

# **OBJECT-ORIENTED RE-ENGINEERING OF THE NURSING DOCUMENTATION SYSTEM HPAD**

by

**Carlos Eduardo Duarte Barrera**

A project submitted in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING  
in  
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO  
MAYAGÜEZ CAMPUS  
2008

Approved by:

---

José A. Borges, Ph.D.  
President, Graduate Committee

---

Date

---

Néstor J. Rodríguez, Ph.D.  
Member, Graduate Committee

---

Date

---

José Navarro, MCpE.  
Member, Graduate Committee

---

Date

---

Viviana Cesaní, Ph.D., P.E.  
Representative of Graduate Studies

---

Date

---

Isidoro Couvertier, Ph.D.  
Chairperson of the Department

---

Date

## **ABSTRACT**

This document describes the process of improving the nursing documentation system, HPAD, by means of a software re-engineering process. The development of the re-engineered system required the study of the existing system to locate both areas for improvement and identify the sections that might be reusable. The redesign process focused on achieving a product easily expandable in the future, with particular emphasis on the software's ability to work independently of the database engine used. A complete rewriting of the existing application was performed while preserving most of the graphical interfaces of the original system. One interface in particular, the Assessment interface, was completely redesigned. The re-engineering process of this project made a significant use of design patterns, particularly the Model-View-Controller was used as a basis for the overall structure of the HPAD software. Other design patterns used to achieve the aim of this project were the Template and Façade patterns. The re-engineered system presents usability improvements such as a change into a MDI interface and data tables sorting; also speed optimizations for some process have been made.

## RESUMEN

Este trabajo describe el proceso de mejora del sistema de documentación de enfermería HPAD a través de la aplicación de procesos de reingeniería del software. El desarrollo de este sistema requirió el estudio del sistema existente para localizar tanto los puntos potenciales de mejora como aquellas secciones que podrían ser reusables. El proceso de rediseño se enfocó en alcanzar un producto fácilmente expandible en el futuro, con un especial énfasis en la habilidad del software para trabajar independientemente del motor de base de datos usado. Se realizó una reescritura completa de la aplicación, conservando la mayoría de interfaces gráficas del sistema existente. Una interfaz en particular, la interface de “*Assessment*”, fue completamente rediseñada. El proceso de reingeniería de este proyecto hace un uso extensivo de patrones de diseño. El patrón más usado fue el “*Model-View-Controller*”, el cual fue usado como base para la estructura general del software HPAD. Otros patrones de diseño usados para lograr el objetivo de este trabajo fueron “*Façade*” y “*Template*”. El sistema resultante presenta mejoras de usabilidad como el cambio a una interface MDI e implementación de ordenación para tablas de datos. Optimizaciones de velocidad de algunos procesos también fueron realizadas.

Copyright © 2008  
By  
Carlos Eduardo Duarte Barrera

*To my family, my exemplary heroes:*

*Mom, Dad,*

*Brother, Sister,*

*And Nana, my real strength*

## ACKNOWLEDGEMENTS

I want to express my sincere appreciation to my advisor, Dr. José Borges for his patience, orientation and support during my studies and especially in this project. His professional and personal qualities have contributed significantly to the educational process at the University of Puerto Rico becomes a valuable experience for me.

Thanks to Dr. Néstor Rodríguez, for his guidance and inputs in the field of Human-Computer Interaction, which were very important not only for the development of this project, but it will be for projects to undertake in the future.

From the Borges-Rodríguez pair remains for me the great impression that one can be an excellent professional and exceptional person at the same time.

Thanks to the members of my graduate committee, Ms. José Navarro, Dr. Celia R. Colón Rivera and Dr. Viviana Cesaní for their valuable contributions and dedication to my project.

I cannot fail to mention my colleague, best friend, unblemished example and companion of battles, Ms Arianna Yamile López Quiroga, who has been my inspiration and impulse engine for achieving the goals that I have proposed and who has been the key player in the achievement I reach with the completion of this work.

Thanks partner.

# TABLE OF CONTENTS

ABSTRACT.....	2
RESUMEN .....	3
ACKNOWLEDGEMENTS.....	6
TABLE LIST.....	9
FIGURE LIST .....	10
<b>1. INTRODUCTION .....</b>	<b>12</b>
2.1 INTRODUCTION.....	18
2.2 RE-ENGINEERING.....	18
2.3 ELECTRONIC RECORD SYSTEMS .....	22
<b>3. SYSTEM DESCRIPTION.....</b>	<b>25</b>
3.1 INTRODUCTION.....	25
3.2 ORIGINAL SYSTEM .....	25
3.3 RE-ENGINEERED SYSTEM .....	26
3.3.1 <i>Login Interface</i> .....	27
3.3.2 <i>Patients List</i> .....	27
3.3.3 <i>Nursing Documentation Interfaces</i> .....	28
3.3.3.1 <i>Orders Interface</i> .....	29
3.3.3.2 <i>Meds Interface</i> .....	31
3.3.3.3 <i>Drips Interface</i> .....	33
3.3.3.4 <i>Vitals Interface</i> .....	33
3.3.3.5 <i>Notes Interface</i> .....	35
3.3.3.6 <i>Assessment Interface</i> .....	37
3.3.3.7 <i>Intake/Output Interface</i> .....	38
3.3.3.8 <i>Pain Interface</i> .....	40
<b>4. ANALYSIS .....</b>	<b>43</b>
4.1 DATABASE ENGINE .....	43
4.2 WINDOW LAYOUT .....	45
4.3 APPLICATION SPEED.....	46
4.4 THIRD-PARTY SOFTWARE .....	49
4.5 OBJECT-ORIENTED IMPLEMENTATION.....	50
4.6 OTHER ISSUES .....	51
<b>5. SYSTEM DESIGN AND IMPLEMENTATION .....</b>	<b>54</b>
5.1 OBJECT-ORIENTED DESIGN.....	54
5.2 DATABASE ACCESS IMPROVEMENTS .....	59
5.3 USABILITY IMPROVEMENTS.....	61
5.4 ALGORITHM IMPLEMENTATION IMPROVEMENTS .....	64
5.5 THIRD-PARTY SOFTWARE USE IMPROVEMENTS .....	64
5.6 DATABASE MODEL IMPROVEMENTS.....	65
5.7 OTHER IMPROVEMENTS .....	68
<b>6. RECAPITULATION AND FUTURE WORK.....</b>	<b>70</b>
6.1 RECAPITULATION .....	70

6.2 FUTURE WORK.....	72
<b>APPENDIX.....</b>	<b>78</b>
APPENDIX A.....	79



## Table List

<b>Tables</b>	<b>Page</b>
Table 5.1 Classes needed in the new vitals saving data process.....	59
Table 5.2 Classes needed in the patient I/O data displaying process.....	60
Table 5.3 Labels table data corresponding to IV Type combo box .....	67
Table 5.4 Process time comparison between original and re-engineered versions.....	68

## Figure List

Figures	Page
Figure 3.1 Login interface.....	27
Figure 3.2 Patients List interface .....	28
Figure 3.3 Orders interface .....	29
Figure 3.4 Acknowledge Order interface.....	30
Figure 3.5 Acknowledge Consult Order interface .....	31
Figure 3.6 Meds interface .....	32
Figure 3.7 Acknowledge Medication Order interface .....	33
Figure 3.8 Vitals interface.....	34
Figure 3.9 New Vitals interface .....	35
Figure 3.10 Notes interface.....	36
Figure 3.11 New Note interface.....	36
Figure 3.12 Assessment interface. ....	37
Figure 3.13 New Assessment interface .....	38
Figure 3.14 Intake & Output interface.....	39
Figure 3.15 New I/O interface .....	40
Figure 3.16 Pain interface .....	41
Figure 3.17 Add Pain Data interface.....	42
Figure 4.1 Original window layout.....	45

Figure 4.2 Patient List interface and Patient Documentation interface .....	47
Figure 4.3 Filter in Notes Tab.....	48
Figure 5.1 Application of Model-View-Controller pattern in HPAD.....	55
Figure 5.2 Application of Template Method design pattern in HPAD.....	56
Figure 5.3 Application of Facade design pattern in HPAD .....	58
Figure 5.4 Windows taskbar with many items for an instance of HPAD.....	62
Figure 5.5 Windows taskbar with one item for an instance of HPAD .....	62
Figure 5.6 Table Column sorting example .....	63
Figure 5.7 IV Type combo box.....	66
Figure 5.8 MS SQL Query Analyzer showing labels table data.....	67
Figure A-1 HPAD class diagram, part 1 .....	79
Figure A-2 HPAD class diagram, part 2 .....	80

# 1. INTRODUCTION

It is impossible to create software, independently of its size, which does not need to be modified. Applications that have served the needs of a company during certain amount of time may become unstable due to the alterations, adjustments, and improvements realized on the software through time. This kind of systems are called Inherited/Legacy Information Systems (LIS), which generally are expensive to support because they lack the documentation necessary for the understanding of the details of the system and its code-debugging process is costly and consumes a lot of time of the software developer. The effort to support an application that has passed through a frequent process of "patching" implies evaluating the pros and cons of such process to determine if it is more favorable to realize a reorganization and modification of the existing code or if, on the contrary, the most advisable thing is to redesign the system from scratch.

Software re-engineering is the process of reorganizing and modifying existing software systems to make them easier to support, by restructuring or rewriting parts or the totality of a legacy system without changing its functionality [Satpathy02]. Re-engineering is applicable to systems in which the following situations take place:

- Frequent failures which are difficult to locate
- Efficiency level is not the maximum
- Difficulties in the integration with other systems
- Resistance to introduce changes
- Few persons qualified to realize modifications

- Difficulties to make tests
- Maintenance consumes many resources
- It is necessary to include new requirements, but to keep the basic ones.

Such situations are present in a nurse documentation software application developed at the University of Puerto Rico at Mayagüez by several research groups of the Human-Computer Interaction area. The development of the software has taken several years and it has involved more than 8 development groups. The system has passed through various requirement changes, usability studies, and performance evaluations that have generated many variations in the code. This makes this application a perfect candidate for a re-engineering process.

The importance of the HPAD project is based on the urgency of changing the way that the documentation of the medical information takes place, turning it from a paper-based to an electronic-based process. Nowadays, most of the hospitals in Puerto Rico and United States support the medical records on paper. The change to a medical electronic record avoids the accumulation of documentation that, in some cases, is repeated or redundant, and that occupy unnecessary physical spaces. Every hospital has big spaces destined to guard clinical histories that are difficult to locate later. The information is treated in an individual way and does not offer the possibility of being employed in order to analyze the information, compile statistics, or conduct epidemiology studies.

Other potential benefits of the medical electronic record are:

- Rapid access to information and precedents of patients that enter to the institutions unconscious without anyone who could alert about previous diseases, allergies, etc.
- More control on the medication and complementary studies and to know how the health resources are spent. There are known incidents of criminal procedures that are made with medicines, with medicine orders that are passed from a person to other one, and with medicines that are charged from beneficence societies without having being administered. Thousands of million dollars are spent in the health system for the lack of real controls [Gordon96].
- Allow the use of medical databases to researchers with statistical purposes.
- Able to know the evolution of diverse treatments and new diagnostic procedures. Allows access to information regarding adverse reactions of diverse medicines.
- Improved legibility in the patient's records. In many cases the clinical histories are illegible which may result in misinterpretation and mismanagement of patients care.
- Access to the records at any time and any place. Several services can access them at the same moment without having to move them from one service to another.

- Confidentiality and security of information. With current encryption systems it is more difficult to be able to extract information from a computer than to do it from a clinical history written in paper that travels weeks in the hospitals.
- Integrity in the information. With well constructed computer systems it is impossible to modify information if the user is not allowed. On the contrary, in clinical histories written in paper it is too easy to extract sheets and to replace them or to eliminate them definitively.
- Longevity of information. It is possible to keep the information stored during long time, without occupying too much space by means of storage media such as CDs/DVDs/Tapes and others.

## 1.1 Objective

The main objective of this project is to improve the original nurse documentation system, called HPAD, by applying re-engineering to obtain a software module with the same functionality and improved design. The modifications to be realized and improvements to be achieved are:

- Redesign of the Assessment module with the objective to improve the experience of the nurses using the interface
- Improvement in the speed of the processes involved with the database access
- Simplification of the code so that the class model used will be easier to understand for a later development groups.

- Improvement of the code to make the software modules easier to modify, debug, and test.
- Facilitate any future migration or upgrade of the platform (technology), the database, and operating system
- Modernization or updating of some components of the existing system
- Correction of the problems detected in the original system
- Change the original method of organization of the windows of the software to a Multiple Document Interface (MDI) that will allow the windows of the software to limit its area of action to the internal zone of a father window.

The methodology used to accomplish these objectives was divided in three phases. The first phase was to conduct a detailed analysis and evaluation of the original system. This involved the study of its deficiencies, disadvantages, and problems; an examination of the source code and its functionality; and the analysis of the object model used. The second step was the design phase, which involved the localization of the potential points of improvements; the evaluation of the consequences of those improvements and its impact on the final version; and the localization of the set of classes that might be used along with the modifications required in the process to make them reusable. The last phase was the implementation of a new version of the system based on the results of the previous phases. The new version was then evaluated and tested to verify if the requirements were achieved.



## **1.2 Project Outline**

The next chapter includes a literature review of previous works related to some projects using re-engineering for different kinds of software systems. In addition, it describes some projects which involve the managing of clinical information and that were the base for the nurse documentation system HPAD. Chapter 3 provides a description of the original system. The analysis of the key sections of the original system requiring improvements is presented in Chapter 4. An explanation of the design and implementation of the re-engineered system is discussed in chapter 5. Finally, the conclusions and future work are presented in Chapter 6.

## 2. PREVIOUS WORKS

### 2.1 Introduction

The high complexity and variability of the software for managing patients' electronic records, added to the peculiarities surrounding this kind of software, make these programs perfect candidates for re-engineering processes that transforms them into applications that satisfy the initial requirements and prepares them for future extensions and variations. This chapter reviews current literature on the technologies involved in the application of re-engineering processes in *legacy systems* (An *Inherited* or *Legacy System* can be defined as any information system that significantly resists to modification and evolution), giving emphasis to the use of software design patterns for the implementation of the necessary adjustments to improve its characteristics and/or capacity of maintenance. Also this chapter discusses the complexity and variability of the systems of electronic record in health institutions using several studies.

### 2.2 Re-engineering

There are several projects or studies related to the application of re-engineering methods to software systems with the objective of obtaining significant improvements. Many of these are based on the use of software design patterns to achieve their objectives. Design patterns treat the design problems that repeat themselves and are present in particular situations of the software design, in order to propose solutions to them. Therefore, design patterns are successful solutions to common problems.

In a research realized by Masuda G. [Masuda00], a case of study is described regarding the redesign of existing software using design patterns. The study quantitatively evaluates the effectiveness of applying design patterns to the redesigning process. The evaluation was based on the C&K metrics suite, which defines six metrics for object oriented design. These metrics have been used in various application domains and its effectiveness has been proven. The referenced study collected C&K metrics values for two versions of a system. One was a prototype version designed without design patterns, while the other one was designed using patterns. The results found significant differences between the different metrics of both versions. They quantified better results in the version designed with patterns. They also showed a successful case in which the use of design patterns, specially Builder, Template, and Strategy [Gamma95], improves flexibility and extensibility of a piece of software.

A similar study conducted by Balazinska, Merlo, and others [Balazinska99] made an exercise of partial redesign of a software written in Java language based on clone analysis. The project presents a new approximation to the redesign of systems developed in Java by means of factorizing the common parts of the cloned methods and parameterizing its differences using the "Strategy" design pattern. The new entities created by such transformations were extracted also from its original context, facilitating the reutilization and increasing the capacity of maintenance of the system. The results of the re-engineering activities conducted showed an increase in the total size of source code due to the numerous but simple methods created. The experiment also showed that it's possible to replace clones

with more maintainable structures totally automatically, i.e. without any input from users, nor any information on the system besides its source code.

A work made by Mancl, under Lucent Technologies sponsorship [Mancl01], presents some design techniques that a design team can use when they must confront the challenge of evolving existing software towards a new system. They demonstrate that the creation of a "wrapper" class that contains groups of functions extracted from the code to re-design is the simplest way of redesign. They also show more complex ways of redesigning by using other design patterns which help to improve the design. The article describes a real example of this approximation that shows how the re-engineering improved the design of software orientated to wireless networks.

The primary use of design patterns is associated with forward engineering and the design phase of the software life-cycle. However, in the work presented by Tahvildari and Kontogiannis [Tahvildari02], they examine design patterns from a different perspective, their classification and use for software re-engineering and restructuring. Specifically, twenty three design patterns originally presented in the "Gang of Four" book [Gamma07] are reclassified for re-engineering purposes into two major categories, primitive and complex. Their classification scheme is based on three primary relationships between patterns such as use, refine and conflict.

Use relationship appears when building a solution for the problem addressed by X, one sub-problem is similar to the problem addressed by Y. Therefore, the design pattern X uses the design pattern Y in its solution. Refine relationship is when a specific pattern X refines a

more abstract pattern Y if the pattern X must deal with a specialization of the problem the general pattern Y addresses, and must have a similar (but more specialized) solution structure. Finally, conflict relationship denotes that the two or more patterns provide mutually exclusive solutions to similar problems.

The author also establish three secondary relationships such as *similar*, which it is used to describe patterns which are similar because they address the same problem, *combine*, which it is the case where two patterns are combined to solve a single problem which is not addressed directly by any other pattern, and *require*, which appears when one pattern requires a second pattern if the second pattern is a prerequisite for solving the problem addressed by the first pattern; all three secondary relationships can be expressed in terms of the primary ones. The resulting classification can assist software engineers with : i) understanding better the complex relationships between design patterns, ii) organizing existing design patterns as well as categorizing and describing new design patterns, iii) building tools which support the application of design patterns during restructuring.

Another case in which re-engineering was applied to an existing system is the one presented by Satpathy and Siebel in [Satpathy02]. In the document they describe the case of the re-engineering of the People Tracking subsystem of a surveillance system written in C++. The original system was developed in a university environment. They discuss the problems, the challenges, and the approaches taken; describing how the module was re-engineered so that it could be used in the “real world”. The main observations of this case study were: i) Tool support or expert advice can aid in identifying that a piece of software is not

maintainable, to help in the decision whether a re-engineering process needs to be carried out. ii) Increasing the maintainability of software by re-engineering techniques makes the addition of new functionality more efficient. iii) Surprisingly, little domain knowledge is necessary for porting or re-engineering the software. Regarding the last observation, they indicate that domain knowledge is mainly needed to extract dynamic information from the code, and partially during the addition of new functionality. They state that the availability of design documents can, to a certain extent, make up for missing domain knowledge.

## **2.3 Electronic record systems**

There are several projects or studies related to the complexity and variability of computer-based patient record in hospitals.

The study conducted by Joan S. Ash [Ash97] from the Oregon Biomedical Information Communication Center, presented a survey of the perceptions of 629 informatics experts representing 67 accredited schools of medicine which identified the most important factors regarding the implementation of a Computer-Based Patient Record (CPR). The factors identified were:

- Innovation Attributes (attributes inherent in the CPR itself) such as voluntariness, image, ease of use, result demonstrability and visibility.
- Organizational Attributes such as communication, decision making, support, planning and rewards.

- Boundary-Spanning Attributes (related to marketing efforts) such as relative advantage, compatibility, generation of marketing intelligence, dissemination of marketing intelligence and responsiveness of marketing.

The survey concluded that the set of Innovation Attributes are important predictors of CPR infusion, with visibility as a significant variable. People need to see both the system itself and other people using the system. Also, it was noted that the system was viewed by many members as a threat to the values of the organization, and their responses to this cultural assault were predictable. It is normal that they feel uncomfortable with the change that an information system involves. Responses of this magnitude should be anticipated, and they must be managed.

The use of electronic patient record systems is essential in any healthcare organization that needs to integrate and centralize patient healthcare information. Several people, processes, and services are involved in this heterogeneous environment stressing the need for information security. Both the patients and healthcare organizations concerned can be seriously damaged if no proper security is provided. In a work conducted by some members of the Faculty of Medicine and Science of LIACC in Porto, Portugal [Ferreira04], a digital signature system was implemented and applied into a healthcare software system made at the Faculty of Medicine of LIACC. The module consists of a database of public keys properly protected, whose key pair sets needs to be changed regularly. Only one user of the system has its own pair and uses it to sign the software reports. They found that the application of digital

signature increases the security and the time spent to execute the processes of signing and verifying the digital signature is not significant and does not compromise system's usability.

A research conducted by Chiang Jao and others [Jao07] approached the subject of the challenges facing the adoption of an electronic charge capture system on a clinical information system. They present the hypothesis that information technology could improve the billing workflow. As part of the study, the researchers developed a semi-automated charge capture system assisted with information technology to improve the billing interface and promote physician productivity. The pilot system was successfully designed to assist physicians to improve productivity. They concluded that clearer changes in physician behavior and organizational management are needed to reduce administrative cost and prevent further errors. Although overall technical improvement may provide access to desired information, major barriers on data entry still hamper the efficient implementation and adoption of clinical information systems.

In one way or another, the previous works described have contributed to the development of the present project by means of the explanation of concepts and its application to real environments. They also demonstrated not only the advantages of the application of re-engineering in legacy software, but contributed with a vision of the problems that can arise in the process of conversion to modern software.



## **3. SYSTEM DESCRIPTION**

### **3.1 Introduction**

HPAD software is an application focused in the management of patient information in hospitals that offers support to the follow-up of the medical orders electronically. It supports the nursing documentation process and provides a mechanism for alerts and reminders.

The Physician and Nurse Documentation system, HPAD, is composed of both desktop and PDA versions. The work described in this project is focused on the redesign of the desktop version of the nursing documentation module. This module was researched and developed with the collaboration of several graduate students of the University of Puerto Rico in Mayagüez. It is based on the graduate work conducted by Gilberto Crespo [Crespo05] and Carlos Perez [Perez05] among others.

This chapter describes the original nursing documentation system, called HPAD, and the new version of the system developed in this project. Both versions are described from the point of view of user interfaces and development technologies.

### **3.2 Original System**

The original system was developed using the Java programming language and the JBuilder Developer environment, Tenth edition, from Borland Enterprise. The database server used is the MySQL Server 4.1.12 from MySQL Corporation. The system was developed applying the object-oriented development methodology.

Because of the graphical interfaces were not modified in the present project, they will be explained in more detail in the following section.

The group of researchers that has led the development of the HPAD project was specialized in the human-computer interaction and the application of usability principles in software design. Therefore, the project has had a strong emphasis in these aspects; providing a product with a high user satisfaction, efficiency potential (productivity level reached by expert users) in most of their tasks, and that exhibited a high degree of learnability (capability of a software product to enable the user to learn how to use it) [Crespo05].

### **3.3 Re-engineered System**

The Java programming language was used for the creation of the system, using the open source development environment Eclipse 3.2 and using as database engine Microsoft SQL Server 2000. The system uses the client-server model and it is prepared to be used in wired and wireless environments.

As explained in the previous section, most of the interfaces of the present version of the system were preserved (front-end). However, the business rules (back-end) were replaced by code developed from scratch that kept the same functionality. In this way, we improved the business rules and preserved the positive usability characteristics of the original version. For the Assessment Interface Module, on the hand, a total reconstruction was required.

The following sections describe the graphical interfaces that compose the desktop module of the nursing documentation system HPAD.

### 3.3.1 Login Interface

Since HPAD is a multiuser system, it is necessary to have a validation process of authorized users. The Login interface, shown in Figure 3.1, is the first window displayed upon activating the application. Nurses are required to enter their username and password to be validated.



Figure 3.1 Login interface

### 3.3.2 Patients List

Once the nurse is validated by the login process, the system will display the Patients List interface that lists all the patients in their corresponding ward. The top left section of the interface displays information about the nurse who entered into the system such as name, and registered number. The icons that appear to the right side of the patient's name and room

number represent reminders of pending tasks alerts associated with that patient. Figure 3.2 shows an example of this interface.

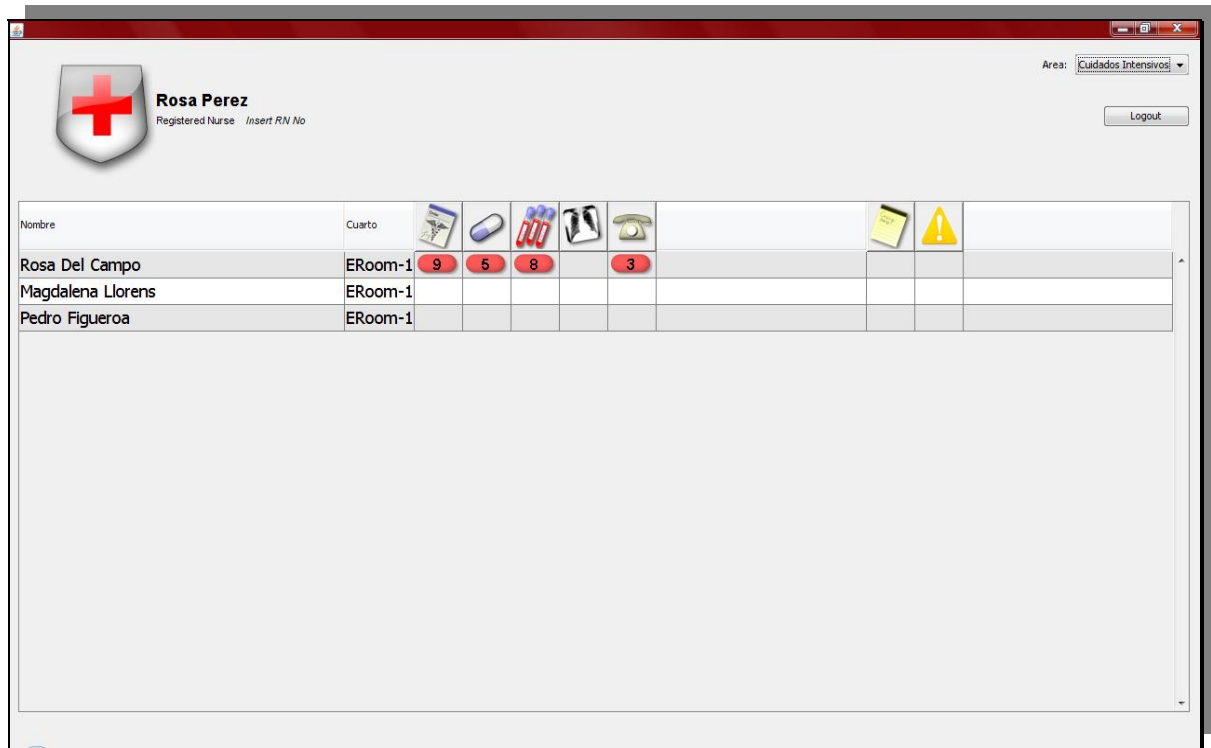


Figure 3.2 Patients List interface

### 3.3.3 Nursing Documentation Interfaces

When a nurse selects one of the patients from the patients list, by clicking on any cell of its row, the system opens the interface of nurses' documentation for that patient. This interface is composed by a header that provides general information of the patient. This includes: name, sex, age, weight, height, primary physician, account, and record number. On the right side of the heading, there are diagnoses fields, as well as some alerts and reminders representing special needs and precautions.

In the main section of this interface, the user can view all the record by navigating through a series of tabs (Orders, Meds, Drips, Vitals, Notes, Assessment, I/O (Intake/Output), and Pain) used to view and manage the clinical information of the patient.

### 3.3.3.1 Orders Interface

The Orders interface (See figure 3.3) displays all the medical orders generated by the different physicians involved in the patient care. The orders are classified by type, physician, status, and creation date. The interface is divided in two sections; the left section lists the orders submitted for the patient; the section on the right shows the details corresponding to any order selected in the right section. This layout is used in most of the interfaces of the nursing documentation system.

**Patient - Rosa Del Campo**  
 Female 52yrs 180lb 129in  
 ERoom-1 Dr. Méndez  
 Account No: 1 Record No:1 Blue Cross

Allegies: Penicillin

Date	Type	Physician	Status
Mar/10/06 5:13PM	Restraint	Jose Rios	Pending
Mar/10/06 5:11PM	Laboratory	Jose Rios	Pending
Mar/9/06 4:31PM	Laboratory	Jose Rios	Pending
Apr/2/07 8:00AM	Consult	Jose Rios	Pending
Apr/1/07 8:00AM	Exam	James Mendez	Pending
Apr/1/07 7:55AM	Laboratory	Jose Rios	Pending
Mar/29/07 11:30PM	Admission	Luis Rivera	Pending
Mar/29/07 11:30PM	Diagnostic	Luis Rivera	Pending
Mar/29/07 11:30PM	Vital Sign	Luis Rivera	Pending
Mar/29/07 11:30PM	Diet	Luis Rivera	Pending
Mar/29/07 11:30PM	Activity	Luis Rivera	Pending
Mar/29/07 11:30PM	Laboratory	Luis Rivera	Pending
Mar/29/07 11:30AM	Laboratory	Luis Rivera	Pending
Mar/29/07 11:30AM	Laboratory	Luis Rivera	Pending
Mar/29/07 11:30AM	Laboratory	Luis Rivera	Pending
Mar/29/07 11:30AM	Laboratory	Luis Rivera	Pending
Mar/29/07 11:30AM	Generic	Luis Rivera	Pending
Mar/29/07 11:30AM	Consult	Luis Rivera	Pending
Mar/29/07 11:00AM	Consult	Jose Rios	Pending
Mar/29/07 11:00AM	X-Ray Study	Jose Rios	Pending

**Details:**

**Admission**  
 Ordered on Mar/29/07 11:30PM by Luis Rivera

Order Comment: **Admit to internal medicine ward under Dr. Jose Rios service.**

Acknowledge

Figure 3.3 Orders interface

The status of an order displayed on the right section of the interface changes according to the actions taken by the nurses regarding that particular order. Once an order is attended by the nurse, the nurse can press the Acknowledge button which opens the Acknowledge pop-up window (See figure 3.4). This window includes a summary of the order and a textbox that allow nurses to enter any comment about the procedure taken. Once the order is acknowledged, the order status in the database is changed from “Pending” to “Acknowledged”.

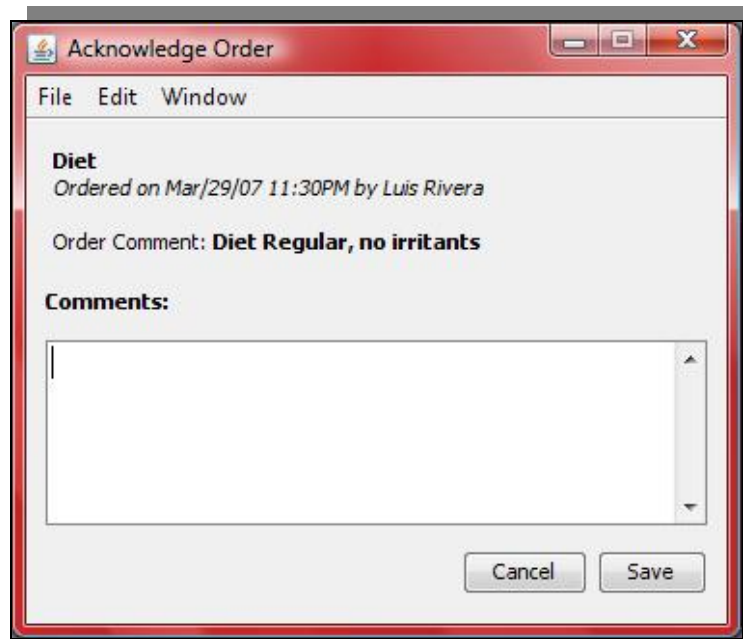
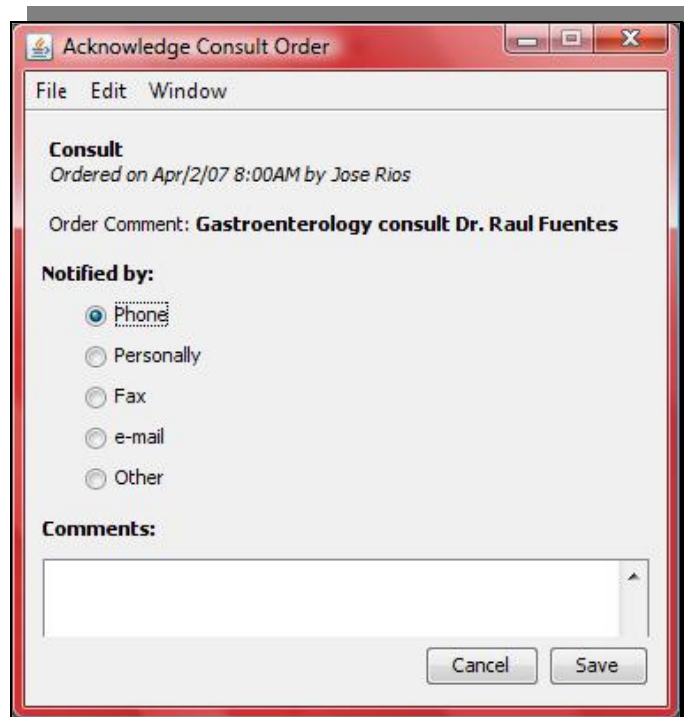


Figure 3.4 Acknowledge Order interface

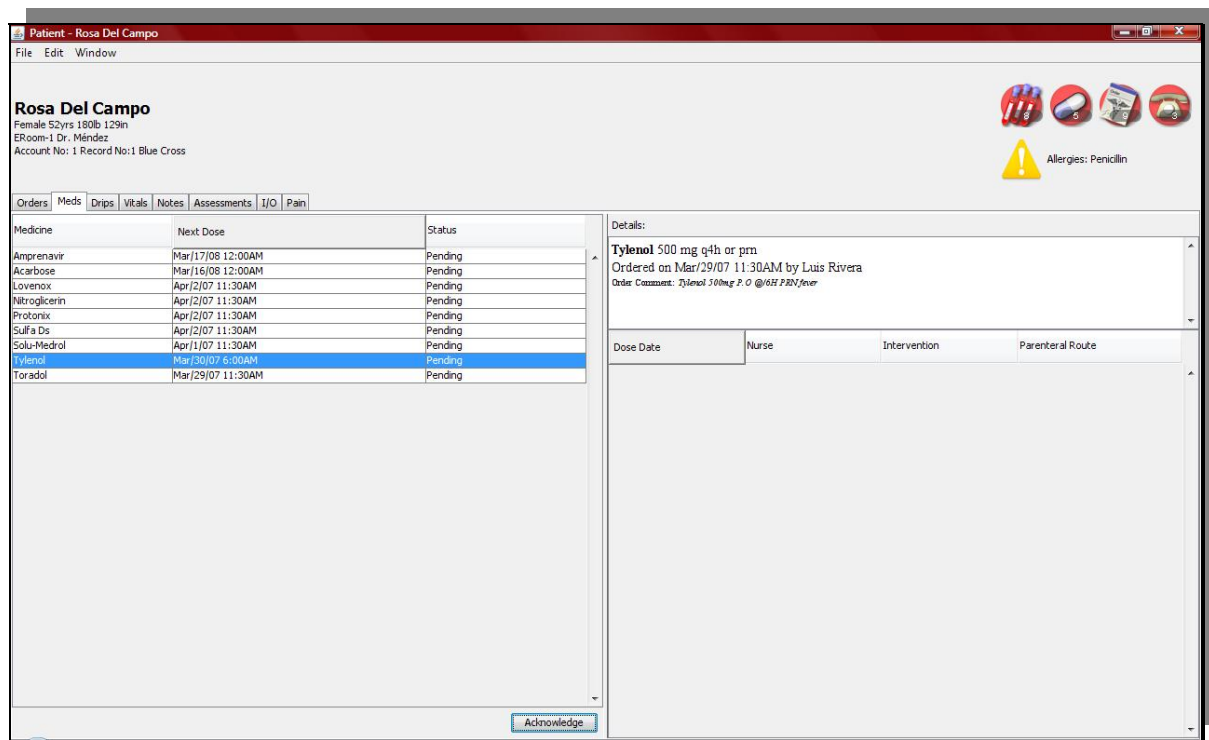


**Figure 3.5 Acknowledge Consult Order interface**

The Acknowledge window displayed depends on the type of order selected. For example, Figure 3.5 illustrates the Acknowledge window displayed if a consultation order is selected.

### **3.3.3.2 Meds Interface**

The Meds interface (See figure 3.6) is used to display the list of medications prescribed to the current patient, classifying them by status and next dose date. This interface shows a list of the medications orders on the left section of the form. Its behavior is very similar to the Orders interface. Once a specific medication order is selected, the right section of the form shows the details of the order and its administration history.



**Figure 3.6 Meds interface**

When a nurse presses the Acknowledge button for a medication order, an Acknowledge window is displayed (See figure 3.7) which provides a summary of the medication order, a textbox to enter comments, and other graphical objects to specify the details regarding the administration of the medicine.



**Acknowledge Medication Order**

File Edit Window

**Details:**

**Tylenol** 500 mg q4h or prn  
 Ordered on Mar/29/07 11:30AM by Luis Rivera  
 Order Comment: Tylenol 500mg P.O @/6H PRN fever

**Intervention:**

☒ Administered      Parenteral Route: Right Deltoid      Amount:  mL  
☐ No Administered      Reason:

**Comments:**

Cancel Save

Figure 3.7 Acknowledge Medication Order interface

### 3.3.3.3 Drips Interface

Drips interface is identical to Meds interface and works the same way. It is provided to separate the orders of drips medications from other medications orders.

### 3.3.3.4 Vitals Interface

The Vitals interface (See figure 3.8) is used to enter new vital signs for a patient and to provide a general sight of the vital signs behavior of the patients through time, by means of both a grid display and a graphical display. The left section presents a list of vital measurements taken: temperature (Temp), blood pressure (BP), pulse, respiration rate (Resp),

O2 saturation (O2Sat), and weight. The right section displays a graphical representation of the vitals measurements specified by the checkboxes at the bottom.

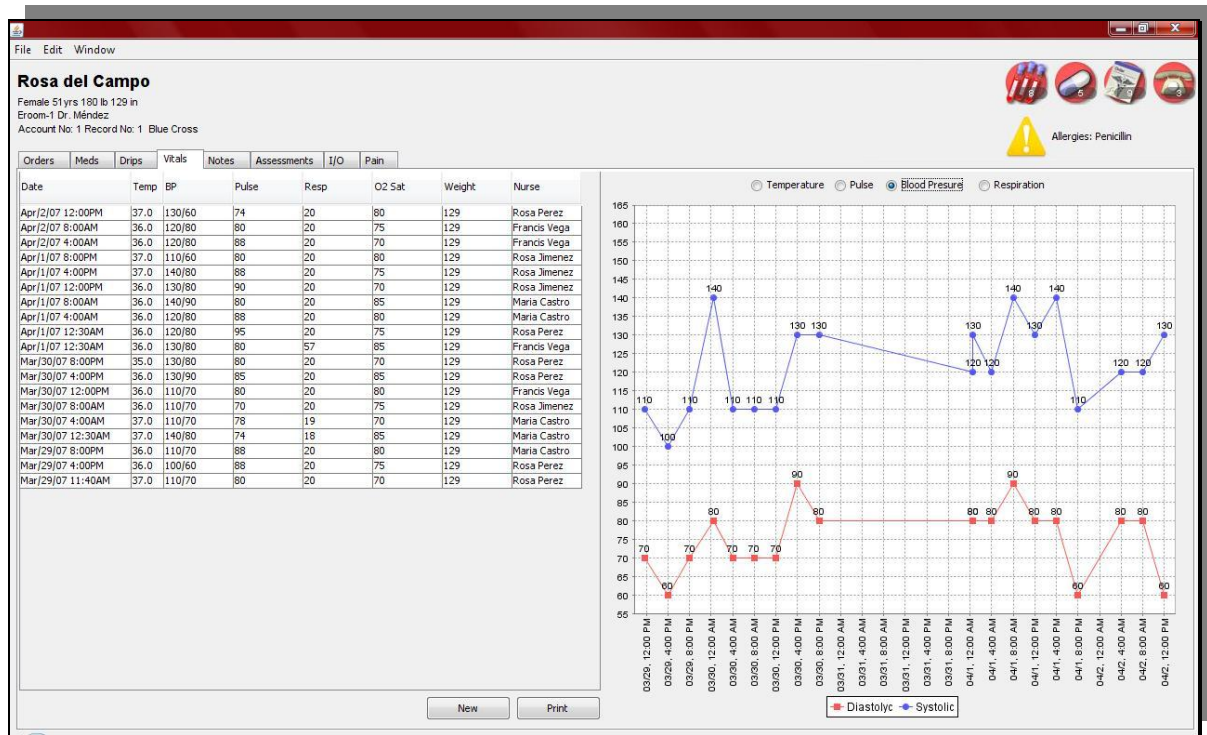


Figure 3.8 Vitals interface

If the New button is clicked, the Add Vital window is opened (See figure 3.9). This window allows the nurse to enter a new set of vital measurements. All the quantities can be entered using the keyboard or using the on screen keyboard provided in the window. Once the new vital signs are saved they are displayed on the list of the left section.

**Vitals Measurement**

Temperature:  °C

Systolic BP:  mmHg

Diastolic BP:  mmHg

Pulse:  BPM

Respiration:  RPM

O<sub>2</sub> Saturation:  %

Weight:  lbs

1 2 3

4 5 6

7 8 9

. 0 Del

Tab

Save Cancel

**Figure 3.9 New Vitals interface**

### **3.3.3.5 Notes Interface**

The Notes interface (See figure 3.10) allows nurses to access and view notes generated for the current patient. The left section of the interface displays a list of the notes written by nurses and doctors regarding the patient. This list specifies the user who created the note, the date, and the type of note. If a note of the list is selected, then the contents of the note are displayed in the right section of this interface.

To create a new note, the user must click the New button. As a result the New Note's window is displayed (See figure 3.11). Once the user enter the type of note, the title of the note, and the contents of the note, he/she can click the Save button to submit the note.

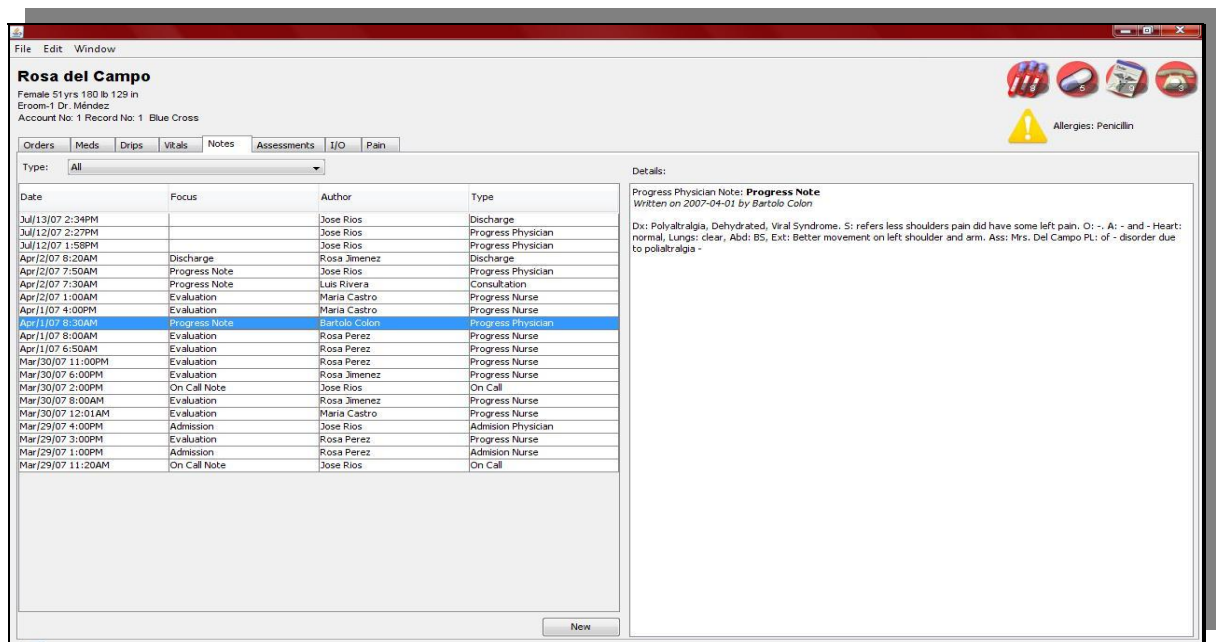


Figure 3.10 Notes interface

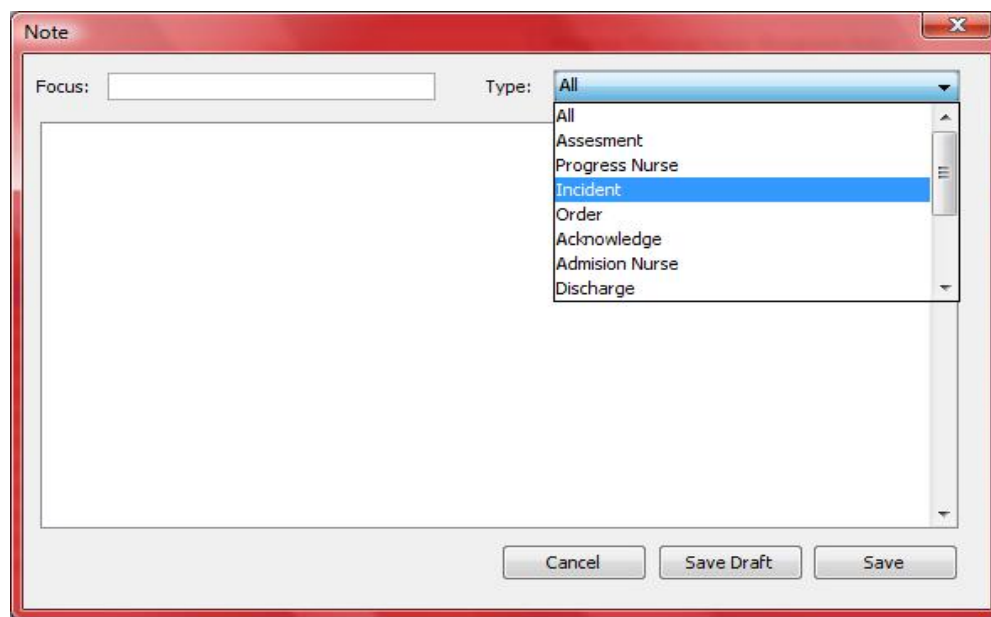


Figure 3.11 New Note interface

### 3.3.3.6 Assessment Interface

The Assessment interface is used for the daily assessments of the patients and to view the previous assessments entered. The left section of the interface displays a list of the different assessments taken from the patient. The list includes the date and the name of the nurse that entered the data. If the user selects a row from the table, then in right section of the interface is displays the details of the assessment selected. Figure 3.12 shows the Assessment interface.

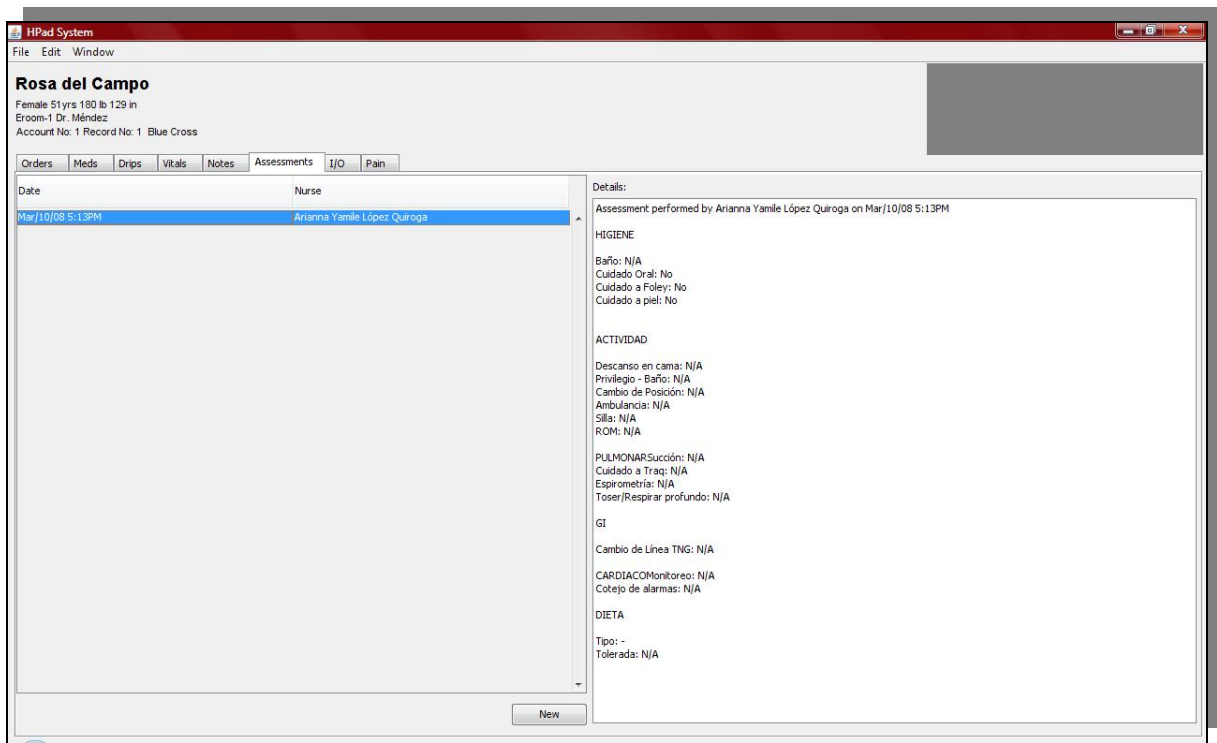


Figure 3.12 Assessment interface

If the New button is clicked, then the Add Assessment window, shown in Figure 3.13, is displayed. This window allows the nurse to enter a new set of assessment data for the patient. The data is entered using a table which has three columns: *Item*, *Value* and

*Other/Comments*. The *Item* column represents the issue that is being evaluated. The *Value* column is used by the user to enter the measured data, which is entered using a combo box, a text box, or radio buttons (according to the item). The *Other/Comments* column allows the nurse to enter any additional data or comments. Once the new assessment data is fully entered, the user can click the Save button in order to store the data to the database.

The screenshot shows a window titled "New Assessment" with a red header bar. Inside, there is a dropdown menu labeled "Area:" with "Piel" selected. Below this is a table with three columns: "Item", "Value", and "Other/Comments". The table contains the following data:

Item	Value	Other/Comments
Temperatura	Tibia	
Edema	<input checked="" type="radio"/> Yes <input type="radio"/> No	
Color	Palida	
Humedad	Hidratada	
Integridad	Laceraciones	
Vendaje	Drenando	

Below the table is a large empty text area for additional comments. At the bottom right of the window are two buttons: "Save" and "Cancel".

Figure 3.13 New Assessment interface

### 3.3.3.7 Intake/Output Interface

The I/O interface shows the list of I/O's measurements taken to the patient (See figure 3.14). The list includes the quantity (ml), a symbol indicating whether it was input or an output, and the clinician that entered the data. The user can filter the information displayed

by type or by a period of time. The total amount of input, output, and balance for the list of measurements displayed are indicated at the bottom of the panel.

**Patient - Rosa Del Campo**  
 Female 52yrs 180lb 129in  
 ERoom-1 Dr. Mendez  
 Account No: 1 Record No:1 Blue Cross

Allegies: Penicillin

Orders | Meds | Drips | Vitals | Notes | Assessments | I/O | Pain

Show: ☒ All ☐ In ☐ Out Type: All Types Period: All Measurements

Date	Type	Dir	mL	Nurse
Apr/2/07 12:30PM	IV		200.0	Rosa Jimenez
Apr/2/07 12:30PM	Other		320.0	Rosa Jimenez
Apr/2/07 6:23AM	IV		102.0	Maria Castro
Apr/1/07 10:25PM	IV		52.0	Maria Castro
Apr/1/07 10:25PM	Other		240.0	Maria Castro
Apr/1/07 3:40PM	Urine		600.0	Rosa Perez
Apr/1/07 3:40PM	Other		600.0	Rosa Perez
Apr/1/07 3:40PM	IV		750.0	Rosa Perez
Apr/1/07 6:35AM	IV		1100.0	Rosa Perez
Mar/30/07 10:31PM	Urine		600.0	Rosa Jimenez
Mar/30/07 10:31PM	Other		800.0	Rosa Jimenez
Mar/30/07 10:31PM	IV		1000.0	Rosa Jimenez
Mar/30/07 2:30PM	Other		420.0	Rosa Jimenez

In: 8264.0 mL Out: 2600.0 mL Balance: In 5664.0 mL

New

**Figure 3.14 Intake & Output interface**

To create a new I/O, the user must click in the New button to open the New I/O window (See figure 3.15). Once the user has provided information about the direction (intake or output), type, and quantity, the nurse must click the Save button to submit the information. The quantity can be entered using the keyboard or using the on screen keyboard provided in the window.

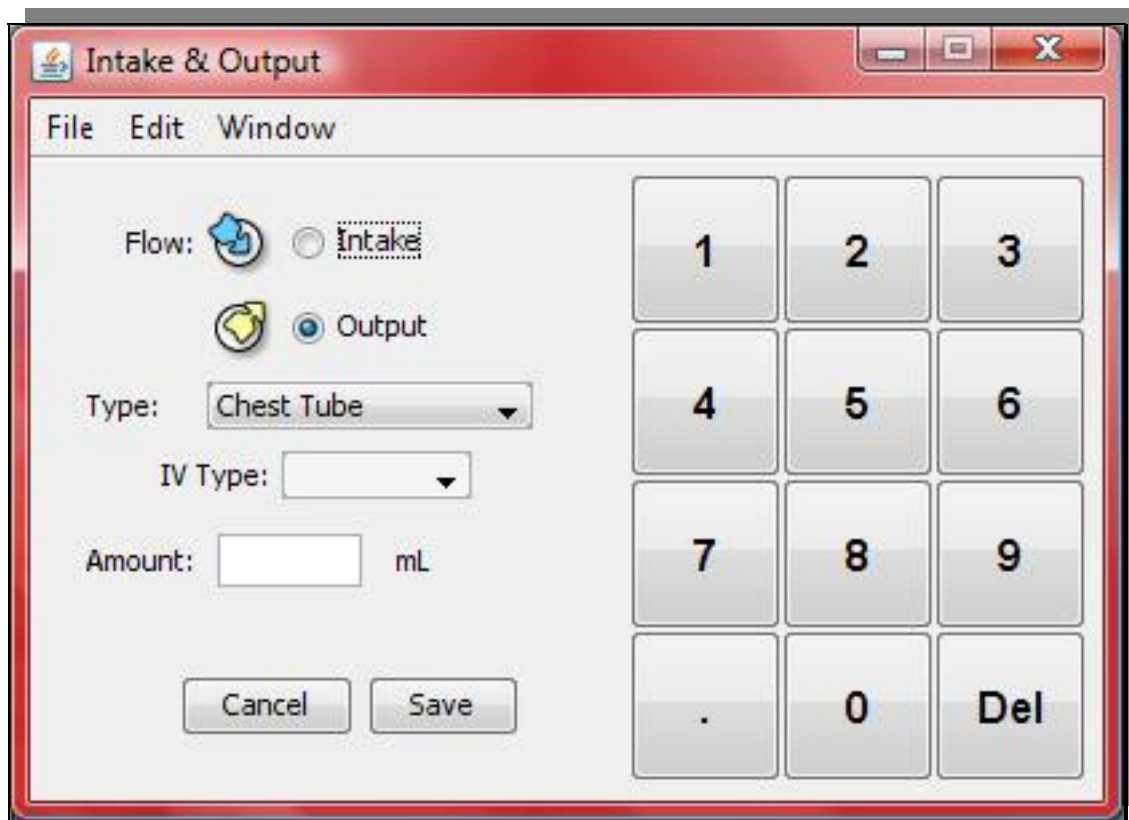


Figure 3.15 New I/O interface

### 3.3.3.8 Pain Interface

The Pain interface was developed to allow nurses to register the information related to the pain of the current patient and to let the user to see clearly its evolution by means of the use of drawings representing the intensity of the pain. In the left section of Pain interface a list of all pain assessments are displayed (See figure 3.16). This list specifies the user that conducted the pain assessment, the date, and the classification of the pain. On having selected a pain assessment of the list, in the right section it is possible to observe detailed information about the selected item.











**Patient - Rosa Del Campo**

File Edit Window

**Rosa Del Campo**  
 Female 52yrs 180lb 129in  
 ERoom-1 Dr. Méndez  
 Account No: 1 Record No:1 Blue Cross

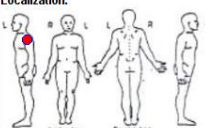
Orders | Meds | Drips | Vitals | Notes | Assessments | I/O | Pain

Date	ID	Classification
Jun/30/07 3:45PM	B	
Mar/30/07 11:35PM	B	
Mar/30/07 7:55AM	B	
Mar/29/07 11:50PM	B	
Mar/29/07 4:00PM	B	
Mar/29/07 1:00PM	B	
Jan/7/07 3:18PM	B	
Jan/7/07 7:33AM	B	

Follow-up New

**Details:**

**Localization:**



**Description:** Ardor, Angustante,  
**Therapy:** Ninguna  
**Medication Administered:** Yes  
**Comments:** Hombro y brazo izquierdo  
**Nurse:** Maria Castro

**Allergies:** Penicilin

**Figure 3.16 Pain interface**

To create a new pain assessment, the user must click in the New button. The New Pain window is displayed for nurses to conduct the assessment (See figure 3.17). The window provides options for specifying the pain level, its location, its description, the treatment being used to lower the pain, and additional comments regarding the pain. The pain location can be specified by pointing and clicking on the specific area of the body image presented at the upper right corner.

**Pain**

File Edit Window

**ID:**

Clasification:

0 1 2 3 4 5

Localization:

Anterior Posterior

Clear Point

**Description**

☐ Opressivo    ☐ Frio  
☒ Ahogamiento    ☐ Caliente  
☐ Lastimado    ☐ Ardor  
☐ Comezón    ☐ Calambre  
☐ Adormecido    ☐ Intermitente  
☒ Punzante    ☐ Continuo  
☐ Angustiante

Clear All

**Treatment:**

Therapy:

Med Administered: ☐ Yes ☒ No

Comments:

Cancel Save

**Figure 3.17 Add Pain Data interface**

The original nursing documentation system HPAD is a software that meets their objectives in terms of functionality, so the change of graphical interfaces is not necessary, but because it has been developed with tools whose versions are regarded as obsolete at this point, it becomes necessary the migration of these tools and a more thorough analysis of the system to define what are the key points of the re-engineering process to implement.

## **4. ANALYSIS**

The analysis phase of the software development process is used to study the needs that final product must cover by means of the exploration of both, the existing problems in the original software, and the new characteristics that want to be implemented. This chapter describes the problems found in the original nurse documentation system and discusses the aspects that can be improved. This represents the main objective of the present project.

### **4.1 Database Engine**

The original software uses the open source database MySQL Server version 4 from MySQL Corporation (now Sun Microsystems property). The principal characteristic of this database engine is that it is multiplatform and designed to being employed with databases of average size (10-100 million rows, or about 100 MB per table) on small computer systems.

The target market of the nurse documentation system HPAD are hospitals and clinics, which handle quantities of information that easily can exceed the limits that the manufacturer of MySQL Server suggests. If the hospital that uses the system grows and the limit recommended by the MySQL manufacturer is exceed, the effect observed by users will be a slow response while working with the software that inevitably will become in a complete collapse of the system.

Equally, thinking about the future development of the HPAD system, it is advisable that the database engine supports advanced characteristics like full support for cursors,

complete views, stored procedures, and foreign keys. It is necessary to clarify that neither the original version, nor the re-engineered version use such characteristics, but it is advisable for later versions.

From the implementation point of view, the method used to access the database is by means of a “jdbc” driver provided by the manufacturer of MySQL Server and a Java Class of initial connection that exposes an instance of the object `java.sql.DriverManager.Connection`. The sentences of SQL language to realize selections, insertions, updates, and elimination of information pass directly to the object `java.sql.DriverManager.Connection` with the consequence that the syntax of the sent commands must be specific for MySQL. The problem with this approach is that the connection is limited to this database engine exclusively.

A change of the database engine would imply checking the whole source code, replacing the connection object with the one that is needed, and transforming the specific MySQL SQL instructions for the required SQL sentences adapted for the new engine. This is not a simple or rapid process. Changes in source code on a system that has already been put in place are much more complicated than to make these changes at any previous development stage.

## 4.2 Window Layout

The window layout of the original system was designed so that whenever the system needs to show a new interface, it creates a window (child) that the operating system treats independently from the window that calls it (parent). The main problem with this approach is that the operating system task bar can be easily filled with different items. As a consequence, the user may be confused or can start to think that those items belong to applications different to the HPAD system. When there is not a modal window system, that is, when the child window does not allow the application to change its focus, the user might open the same window many times.

The user can get confused with this quantity of items on the taskbar and, more critically, the duplicated windows can lead to possible consistency problems in the information by allowing the user to save the same information several times (See figure 4.1).

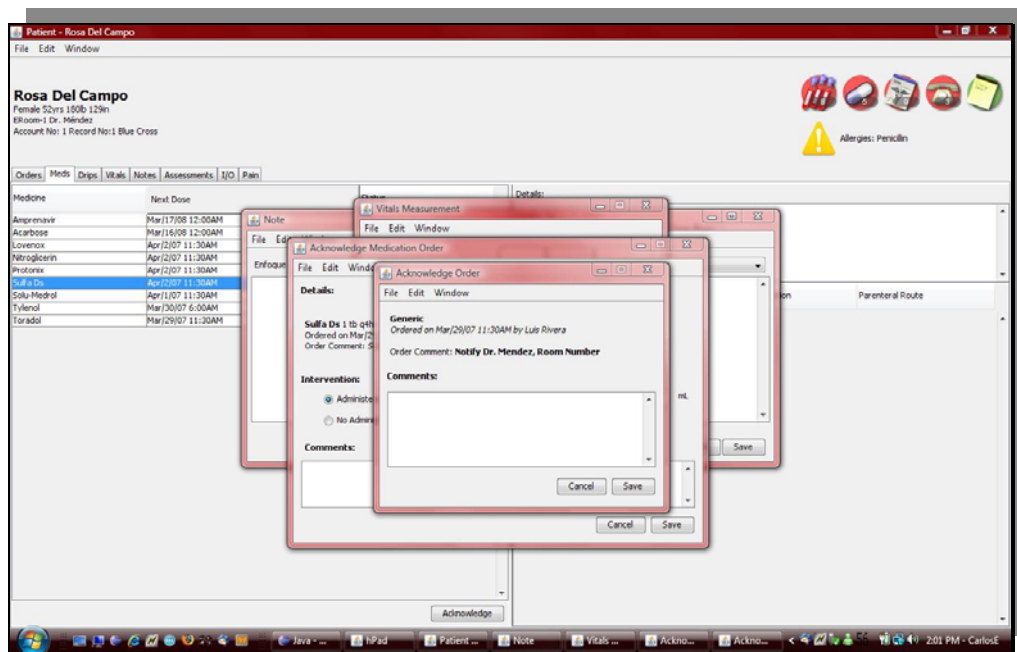


Figure 4.1 Original window layout

### **4.3 Application Speed**

One important aspect that a developer must take in consideration when creating a new software application is its speed. The present version of HPAD is limited or inadequate in that aspect. For example, the time it takes to load the information between the Patients List interface and the Nursing Documentation interface is 9.23 seconds in average with a wireless connection to the database server (See figure 4.2).

This time is considered excessive in a computer system, especially if we keep in mind that in the test, the HPAD application loaded only the information of one test patient which implies the load of about 100 records associated to him. In a real hospital environment, both the number of patients stored in the HPAD database and the number of records associated with each patient will be significantly higher. Thus, this will increase the processing times taken to load the information of the Nursing Documentation interface.

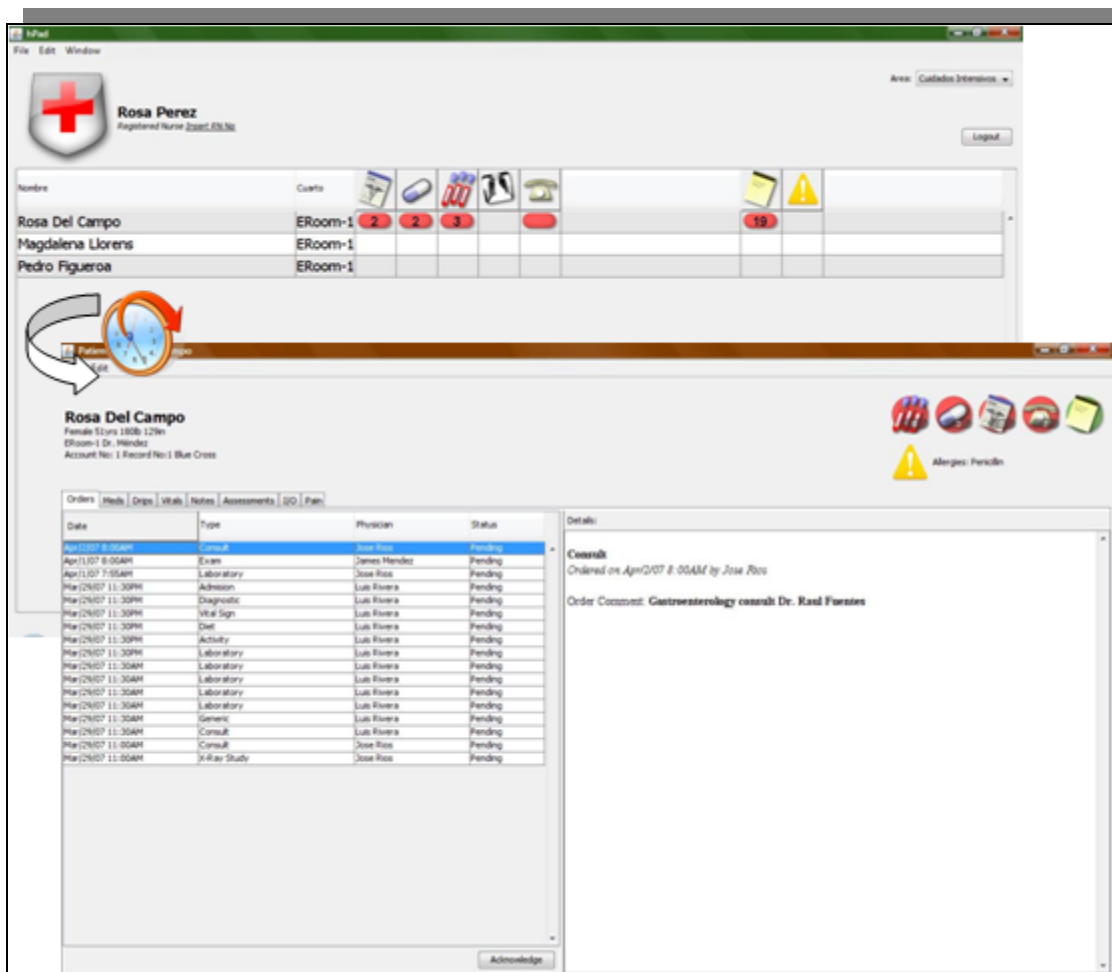


Figure 4.2 Patient List interface and Patient Documentation interface

Another example of inadequate processing time is found in the Notes and IO tabs of the Nursing Documentation interface (See figure 4.3). Both of these interfaces use combo boxes in order to filter lists according to several parameters. For example, the Notes tab has a combo box named Type which is used to filter the list of notes of the current patient based on the following criteria: assessment, progress note, incident, etc. This filtering process shows abnormal slowness.

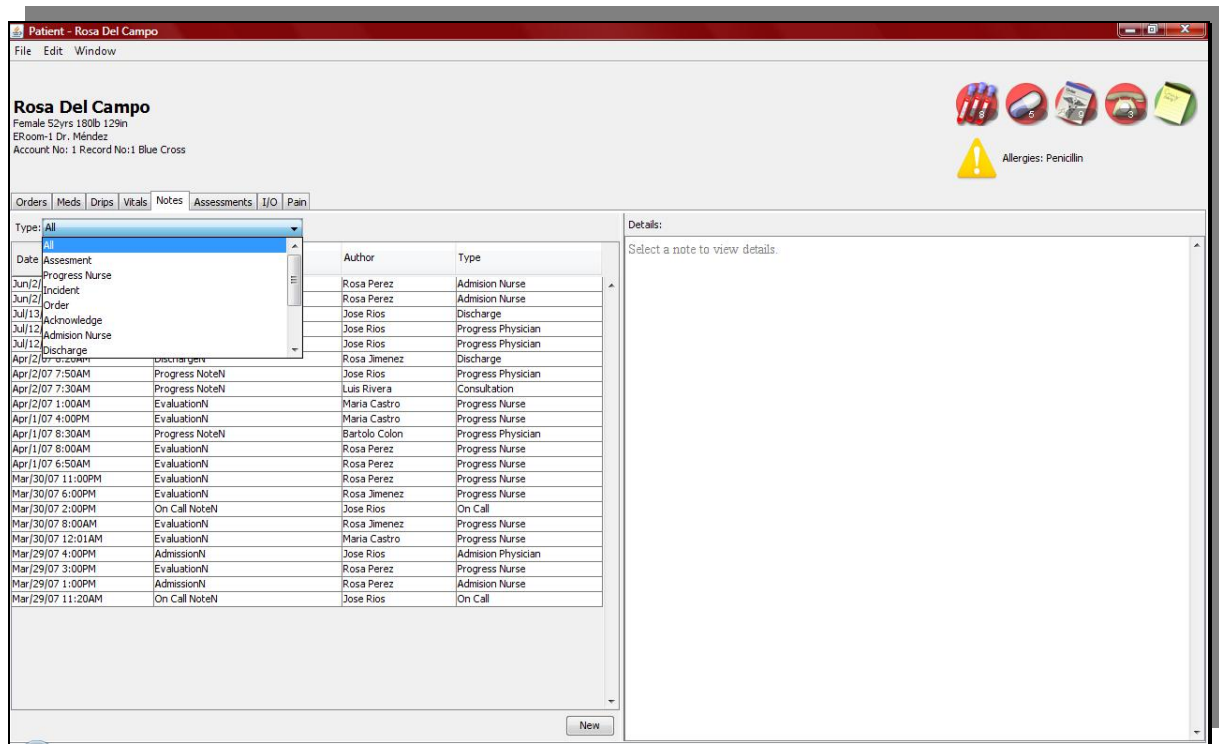


Figure 4.3 Filter in Notes Tab

The filtering process is executed by making a request to the database specifying the parameters of the filter. This procedure is realized whenever the user selects an item in any of the filtering combo boxes the application provides (Notes or IO tabs). This method of record selection in connected systems via wireless or that possess a high quantity of stored records is considerably expensive in terms of time and bandwidth consumption of the transmission media.



## 4.4 Third-party Software

The original HPAD application uses two types of Java classes that were designed by external sources: mysql-connector-java version 3.1.11 and jfreeChart version 0.9.21. The Java Database Connectivity “jdbc” driver mysql-connector-java is provided by MySQL Corporation and it is used to achieve the connection between the applications developed in the Java language and the MySQL Server database engine. At this moment this library is declared as deprecated by the manufacturer and has been replaced by the jdbc library MySQL Connector/J version 5.1.

The new version of this library solves several security problems, gives support to XML processing, provides support for setting per-connection client information, and support for JDBC-4.0 NCHAR, NVARCHAR and NCLOB types that previous versions were not supporting.

Another instance of the use of third-party software in HPAD is found in the library jfreeChart. The Vitals interface in the original system displays a graphical representation of the following vitals: temperature, pulse, blood pressure, and respiration. These graphs are created using the version 0.9.21 of the jfreeChart library. These set of open source Java classes is distributed under GNU license, and its main objective is to facilitate the dynamic creation of graphs to Java software developers.

The main problem of version 0.9.21 of the library is that it contains a bug that causes it not to clean the memory completely after the graphics were generated and eventually might freeze the computer where the program is running. In addition, the performance of the screen drawing process in version 0.9.21 is not optimal, which becomes a limitation when bigger graphics would be needed in a future version of HPAD.

## **4.5 Object-Oriented Implementation**

One of the important findings uncovered during the analysis process of the source code of the original system was extreme fragmentation of the classes that compose HPAD. The tasks implemented by the software modules have been subdivided in a lot of small parts that makes it very difficult to debug or to trace the code of a process. The quantity of classes that are called for each task makes it hard for the developer to understand the logic of the processes executed.

In total, the application uses 98 classes to manage all its functions (without counting libraries provided by the Java Development Kit) showing an excessive fragmentation of the tasks.

## 4.6 Other Issues

Some other problems or limitations that have been found are:

- Some lists are fixed or hardwired in the Java code of the program HPAD. In the event that any of these options would need a modification, then an entire deployment process would be mandatory. For example, the class *MedAckWindow* includes the lists of items to be displayed in the Combo-boxes of *Parenteral Route* and *Reason*.

The source code used for this process is as follows:

```
String[][] parenteralsString = { {"W", " "}, {"N", "N/A"}  
, {"A", "Right Deltoid"}  
, {"B", "Left Deltoid"}  
, {"C", "Right Dorsogluteal"}  
, {"D", "Left Dorsogluteal"}  
, {"E", "Right Ventrogluteal"}  
, {"F", "Left Ventrogluteal"}  
, {"G", "Right Vastus Lat"}  
, {"H", "Left Vastus Lat "}  
, {"I", "Right Rectus Temoris"}  
, {"J", "Left Rectus Temoris"}  
, {"K", "Right Abdomen"}  
, {"L", "Left Abdomen"}  
};  
String[][] reasonString = { {"W", " "}, {"N", "N P O"}  
, {"D", "Dx Study"}  
, {"G", "GI Disturbance"}  
, {"O", "OR"}  
, {"A", "Adverse Reaction"}  
, {"R", "Reject By Patient"}  
, {"T", "Not Needed"}  
, {"M", "Medical Order"}  
, {"P", "Physical Therapy"}  
, {"I", "Out of Institution"}  
, {"C", "According to Norm"}  
, {"S", "In Process"}  
};
```

In cases when a change is required on the items of these lists, it is necessary to update the definition of the arrays in the source code, compile the complete code of the software module, and copy the new file in the computer of every user of the system. This procedure is unacceptable in the nurse documentation process because it would

require putting computers offline in order to have the update process done. In a hospital environment, there would probably be many people who would see their work temporarily suspended. Of course, many risks could be associated with such situations.

- The implementation of insertions and modifications of database records is made by means of *Updatable ResultSets*. Once these are brought from the database, they can be modified in memory through the *ResultSet* object and returned to the database for its later storage. The main problem with this approach is that they impose certain restrictions. For example, the query of an *Updatable ResultSet* must specify the primary key as one of the selected columns and to select from only one table. Clearly, to oblige the developer to work with only one table at a time is not the best way to optimize database accesses. This forces various database calls for one task which affects the performance of the applications.

*Updatable ResultSets* do not behave consistently for all database engines. For example, the query "SELECT \* FROM my\_table" working with the MySQL jdbc driver returns a *ResultSet* that behaves standard (updatable), but if the driver used is the MS SQLServer jdbc it returns a read-only *ResultSet*. This may be a rather complicated problem when the application administrators want to migrate to another type of database because if they are using a driver that returns read-only *ResultSets*, the application will cease to operate as intended and will have to make the respective changes in the data access logic.

It is clear that the number and magnitude of the problems found along the analysis stage deserve to be reviewed so that at the end of the development process of the re-engineered version the result is an application that meets the objectives set forth in this work.

## 5. SYSTEM DESIGN AND IMPLEMENTATION

The previous chapter described some of the problems present in the original nurse documentation system. It also discussed the aspects that should be improved to have a stable product that would be easy to test, portable, extendable, and at the same time, to preserve its usability characteristics. The objective of this chapter is to detail the ways in which every aspect was addressed and managed from the point of view of software re-engineering.

### 5.1 Object-oriented Design

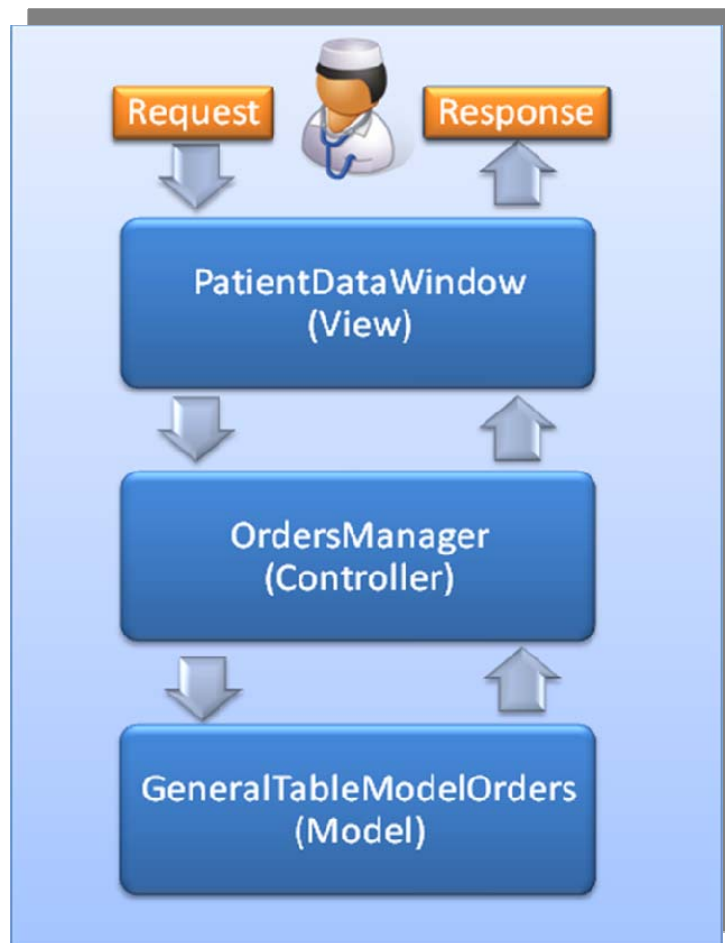
The re-engineered system was developed using the object-oriented methodology with a strong emphasis in the use of design patterns. As described in Chapter 2, design patterns are solutions proved to common problems in software development.

*"A well constructed object-oriented architecture is full of patterns. The quality of an object-oriented system is measured for the attention that designers have given to collaborations between its objects. Patterns lead to smaller, simpler and more understandable architectures"* (Grady Booch) [Booch07]

The most used design pattern in this project was the *Model-View-Controller* pattern. This pattern separates the information of an application, the user's interface, and the control logic into three different components.

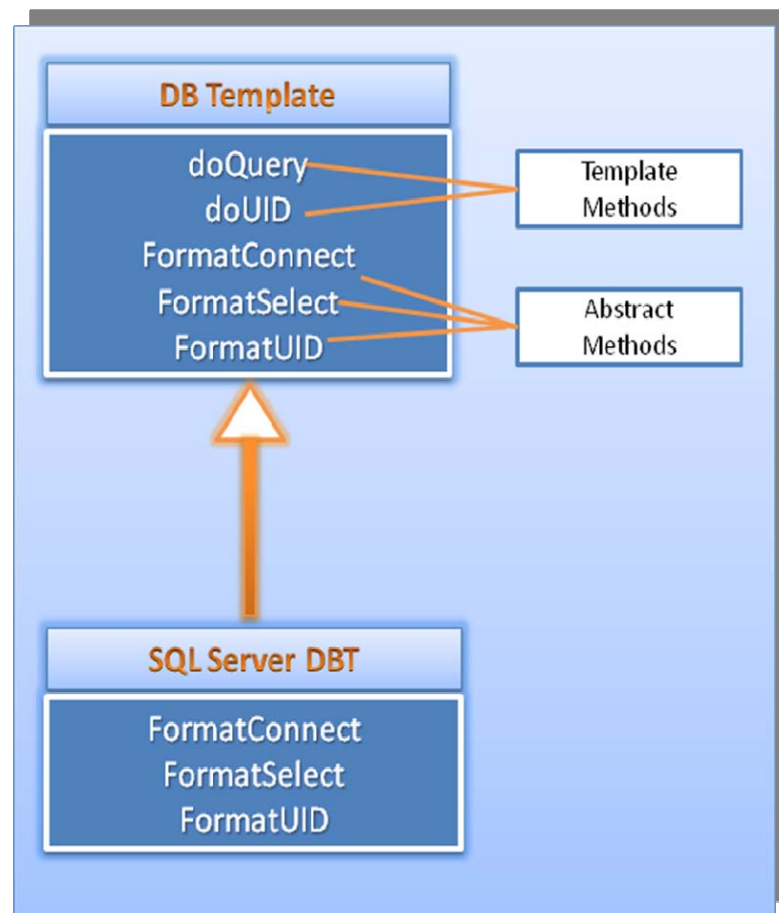
- **Model:** This represents the classes that define and include the information that the system manages.
- **View:** These are the classes represent model in a format adapted to interact, usually the user interface.
- **Controller:** This represents the classes that responds to events, usually user actions, and invokes changes in the model and probably in the View.

An example of the application of this design pattern can be seen in the figure 5.1:



**Figure 5.1** Application of Model-View-Controller pattern in HPAD

Another pattern used in the redesign process was the *Template* pattern. According to the Gang of Four book, the intent of this pattern is “to define the skeleton of an algorithm in an operation, deferring some steps to subclasses” [Gamma07]. This pattern was used in the present project for the database connection process. The Figure 5.2 illustrates how the *Template* was implemented in the nurse documentation application.



**Figure 5.2 Application of Template Method design pattern in HPAD**

The main advantage of using the template pattern in this way is that it facilitates future migration to another database engine because it is only necessary to create a class *NewSQLEngine* for the new engine, which extends from the abstract class *DBTemplate*,

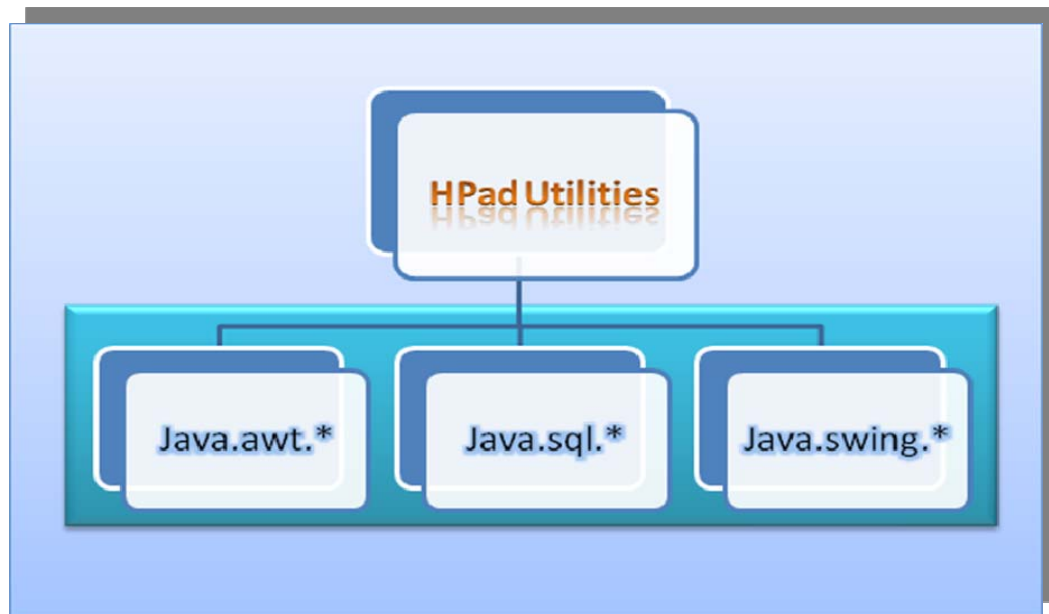


implementing the following methods: *FormatConnect* (to execute the connection), *FormatSelect* (for data recovery) and *FormatUID* (for updates, insertions and erasing data).

With this approach it is not necessary for the developer to go around looking in the source code and changing the lines that correspond to database access. Thus, the process is significantly simplified because the developers only need to create a new *NewSQLEngine* class and replace the instance of the earlier class of database access by the new class.

The *Facade* design pattern is used “to provide a unified interface to a set of interfaces in a subsystem. *Facade* defines a higher-level interface that makes the subsystem easier to use. This can be used to simplify a number of complicated object interactions into a single interface” [Gamma07].

This pattern was used in the HPAD project to simplify the access to the database and to the graphs and window management classes provided by the *Java Development Kit* “JDK”. This change makes easier both to interact with those classes and the future modifications of the software. Figure 5.3 illustrates how the Façade pattern was applied.



**Figure 5.3 Application of Facade design pattern in HPAD**

To solve the extreme fragmentation problem that the class diagram of the original version presents, the re-engineered version of HPAD attempts to simplify the way the classes interact each other so that the new class diagram is easy to understand, debug and test.

Table 5.1 shows the difference in the amount of classes needed to perform the saving process of patient's vital signs data. It may be noticed that the amount of classes needed in the re-engineered version is significantly less than those required in the original version.

**Table 5.1** *Classes needed in the new vitals saving data process*

Original Version	Re-engineered Version
VitalsWindow	VitalsWindow
VitalsController	GeneralTableModelVitals
PatientTableModel	SQLServerDBT
DBEditorWindow	
WindowMarshal	
DBComboBoxModel	
DBSupport	

Table 5.2 shows the classes necessary for displaying patient I/O data in the original version and in the re-engineered version. As in the previous case, the amount of classes needed for this operation was excessive in the original version.

The class diagram of the nurse documentation system HPAD can be seen in Appendix 1.

## 5.2 Database Access Improvements

One of the requirements of this software version was that it must facilitate the developer to use any kind of database. In fact, the new version replaces the use of the *MySQL Server* database and supports the *Microsoft SQL Server* database engine.

When the new version of the nurse documentation application was proposed, it specified the need to re-design the software in such a way that a change from one database engine to another one would not be a labor so complicated. For this reason, the *Template* design pattern was used. By using the pattern, the developer will only have to create an object that extends from *DBTemplate* class and implement the *FormatConnect*, *FormatSelect* and *FormatUID* methods for the database engine support.

**Table 5.2** *Classes needed in the patient I/O data displaying process*

Original Version	Re-engineered Version
IO Controller	IOManager
IO Order	ImageIOController
IO TableAmount	GeneralTableModel
IOTableIcon	PatientDataWindow
IOWindow	PatientDataController
PatientAdmissionController	
PatientAdmissionModel	
PatientAdmissionWindow	
PatientIOPanel	
PatientTableModel	
UserController	
UserModel	
Window	
....(others)	

The structure of the abstract class *DBTemplate* is as follows:

```

public abstract class DBTemplate {
    public DBTemplate() {
    }

    public final ResultSet doQuery (String QueryString) {
        ResultSet rs;
        FormatConnect();
        rs=FormatSelect(QueryString);
        return rs;
    }

    //UID: Update Insert and Delete
    public final int doUID (String UIDString) {
        int resultado;

        FormatConnect();
        resultado=FormatUID(UIDString);
        return resultado;
    }

    public abstract void FormatConnect();
    public abstract ResultSet FormatSelect(String query);
    public abstract int FormatUID(String query);
    public abstract void setServer(String server);
    public abstract void setDBname(String DBname);
}

```

Another issue related to the database, described in section 4.6, is the use of updatable *ResultSets* for the implementation of the insertion and modification process of the database records. Due to the problems explained in the section 4.6, specifically the issues related to the migration to other database engines, the logic was changed in order to use SQL (*Standard Query Language*) sentences which can be understood by 100% of the database engines on the market.

### **5.3 Usability Improvements**

The original nurse documentation system HPAD implements a *Single Document Interface* (SDI), in which the task bar of the window manager shows each one of the windows opened by the application (See figure 5.4). Due to the disadvantages enunciated in the section 4.2, it was decided to change the design to a *Multiple Documents Interface* (MDI). In this approach the program has a main window and the secondary graphical interfaces (other windows and message boxes) are always going to depend on this window. In this way, the task bar will display just one item for the application HPAD (See figure 5.5).

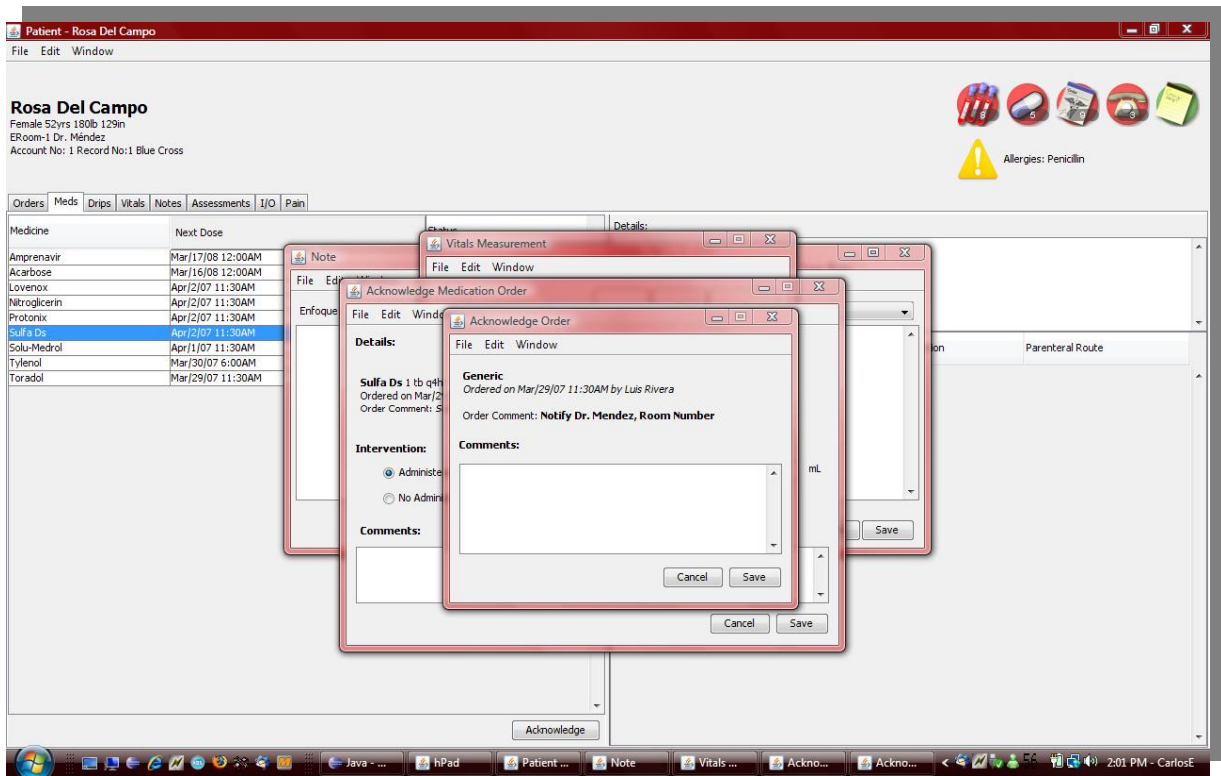


Figure 5.4 Windows taskbar with many items for an instance of HPAD

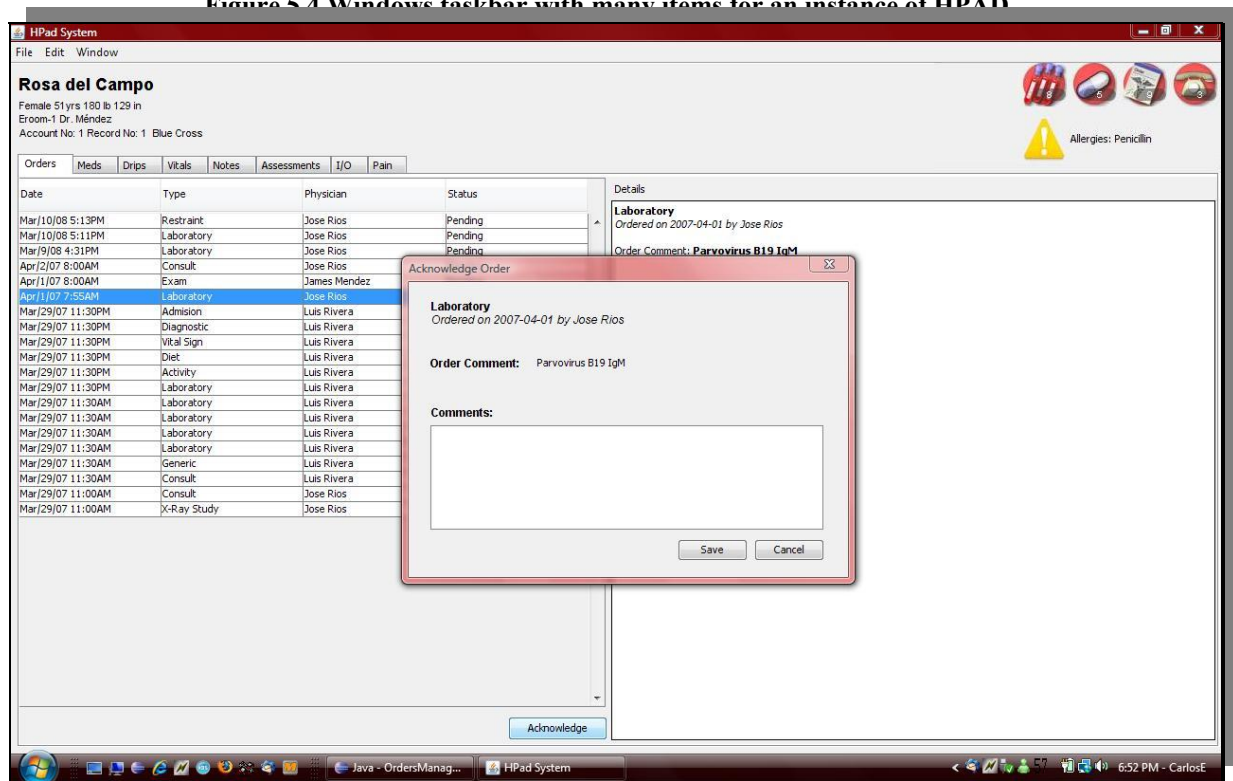


Figure 5.5 Windows taskbar with one item for an instance of HPAD

Another feature included in the new version that will enhance the usability and information access to users was the capacity to sort lists of items. Sorting was implemented as a standard characteristic in all tables used to display lists of items in the application. In this way, the user can order the list's contents by clicking in the name of any column of the table. The default behavior is to order initially in an ascending way and then descending (alternating by successive clicks). If the table is sorted in ascending way, it will show the icon “^” next to the column name. If the classification method is descending, the icon “v” will appear (See figure 5.6).

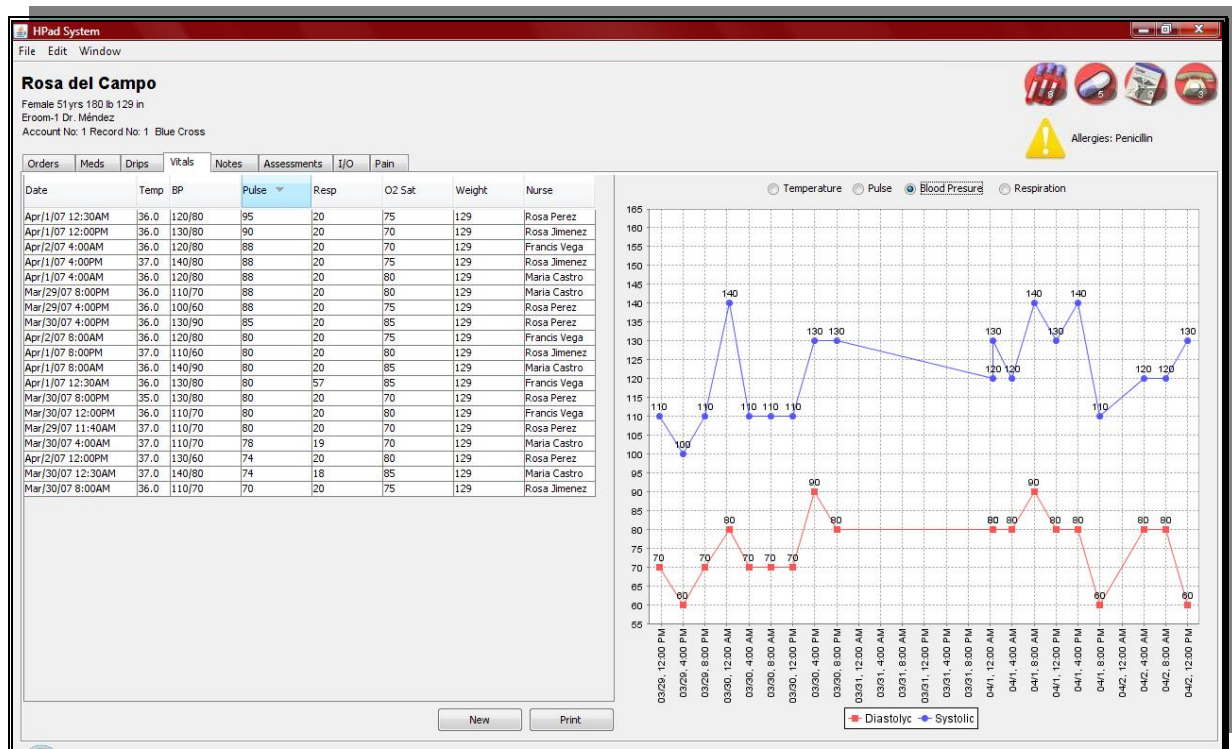


Figure 5.6 Table Column sorting example

## 5.4 Algorithm Implementation Improvements

The complete source code for the original nurse documentation software HPAD was rewritten from scratch with the exception of the alerts and reminders set of classes. This allowed the implementation of some algorithms from a different point of view, hoping to reach some performance improvements.

One of the improved aspects was the method used to make the ordering of the lists showed in the software (see previous section). Whenever the user clicks on a column to sort the items of a list using as criterion the above mentioned column, the original system makes a request to the database adding the corresponding clause `ORDER BY (ORDER BY column ASC/DESC)` and then it re-draws the table with the returned information. The new system uses the data that it already keeps in memory, sorting it, and then re-accommodates the table. This resulted in a much higher ordering speed, especially in environments with wireless connection.

## 5.5 Third-party Software Use Improvements

In order to give support to any database engine in a Java application, it is necessary to attach the driver that supports the API (*Application Programming Interface*) `JDBC 2.x` or superior, which normally is provided by the database vendor. The *jdbc* driver used by the re-engineered version of HPAD is the *SQL Server 2005 JDBC Driver 1.2* that provides compatibility with *SQL Server* previous versions. It is `JDBC 3.0` standard compliant and runs on the *Java Runtime Environment* (JRE) 1.4 and later versions. Among the improvements



that this driver version provides is a better performance and solutions to different security problems.

The libraries used to plot vital signs information are called *jfreeChart*. The original nurse documentation system HPAD uses version 0.9.21. However, the manufacturer (*Object Refinery Limited Company*, has made available the version 1.0.6. The updated version contains improvements in the render algorithms, checking of input data and managing of tooltips in graphs, as well as fixing of problems of memory consumption when the application that contains the graph remains a lot of time working on a computer system. The manufacturer recommends the urgent update of any previous version to 1.0.6. For this reason, the redesigned nursing application replaced the previous version of these libraries for the most current. This change required some changes in the methods of the classes responsible of plotting the vital signs information.

## **5.6 Database Model Improvements**

As was explained in section 4.5, some options lists are fixed in the source code of the program, which would force to re-compile and redistribute the application whenever a change was made in the items that compose them. To avoid this potential limitation and lack of flexibility in the software, one table has been created in the database that stores these types of lists and which is read by the software in the interface creation step.

Table name:

Labels

Columns:

lbl\_type (varchar(10))  
 lbl\_code(vvarchar(10))  
 lbl\_text (text)

To illustrate the use of this table we can take the combo box called *IV Type* showed at the interface of *New Intake & Output* (See figure 5.7). This list has two items: *Subcutane* and *Others*. This list may change, depending on the needs of the hospital that is using HPAD, by including new items, removing existing items, renaming them in different ways or changing them to their equivalent in the Spanish language.

**Figure 5.7 IV Type combo box**

In order to become independent these list items from the program source code were included in two rows of the Labels table as can be seen in table 5.3:

**Table 5.3 *Labels* table data corresponding to IV Type combo box**

Lbl_type	Lbl_code	Lbl_text
IOIVTYPE	O	Others
IOIVTYPE	S	Subcutane

Column *Lbl\_type* contains the name that groups those items: *IOIVTYPE*; column *Lbl\_code* contains the code corresponding to each item of the list which will be used in other parts of the program to refer to it; and column *Lbl\_text* contains the text string to be shown in the list.

An image of the *MS-SQL Server Query Analyzer* showing the data in the *Labels* table can be seen in figure 5.8:

	lbl_type	lbl_code	lbl_text
1	IOIVTYPE	O	Others
2	IOIVTYPE	S	Subcutane
3	MEDPAREN	A	Right Deltoid
4	MEDPAREN	B	Left Deltoid
5	MEDPAREN	C	Right Dorsogluteal
6	MEDPAREN	D	Left Dorsogluteal
7	MEDPAREN	E	Right Ventrogluteal
8	MEDPAREN	F	Left Ventrogluteal
9	MEDPAREN	G	Right Vastus Lat
10	MEDPAREN	H	Left Vastus Lat
11	MEDPAREN	I	Right Rectus Temoris
12	MEDPAREN	J	Left Rectus Temoris
13	MEDPAREN	K	Right Abdomen
14	MEDPAREN	L	Left Abdomen
15	MEDPAREN	N	N/A
16	MEDREASON	A	Adverse Reaction
17	MEDREASON	C	According to Norm
18	MEDREASON	D	Dx Study
19	MEDREASON	G	GI Disturbance
20	MEDREASON	I	Out of Institution
21	MEDREASON	M	Medical Order
22	MEDREASON	N	N P O
23	MEDREASON	O	OR
24	MEDREASON	P	Physical Therapy
25	MEDREASON	R	Reject by Patient
26	MEDREASON	S	In Process

**Figure 5.8 MS SQL Query Analyzer showing *labels* table data**

## 5.7 Other Improvements

As a result of the set of changes implemented in the application re-design, a substantial improvement has been registered in the times of some key processes of the system. One aspect that had an important improvement is the loading time of the patient's information after having being chosen in the *List of Patients* interface (See Figure 4.2).

The time that took the original system to load the patient information was around 9.23 seconds, whereas the time of the re-engineered system is 4.55 seconds in average, being tested in a wireless environment.

Some other processes that experienced a significant speed improvement are shown in Table 5.4.

**Table 5.4** *Process times comparison between original and re-engineered versions*

Process	Original version (secs)	Re-engineered version (secs)
Notes tab / filtering by Type	1.22	0.08
I/O tab / filtering by In/Out	0.99	0.04
I/O tab / filtering by Type	0.40	0.09
I/O tab / filtering by Period	1.13	0.12

The re-engineered version of HPAD has been developed using object-oriented programming placing great emphasis on the use of design patterns and looking to improve

those aspects that, according to the analysis phase, deserved to be reviewed to achieve a more modern software and prepared to face changes that will be required in the future.

## 6. RECAPITULATION AND FUTURE WORK

This document described the process of improving the nursing documentation system HPAD through the application of software re-engineering process. The development of the re-engineered system required the study of the existing system to locate both points of improvement and those sections that might be reusable. The redesign process focused on achieving a product easily expandable in the future, with particular emphasis on the software's ability to work independently of the database engine used.

### 6.1 RECAPITULATION

The software was implemented in *Java* language, which supports a wide variety of operating systems. The graphical interfaces of the original system were preserved, with the exception of the *Assessment* interface, which was completely redesigned.

The re-engineering process of this project made a significant use of design patterns. The most used pattern was the *Model-View-Controller*, which was used as a basis for the overall structure of the HPAD software. To provide a high level interface for some general functions used by the software, the *Façade* pattern was implemented. As a result of this pattern, an easy access for the developer to these functions was achieved, accelerating the implementation process.

The *Template* pattern was used so that the re-engineered nursing documentation system would not be linked at code-level with any specific database. The aim was that with very little code changes from developers, the HPAD software can work with almost any database in the current market. The *updatable ResultSets* from HPAD implementation were removed because of their limitations and were replaced by standard SQL sentences guaranteeing its operation regardless of the database vendor.

Some usability improvements were also implemented in this process. The change from a *Single Document Interface* (SDI) to a *Multiple Documents Interface* (MDI) was implemented in order to achieve greater consistency between the different interfaces and enhance the experience of the user when using the system HPAD. Another feature implemented was to allow sorting in all the display tables by clicking on the title bar. For this feature a graphic indicator was added to provide an indication about the type of sorting: ascending or descending.

The source code of the nursing documentation system was almost completely rewritten, with the exception of the set of classes of the *Alerts and Reminders* module, which were partially adapted and reused. All sorting algorithms were rewritten to use the resources of the local computer (processor and memory) rather than leaving the job of sorting to the database server. This led to a substantial improvement in the speed of these procedures. Other sections of the software were also benefited by the algorithms that were rewritten. For example, the loading time of the patient's information interface after having being chosen in the *List of Patients* interface was significantly improved.

Some libraries provided by Third-party companies were updated to their latest version in order to provide the application the benefits from its improved performance, security, and stability. These libraries are *jdbc driver SQL Server 2005* version 1.2 from *Microsoft Corporation* and *jfreeChart* 1.0.6 from *Object Refinery Company Limited*.

A table, called Labels, was added to the database. This table is used to store the lists of options that the system provides and displays. The advantage of using a database table instead of adding items to the lists in the source code is that these can be modified without changing the program code and having to compile and copy each client computer. This makes the update process easier for developers and users of the system.

The result of the re-engineering process applied in this project was a software that is faster, and easier to test, debug, modify, and extend. The new version of the HPAD software includes more modern components and the correction of several problems identified during the analysis phase. The software retains the functionality and the positive aspects of the interfaces of the original system by adding usability improvements.

## **6.2 FUTURE WORK**

In previous projects a usability test was conducted for the HPAD interfaces [Perez05]. It is necessary to conduct further usability tests for the *Assessment* interface because it has been completely redesigned.



The *Alerts and Reminders* set of classes should be revised, specifically the logic generating alerts to fulfill its objectives. These modules were not studied in this project because they were outside the objectives.

In order to formally conclude that the improvements generated by this project had a significant positive impact, a more detailed study of the execution times of the processes should be conducted of both, the original, and the re-engineered version of the software. The tests should include an environment closer to reality in terms of the number of patients, the data patient's records, the number of simultaneous users, and the simulation of a peak load of the database.

There must exist a module for reports in which the users could consult statistics and realize consultations on the stored information.

A complete review of the data model of the system is also recommended. In at least two cases, redundancy of information was detected: the tables *note\_type* and *note* contain *noty\_name* field, and tables *i\_o* and *io\_type* contain *io\_direction* field. These cases should be modified to a reference by code. These changes were not made by disorders that may lead to the other modules that compose HPAD (mobile and desktop).

The general performance of the system can be further improved with a study of the database tables that experience frequent readings and few writings. This study could indicate the appropriate indexes that accelerate the searches requested by the application.

## REFERENCES

- [Ash97] Ash, Joan S. "Factors Affecting the Diffusion of the Computer-Based Patient Record". Proceedings Conference of the American Medical Informatics Association. Annual Fall Symposium. (1997)
- [Balazinska99] Balazinska, Magdalena., Ettore Merlo, Michel Dagenais, Bruno Lague and Kostas Kontogiannis. "Partial Redesign of Java Software Systems Based on Clone Analysis". IEEE, Proceedings. Sixth Working Conference on Reverse Engineering. (1999), pp. 326-336.
- [Booch07] Booch, Grady. "Object-Oriented Analysis and Design with Applications". The Benjamin/Cummings Series in Object-Oriented Software Engineering, 2nd Edition. (2007)
- [Crespo05] Crespo, G. "A Comparative Study of Nurses Accessing Electronic Patient Record Systems with PDAs and Tablet PCs". M.S. Thesis University of Puerto Rico, Mayagüez, P.R. (2005)
- [Ferreira04] Ferreira, Ana., Ricardo Correia, Luis Antunes, Ernesto Palhares. "Integrity for Electronic Patient Record Reports". Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems. 2004.

- [Gamma07] Gamma, Erich, Richard Helm, Ralph Johnson and John Vlissides. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley books, 36<sup>th</sup> printing. (2007)
- [Gordon96] Gordon, Randolph L., Edward Baker, William Roper and Gilbert Omenn. "Prevention and the Reforming U.S. Health Care System: Changing Roles and Responsibilities for Public Health". Annual Review of Public Health. 1996.
- [Jao07] Jao, Chiang S., Cathy M. Helgason and Donna Zych. "The Role of Clinical Information Systems in Improving Physician Productivity: Challenges Facing the Adoption of an Electronic Charge Capture System". IEEE International Conference on Systems, Man and Cybernetics. (2007)
- [Mancl01] Mancl, Dennis. "Refactoring for Software Migration". IEEE Communications Magazine. (2001), pp. 88-93.
- [Masuda00] Masuda, Gou., Norihiro Sakamoto and Kazuo Ushijima. "Redesigning of an Existing Software Using Design Patterns". IEEE International Symposium on Principles of Software Evolution (2001).
- [Nielsen93] Nielsen, J. "Usability Engineering". Academic Press A Harcourt Science and Technology Company. 1993.

- [Perez05] Pérez, Carlos G. “Usability Study of a Patient Alerts and Reminders Display Component for an Electronic Medical Record System”. M.S. Thesis University of Puerto Rico, Mayagüez, P.R. (2005)
- [Rodríguez02] Rodríguez, Néstor J., José A. Borges, Viviam Murillo and Johanna Ortiz. “A Study of Physicians Interactions with Text-Based and Graphical-Based Electronic Patient Record Systems”. Proc of the 15th IEEE Symp. on Computer-Based Medical Systems (2002).
- [Rodríguez03] Rodríguez, Néstor J., José A. Borges, Yajaira Soler, Viviam Murillo y Celia R. Colón Rivera. “PDA vs. Laptop: A Comparisson of Two Versions of the Nursing Documentation Application”. Proc of the 16th IEEE Symp. on Computer-Based Medical Systems. (2003).
- [Rodríguez04] Rodríguez, Néstor J., José A. Borges, Yajaira Soler, Viviam L. Murillo, D. Z. Sands, “A Usability Study of Physicians Interaction with PDA and Laptop Applications to Access an Electronic Patient Record System”, 17th. IEEE International Symposium on Computer-Based Medical Systems. (2004).
- [Satpathy02] Satpathy, Manoranjan, Nils T. Siebel and Daniel Rodríguez. “Maintenance of Object Oriented Systems through Re-engineering: A Case Study”. Proceedings of the International Conference on Software Maintenance. (2002)

[Tahvildari02] Tahvildari, Ladan and Kostas Kontogiannis. “On the Role of Design Patterns in Quality-Driven Re-engineering”. IEEE, Proceedings. Sixth European Conference on Software Maintenance and Re-engineering. (2002)

## **APPENDIX**

## APPENDIX A

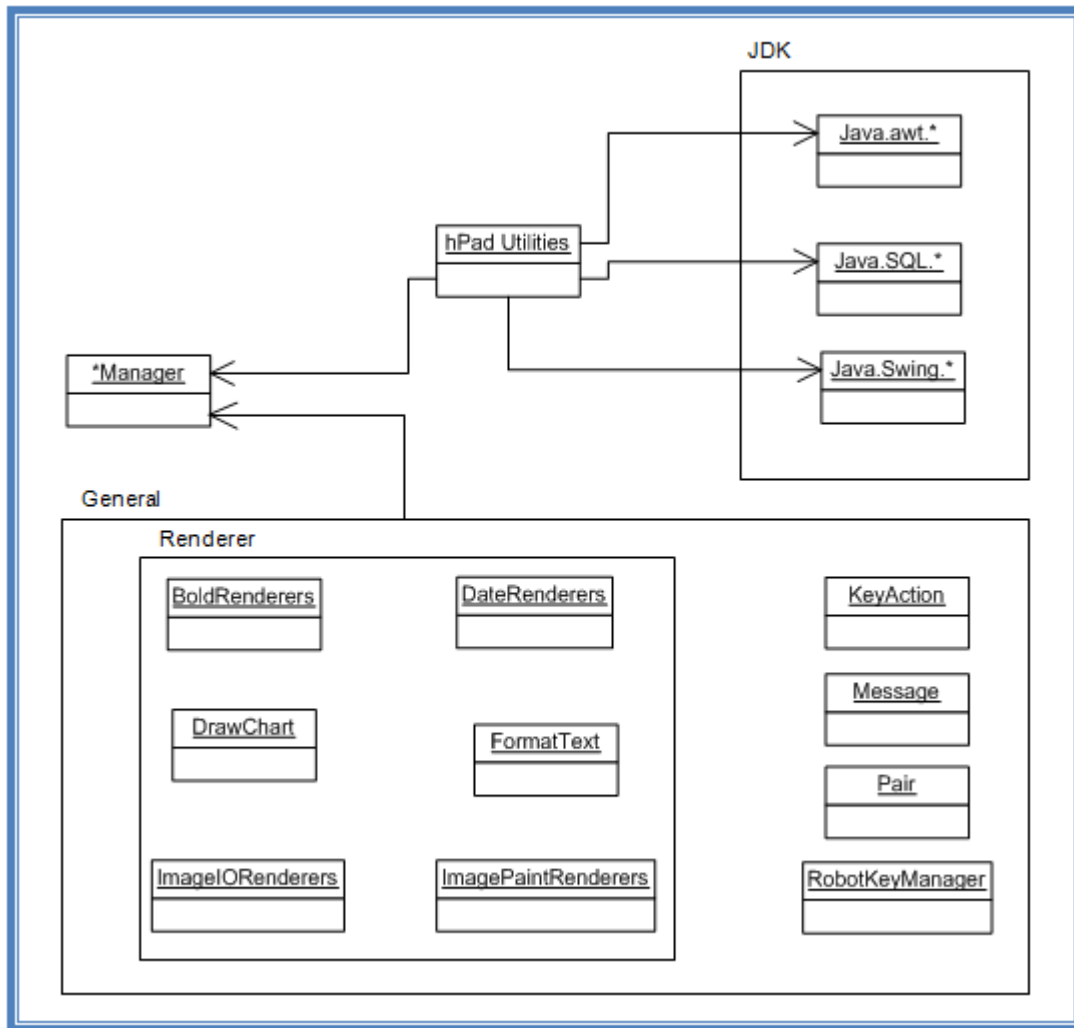


Figure A-1 HPAD Class Diagram part 1

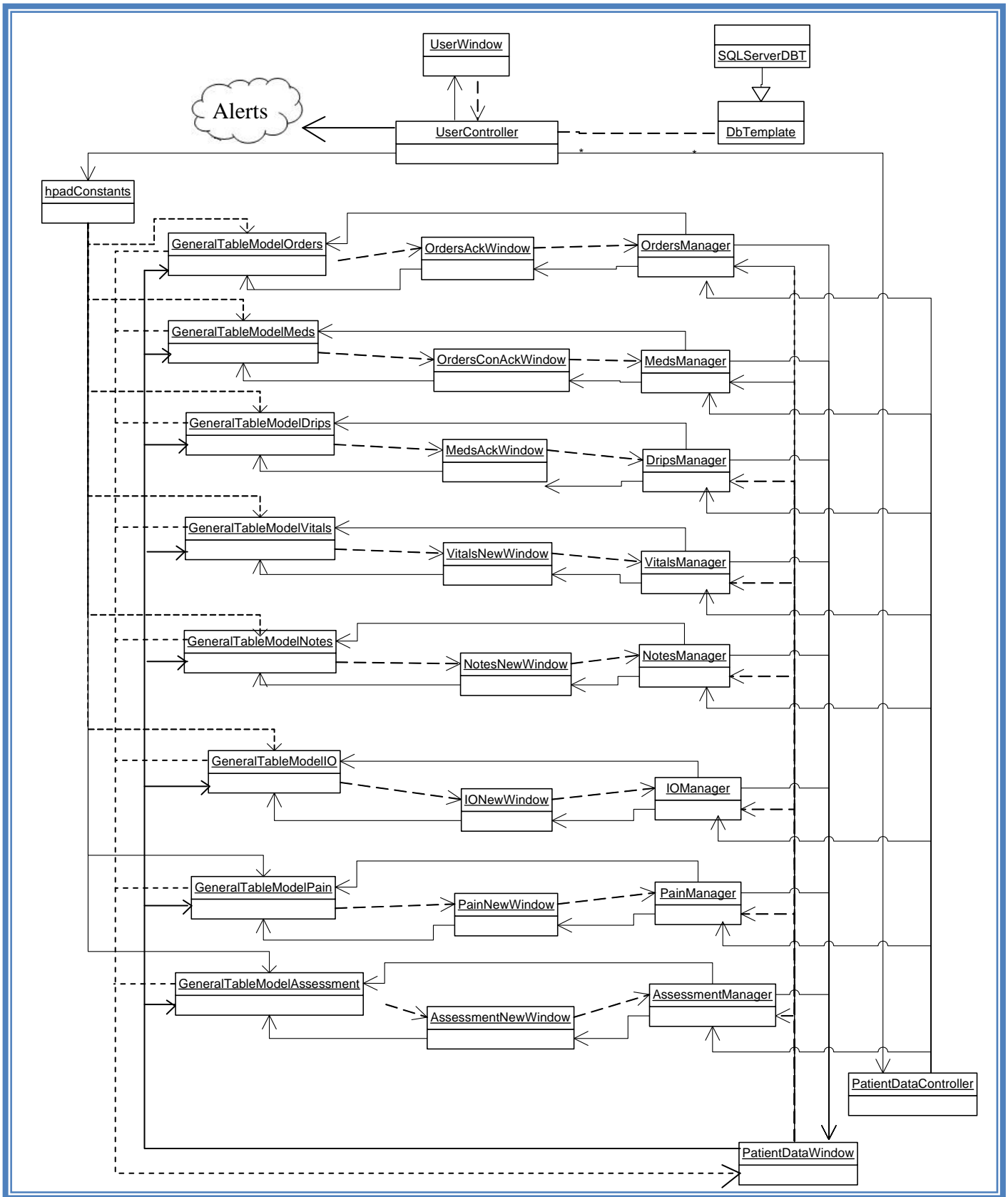


Figure A-2 HPAD Class Diagram part 2