities

Short term activities include activities that range from those that run for one day *e.g.*, adding a new patient to the system, to those which run for a year or more, during which time new regimens are created or physicians feel it necessary to make changes in existing regimens. Short-term activities include adding new patients (Section 4.4.2.1), creating new regimens (Section 4.4.2.2), and altering existing regimens (Section 4.4.2.3).

4.4.2.1 Add Patient

A user enters patient's personal information to add him into the system. If a patient is already in the database the user is brought back to the add patient page with an error message. A workflow diagram for this function is shown in Figure 4.9, and a screenshot of the implemented patient registration page is shown in Figure 4.10.

The following decisions were made during the design of different features involved in the *Add Patient* function:

1. **Selection of patient's unique identifier:** To uniquely identify a patient in a patient database there is a need for a unique key which will help to search

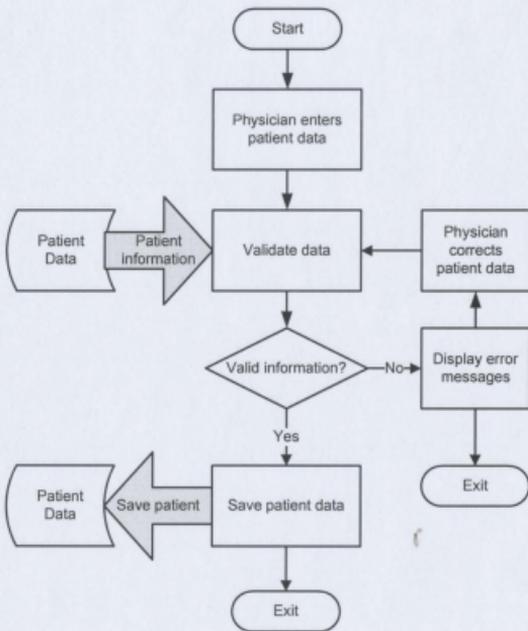


Figure 4.9: Workflow Diagram: Add Patient to Database

Patient Registration

Quick Links

- ▶ Patient Search
- ▶ Prescribe Regimen
- ▶ Add Patient
- ▶ Create Regimen
- ▶ Alter Regimen
- ▶ Import Regimen (0 New)
- ▶ View Deleted Regimens
- ▶ View Regimens
- ▶ Admin
- ▶ Logout

MCP Number :

First Name :

Last Name :

Gender :

Date of Birth :

Street Address :

City :

Province :

Postal Code : (Format: AAAAA)

Figure 4.10: Screenshot: Add Patient to Database

patients in that database. For this purpose, the patient's MCP number is used (see Footnote 4).

- 2. Use of demographic information to identify patient:** Different medical systems use different types of keys to identify a patient in their patient databases. Patient MCP numbers and demographic information are currently implemented as keys in our system. The question naturally arises as to how much demographic information we are going to store. Storing more information might be better for futures uses, such as linking with other datasets or communicating with other systems. However, in the interests of patient privacy and saving memory space, we have elected to store what we consider the minimum information necessary for identification by demographic information.

4.4.2.2 Create New Regimen

In order to create a new regimen, the user suggests a unique name for the regimen, the number of chemotherapy drugs, and defines the scope of regimen, *i.e.*, public or private (see below). The user then enters the details of chemotherapy drugs and the system validates chemotherapy drugs against the drug database. If the chemotherapy drugs are not in the database the system prompts the user to add the new drugs and their emetogenicity levels (see Section 4.4.3.1 for details). If all entered drugs are in the database, or have been added to the database, then the system suggests the possible list of antiemetic drugs based on the emetogenic levels of all chemotherapy drugs (see below). Here users can make changes to the list of antiemetic drugs by adding or deleting drugs/doses from a group or changing the order of drugs/doses.

The user then requests the final regimen and the system develops a regimen based on the information entered by the user and allows the user to save the regimen to the database according to the scope mentioned by the user. A workflow diagram for this function is shown in Figure 4.11. Screenshots of the implemented initial pages to initialize the regimen, collect the details of chemotherapy drugs and to select antiemetic drugs are shown in Figures 4.12, 4.13, and 4.14 respectively.

The following decisions were made during the design of different features involved in the *Create New Regimen* function:

1. **Public and private regimens:** While creating a new regimen the physician is asked to set the scope of that regimen. The physician can make the new regimen available just to himself, *i.e.*, a private regimen, or he can make it public so that every physician in the system will be notified about the new regimen and allowed to import it to their private list. For details of the *Import Regimen* function, see Section 4.5.2.1.
2. **Automatic selection of antiemetic:** A decision support feature is incorporated into the system which helps physicians during the selection of antiemetic drugs. To make the system automatically select antiemetic drugs based on the emetogenicity potential of chemotherapy drugs, an algorithm for predicting the acute emetogenicity of chemotherapy regimens was used. Recall that emetogenicity potential of chemotherapy drugs is between 0–5, with 0 having no effect and 5 having maximum effect. This algorithm, developed by the VHA Pharmacy Benefits Management Strategic Healthcare Group and Medical Advisory Panel [79], is as follows:

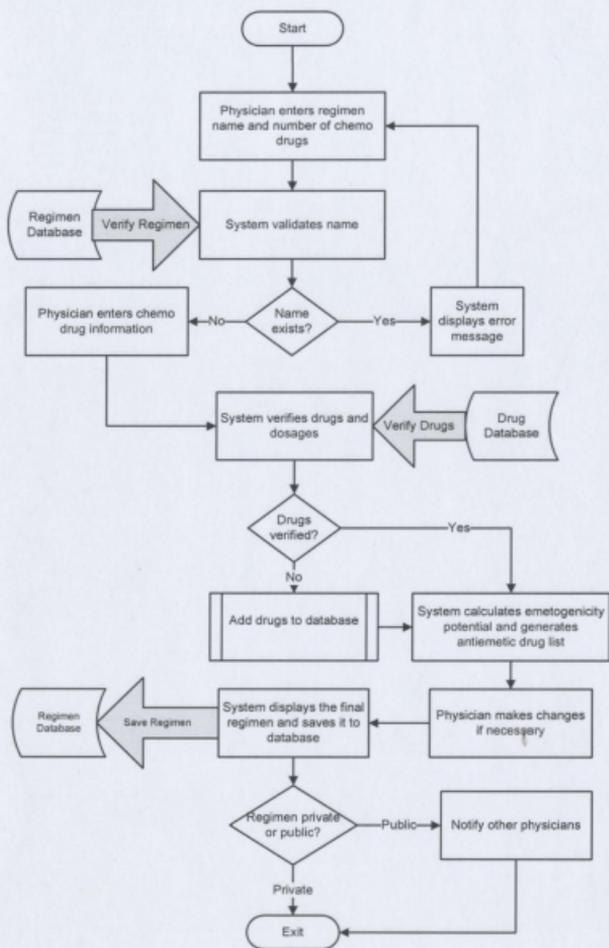


Figure 4.11: Workflow Diagram: Create New Regimen

Create Regimen

Quick Links

- Patient Search
- Prescribe Regimen
- Add Patient
- Create Regimen
- Alter Regimen
- Import Regimen (0 New)
- View Deleted Regimens
- View Regimens
- Admin
- Logout

Please enter following information

Name of Regimen:
Number of chemoDrugs:
Regimen scope Private Public

(You may enter up to 80 characters.)

Comments:
 characters left

Figure 4.12: Screenshot: Create Regimen - Regimen Name

Regimen detail

Quick Links	Chemo Drugs Information					
	Name	Dose	Max Dose*	Route	Frequency	Instructions
<ul style="list-style-type: none"> > Patient Search > Prescribe Regimen > Add Patient > Create Regimen > Alter Regimen > Import Regimen (0 New) > View Deleted Regimens > View Regimens > Admin > Logout 	PHOSPHAMIDE	750 [mg/m ² ▼]	[]	IV ▼	over 1 hour ▼	[]
	DOXORUBICIN	50 [mg/m ² ▼]	[]	IV ▼	[] ▼	(in a syringe) push ▼
	VINCRIStINE	1.4 [mg/m ² ▼]	2	IV ▼	[] ▼	20 ml ^s Normal Saline ▼ (in a syringe) push ▼
	PREDNISONE	100 [mg ▼]	[]	PO ▼	qd for 5 days ▼	[] ▼

AntiEmetics: Use default Change default

* Leave maximum dose field blank for no maximum limit

Figure 4.13: Screenshot: Create Regimen - Chemo Drugs Details

Antiemetic Drugs

Quick Links

- ▶ Patient Search
- ▶ Prescribe Regimen
- ▶ Add Patient
- ▶ Create Regimen
- ▶ Alter Regimen
- ▶ Import Regimen (0 New)
- ▶ View Deleted Regimens
- ▶ View Regimens
- ▶ Admin
- ▶ Logout

Antiemetic Drugs Information

Create New Group

Group Name:

CRC Blockers

- | | | |
|--|--|--|
| <input type="radio"/> none | <input type="radio"/> <input type="text"/> | <input type="radio"/> <input type="text"/> |
| <input checked="" type="radio"/> Stemetil (Prochlorperazine) | <input checked="" type="radio"/> 10 | <input checked="" type="radio"/> po/iv |
| <input type="radio"/> Maxeran (Metoclopramide) | <input type="radio"/> 20 | <input type="radio"/> po |
| <input type="radio"/> Gravol (Dimenhydrinate) | <input type="radio"/> 25 | <input type="radio"/> iv |

[Add drug to CRC Blockers](#)

[Delete drug from CRC Blockers](#)

5-HT3 Receptor Antagonist

- | | | |
|---|--|--|
| <input type="radio"/> none | <input type="radio"/> <input type="text"/> | <input type="radio"/> <input type="text"/> |
| <input checked="" type="radio"/> Zofran (Ondansetron) | <input type="radio"/> 2 | <input type="radio"/> po |
| <input type="radio"/> Anzemet (Dolasetron) | <input checked="" type="radio"/> 8 | <input type="radio"/> iv |
| <input type="radio"/> Kytril (Granisetron) | <input checked="" type="radio"/> 100 | <input type="radio"/> po/iv |

[Add drug to 5-HT3 Receptor Antagonist](#)

[Delete drug from 5-HT3 Receptor Antagonist](#)

Steroids

- | | | |
|---|--|--|
| <input type="radio"/> none | <input type="radio"/> <input type="text"/> | <input type="radio"/> <input type="text"/> |
| <input checked="" type="radio"/> Decadron (Dexamethasone) | <input type="radio"/> 8 | <input type="radio"/> po |
| | <input type="radio"/> 10 | <input type="radio"/> iv |
| | <input checked="" type="radio"/> 20 | <input checked="" type="radio"/> po/iv |

[Add drug to Steroids](#)

[Delete drug from Steroids](#)

Anxiolytics

- | | | |
|--|---|--|
| <input checked="" type="radio"/> none | <input checked="" type="radio"/> <input type="text"/> | <input type="radio"/> <input type="text"/> |
| <input type="radio"/> Ativan (Lorazepam) | <input type="radio"/> 0.5 | <input type="radio"/> po |
| | <input type="radio"/> 1.0 | <input type="radio"/> iv |
| | <input type="radio"/> 2.0 | <input type="radio"/> po/iv |

[Add drug to Anxiolytics](#)

[Delete drug from Anxiolytics](#)

Figure 4.14: Screenshot: Create Regimen - Antiemetic Drugs Details

- (a) Identify the most emetogenic chemotherapy drug in the regimen.
- (b) Assess the relative contribution of other chemotherapy drugs to the emetogenicity of the regimen. When considering other drugs the following rules apply:
 - Level 1 drugs do not contribute to the emetogenicity of a given regimen
 - Adding one or more level-2 drugs increases the emetogenicity of the regimen one level above the most emetogenic agent in the regimen
 - Adding level-3 or level-4 drugs increases the emetogenicity of the regimen by one level per such drug.

Note that regardless of the final computed value, if it is greater than 5 then it is set to 5. In this system, physicians can make changes in the antiemetic drugs and their dosages as suggested by the algorithm.

4.4.2.3 Alter Existing Regimen

In order to alter an existing regimen the physician selects an existing regimen from his private list. Because the new regimen could be confused with the old one, the physician is required to rename that regimen and select its new scope, *i.e.*, public or private (see Section 4.4.2.2). The system displays the list of all chemo-drugs in the existing regimen. The physician makes changes in the chemotherapy drugs, *i.e.*, adding or deleting chemotherapy drugs to the regimen, and can change the doses, routes, frequencies or instructions associated with drugs (see Section 3.2.1). The system then recommends antiemetic drugs based on the new chemo drugs by using the algorithm described in Section 4.4.2.2. After this, the physician makes changes or

approves the suggested antiemetic drugs and the system displays the final regimen based on changes made by the physician who then saves the regimen. The workflow diagram for this function is shown in Figure 4.15. Screenshots of the implemented initial pages to rename the existing regimen, and to collect details of new chemotherapy drugs and/or to delete existing drugs are shown in Figures 4.16 and 4.17 respectively. The page to select antiemetic drugs in case of altering an existing regimen is the same as shown in Figure 4.14 for creating a new regimen.

The following decisions were made during the design of different features involved in the *Alter Existing Regimen* function:

1. **Renaming modified regimen:** To modify an existing regimen the system creates a new copy of that regimen in the database and the existing regimen remains unchanged. Changes are not permitted in an existing regimen for the following reasons:

- An existing regimen might be in use by other physicians who are not willing to make changes in that regimen.
- An existing regimen might already have been prescribed by the physician. In order to display an old prescription, the system maps prescription drug data to the regimen template stored in the database. Thus, altering that regimen template can cause problems when the system attempts to display old prescriptions.

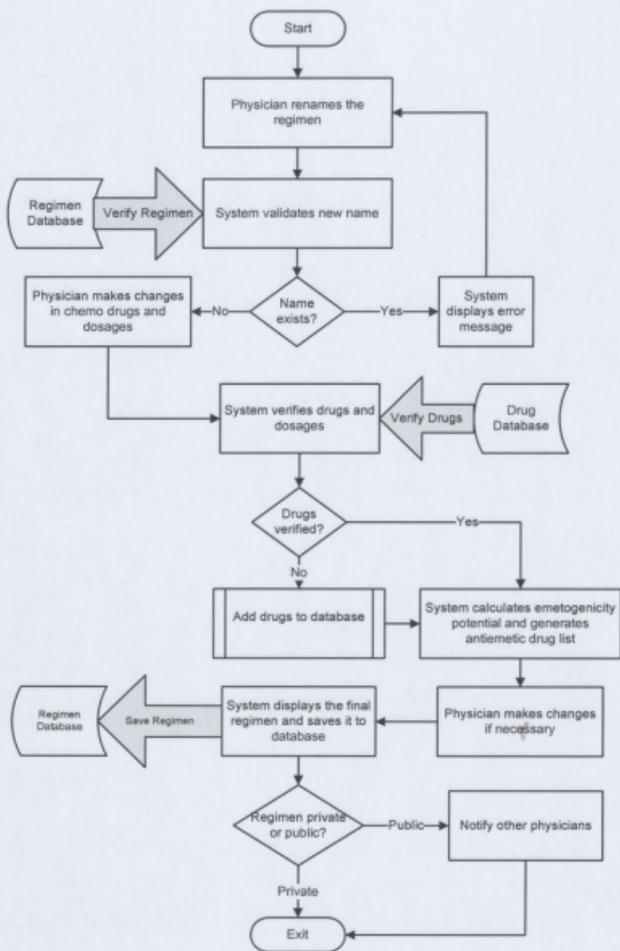


Figure 4.15: Workflow Diagram: Alter Regimen

Quick Links

- › Patient Search
- › Prescribe Regimen
- › Add Patient
- › Create Regimen
- › **Alter Regimen**
- › Import Regimen (0 New)
- › View Deleted Regimens
- › View Regimens
- › Admin
- › Logout

Please enter modified name of CHOP

Name of Regimen:

Number of new
chemoDrugs:

Regimen scope Private Public

(You may enter up to 50 characters.)

Comments:

36 characters left

Figure 4.16: Screenshot: Alter Regimen - Regimen Name

HealthCare
Corporation of St. John's

Regimen detail

Quick Links

- > Patient Search
- > Prescribe Regimen
- > Add Patient
- > Create Regimen
- > Alter Regimen
- > Import Regimen (to New)
- > View Deleted Regimens
- > View Regimens
- > Admin
- > Logout

Chemo Drugs Information						
Name	Dose	Max Dose*	Route	Frequency	Comments	
<input type="checkbox"/> CYCLOPHOSP	750 [mg/m ² ▼]	<input type="text"/>	[IV ▼]	over 1 hour [▼]	<input type="text"/>	
<input type="checkbox"/> DOXORUBICIN	50 [mg/m ² ▼]	<input type="text"/>	[IV ▼]	<input type="text"/>	(in a syringe) push [▼]	
<input type="checkbox"/> VINCRIStINE	1.4 [mg/m ² ▼]	<input type="text"/>	[IV ▼]	<input type="text"/>	(MAXIMUM 2 mg) in 20 ml Normal Saline [▼]	
<input type="checkbox"/> PREDNISONE	100 [mg/m ² ▼]	<input type="text"/>	[PO ▼]	od for 5 days [▼]	<input type="text"/>	
<input type="checkbox"/>	<input type="text"/> [mg/m ² ▼]	<input type="text"/>	[PO ▼]	<input type="text"/>	<input type="text"/>	

AntiEmetics: Use default * Change default

* Leave maximum dose field blank for no maximum limit

Figure 4.17: Screenshot: Alter Regimen - Chemo Drugs Details

4.4.2.4 Manage Regimens

Physicians can manage their private regimens by moving them to primary or secondary lists (see below). To move a regimen from one list to another the physician selects the regimen to be moved and the system adds that regimen to the second list and deletes it from the first. A workflow diagram for this function is shown in Figure 4.18, and a screenshot of the implemented manage regimen page is shown in Figure 4.19.

The following decisions were made during the design of different features involved in the *Manage Regimens* function:

1. **Purpose of regimen management:** In order to make it easier for physicians to find the most frequently used regimens out of their all private regimen list, the list is divided into primary and secondary regimens lists. The purpose of the primary list is to hold all regimens that are used frequently in prescribing and display them first for regimen selection during prescribing. The physician can move a regimen from one list to another at any time.

4.4.3 Mid-Term Activities

Mid-term activities include activities that range from one to five years, where new drugs are introduced and are used as replacements for existing drugs in regimens. Mid-term activities involve the addition of new chemo-drugs to the database (Section 4.4.3.1) and the retrieval of lost passwords (Section 4.4.3.2).

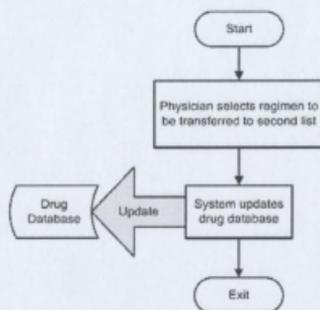


Figure 4.18: Workflow Diagram: Manage Regimens

		Regimen List	
Quick Links	Regimen Name	Comments	
<ul style="list-style-type: none"> > Patient Search > Prescribe Regimen > Add Patient > Create Regimen > Alter Regimen > Import Regimen (0 New) > View Deleted Regimens > View Regimens > Admin > Logout 	<input checked="" type="checkbox"/> CHOP	Test Regimen	
	<input type="checkbox"/> Modified Chop	Modified version of Chop	
	<input type="checkbox"/> DHAP	Test Regimen	
	<input type="checkbox"/> ALL Induction	Test Regimen	
	<input type="checkbox"/> NOVE	Test Regimen	
	<input type="checkbox"/> STEM_CELL_COLLECTION	Test Regimen	
		Move selected to secondary list	View Secondary List

Figure 4.19: Screenshot: Manage Regimens

4.4.3.1 Add New Chemotherapy drug

While creating a new regimen (see Section 4.4.2.2) or altering an existing regimen (see Section 4.4.2.3), if the physician enters a drug name which is not in the drug database he/she is prompted to enter the new drug into the database along with the emetogenic potential of that drug. The physician enters the drug name and the system then generates a form to enter the emetogenicity potential. Next, the physician enters ranges of dosages for all emetogenic levels of the drug (see Section 3.2.1), which are then validated by the system and stored in the database. The workflow diagram for this function is shown in Figure 4.20, and a Screenshot of a the implemented page to add new chemotherapy drugs to the system is shown in Figure 4.21.

The following decisions were made during the design of different features involved in *Add New Chemotherapy drug* function:

1. **Automatic error checking for possible mistakes by the user:** The range of possible dosages of a drug is broken down in such a way that there are sub-ranges, each with their own emetogenicity level. The union of these sub-ranges must cover the entire range, and the emetogenicity level must increase as the dosage goes up. When adding a new drug, the system automatically checks for the any possible errors in dosage ranges and emetogenic levels. There are four possible types of errors: The emetogenicity level can be out of range or non-decreasing and ranges can overlap or have gaps between them. For this purpose, the physician has to enter the correct details of one emetogenic level for a particular drug, which is then examined by the system before the physician is allowed to proceed to the next level.

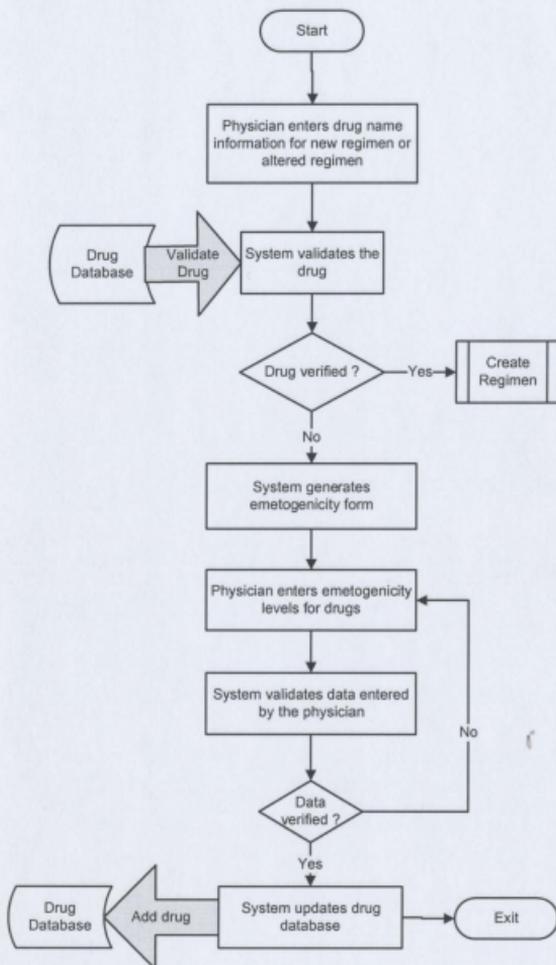


Figure 4.20: Workflow Diagram: Add New Chemotherapy Drug

Add Drug

Quick Links

- › Patient Search
- › Prescribe Regimen
- › Add Patient
- › Create Regimen
- › Alter Regimen
- › Import Regimen (0 New)
- › View Deleted Regimens
- › View Regimens
- › Admin
- › Logout

Following drug was not found in chemotherapy emetogenic potential database. Please enter the emetogenic potential for this drug on a scale of 0-5, where 0 means no emetogenic potential & 5 means severe emetogenic potential

Enter Emetogenic levels in **decending order**.
*Leave Maximum dose field empty for no upper limit.

Agent	Minimum Dose	Maximum Dose*	Level
PRECNISONE	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Submit

Figure 4.21: Screenshot: Add New Chemotherapy Drug

2. Direct addition of drugs to chemotherapy drug database without passing through administration: Physicians are allowed to enter new drugs directly into the drug database without any approval from the system administrator. The addition of another step, *i.e.*, approval from the administrator, could cause unnecessary delays in the creation of a new regimen or alteration of an existing regimen, which is not tolerable due to the time-sensitive nature of the work physicians perform.

3. Automatic setting of missing limits for Emetogenicity levels of chemo drugs: Any drug with no specified upper limit for the highest emetogenic level entered has no upper limit of dosages for that particular level. This means that, any dose entered above the lower limit of that level is assigned the emetogenicity potential of that level. Moreover, the range starting from 0 up to the minimum range specified for the lowest level entered by the physician is automatically assigned an emetogenic potential of 1. Note that level 1 does not affect the selection of antiemetic drugs for a regimen (see Section 4.4.2.2 for details).

4.4.3.2 Password Retrieval

In order to retrieve his/her password, a user needs to enter his username or email address. The system validates the username/email of the user and mails the login information to the user email address stored in the database. In the case of an invalid username/email address entered by the user, that user is taken back to the password

retrieval page with an error message. The workflow diagram for this function is shown in Figure 4.22, and a screenshot of the implemented password retrieval page is shown in Figure 4.23.

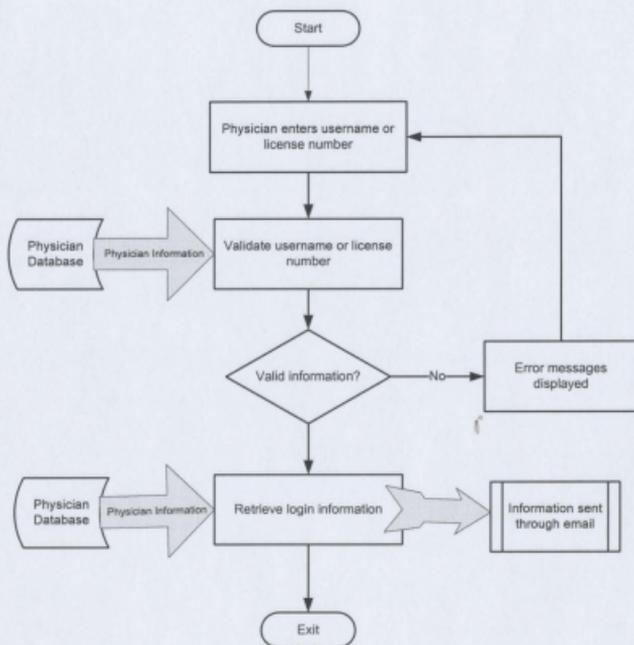


Figure 4.22: Workflow Diagram: Password Retrieval

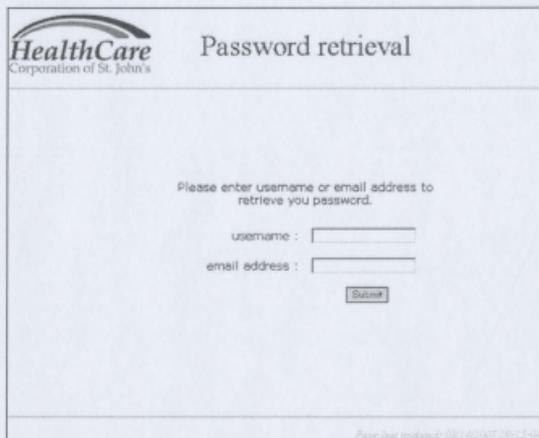


Figure 4.23: Screenshot: Password Retrieval

The following decisions were made during the design of different features involved in the *Password Retrieval* function:

1. **Information required to retrieve username/password:** To make it simpler for users to retrieve their passwords, they can either use their usernames or their email addresses to retrieve them; where the system sends a password along with the username to the email address of the user in question. This method is also helpful in cases where users have forgotten their usernames allowing them to retrieve them by entering their email address into the system.
2. **Use of email address to retrieve username/password:** We have looked into different commonly used ways to provide a user his password [46]. The basis for selection of a method is that it should be as simple as possible for the

user, and it should be secure enough that no one can misuse it. One possible way to give a password to a user is to send it to the email address which he/she provides during the registration process. Another is by answering some secret question which only the user can answer. This secret question is saved to the database, along with other information, during the registration process. While making decisions about the method for password retrieval to be employed, it was considered that it would be an extra burden on the user to remember the answer to the secret question. Hence, the email address method is used here.

4.5 Multi User Design

In the previous section, we noted that single-user activities were divided into session, short-term, and mid-term activities. Multi-user activities can have duration from single-session to mid-term. However, an important aspect of multi-user activities is the type of interaction among users; hence, in the case of a multi-user design, we have chosen to divide activities into three categories based on the interaction between the users, namely indirect (Section 4.5.1), limited direct (Section 4.5.2), and direct interaction (Section 4.5.3). These categories, along with their corresponding tasks, are shown in Table 4.2. Note that we are using the same task, subtask, and class conventions for Table 4.2 as were described in introduction Section 4.4 for Table 4.1.

4.5.1 Indirect Interaction

Indirect interaction between the users of a chemotherapy prescription system happens when a patient is treated by more than one physician. Here, physicians can not

Table 4.2: Multi User Design Table

User Interaction Type	Task (Section #)	Sub-Task (Diagram #)	Associated Classes
Indirect	View patient history (4.5.1.1)	View patient history (4.24)	Standard + Patient + Prescription
Limited direct	Import Regimen (4.5.2.1)	Import Regimen (4.26)	Standard + Regimen
Direct	Add Proposal (4.5.3.1)	Add Proposal (4.28)	Standard + Administrator
	Vote Proposal (4.5.3.2)	Vote Proposal (4.30)	Standard + Administrator
	View Proposals (4.5.3.3)	View Proposals (N/A)	Standard + Administrator

only view the history of a patient and the way that patient is treated by the other physicians, but also make changes in the prescriptions assigned by other physicians by deleting the existing prescription. The only task that falls under indirect interaction is viewing patient history (Section 4.5.1.1).

4.5.1.1 View Patient History

After a successful search for a patient (see Section 4.4.1.2) the system displays the patient's history page. Here the physician can cancel, modify, or add comments to an old prescription, or can prescribe a regimen to the patient (see Section 4.4.1.3). Note that viewing patient history partially falls under the multi-user environment, where a physician can view the prescriptions made by other physicians and can cancel them. A workflow diagram for this function is shown in Figure 4.24, and a screenshot of the implemented patient history page is shown in Figure 4.25.

The following decisions were made during the design of different features involved in the *View Patient History* function:

1. **Alteration of past prescription:** Physicians can change any prescription. Such changes can only be made by the physician who made that prescription.
2. **Conditions under which a Physician can cancel a prescription:** Any physician can cancel any prescription prescribed by himself or any other physician within one day of the prescription being issued. Moreover, any future prescription can be cancelled at any time by any physician. Note that it is only in the second scenario where a future prescription can be cancelled. Under *Prescribe Regimen* (see Section 4.4.1.3) we see that if a physician prescribes a

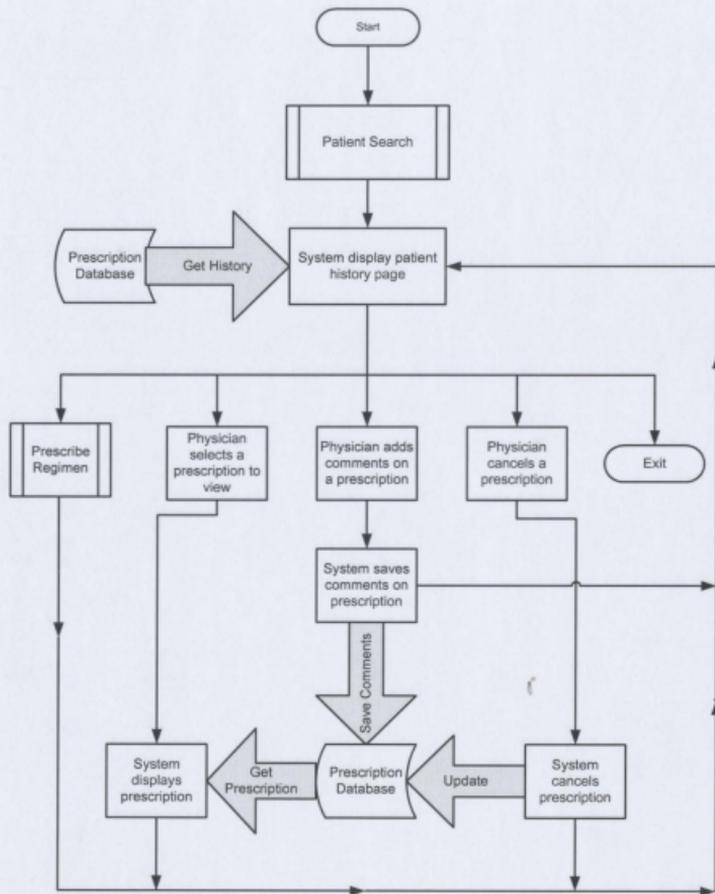


Figure 4.24: Workflow Diagram: View Patient History

Quick Links <ul style="list-style-type: none"> > Patient Search > Prescribe Regimen > Add Patient > Create Regimen > Alter Regimen > Import Regimen (0 New) > View Deleted Regimens > View Regimens > Admin > Logout 	NAME: JOHN HOPKINS MCP NUMBER: 234543234 AGE: 28 GENDER: MALE ADDRESS: 57 ALLANDALE ROAD, ST. JOHN'S, NL POSTAL CODE: A1B3C7																								
	Prescription History																								
	<table border="1"> <thead> <tr> <th>Name</th> <th>Date</th> <th>Status</th> <th>Comments</th> </tr> </thead> <tbody> <tr> <td>● CHOP</td> <td>08/12/2007 04:13:36</td> <td>CANCEL</td> <td><input type="text"/> add comments</td> </tr> <tr> <td>● CHOP</td> <td>08/12/2007 04:12:33</td> <td>CANCELLED BY SYED 08/12/2007 04:14:30</td> <td><input type="text"/> add comments</td> </tr> <tr> <td>● CHOP</td> <td>08/12/2007 04:12:15</td> <td>CANCEL</td> <td><input type="text"/> add comments</td> </tr> <tr> <td>● CHOP</td> <td>08/12/2007 04:11:05</td> <td>CANCEL</td> <td><input type="text"/> add comments</td> </tr> <tr> <td>● CHOP</td> <td>08/12/2007 04:04:26</td> <td>CANCELLED BY SYED 08/12/2007 04:11:18</td> <td><input type="text"/> (1) add comments(1)</td> </tr> </tbody> </table>	Name	Date	Status	Comments	● CHOP	08/12/2007 04:13:36	CANCEL	<input type="text"/> add comments	● CHOP	08/12/2007 04:12:33	CANCELLED BY SYED 08/12/2007 04:14:30	<input type="text"/> add comments	● CHOP	08/12/2007 04:12:15	CANCEL	<input type="text"/> add comments	● CHOP	08/12/2007 04:11:05	CANCEL	<input type="text"/> add comments	● CHOP	08/12/2007 04:04:26	CANCELLED BY SYED 08/12/2007 04:11:18	<input type="text"/> (1) add comments(1)
Name	Date	Status	Comments																						
● CHOP	08/12/2007 04:13:36	CANCEL	<input type="text"/> add comments																						
● CHOP	08/12/2007 04:12:33	CANCELLED BY SYED 08/12/2007 04:14:30	<input type="text"/> add comments																						
● CHOP	08/12/2007 04:12:15	CANCEL	<input type="text"/> add comments																						
● CHOP	08/12/2007 04:11:05	CANCEL	<input type="text"/> add comments																						
● CHOP	08/12/2007 04:04:26	CANCELLED BY SYED 08/12/2007 04:11:18	<input type="text"/> (1) add comments(1)																						

Page last updated: 08/12/2007 04:14:25

Figure 4.25: Screenshot: View Patient History

regimen when an outstanding prescription is already in the system, the future prescription is automatically cancelled by the system.

3. **Addition of comments with each prescription entry:** In order to enhance communication between the users of the chemotherapy prescription system, any physician can add comments on any prescription. This feature can help the physician in informing other physicians about the reasons for his/her actions, *e.g.*, his/her reasons for canceling a prescription.

4.5.2 Limited Direct Interaction

Limited direct interaction between the users of the chemotherapy prescription system occurs when a physician shows his interest in using a regimen created by the other physician. The only task that falls under limited direction interaction is the *Import Regimen* function (Section 4.5.2.1).

4.5.2.1 Import Regimen

In order to import a public regimen, the system displays the list of public regimens available to the user. The user then selects the regimens he wants to import and the system adds them to the user's private list. A workflow diagram for this function is shown in Figure 4.26, and a screenshot of the implemented page to import regimen is shown in Figure 4.27.

The following decisions were made during the design of different features involved in *Import Regimen* function:

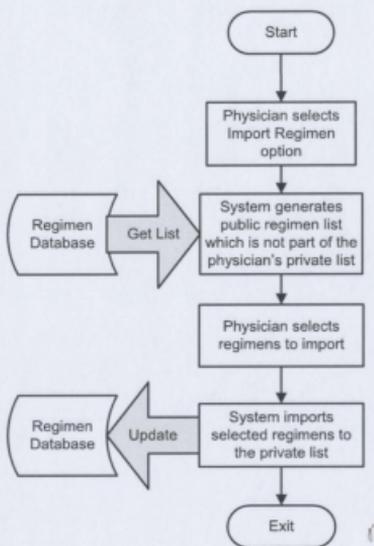


Figure 4.26: Workflow Diagram: Import Regimen

Regimen List

Quick Links

- > Patient Search
- > Prescribe Regimen
- > Add Patient
- > Create Regimen
- > Alter Regimen
- > Import Regimen (2 New)
- > View Deleted Regimens
- > View Regimens
- > Admin
- > Logout

New Regimens	
Regimen Name	Comments
<input type="checkbox"/> Modified Chop	Modified version of Chop
<input type="checkbox"/> NOVE	

Previously viewed regimens	
Regimen Name	Comments
<input type="checkbox"/> DHAP	
<input type="checkbox"/> ALL Induction	

Figure 4.27: Screenshot: Import Regimen

1. **Importing a regimen from a public list to a private list:** In order to increase collaboration and sharing among users, physicians can make regimens public while creating a new regimen (see Section 4.4.2.2). The public list is always available to all physicians and displays all public regimens that are not already part of a physician's private lists. Each physician can, at any time, import a regimen from his/her private list to the public list.
2. **Alert physician of new regimen:** To notify physicians of a new regimen in a public list, each time a new regimen is added to the database, all physicians in the database are notified the next time they log in to the system by a display of the number of new regimens available to import in front of the link to the import regimen page in the subtask sidebar(see Figure 4.27).
3. **Secondary list for viewed regimens:** In order to differentiate between the viewed and unviewed public regimens, every public regimen that is not imported by the physician, after looking at his primary list of public regimen, is sent to a secondary public regimen list and can be imported at any later time.

4.5.3 Direct Interaction

Direct interaction is where the majority of system administration is done. System administration under the democratic model is done by issuing and voting on proposals (see Section 4.2.1). The types of proposals currently implemented are described in Section 4.5.3.1. This process involves several functions. To make the voting mechanism simpler, we have introduced a bulletin board structure where physicians can either make a new proposal (See Section 4.5.3.1), vote on a proposal (See Section

4.5.3.2), or view the status of an existing proposal (See Section 4.5.3.3).

The democratic model of system operation has the following features:

1. **Bulletin board:** In order to accommodate all types of system administration features, a bulletin board structure was designed. This bulletin board can display all pending proposals which are not resolved (see Section 4.5.3.2 for more details on proposal resolution). Note that all proposals that are resolved are moved to a separate list. Here physicians can either add a new proposal to the bulletin board or can vote on an existing proposal.
2. **Ability to issue proposals:** Every physician who is registered in the system has the ability to make a proposal. Moreover, there is no maximum limit on the number of proposals that can be made by a particular physician.
3. **Resolution mechanism for proposals:** All proposals are resolved in a democratic fashion, where physicians vote to either accept or reject the change requested. However, the percentage of votes necessary for the acceptance of a proposal depends on the criticality of the change requested in that proposal (see Section 4.5.3.1). Once a proposal is accepted or rejected it is sent to the resolved proposals list, and any actions requested by an accepted proposal are automatically taken by system.

4.5.3.1 Add Proposal

In order to make a proposal, the user selects the new proposal option. The user is asked to select a category of proposal and the system displays different proposal types under the selected category. The user then selects the proposal type and enters

any required data for that proposal. The system validates the data and saves the proposal to the proposal database. A workflow diagram for this function is shown in Figure 4.28 and a screenshot of the implemented page to add a proposal to change emetogenicity potential is shown in Figure 4.29.

The following decisions were made during the design of different features involved in the *Add Proposal* function:

1. **Types of Proposals:** There are many types of tasks that can be performed using the democratic model of system operation. However, in the system developed in this thesis, the available tasks are as follows:
 - (a) *Add user:* This proposal is used to add a new user to the system. Any existing user may make such a proposal, in which that existing user selects a unique username and enters his/her personal information required to create an account⁵. The criteria for adding a new user is acceptance by two existing users. Upon successful acceptance of the proposal the new user is automatically added to the system.
 - (b) *Delete user:* This proposal is used to delete an existing user from the system. Any existing user may make such a proposal, in which that user enters the username of the user to be deleted, as well as optional reasons for deletion. The criterion for deleting a user is acceptance by 50 percent of

⁵This is actually an over simplification. The current system does not allow usernames or passwords to be changed once entered. Hence, we are assuming that though the existing user triggers the function, the new user is actually present and enters his username and password in such a way that the existing user does not see it, allowing it to remain private. We realize that this is awkward and it should be fixed in the future versions of the system

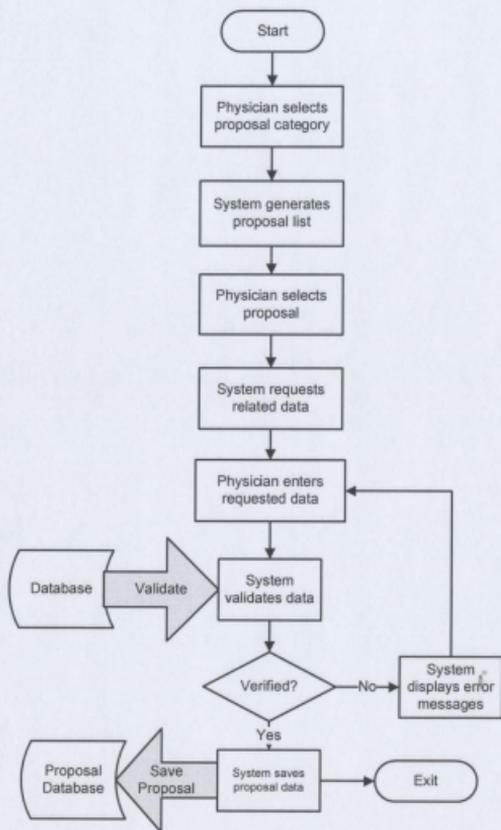


Figure 4.28: Workflow Diagram: Add Proposal

- Quick Links**
- > Patient Search
 - > Prescribe Regimen
 - > Add Patient
 - > Create Regimen
 - > Alter Regimen
 - > Import Regimen (0 New)
 - > View Deleted Regimens
 - > View Regimens
 - > Admin
 - > Logout

Existing Emetogenic Levels			
Agent	Minimum Dose	Maximum Dose	Level
Cyclophosphamide	1501	No Max. Limit	5
Cyclophosphamide	751	1500	4
Cyclophosphamide	0	750	3

New Emetogenic Levels

Enter modified Emetogenic levels in **decending order**.
*Leave Maximum dose field empty for no upper limit.

Agent	Minimum Dose	Maximum Dose*	Level
Cyclophospham	1501		5
Cyclophospham	751	1500	4
Cyclophospham	500	750	3
Cyclophospham	0	499	2

Comments:

60 character left

Submit

Figure 4.29: Screenshot: Add Proposal - Change Emetogenicity Potential

the existing users. Upon successful acceptance of the proposal the selected user is automatically deleted from the system.

- (c) *Change chemotherapy drug name:* This proposal is used to change the name of chemotherapy drugs in the drug database. This feature is used to correct any misspelled drug name in the drug database. Any existing user can make such a proposal, in which the user enters both the existing and modified name of the drug. The criterion to change the name of a chemotherapy drug is acceptance by 60 percent of the existing users. Upon successful acceptance of the proposal, the drug name is automatically changed in the drug database. Note that this change of chemotherapy drug name will only affect future regimens; for existing regimens, the name of the drug will remain the same.
- (d) *Change emetogenicity potential of chemotherapy drug:* This proposal is used to change the emetogenicity levels of chemotherapy drugs. This feature is used to correct the range of any level or to add a new level to a drug. Any existing user can make such a proposal and enters the modified details of the all emetogenicity levels of the chemotherapy drug. The criterion to change emetogenicity potential of a chemotherapy drug is acceptance by 60 percent of the existing users. Upon successful acceptance of the proposal, the old emetogenicity levels of the chemotherapy drug are replaced by the proposed levels. Note that emetogenicity levels are used only while creating a new regimen. Hence, a change in emetogenicity levels will not affect the existing regimens.

Note that each of the acceptance criteria above also has an implicit rejection criterion. If X percent is the acceptance criterion for a proposal then $(100 - X)$ percent is the rejection criterion of that proposal.

4.5.3.2 Vote Proposal

In order to vote on a proposal, a user either accepts or rejects it. The system saves the vote in the proposal database, and when the acceptance or rejection criterion associated with the proposal is met, that proposal is considered resolved. The system performs the administrative task if the proposal is accepted, and in either case, the proposal is sent to the resolved proposal list. A workflow diagram for this function is shown in Figure 4.30; as voting on a proposal is done on the same screen where proposal status is viewed, see Section 4.5.3.3 for screenshot.

The following decisions were made during the design of different features involved in the *Vote Proposal* function:

1. **User accepts a proposal when he/she adds it:** A user who adds a proposal does not need to vote on it. New proposals added by the user are considered to be accepted by the user who adds them.
2. **Criteria for acceptance or rejection of proposal:** A proposal is considered resolved as soon as its acceptance or rejection criterion is met. Hence, not all the users of system are required to vote on a proposal when an acceptance or rejection criterion is already fulfilled. Once a proposal has been resolved it is moved to the resolved proposal list.

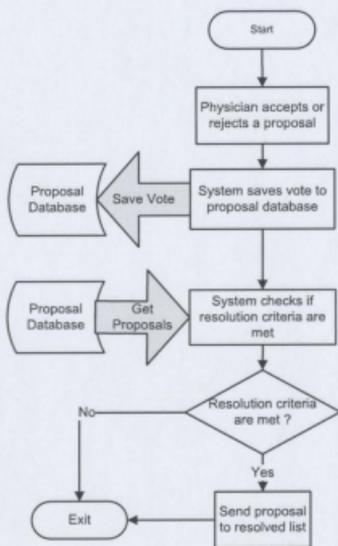


Figure 4.30: Workflow Diagram: Vote Proposal

4.5.3.3 View Status of Proposals

When a user selects the system administration option the system displays all the outstanding proposals that remain unresolved. A user can add a new proposals (see Section 4.5.3.1) or vote on existing proposals (see Section 4.5.3.2). A workflow diagram of this function is shown in Figure 4.31, and a screenshot of the implemented page to view the status of proposals is shown in Figure 4.32.

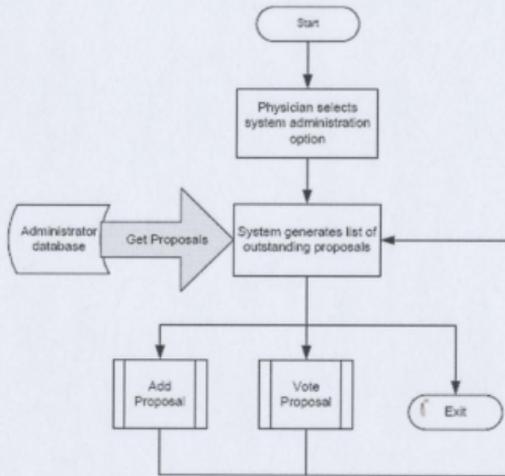


Figure 4.31: Workflow Diagram: View Proposals

Quick Links

- › Patient Search
- › Prescribe Regimen
- › Add Patient
- › Create Regimen
- › Alter Regimen
- › Import Regimen (0 New)
- › View Deleted Regimens
- › View Regimens
- › Admin
- › Logout

Proposals

PROPOSAL TYPE	PROPOSED BY	DATE	COMMENTS	RESPONSE
ADD USER SNAQVI	TWAREHAM	2007-08-14	● VISITING	ACCEPTED
DELETE USER TWAREHAM	EYEDINAQVI	2007-08-14	●	ACCEPT ↕ REJECT ↕
● CHANGE EMTOGENIC POTENTIAL	EYEDINAQVI	2007-08-14	●	ACCEPT ↕ REJECT ↕

Submit

[Add Proposal](#)

[View Event Log](#)

Page last updated: 08/14/2007 22:26:34

Figure 4.32: Screenshot: View Proposals

4.6 Database Design

To implement the Entity objects that were described in Section 4.3 and used in previous sections, we have designed a database for our system consisting of 18 schemas. The relationships among these schemas, as well as their associated Entity objects, is shown in Figure 4.33. As we can see from this figure, an Entity object may be mapped onto a collection of several schemas.

The description of each schema is given below. Each schema consists of a collection of attributes, where each attribute has an associated basic datatype. In each schema the attribute that is the primary key is bold faced and a foreign key attribute is italicized. An attribute is both a primary and a foreign key if it is bold faced and italicized. A short description will be given at the end of each schema for any attribute marked by a star (*). Schemas are listed alphabetically by the objects they are associated with as follows:

- **Administrator**

- Administrator (**ProposalID**: integer, *CreatorUsername*: string, Status*: enum, CreationDate: date/time, Comments: string, Category: string, AcceptanceCriteria: integer)

*Status**: Status can be pending, rejected, or accepted.

- AddUserProposal (*ProposalID*: integer, *Username*: string)
- DeleteUserProposal (*ProposalID*: integer, *Username*: string)
- ChangeEmetogenicityPotentialProposal (*ProposalID*: integer, *DrugID*: integer, Level: integer, DoseUpperLimit: integer, DoseLowerLimit: integer)

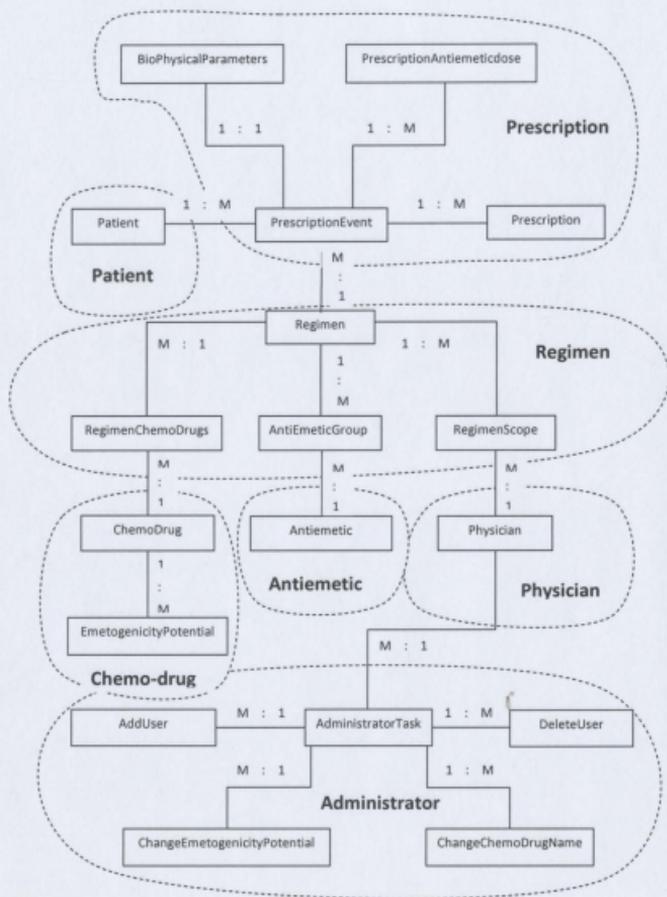


Figure 4.33: Database Entity-Relationship Diagram

- ChangeChemoDrugNameProposal (ProposalID: integer, DrugID: integer, NewName: string)

- **Antiemetic**

- RegimenAntiEmetics(DatabaseGroupID: integer, TradeName: string, GenericName: string, Dose: float, Route: string)

- **ChemoDrug**

- ChemoDrug (DrugID: integer, Agent: string)
- EmetogenicityPotential (DrugID: integer, Level: integer, DoseUpperLimit: integer, DoseLowerLimit: integer)

- **Patient**

- Patient (DatabasePatientId: integer, FirstName: string, LastName: string, MCPNumber: string, Gender: string, Birthday: date/time, StreetAddress: string, City: string, Province: string, PostalCode: string)

- **Physician**

- Physician (Username: string, Password: string, LicenseNumber: string, FirstName: string, LastName: string, DateAdded: date/time, EmailAddress: string, Active*: boolean)

Active: Only physicians for which attribute active is set to true can use the system.*

• Prescription

- PrescriptionEvent (**DatabasePrescriptionID**: integer, *DatabasePatientID*: integer, *DatabaseRegimenID*: integer, CycleNumber: integer, PrescriptionDate: date/time, Comments: string, PhysicianUsername: string, DoseReduce: integer, Cancelled: boolean, CancelledBy: string, CancellationDate: date/time)
- Prescription (*DatabasePrescriptionID*: integer, DrugName: string, CalculatedDose: float, PrescribedDose: float)
- BioPhysicalParameters (*DatabasePrescriptionID*: integer, Height: integer, Weight: integer, CalculatedBSA*: float, UsedBSA: float)
*CalculatedBSA**: BSA is Body Surface Area, calculated on basis of patient height and weight (see Section 4.4.1.3 for details).
- PrescriptionAntiEmeticsDose (*DatabasePrescriptionID*: integer, DrugName: string, DrugDose: float, DrugRoute: string, Frequency: string)

• Regimen

- Regimen (**DatabaseRegimenId**: integer, RegimenName: string, *Username*: string, DateEntered: date/time, Comments: string)
- RegimenScope (*DatabaseRegimenID*: integer, *PhysicianUsername*: string, CreatorLicenseNumber: string, Scope*: enum, View*: boolean, CreationDate: date/time)
*Scope**: scope can be public, private or default.

*View**: view is true if regimen has been viewed by physician, and false otherwise.

- AntiEmeticsGroup (DatabaseGroupID: integer, DatabaseRegimenID: integer, GroupName: string)
- RegimenChemoDrugs (DatabaseRegimenId: integer, DrugID: integer, Quantity: float, Measure: string, MaxDose: float, Instructions: string, Frequency: string, Route: string)

4.7 Summary: Implications for Medical Informatics Development

We have learned the following lessons in the course of designing our system:

- When designing medical systems, a key to success is to pay attention to both the workflow and the interface of the new system. Attention to workflow for the computerization of healthcare is important because of its user guided acceptance nature (see Section 6.1). This will involve the physician champion who will both ensure that the new system mimics the old system and guide workflow and interface design decisions of the new system. Moreover, where possible, developers themselves should observe the existing workflow. In doing this on our own system, we recognized that much of the system design consists of guidelines rather than rigid requirements; physicians tend to break the rules in the interest of patient care and any requirement can rapidly become a guideline which can be ignored at doctors' discretion. Hence, a flexible workflow and

interface is very important for a successful computerization of healthcare.

- Our second lesson is based on what we have noticed while computerizing a small healthcare workflow, *i.e.*, chemotherapy. Our experience on this project suggests that focusing on a small healthcare workflow has benefits over working on a larger and more general system:
 - We can pay attention to and capture the details of the existing workflow in a much better way that we can do it for a larger, more complex workflow.
 - Though keeping it small results in the system operating under the Small Organizational Model, which is a problem because of limited resources, by adopting the democratic model, we can get around this resource problem to a large degree.
 - Last but not least, because we are dealing with a small and focused user group with very well-defined needs and tasks, this approach allows us to map system evolution much further forward into the future.

From the above, it is obvious that having one or more physician champions involved in the design process is very important. Although it is desirable that wherever possible computer literate physicians be involved in the design process, this level of computer literacy should be fairly high. Physicians with little knowledge of computers can actually be dangerous to the design process, as they may insist on certain decisions that they believe are correct based on their limited experience, and they may accidentally limit the design process by omitting features that they believe (on the basis of their limited experience) are too difficult to put into the system.

We make following conjectures based on the lessons given above:

- **Conjecture 1:** Our “aim to small workflow” philosophy in conjunction with the democratic model will work in other focused healthcare organizations.
- **Conjecture 2:** Our “aim to small workflow” philosophy leads to a “bottom-up” philosophy for overall computerization of the healthcare organization: Instead of starting computerization for all departments of a healthcare organization at once, one should instead develop a collection of carefully selected focused groups within that organization that cover the range of possibilities of variation in the whole system and from there develop the whole system.
- **Conjecture 3:** In this thesis, we viewed several mechanisms for involving physician champions in the system development process. Our mechanism of choice has been face-to-face meetings. However, more flexible schemes are possible. For instance, one can have an on-line forum structure which can be used for discussion on different screens or as an error reporting tool. We implemented such a structure at the early stage of the project. Though we did not use it, we conjecture that in situations where representative champions must be involved and their schedules do not allow frequent face-to-face meetings; such a system may work well.

Conjecture 1 brings up an interesting point: does the democratic model works as well for software systems in organizations other than healthcare, where instead of having free agents like physician, we have more traditional-type employees? This will be addressed more in Chapter 7.

Chapter 5

System Development

In this chapter we will describe the overall software architecture and technologies underlying the implementation of our system as described in Chapter 4. We realize that this design may be implemented differently relative to more advanced current technologies or technologies that will evolve in future. However, we wish to describe the choices we have made and the justification of these choices not only as documentation of our system, but also as an example of how such technologies choices should be made with respect to a medical informatics system. After a brief discussion of the development process (Section 5.1), we will describe the MVC model (Section 5.2). We will then move on to the various technologies used in this system (Section 5.3), where we will describe both the available options and evaluate these options for their suitability for medical informatics software. Finally, we will have a section on the lessons learned during system development (Section 5.4).

5.1 Development Process

As the Spiral model (Section 2.3.2.4) was selected for the development of the system, the breakdown of system tasks in the initial planning phase into single-user features and multi-user features was helpful in dividing the system into subtasks. First, single-user features were developed and after the successful development of single-user features, multi-user features were added to the system. Each iteration included the initial planning, requirement gathering, design, development, testing, and evaluation phases as shown in Figure 5.1.

5.2 Development Model

In this section, we will discuss the overall system development model. As this system is web-based, after a brief introduction to web based technologies, we will describe different models and evaluate them.

Consider the evolution of the web and its relationship to the software development model [66]. In the early days of the web, web pages were only used as a source of information. They were developed as static HTML pages, which were accessible over the internet. In the past few years web development tools and technologies like J2EE, which is used to execute Java applications on web servers, have enabled developers to design multi-tier web applications for large organizations. Web pages have now become dynamic entities that allow the input and display of information and can be created or modified dynamically. Moreover, these web pages function as forms, interacting with deeper logic in the system.

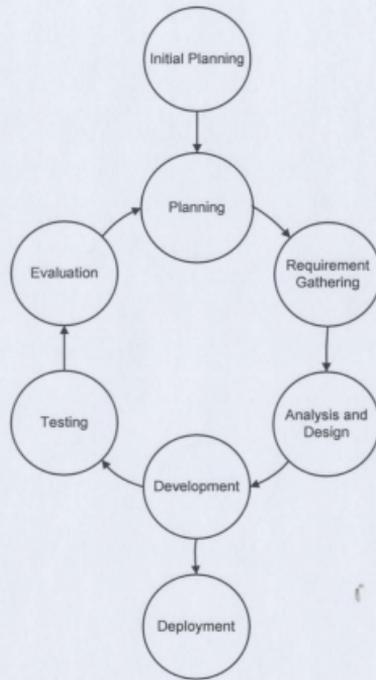


Figure 5.1: Spiral Development Model

As we see above, the evolution of web based systems has gone from static pages to a fusion of display pages, which may be dynamic, and underlying code. This is managed by the use of multi-tier models. There are two such tier models: the 2-tier model, and the 3-tier model. The 2-tier model naturally models a distributed application. A distributed application is naturally split into two layers: the interaction layer and the processing layer. The interaction layer is used as the interface for the user, where it displays data and receives commands from the user, translating data or commands received from the user and delegating them to the processing layer. The processing layer processes data and sends it back to the interaction layer. For many applications, a 2-tier model is sufficient, but if accessing data in the processing layer is complex then we should split the processing layer further into two more layers: the business logic layer and the data access layer, where the business logic layer encompasses all processing and the data access layer handles the transfer of data in and out of the database. The resulting three-layer approach is used in the Model-View-Controller (MVC) model of system development, where the developer separates functions associated with data (Model), user interface (View) and business logic (Controller) into three different layers. As healthcare organizations often involve complex data processing, we are using the MVC model as the development model of our project.

5.3 Technology Decisions

Given our chosen development model, technological decisions fall under four components. We need a web development framework for building under MVC model, an

IDE to manipulate things easily, a database, in order to store and access information effectively, and a web server to run the application. In this section, we will list the choices for each of these four components and then see which are most appropriate for our project.

5.3.1 Development Framework

There have been many web development frameworks created under the scope of MVC modeling. Every development framework has its primary advantages which force developers to use the MVC model in web application development. Along with this primary advantage, each framework has its own set of advantages and disadvantages. In this section we will discuss the two major web development frameworks, Jakarta Struts and the Spring MVC Framework. Note that both of those frameworks are based on J2EE technology. We will look at the advantages and disadvantages of these frameworks, and we will discuss which framework is best for the development of our chemotherapy prescription system.

5.3.1.1 Jakarta Struts

Jakarta Struts [4] is an open source project by the Apache Software Foundation. It is a Java implementation of the MVC model. This project was originally created by Craig McClanahan, but it was later taken over by the open source community. Struts provides additional benefits, offering a collection of utilities to handle many of the most common tasks of web application development.

Jakarta Struts offers the following advantages during application development [66, 29, 28]:

1. **Centralized XML Configuration:** Most configuration values are part of XML instead of being hard coded in Java files. This helps developers to concentrate on development tasks rather than caring for the system layout.
2. **Struts Tags:** Struts provides custom tag libraries for HTML forms, which can implement iteration on different types of data structures, *e.g.*, arrays of objects.
3. **Form Validation:** Struts offers an additional layer of form field validation, which is used to validate the form data before any type of processing in the business logic layer. This extra layer helps in breaking down the application into different components and also makes it faster, as validation is done before any type of processing is done on form values.
4. **Consistent Approach:** Struts enforces a consistent MVC approach throughout the web application.
5. **Developer Support:** Struts has a relatively larger user community and a large knowledge base, both on the web and in the form of books.
6. **Testability:** Testing can be performed in Struts using `StrutsTestCase`, an in-built unit testing facility to test different objects in the system [64].
7. **IDE Support:** Struts has a lot of IDE support and even has IDEs built on top of it (see Section 5.3.2).

The major disadvantage of the Struts framework is that Struts is relatively hard to understand for a new developer. It has a steep learning curve, as the developer needs to know both Java Server Pages (JSP) [72], a Java technology to dynamically generate

HTML and XML, and Servlet [73], which can run Java code on the server and send HTML pages to a browser, before learning the struts framework.

5.3.1.2 Spring MVC Framework

The Spring MVC framework [33] is also an open source framework for the Java platform. It was first developed by Rod Johnson, and was first released under an Apache 2.0 license in 2003. The Spring framework provides a fully functional MVC module for web application development. The Spring MVC framework provides the following advantages [33]:

1. **Testability:** Spring allows easy testing with Spring Mocks, which is an inbuilt unit testing facility used to test objects in the system
2. **Highly configurable:** Spring is designed in such a way that every piece of logic and functionality is highly configurable. This makes it capable of integrating effortlessly with other popular frameworks.
3. **JSP tag library (also known as the Spring tag library):** Spring provides a comprehensive set of tags for handling form elements when using JSP and Spring Web MVC. Each tag provides support for the set of attributes of its corresponding HTML tag counterpart, making the tags familiar and intuitive to use.

The major disadvantage of the Spring MVC framework is that as the Spring MVC framework is relatively new, it has a comparatively smaller user community and fewer resources, such as books and tutorials.

5.3.1.3 Evaluation of Struts and Spring frameworks

We have seen that both MVC frameworks have advantages and disadvantages attached to their usage. As the Spring framework is relatively new and has a smaller user community and less support for the developers, the Struts framework seems like a rational choice for an academic project. Moreover, for large applications, Struts enforces MVC, which is helpful in the division of applications into different layers and best for an iterative development approach such as the Spiral model (see Section 2.3.2.4), which was used for this project. Hence, we will use Struts as a development framework for the development of our chemotherapy prescription system.

5.3.2 Development IDE

A highly encouraging aspect of Struts is its massive support from different IDEs, including Eclipse, NetBeans, IBM WebSphere, Borland JBuilder X, Struts Console and Struts Studio [76]. Out of these the most popular are Eclipse¹ and NetBeans. Both NetBeans and Eclipse offer several wizards to automate the process of development and simplify the complex struts environment. Netbeans has a gentler learning curve than Eclipse. However, Eclipse is famous because of its light weight, *i.e.*, minimal usage of memory and processing time. Moreover, Eclipse has great support for unit testing, *i.e.*, the testing of individual classes or collections of classes in isolation. Given the benefits of Eclipse over NetBeans, Eclipse is the natural choice for the development of our chemotherapy prescription system.

¹Eclipse alone does not support Struts, but Eclipse with the plug-in MyEclipse makes it the most popular IDE used for the Struts development framework. MyEclipse was developed in 2003 and has continuously improved in each version [16].

5.3.3 Database

Though there are many available database technologies, we have restricted our comparison to two of the most popular, MySQL [48] and Oracle [55]. MySQL is selected for comparison because it is the best free, *i.e.*, open source, database and Oracle because it is the best commercially-available database.

The Oracle database is a full featured database engine that is well known for its security and performance and is an obvious choice for any large application where security and performance are primary concerns [57]. Moreover, due to its built-in support for Java, developers can develop stored procedures, triggers, and functions that can be executed in the database. On the other hand, MySQL, a free open source database, offers most of the services that Oracle provides and also provides security in terms of individual column locking. For any application, such as healthcare, that has a strong need for security we recommended Oracle; however, as we are working on an academic project, where cost is more important than security, a free database like MySQL is more appropriate for our project.

5.3.4 WebServer

There are many available web-servers, but we have selected the two most commonly used web servers that support J2EE for comparison, namely JBoss [62] and Apache Tomcat [4]. The basic difference between these two servers is that JBoss is a complete Java application server; it supports EJB (Enterprise Java Beans) [71], is a component architecture for the development and deployment of object-oriented, distributed, enterprise-level applications, and also includes the Tomcat server. On the other hand,

Apache Tomcat is used for the support of Java Servlet and Java Server Pages. Apache Tomcat was chosen for our chemotherapy prescription system due to its faster load time and lower memory usage. Otherwise, both servers have similar features except that JBoss is capable of running EJBs, which we are not using in this project.

5.4 Summary: Implications for Medical Informatics Development

We have learned the following lesson in the course of developing our system:

- The MVC model is the most widely used development model in web applications. It is also the most appropriate for medical informatics as well. Due to the continuously evolving health field, the MVC model makes it easier for an enterprise or software packager to continually evolve an application as new needs and opportunities arise. Moreover, medical informatics tools and applications are usually complex systems. For such systems the clear division of the system into layers under MVC model is very important and helpful to the development and maintenance of the system.
- There are factors affecting technology choices in medicine. Given that we want to ease acceptance and have the familiar look and feel of older systems, technologies should be chosen to be consistent with those used by these older systems. Moreover, there are additional factors related to medical technology in general and the small organizational setting. Security is paramount, regardless of what level of organization you are working with and in the small organizational set-

ting the technology must be able to function semi autonomously and be cheap because of limited resources.

During the system development process, physician champions can still be a guide to technology choices. The advice of physicians is useful to the extent that it helps you find those factors in the workplace environment that dictate these choices. These factors may involve types of technologies physicians are more familiar with and are more comfortable using. However, we should pay attention to physician advice on technology to the extent that the physician's knowledge and experience does not accidentally hamper the development process (see Section 4.7).

Chapter 6

System Acceptance and Implementation

In Chapters 4 and 5, we discussed the design of our chemotherapy prescription system and the various technologies used during the development of the system. We observed how the development of a software tool for a healthcare organization is different from similar developments for other organizations and requires special considerations (see Section 2.2.1.4). Like other software development phases, the implementation of such a system in a target workplace is also affected by these factors.

Though, as shown in Chapter 5, we have fully implemented and tested the system described in Chapter 4, we have not been able to complete the user acceptance process, *i.e.*, the system has not yet been implemented in the clinical setting or tested by clinicians not involved in the development process. However, this chapter contains our notes on how this process should proceed. We will first discuss different models for acceptance of new technologies relative to types of user communities (Section 6.1), and

we will see which model is applicable to the acceptance of medical informatics systems. We will then discuss how a medical informatics system should be implemented in any healthcare organization (Section 6.2). Finally, we will discuss different lessons learned for medical informatics software in general (Section 6.3).

6.1 System Acceptance Model

In this section we will first discuss models for the acceptance of new technologies relative to the different types of user communities, *i.e.*, user-driven acceptance vs. organization-driven acceptance and then discuss which model is most appropriate for healthcare organizations.

The most popular acceptance model in existence is TAM, developed by Davis in 1989 [22], which was explicitly phrased in term of users in general. However, for the medical informatics environment we would like to distinguish between two forms of TAM: the user-driven acceptance model, which corresponds to the original TAM model by Davis, and an organization-driven acceptance model [58] extended from the original TAM model:

- **User-Driven Acceptance:** This model describes different factors that are involved in the adoption of new software by individuals. The two main factors that are identified in the model are perceived usefulness and perceived ease of use, Where **perceived usefulness** is defined by the model as: *“The degree to which a person believes that using a particular system would enhance his or her job performance.”*[22, pp. 320] and **perceived ease of use** is defined as: *“The degree to which a person believes that using a particular system would be*

free of efforts" [22, pp. 320]. In the real world, different constraints that are applied to the TAM include time constraints, limited abilities, organizational limits, and unconscious habits, all of which affect the individual acceptance of new technology [7].

- **Organization-Driven Acceptance [58]:** As noted above, different constraints on TAM also include organizational limits. The acceptance of a new software or system by an organization is mostly dependent on the management or IT department of the company, who dictate to employees the adoption of that new system. Individual employees consent is overridden by the decree of top management in the organization, who are responsible for allocating resources, or the IT division who are more familiar with the computer field and are considered in a better position to make a decision on the selection of appropriate systems to bring about the desired changes. Hence, corporate culture has a major impact on the decision of individual users towards the adoption of new systems.

One could further sub-divide organization-driven acceptance, but this is not relevant for the purpose of our thesis.

Many studies have extended, and used, TAM in different types of systems and different fields which also include the examination of healthcare professionals under the TAM model [70, 18, 17]. Unlike other organizations, where decisions are made by management and the IT department and employees follow what management decides, healthcare organizations are different due to the fact that decisions about the selection of any new technology are independent of organizational management or IT divisions, preferring instead to use only those systems that are proven to be helpful

for enhancing the process of patient care (see Section 2.2.1.4). Hence, the selection of a new tool/software for a physician falls under the individual user acceptance model, such that the major factors that affect this decision are perceived usefulness, perceived ease of use, *and* the values of the medical profession.

6.2 Implementing Medical Informatics Software in the Workplace

In the previous section, we noted that healthcare organizations are different from many other organizations, in that, acceptance of a new system by healthcare organizations is greatly dependent on the approval of the users of the system. In this section we will discuss how physicians should be made part of the development process in the hope that they will champion the resulting system. The adoption process falls under two parts: software adoption and implementation, which we will discuss in separate subsections.

6.2.1 Software Adoption

As noted in Section 2.3.2, the initial stage of many software process models is getting requirements from potential users of the system under development. For this purpose, users are involved to educate the technology experts about their requirements. However, in the case of a medical informatics software development, what is really required is a partnership between the developers and the users of the systems. Here, users of systems may involve users other than physicians, *e.g.*, pharmacists. In

that case, all types of users must be consulted in every phase of system development. To better understand the system IT experts need to examine the target workplace and watch how people interact with each other in order to make the system flexible enough that it fully follows the actual healthcare workflow (see Section 3.1.4). Moreover, there is a need to involve all users of the system during the design decisions, so that they all feel invested in the final solution. In our case, the development of the chemotherapy prescription process, we involved physician champions not just in the requirement gathering phase but in every development phase of the system, as seen in previous chapters.

6.2.2 Software Implementation

For the implementation of the system in the target workplace, there is to our knowledge no model available in the software engineering literature. Under the Spiral model (see Section 2.3.2.4), which we are using for the development of our chemotherapy prescription system, end users start evaluating different builds of the system from the beginning of software development, but during the final implementation of the system in the workplace we need to provide training to all^f users, especially to users who were not involved in the system evaluation process. After this training, there are two ways to proceed: (1) have a system live date when users stop using the old system and shift to the new system, or (2) have the system implemented in a trial process¹. If the workplace acceptance is organization-driven, setting up a specific im-

¹The second option is similar to beta testing [34], in that the new system operates in parallel with the old system. However, in the case of beta testing the purpose is to establish correctness of the new system, whereas here the purpose is to promote user acceptance.

plementation date is possible and is the easiest way to implement a system; however, if the workplace has user-driven acceptance, as in healthcare organizations, systems should be implemented in a number of trials before the actual implementation of the system. Here, the purpose is to make sure that the new system adequately deals with all possibilities which are part of the healthcare workflow and ensure that the system is error free before it is implemented. Otherwise, if users find errors, they will lose confidence in, and stop using, the system.

6.3 Summary: Implications for Medical Informatics Development

We have learned the following lesson in the course of preparing for the acceptance and implementation of our system:

- As described in the lessons learned in Chapters 4 and 5, we chose representative physicians champions based on their knowledge. In the acceptance and implementation phase, we need to involve every user of the system. Given that healthcare has user guided acceptance, getting a system accepted by all users will then help with any problems with the acceptance from IT department and bureaucracy.
- After getting the whole user community involved, the next step is to make sure not to lose them. Users can still reject the system if they lose their faith in the system due to the wrong types of training or implementation methods. We must make sure that software tools are error free before they are implemented.

Moreover, before implementation, running a system in trial mode parallel with the existing system (rather than running the new system alone) can help in user testing and can identify the problems without losing the faith of users on the new system.

The role of physician champion at this point is now particularly critical; moreover, it is mainly political. If the representative physician champions have been chosen correctly, they are people who are deeply embedded in the user community and have the political know-how and personal relationships that will ensure that the new system is acceptable to the whole user community. Note how this situation is different from the typical implementation of software in business organizations, where bureaucracy and IT departments are in control and users have virtually no input or role in the acceptance process.

Chapter 7

Conclusion and Future Directions

In this thesis, we have developed an on-line web-based system for chemotherapy prescription. This system takes into account a number of factors that are involved in the development of medical systems. In the course of developing this system, we have proposed the concept of system operational models. We have noted that our system falls under the Small Organizational Model, and we have proposed a way of dealing with the associated problems of limited resources for maintenance and evolution in the design of such a system. We have also described a number of lessons learned while developing this system which are applicable both to developing similar systems and to developing medical software in general.

There are number of directions for future research. The most obvious of these involve extensions to the system developed in this thesis:

- Different reporting tools can be added to the system, which will give people the ability to analyze medical data. This process of adding reporting tools to the system will need to go through another requirement gathering phase to collect

information about the types of report that the end users want from the system.

- The system can be integrated to work with existing computerized systems and databases. It can be integrated with existing patient or drug databases; it could also be integrated with any existing pharmacy systems to aid sending prescriptions in electronic form, providing appropriate legislation is put in place (see Section 2.2.1.3).
- Along with the reports for primary care, administrative and research-oriented based reports can be generated from the patient and prescription databases. As such reports are no longer primary-care related and will not be read by the doctors themselves, we must implement dataset anonymisation techniques to ensure patient privacy (see Section 2.2.1.3).

More radical extensions of this system involve fundamental redesign. For example, as mentioned in Section 4.7, medical workflow consists of guidelines rather than requirements. Hence, we can attempt to build a much more free-form system, in which not only the workflow which we have seen but the full variability of that workflow is implemented in the system. This might also include a more flexible user interface, which physicians could customize according to their individual needs and preferences.

The work in this thesis also suggests a possible line of software engineering research. We have identified what we believe to be a gap in current practice related to system operational models. In particular, we have identified a type of system (falling under the Small Organizational Model) which is not being handled well by current software development practice. We have proposed one way to getting around problems associated with this model, namely, the democratic model of system oper-

ation. However, as noted in Section 4.7, this model may not be applicable outside of a user-oriented environment like that in healthcare. Hence we need to investigate alternatives to the democratic model of system operation in healthcare system, evaluate the usefulness of these models outside healthcare, and further explore the full spectrum of system operational models that exist in software applications in general.

Bibliography

- [1] Allan, A. (2004) "Passwords Are Near the Breaking Point." *Gartner, Research Note*. http://www.indevis.de/dokumente/gartner_passwords_breakpoint.pdf
- [2] American Cancer Society. (2007) "What Is Chemotherapy And How Does It Work?" <http://www.cancer.org>
- [3] American Medical Network. (2006) "What is Chemotherapy?" <http://www.health.am>
- [4] Apache Software Foundation. (2007) "Struts." <http://struts.apache.org/>
- [5] Apache Software Foundation. (2007) "Apache Tomcat." <http://tomcat.apache.org/>
- [6] Ash, J.S., Stavri, P.Z. and Kuperman G.J. (2003) "A Consensus Statement on Considerations for a Successful CPOE Implementation." *Journal of the American Medical Informatics Association*, **10(3)**, 229-234.

- [7] Bagozzi, R. P., Davis, F. D., and Warshaw, P. R. (1992) "Development and Test of a Theory of Technological Learning and Usage." *Human Relations*, **45(7)**, 659-686.
- [8] Basili, V.R. and Turner, A.J. (1975) "Iterative Enhancement: A Practical Technique for Software Development." *IEEE Transactions on Software Engineering*, **1(4)**, 390-396.
- [9] Bates, D.W., Leape, L.L., Cullen, D.J., Laird, N., Petersen, L.A., Teich, J.M., Burdick, E., Hickey, M., Kleefield, S., Shea, B., Vander, V.M., and Seger, D.L. (1998) "Effect of Computerized Physician Order Entry and a team Intervention on Prevention of Serious Medication Errors." *Journal of the American Medical Association*, **280(15)**, 1311-1316.
- [10] Bell, D. (2001) *Software Engineering, A programming Approach*. (Third Edition). Addison Wesley; Reading, MA.
- [11] Bobb, A., Gleason, K., Husch, M., Feinglass, J., Yarnold, P.R., and Noskin, G.A. (2004) "The Epidemiology of Prescribing Errors, The Potential Impact of Computerized Prescriber Order Entry." *Archives of Internal Medicine*, **164(7)**, 785-792.
- [12] Boehm, B.W. (1988) "A Spiral Model of Software Development and Enhancement." *IEEE Computer*, **21(5)**, 61-72.
- [13] California Healthcare Foundation (2001) "A Primer On Physician Order Entry." <http://www.chcf.org/documents/hospitals/CP0Ereport.pdf>

- [14] Canada Health Infoway (2006) "BEYOND GOOD INTENTIONS: Accelerating the Electronic Health Record in Canada." A Policy Conference Held on June 11-13, 2006 Montebello QC. [http://www.infoway-inforoute.ca/Admin/Upload/Dev/Document/Conference Executive Summary_EN.pdf](http://www.infoway-inforoute.ca/Admin/Upload/Dev/Document/Conference%20Executive%20Summary_EN.pdf)
- [15] Canada Health Inforway (2007) "Canada Health Infoway." <http://www.infoway-inforoute.ca/>
- [16] Carnell, J., and Harrop, R. (2004) *Pro Jakarta Struts*. Apress; Berkeley, CA.
- [17] Chau, P.Y.K. and Hu, P. J. (2001) "Information Technology Acceptance by Individual Professionals: A Model Comparison Approach." *Decision Sciences* **32(4)**, 699-719.
- [18] Chismar, W.G. and Wiley-Patton, S. (2003) "Does the Extended Technology Acceptance Model Apply to Physicians." in *Proceedings of the 36th Hawaii International Conference on System Sciences*, IEEE Press; Los Alamitos, CA.
- [19] Collen, M.F. (1970) "General requirements for a Medical Information System (MIS)." *Computers and Biomedical Research*, **3(5)**, 393-406.
- [20] Colpaert, K., Claus, B., Somers, A., Vandewoude, K., Robays, H., and Decruyenaere, J. (2006) "Impact of computerized physician order entry on medication prescription errors in the intensive care unit: a controlled cross-sectional trial." *Critical Care*, **10(1)**, R21.

- [21] Cutler, D.M., Feldman, N.E., and Horwitz, R.J. (2005) "U.S. Adoption Of Computerized Physician Order Entry Systems." *Health Affairs*, **24(6)**, 1655-1663.
- [22] Davis, F. D., (1989) "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology." *MIS Quarterly*, **13(3)**, 319-340
- [23] de Dombal, F.T., Leaper, D. J., Staniland, J.R., McCann, A.P. , and Horrocks, J.C. (1972) "Computer-aided diagnosis of acute abdominal pain." *British Medical Journal*, **5804(2)**, 9-13.
- [24] Dennis, G. (2005) "GOING THE EXTRA MILE Interoperability is key to EHR development in Canada" *Healthcare Information Management and Communications Canada*, **19(4)**, 44-46.
- [25] Evans, R.S., Pestotnik, S.L., Classen, D.C., Clemmer, T.P., Weaver, L.K., Orme, J.F., Lloyd, J.F., and Burke, J.P. (1998) "A Computer-Assisted Management Program for Antibiotics and Other Antiinfective Agents." *The New England Journal of Medicine*, **338(4)**, 232-238.
- [26] Floyd, C. (1984) *A Systematic Look at Prototyping*. In: Budde, R., Kuhlenskamp, K., Matthiassen, L., and Zllighoven, H. (eds.) "Approaches to Prototyping." Springer Verlag; New York. pp. 1-17.
- [27] Garets, D., and Davis, M. (2006) "Electronic Medical Records vs. Electronic Health Records: Yes, There Is a Difference." *HIMSS Electronic Health Record Committee*. http://www.himssanalytics.org/docs/WP_EMR_EHR.pdf

- [28] Goodwill, J. (2002) *Mastering Jakarta Struts*. Wiley; Indianapolis, IN.
- [29] Goodwill, J., Hightower, R. (2004) *Professional Jakarta Struts*. Wiley; Indianapolis, IN.
- [30] Handler, T., Holtmeier, R., Metzger, J., Overhage, M., Taylor, S., and Underwood, C. (2003) "HIMSS Electronic Health Record Definitional Model Version 1.1" *HIMSS Electronic Health Record Committee*. <http://www.himss.org/content/files/ehrattributes070703.pdf>
- [31] Health Level Seven, Inc. "Health Level 7." <http://www.hl7.org/>
- [32] Jha A.K., Ferris T.G., Donelan K., DesRoches C., Shields A., Rosenbaum S., Blumenthal D. (2006) "How Common Are Electronic Health Records In The United States? A Summary Of The Evidence." *Health Affairs (Project Hope)*, **25(6)**, 496-507.
- [33] Johnson, R., Hoeller, J., Arendsen, A., Sampaleanu, C., Harrop, R., Risberg, T., Davison, D., Kopylenko, D., Pollack, M., Templier, T., Vervaeet, E., Tung, P., Hale, B., Colyer, A., Lewis, J., Leau, C., and Evans, R. "The Spring Framework - Reference Documentation." <http://www.springframework.org/docs/reference/>
- [34] Kaner, C., Falk, J., and Nguyen, H. (1999) *Testing Computer Software*. Wiley; Indianapolis, IN.

- [35] Kaushal, R. and Bates, D.W. (2001) "Computerized Physician Order Entry (CPOE) with Clinical Decision Support Systems (CDSSs)." <http://www.ahrq.gov/clinic/ptsafety/chap6.htm>
- [36] Kohn, L.T., Corrigan, J.M. and Donaldson M.S. (2000) *To Err Is Human: Building a Safer Health System*. Institute of Medicine, The National Academies Press.
- [37] Kuperman G.J. and Gibson R.F. (2003) "Computer Physician Order Entry: Benefits, Costs, and Issues." *Annals of Internal Medicine*, **139**(1), 31-39.
- [38] Kuperman, G.J., Teich, J.M., Gandhi, T.K. and Bates, D.W. (2001) "Patient Safety and Computerized Medication Ordering at Brigham and Womens Hospital." *Joint Commission Journal on Quality and Patient Safety*, **27**(10), 509-521.
- [39] Laerum, H., Karlsen, T.H., Faxvagg, A. (2003) "Effects of Scanning and Eliminating Paper-based Medical Records on Hospital Physicians Clinical Work Practice." *Journal of the American Medical Informatics Association*, **10**(6), 588-595.
- [40] Langberg, M.L. (2003) "Challenges to implementing CPOE: A case study of a work in progress at Cedars-Sinai." *Modern Physician*, **7**(2), 21-22.
- [41] Legislative Renewal Staff (2003) "Legislative Renewal-Issue Paper, Prescription Drugs (Schedule F)." http://www.hc-sc.gc.ca/ahc-asc/alt_formats/hpb-dgps/pdf/legren/prescription-ordonnance_e.pdf

- [42] Markle Foundation. (2005) "Linking Health Care Information: Proposed Methods for Improving Care and Protecting Privacy." *Working Group on Accurately Linking Information for Health Care Quality and Safety* <http://www.connectingforhealth.org/assets/reports/linkingreport.2.2005.pdf>
- [43] Marion, J.B., David, E.G., and Thomas, J.H. (2003) "Leveraging IT to Improve Patient Safety." *Yearbook of Medical Informatics of the International Medical Informatics Association (IMIA)*.
- [44] Meditech. (2007) "MEDITECH Shaping the World of Health Care." <http://www.meditech.com/>
- [45] Middleton, B., Hammond, W.E., Brennan, P.F., and Cooper, G.F. (2005) "Accelerating U.S. EHR Adoption: How to Get There From Here. Recommendations Based on the 2004 ACMI Retreat." *Journal of the American Medical Informatics Association* **12(1)**, 12-19.
- [46] Miller, C. (2002) "Password Recovery." <http://fishbowl.pastiche.org/archives/docs/PasswordRecovery.pdf>
- [47] Misys Healthcare Systems "Misys In the News." <http://misyshealthcare.com/press+room/in+the+news.htm>
- [48] MySQL AB. "MySQL The world's most popular open source database." <http://www.mysql.com/>

- [49] Naqvi, S. (2003) *Design and Implementation of a Web-Based Chemotherapy Prescription System*. B.Sc.h. Dissertation, Department of Computer Science, Memorial University of Newfoundland.
- [50] National Academies. (2007) "Institute of Medicine of the National Academies." <http://www.iom.edu/>
- [51] National Association of Pharmacy Regulatory Authorities (NAPRA) (2007) "Transfer of Authority to Fill Prescriptions by Electronic Transmission." <http://www.napra.org/practice/electronic.pdf>
- [52] Newfoundland and Labrador Medical Care Plan. (2007) "Medical Care Plan, Government of Newfoundland and Labrador - Canada." <http://www.health.gov.nl.ca/mcp/>
- [53] Odaka, T., Takahama, T., Wagatsuma, H., Shimada, K., and Ogura, H. (1994) "A Visual Data Analysis System for the Medical Image Processing." *Journal of Medical Systems*, **18(3)**, 151-157.
- [54] Opis, (2006) "Clinical Freedom" Opus Healthcare Solutions, Inc. <http://www.opushealthcare.com/>
- [55] Oracle Technology Network "Oracle Database 11g." <http://www.oracle.com/technology/products/database/oracle11g/index.html>
- [56] Perreault, L.E. and Metzger, J. (1999) "A pragmatic framework for understanding clinical decision support." *Journal of Healthcare Information Management*, **13(2)**, 5-21.

- [57] Petri, G., (2005) "A Comparison of Oracle and MySQL." *SELECT Journal*, **1st Qtr(1)**, 41–48.
- [58] Poku, K., Vlosky, R. (2003) "A Model of Marketing Oriented Corporate Culture Influences on Information Technology Adoption." Louisiana Forest Products Development Center Working Paper no. 63. <http://www.rnr.lsu.edu/lfpdc/publication/papers/wp62.pdf>
- [59] Poon, E.G., Blumenthal, D., Jaggi, T., Honour, M.M., Bates, D.W., and Kaushal, R. (2004) "From The Field, Overcoming Barriers To Adopting And Implementing Computerized Physician Order Entry Systems In U.S. Hospitals." *Health Affairs*, **23(4)**, 184–190.
- [60] Potts, A.L., Barr, F.E., Gregory, D.F., Wright, L. and Patel N.R. (2004) "Computerized Physician Order Entry and Medication Errors in a Pediatric Critical Care Unit." *American Academy of Pediatrics*, **113(1)**, 59–63.
- [61] Protti, D.J. (2005) "The Use of Computers in Health Care Can Reduce Errors, Improve Patient Safety, and Enhance the Quality of Service - There Is Evidence." <http://www.connectingforhealth.nhs.uk/worldview/protti2>
- [62] Red Hat Middleware. (2007) "JBoss Application Server." <http://www.jboss.org/products/jbossas>
- [63] Royce, W.W. (1970) "Managing the Development of Large Software Systems: Concepts and Techniques." In: WESCON Technical Papers, Western Electronic Show and Convention; v. 14 Reprinted in *Proceedings of the Ninth International Conference on Software Engineering*, (1989) ACM Press, pp. 328–338.

- [64] Seale, D. (2004) "StrutsTestCase for JUnit v2.1.3" <http://strutstestcase.sourceforge.net>
- [65] Sengstack, P.P. and Gugerty, B. (2004) "CPOE Systems: Success Factors and Implementation Issues." *Healthcare Information Management*, **18(1)**, 36-45.
- [66] Shenoy, S., and Mallya, N. (2004), *Struts Survival Guide: Basics to Best Practices*. ObjectSource LLC; Austin, TX.
- [67] Shortliffe, E.H., Perreault L.E., Wiederhold, G., and Fagan, L.M. (2001), *Medical Informatics, Computer Applications in Health Care and Biomedicine* (Second Edition). Springer; New York.
- [68] Sitting, D.F. and Stead, W.W. (1994) "Computer-based physician order entry: the state of the art." *Journal of the American Medical Informatics Association*, **1(2)**, 108-122.
- [69] Sommerville, I., (2001) *Software Engineering* (Sixth Edition). Addison Wesley; Reading, MA.
- [70] Succi, M.J. and Walter, Z.D. (1993) "Theory of User Acceptance of Information Technologies: An Examination of Health Care Professionals." in: Proceedings of the 32nd Hawaii International Conference on System Sciences, IEEE Computer Society; Big Island, HI.
- [71] Sun Microsystems, Inc. "Java Platform, Enterprise Edition (Java EE) Enterprise JavaBeans Technology." <http://java.sun.com/products/ejb/>

- [72] Sun Microsystems, Inc. "J2EE JavaServer Pages Technology."
<http://java.sun.com/products/jsp/>
- [73] Sun Microsystems, Inc. "J2EE Java Servlet Technology."
<http://java.sun.com/products/servlet/>
- [74] Sun Microsystems, Inc. "Desktop Java JavaBeans."
<http://java.sun.com/products/javabeans/>
- [75] Sweeney, L., (2001) *Computational Disclosure Control for Medical Microdata: The Datafly System*. PhD Thesis, Massachusetts Institute of Technology.
- [76] Taylor, A. (2003) *J2EE and Beyond: Design, Develop, and Deploy World-Class Java Software*. Prentice Hall; Upper Saddle River, NJ.
- [77] Teich, J.M., Osheroff, J.A., Pifer, E.A., Sittig, D.F., Jenders, R.A. (2005) "Clinical Decision Support in Electronic Prescribing: Recommendations and an Action Plan." *Journal of the American Medical Informatics Association* **12**(4), 365–376.
- [78] The 2001 Menucha Conference List. (2001) "Considerations Concerning Computerized Physician Order Entry Implementation." http://ehr.medigent.com/assets/collaborate/2004/04/01/CP0Emenucha_2001.pdf
- [79] VHA Pharmacy Benefits Management Strategic Healthcare Group and Medical Advisory Panel. "Protocol for the Use of Antiemetics to Prevent Chemotherapy-induced Nausea and Vomiting." <http://www.pbm.va.gov/monitoring/antiemeticdosing.pdf>

- [80] William, V.M. (1978) "MYCIN: A knowledge-based consultation program for infectious disease diagnosis." *International Journal of Man-Machine Studies*, **10(3)**, 313-322.

WILSON V. M. (1978) MYCOTA: A knowledge-based consultation program for
diagnosing disease organisms. International Journal of Microbiology, 2(1): 1-12

1978-1979 (2) 01

