



Universidad
Zaragoza



Software Application for Operation Planning in a Surgery Department

MASTER THESIS

This thesis is submitted as a completion of the Erasmus+ Program at Universidad de Zaragoza, SPAIN, along with the source code of the software application developed during the study period

Author
Diana BOTEZ
TUIASI

Supervisors
Ass. Prof.
Cristian MAHULEA, UNIZAR

Ș.l.dr.ing.
Alexandru ARCHIP, TUIASI

September 11, 2017

This page is intentionally left blank.

Chapter 1

Abstract

Currently, the decision makers in a hospital spend more time in performing management tasks than taking care of the patients (the job that they signed up for when they chose Healthcare domain). Although some of them might be doing this for a long time which gives them experience in such decisions, the efficiency can be improved with the help of some technical solutions that are the result of advance research in planning and scheduling surgeries [1]. Such technical solutions can also decrease the overtime hours and reduce the under/over utilization of operating rooms.

This thesis' focus is to create such a solution that will help doctors to spend less time in managing tasks and have a better occupancy rate within the available capacity. It will also decrease the stress level and allow the doctors to offer more of their time to patients. The technical solution that comes next to this thesis is an implementation of algorithms studied and developed before [1]. It also has an in-built management system for medical teams and patients. The interface allows different levels of clearance for a better security of the medical data and it also comes in two languages (English and Spanish).

This page is intentionally left blank.

Contents

1	Abstract	2
2	Introduction	6
2.1	Motivation	7
2.2	History and previous research	8
2.3	Methods of application	10
2.4	Organizational structure of medical department	11
2.5	Project description	13
3	Planning and Scheduling	16
3.1	General Aspects	16
3.2	Current available tools	19
3.3	Other possible use-cases	20
4	Used platforms and mathematical models	22
4.1	Platform(Solvers): CPLEX	23
4.2	Mathematical models	24
4.2.1	Algorithm: MILP	25
4.2.2	Algorithm: MIQCP	29
5	Project Architecture and Implementation	34
5.1	Minimum requirements & Dependencies	35
5.2	Application set-up	36
5.3	Platform, Back-end, Dependencies	38
5.4	Database	47
5.5	User Interface	53
6	Simulation Results	70
6.1	MILP	72
6.2	MIQCP	75
7	Conclusions & Recommendations	79
	Bibliography	81

This page is intentionally left blank.

Chapter 2

Introduction

This thesis is a research about planning and scheduling in a medical department. There are already many studies in this field as there are many subjects to focus on also. Some of the studies come with a physical solution, not just a mathematical algorithm. Despite this fact, not many hospitals adopted such a solution that implements a decision model. This, plus the constant increase of the waiting list forces the most expensive resource in a hospital to become a bottleneck, stop the hospital from getting the proper revenues and make the patients wait in a list that soon will seem endless.

In the next chapters of this thesis, we will talk about why this topic, what is it and how is it used. General aspects about the current algorithms, platforms that implement these algorithms and what directions to take in the near future. We will also talk about the proposed algorithms, their models and how are they integrated in the software application that comes with this thesis. The application and its user interface, the algorithms and their simulation results will be also presented in the second half of this thesis, followed by conclusions about the research.

2.1 Motivation

Technology is now everywhere. We all use it. Even if we just wash clothes, play video-games, drive a car or make robots to help us. We can find the latest available technology starting from smartphones to smart sidewalks or from smart cars to smart homes. Everything is technologized nowadays. So, why not use this power to help ourselves? To give us more time with our family by enjoying the one we have instead of wandering every second if people from the hospital forgot to call us for the surgery or not. The number of surgeries needed by the patients increased in the last decade because of lifestyle, stress, technology, comfort, and so on. Many researches were made for improving and optimizing the results in medical departments. Some of them were focused on lowering the costs, improving the occupation rate or balancing under and over time of surgeons with under and over utilization for the operating rooms.

From a management point of view, operating rooms are critical resources that needs to be occupied efficiently in order to obtain the maximum cost coverage. This also includes proper loading of the ORs, the use of medicines, medical instruments, devices and the human resources. From a patients' point of view, they are waiting for a surgery of a pathology that may or may not be more critical than other patient's pathology. Furthermore, if the critical level of the pathology is under a certain level, they all wait in an ordered list. Both surgeons and patients want to keep this order for the surgeries as much as possible.

2.2 History and previous research

Hundred of years ago, when surgeries as a practice were at the beginning, they were a terrifying thing for both patient and surgeon. Many people chose to die rather than go to a surgeon. Now, looking back at that time, it's understandable why people avoided surgeons. They didn't use any kind of pain reliever, although some of them used ice to numb the area. The patients, in all the cases were awake and seeing everything that was happening to them. Some of them needed to be hold by strong man because of the physical pain and mental stress they were in. Furthermore, the surgeons were not that caring about cleanliness in the operation room. They were mostly protecting themselves from the patient's blood rather than using a clean and safe environment and instruments. That is a bit understandable when we think that back then, there were not too many knowledges about microorganisms. Because of this lack of knowledge and the environment was not safe for an open wound, causing a large number of deaths. The patients that survived the surgery (either they were lucky or they had a great immune system) did not have a supervised post-operative recovery. In the mid-19th Century, surgeries had been done as fast as possible. For example, a leg amputation surgery could have been performed in 3 minutes or less, this of course, with the cost of accuracy. Although the anesthesia has been used since late 1800s, sterilizing the wound, air/room, and using clean white clothes and bed linings had been used starting with 20th Century. Even so, in around 4000 years since Egyptians were curing the migraine by drilling small holes in patient's head, in 3000 BC, the surgeries evolved very much, first by studying on animal body (16th Century) and then on dead human bodies (late 19th Century). Since then until nowadays, surgeries techniques and instruments kept improving thanks to all the studies and researches that were made in this field.

In year 2010, a paper called "Operating room planning and scheduling: A literature review" [2] was published as a review of current available papers. They studied over 246 manuscripts on operating room planning and scheduling that were published between *1950 and 2010. Nearly half of them were after January 1st, 2000*, which shows an increasing research of this domain for helping and improving the medical system. All this research is made only to bring the system up to date and be able to help more efficiently the patients that are in need. The authors split the paper in 7 sections, studying the documents after the patients they considered, performance measurement, precision, type of analysis, solutions, uncertainty and applicability. It gives a good look of what is mostly the focus of researchers, where are the gaps in research and what might be a direction for future studies. The review above shows that more than half of the studies focus on elective patients, no matter if they are inpatients (patients that need to stay in the hospital after the surgery – with or without being in the hospital before the surgery) or outpatients (patients

that usually come in and leave the hospital in the same day in which the surgery is scheduled). Elective patients are not considered emergencies. Emergency patients are non-elective and therefore they cannot be foreseen. There are some researches for emergency patients that use heuristic approaches to partial schedule resources. Elective patients are ordered in a list after the incoming date (the date in which the patient and the doctor decided the patient needs a surgery) and the critical level (low, normal or high). When a planning and scheduling is done, the selected patients for the current planning are announced that they have been scheduled for the surgery in a certain day. As in half of the papers reviewed in the article, this thesis is also considering only the operation theaters alone, without using any other facilities from the hospital, like ICU. This is because we consider only non-emergency patients which have a lower probability of needing to use the ICU than the emergency patients. Another study focus in the previous researches was, the waiting time - either considered for patient or for surgeons. This thesis considers only the waiting time for patients because the medical teams will assign a certain operation room in certain days, so they will know from before, the time and operation room to use.

Other similar studies also show that the majority of research had focused mainly on the surgical department due to high complexity of including other departments or even other facilities from the surgical department like PACU (Post Anesthetic Care Unit) or ICU. They also study current problem from the view of time horizon involved in the scheduling. Some focuses on the long and very long time planning (one year or more) while others focus on short and very short time planning (a few days to a few weeks). When using a long and very long time span the issue of capacity planning and allocation is addressed while in a short time span, a schedule can be constructed and executed with a predictable workload[3].

2.3 Methods of application

Planning and scheduling surgical operations is a process with 2 big steps. First step contains a plan (weekly or monthly) to assign a specific date to a specific patient. The second step is about ordering the patients scheduled on a certain day. In the first step, each patient is in a waiting list of a medical team. To assign a date to a surgery, we first need to assign an operation room to a medical team for each day in the current plan. After a medical team has an operation room assigned for each day of the plan, the patients from its waiting list can be scheduled. Also, in this step, a plan is not complete until all the patients that were scheduled confirm that they can be present on the specified date. The patients which do not confirm that they will be there on the allocated date, will be removed from the current plan and the next patients in the waiting list will be scheduled in their place. This step is repeated as long as there are free places in the current plan and there are patients in the waiting list who haven't been scheduled yet. The patients who were removed from the current plan because they cannot attend on the allocated date, they will be scheduled as soon as possible, manually, in a common agreed date or in a future plan. In the second step, each day from the current plan, will go through an ordering or sorting phase. This step allows giving a bigger priority to more urgent surgeries, complex/easy surgeries or children/old people depending on the surgeon's preferences or even patient's preferences. Also, patient that will come to surgery from a longer distance, have the possibility to be scheduled after a certain hour. This project focusses on the first part. The second step will be completed by each medical team, manually, for the time being.

2.4 Organizational structure of medical department

The project is created to be applied (first, but not only) in the Orthopedic Surgery Department of the local hospital, “Lozano Blesa”, Zaragoza, Spain. The targeted department is composed by a Head of Department (also a doctor) and 5 medical teams. Each team has a coordinator and each doctor has its own waiting list of patients. The waiting list of each team is composed from all the waiting lists of the team members. The first patient in the list is the one that has the biggest waiting time and the last patient in the list is the last patient added to one of the team members’ waiting lists. The application can be easily installed and used in another hospital or department, because:

1. it has bilingual interface (English and Spanish)
2. unrestricted number of Operating rooms
3. unrestricted number of medical teams

The medical department has an availability of only 2 operating rooms per day for non-urgent surgeries. Each of them have an active schedule of 7 hours, usually. Each OR is daily scheduled for only one team. The Head of Department has the following responsibilities:

1. assign teams for each available OR – one team per day
2. assign teams for external consultations
3. assign teams for emergency service

The assignment is made within a time period of two to six months. This way, the teams know in advance in which day they can use which OR. After the medical leaders know the scheduling for the OR, they start planning and scheduling the patients from the team’s waiting list guided by their own intuition and experience. This planning and scheduling is also made manually as the team assignments in the operating theatre, made by the head of department.

Studying the occupancy rate of the ORs from the last years, the average obtained rate was around 72% [1]. The structure of the department is also explained, in Figure 2.4.

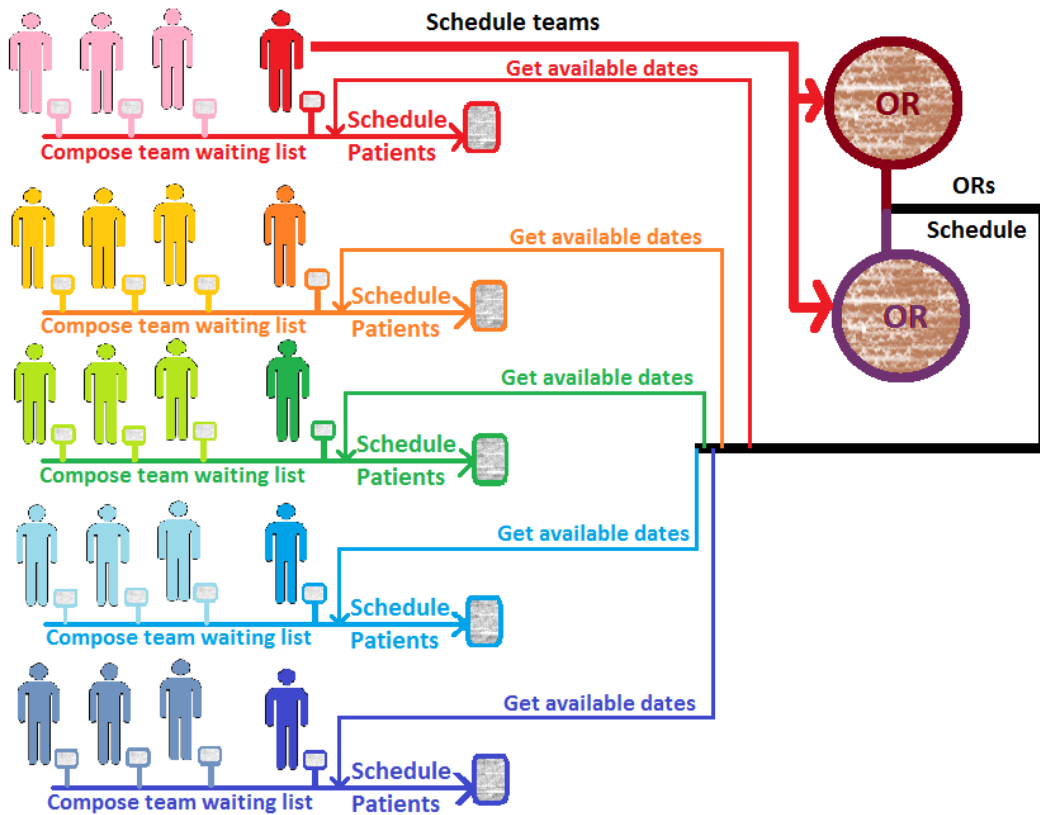


Figure 2.1: Department workflow

In Figure 2.4, each bold color represents a coordinator and faded colors represent team members. The small squares are each doctor's waiting list. The circles are the available Operating Rooms. The red doctor is also the head of department so, this doctor is scheduling daily medical teams in each OR. The coordinators (in strong colors) take the scheduled days for their team from the ORs' time table and create the planning and scheduling of the patients from the team's waiting list (also composed by them), represented by the big squares.

2.5 Project description

This application is made to be used by the medical staff of the Surgery Department. Its interface is designed to help managing the medical teams and patients in a more clear, efficient and easy way. It also gives clearance level to every user. So, not every user is allowed to do everything in the application. This way, it also offers a safe environment to work on. The application will be installed in a local computer in hospital's surgery department. This application is available in two different languages, English and Spanish, option available in the login window. This application also uses a database. So, every modification made will be saved and ready to be accessed and used in the current session or a future one, regardless the language preferences. The user will have to choose the desired language version in the login form and keep it until the end of the session. If the user wants to change the current language, the user must log out, switch the language, then log back in again. After the user has logged in, depending on the clearance, he can use the application for:

1. Managing medical teams
 - Add/remove doctor from team or medical teams
 - Move doctor to another medical team
 - Change the medical leader of the team or the team name
2. Managing patients
 - Add/remove patients
 - Add/remove/change surgeries from patient's medical history
 - Manually schedule a patient to a certain medical team and doctor and to a certain day in which that team is available
 - Unschedule a patient that cannot present himself to the hospital in the assigned day
3. Managing operating rooms
 - Add/remove operating rooms from the operating theatre
 - Schedule a medical team in a certain day for each operating room
 - Change data in the current available time table

4. Planning and scheduling patients from a certain waiting list for a given number of working days
5. Adding the information of the performed surgery for each patient to whom the scheduled surgery is completed

Once a patient is unscheduled, because the patient cannot confirm the availability for the surgery, the patient will be added in the top of the list for the following planning if a manually schedule in another available day is not done until then.

This page is intentionally left blank.

Chapter 3

Planning and Scheduling

3.1 General Aspects

Planning is used in our lives daily. No matter if we are setting our goals as “pass the current study year with over 9 out of 10 CGPA” or “in this vacation I want to see the Black Sea and Alps”, we are making plans. This is a normal thing for humans because one of the executive functions of the brain is planning. Please notice that there is no time notion involved in planning. After we set a time and place to the entries in our plan, it becomes a schedule. In other words, scheduling is the process of arranging/ordering and optimizing the workload of a process/plan. Planning and scheduling, both together – as a team, have a major impact on the productivity. Their purpose is to minimize the allocated time, resources and costs. When making a plan, usually, we need to answer 3 questions: What? How? and Who?, but when we’re making a schedule the answer of other 2 questions is important: When? and Why?. Planning means taking the target (let’s say: cross platform adventure game) and braking it down to smaller pieces (for our example, they can be: User Interface, user connectivity to server, quests, items, levels). Every piece needs to be broken down in smaller parts as long as it cannot be explained short and clear. This is a top-down planning. Another approach can be bottom-up planning, but it is used when there is already at least another similar successful project. For a good planning, it is recommended to have:

- (1) a Work Breakdown Structure (WBS),
- (2) a project execution plan,
- (3) usage of resources and
- (4) project milestones.

After everyone in the team has a good idea of what it should be in the end, the scheduling can be performed.[4] Scheduling means:

- (1) determining when exactly, the tasks from the previously determined plan will be performed (starting time and end time – based on a time estimation of the details

found in each task's description),
(2) in what sequence will they be performed.

To order the entries in a plan are usually used one of the following priority sequencing rules:

- First come, first served
- Earliest due date
- Shortest processing time first
- Longest processing time first

Depending on the process, accepted past due date time, the critical level of the tasks in the plan and dependencies, one of the above rules can be applied. If the plan is well structured, different rules for sequencing can be applied for different independent parts of the project. Though, if the plan includes more than one site or team than, a more complex solution should be considered. In this case, there are algorithms studied and developed ready to be implemented and used depending on the needs or the desired optimization. Almost in all cases, not all targets can be optimized. Therefore, each study focus on the most important tasks at the moment and compromise (but not too much) the optimization of the other tasks. Common scheduling failures (that occur usually to less experienced teams) are: lack of planning (there are tasks that haven't been foreseen. That means there are tasks that haven't been included in the planning, therefore they have not been included in the scheduling. This leads to delays), poor defined activities (it can lead to a misunderstanding of the task which will be reflected in delays), poor duration estimation (estimating less than the actual time, mostly caused by underestimating the complexity of the task).

The surgery planning and scheduling started with the nurse rostering problem. This problem refers to finding an optimal way to determine shifts and holidays for nurses. Both nurses and hospital have their own constraints. The problem is described as finding a schedule that fulfils both types of constraints. Nowadays, this is not the only problem of hospitals as the number of surgeries increased but the number of human resources (nurses, doctors, anesthesiologists and so on), logistics (rooms, beds) and medical instruments are limited. Also, because of this increase, the management team of an operation theatre is now formed from surgeons, anesthesiologists and nurseries unlike before when nurses have been chiefly responsible for the daily function of the surgical suite. A good management of the operation theatre helps the doctors and nurses to focus only on the patient on the surgery day, without losing time and effort to compensate a poor management and compromising patient safety. Operation room efficiency is of the most important constraint that the management team is trying to maximize. For these purpose, they use charts to analyze the under-utilized and over-utilized rooms as well as the overtime medical teams

have. In operation theatres that have more than two operation rooms, it is possible to have under-utilized rooms and overtime for medical team. For example, we have 2 operation rooms, each with 5 hours of time span, and the first operation room is used by a medical team for 3 hours and the second operation room is used by another medical team for 6 hours. That means we have 9 out of 10 hours occupied in the operation theatre (under-utilized operation theatre) and 1 hour of overtime performed by the second medical team in the second operation room.

Another problem which can generate overtime is start-time tardiness. It means that from various reasons, one or more of the surgeries scheduled in one day (referring to elective patients) starts at a later time than the one established in the schedule. This will also generate an increasing waiting time for the following patients besides the overtime for the medical team. The increase number or repeated days with overtime will also increase the stress level of the medical team members which will reflect in a decreasing attention for the patient safety. Not only overtime is a current problem but also the case cancellation in the day of surgery which will generate under time and under-utilization of the operating room. This problem is as important as overtime or over-utilization of the operating room because all of them generate revenue lost for the hospital. Therefore management teams have to consider all the problems and find and apply an optimized solution that suits best. Many of the cancellations are due to non-medical problems like surgeon unavailability, full ICU or bad weather that stops the patient to come in the scheduled day. Depending on the region, country and geolocation, a hospital can deal more with overtime than cancellations or vice-versa. A good management team does not adopt an optimization plan only after the results obtained from simulations or from other hospitals but also regarding its own medical history. Unfortunately, it is hard to obtain charts with medical load in past decades due to late adoption of technology in many sites.

3.2 Current available tools

An editorial on Artificial Intelligence planning and scheduling[5] shows a roadmap of the published research in Artificial Intelligence (AI) planning and scheduling, also giving some examples of prototypes. AI research in the field of planning and scheduling started to be very active in 1960s when many of the papers focused on the development of algorithms on planning as the only goal. In the mid '70s the researchers started to consider the achievement of sub-goals favorizing the appearance of partial order planners. In the mid '80s, the models were expanded to consider time as a major factor in the planning process. By the end of 1990s, there were available prototypes. Some of the prototypes were meant to be used in managing the chemical processes in plant operating procedures (CEP) or a planning and scheduling system used for mobile robots, beer production or military operation planning (Cypress). Also, for supporting physicians' diagnosis and therapy planning, there were created tools like MYCIN or EXPERT. In the medical field, current available software is basically focus on creating the time table for doctors and nurses. Some are available on mobile and allow tracking of working and vacation hours or comparing schedules¹ or web-based solution for managing teams and schedule shifts². The existing solutions of planning patients are not used because they do not consider the patient's preferences for a certain doctor or day. They also have the same problem on doctor's side. They are not available to choose the type of patients they want to have at a certain moment. Therefore, *the return to pen and paper* solution or not adopting a technical option *was unavoidable*.

In the proposed software solution, the patient will have a certain doctor and medical team assigned for each surgery in his medical history. *If him or the doctor in charge want to change it, they are able to do it*. Also, if the patient asks and the doctor in charge and medical team have a free slot in the allocated working hours, the patient can be scheduled for surgery in the day he desires.

¹Bio-optronics software – <http://bio-optronics.com>

²SEIDA – <http://www.optimum-choices.com>

3.3 Other possible use-cases

A possible direction in the future was proposed by Sandberg in the paper published in 2005[6]. He and his colleagues propose a design called “OR of the future (ORF)” where some of the preoperative and postoperative process can be done in parallel, if the OR have intake and recovery rooms. This way, the nonoperative time and the probability of overtime are reduced, by not being assigned to the OR occupation time. In this proposed solution, the anesthesia and other preoperative process are done in the intake room while the surgery room is prepared for the operation and, after the surgery is finished, a nurse will provide the early recovery attention in the post operatory room, while the surgery room can be prepared for the next patient that is in the intake room with the anesthesia personnel. Artificial Intelligence solutions might come in front in a near future by combining the studies and results from this field with the algorithms developed in the field of planning and scheduling, therefore making the planning and scheduling adaptable by learning from its own past decisions.

This page is intentionally left blank.

Chapter 4

Used platforms and mathematical models

Production planning problems (*not* production scheduling), started to implement linear programming since 1940s[7]. One of the powerful and general techniques used in linear programming problems was the simplex method, invented by George Dantzig in 1947¹. Later, in 1950s, production scheduling problems led to researches into sequencing problems. This led to creation of algorithms like: the earliest due date rule (EDD) and shortest processing time rule (SPT). For solving more complex problems, in 1960s, appeared algorithms based on Branch-and-Bound techniques that allowed finding the possible solutions and determine the optimal one. Also, column generation techniques and constraint programming have been developed to solve integer programming problems. In early '70s, complexity theory showed that algorithms able to find the optimal solution in a reasonable time are unlikely to exist. That's why, in '80s and '90s, algorithms that could find a near-optimal solution come into focus.

¹Wikipedia Simplex algorithm's page: https://en.wikipedia.org/wiki/Simplex_algorithm

4.1 Platform(Solvers): CPLEX

IBM is currently developing and a platform that implements algorithm for solving linear programming problems and other related problems (for example: Quadratic Constraint Programming) and it is called “IBM ILOG CPLEX Optimization Studio” [8]. Currently, it offers libraries for the most used programming languages like C, C++, Java, .Net and Python available on Windows environments as well as on Linux environments. CPLEX is crated over Concert Technology which is a set of libraries that allows the programmer to embed CPLEX in the above specified programming languages. Concert Technology also makes use of Callable Library which is a C library that allows the programmer to embed CPLEX in any proگرامing language that can call C functions. IBM describes CPLEX as “a tool for solving, first of all, linear optimization problems”.

The algorithms chosen for this thesis’ purpose are: **MILP** – because of (1) the binary variables defining if the surgeries scheduled or not for each day and (2) the real variables that describes the deviation from the desired output and **MIQCP** because of the quadratic conditions from the second model. Regarding the study above about results of available solvers for these problems, CPLEX was the one that fit better with the requirements of the models. Also, it is available cross-programming language and it offered us the possibility to reimplement the models in Java (they were implemented in Matlab by my colleague for testing purposes), where we could also offer a graphical interface to the end-user.

4.2 Mathematical models

In this thesis, we consider only the elective patients. The testing of the software application is done in the orthopedic department of local hospital “Lozano Blesa”, Zaragoza, SPAIN. There are 2 available operating rooms for the selected group of patients, each operating room being available for only one team per day. Also, the operating rooms’ time table varies daily. Based on the history of completed surgeries in the department, a list of average durations of the performed surgeries has been computed and used in these mathematical models. The purpose of the mathematical models is to optimize the use of the operating rooms without exceeding the maximum available time per day and without making use of unnecessary resources. Because there still are uncertainties about the surgeries’ durations due to unforeseen events of the different nature of the human bodies, and because the total day length of an operating room also includes the cleaning time, the optimal occupancy rate of each operating room, daily, is considered 80%. Before starting the project, “Daniel Clavel Villagrasa” studied these details and composed the models implemented in this thesis[1]. In order to obtain the desired occupancy rate, the first model offers an approach that makes use of integer programming (MILP) – a model that respects the desired occupancy rate with a specified tolerance and preserves the order of patients from the waiting list as much as possible. The patients from the waiting list have been ordered before adding their details as input to the model, after some criteria established by the hospital (for example: date of adding in the list, critical level). The other approach uses quadratic constraints (MIQCP) - a model that computes the probability of overcoming the total time and offers a minimum confidence level for it.

The results of the first algorithm, MILP, (as they can be seen in chapter 6), show that minimizing the absolute deviation can also lead to a bigger percentage rate than the desired one. This will lead to overtime for the medical teams and overload for the operation room, which is not desirable. The second approach comes in order to obtain a safe planning – with a lower probability of overcoming the total time in any of the day scheduled. Both models (MILP and MIQCP) use the duration of the surgeries and the cleaning time (time needed to clean any possible bacteria from the previous surgery and to prepare the operation room for the next surgery) as random variables with normal probability density function. MIQCP model will be used to obtain optimal solutions of the model. Using these data, we’ll obtain a minimum confidence level of not overcoming the total time by computing the average occupation rate.

4.2.1 Algorithm: MILP

The first model designed for the problem studied in this Thesis was implemented using **MILP** algorithm. As specified earlier, this model schedules the surgeries assuming a *desired occupancy rate* for the operating rooms while respecting as much as possible the patients order in the waiting list.

The objective of this model is expressed by two contradictory smaller objectives:

1. obtain the desired occupancy rate
2. maintain the order from the waiting list

This model has the following variables:

- n number of patients in the waiting list
- m number of days to schedule
- *order* vector of size n with the preferred order of each patient in the waiting list
- μ_{srg} vector of size n with the average duration of patients' surgeries (for n patients to schedule, this vector will have n average duration values in the same order)
- S_1, S_2, \dots, S_m binary vectors of size n , that are used for planning and scheduling patients in each day of the current schedule. It is implemented as a matrix of size $m \times n$, each row representing a working day
- β a parameter used to give more importance to one of the two contradictory objectives of this model
- $\alpha \in R_0^m$ a vector of absolute deviation in minutes for each day ($size(\alpha) = m$), regarding the objective ($Obj = dayTime \cdot \frac{occupancyRate}{100}$)

The conditions in this model are:

- It is mandatory that any patient to be scheduled once and only once in the whole plan. In other words, the sum of every column (from 1 to n , each of size m , to be less or equal with 1). This is implemented in the model as the following:

$$\sum_i S_{i,j} \leq 1; \quad \forall j = \overline{1, n} \quad (4.1)$$

- Each element form the vector of absolute deviations is defined as following:

$$\alpha_i \geq \left| \sum_{j=1}^n (\mu_{srg_j} \cdot S_{i,j}) - Obj \right| ; \quad \forall i = \overline{1, m} \quad (4.2)$$

If the accepted deviation from the desired occupation rate is 0, the requirement of keeping the order of the patient list cannot be fulfilled and vice versa. To solve this, the main objective of the model was composed by a linear cost function with two terms. The first one is related to the desired occupancy rate and the second one, with the order of the scheduled patients. The two terms are balanced by parameter β which allows us to set a compromise between the two objectives. The objective is defined as following:

$$\min \left(\sum_{i=1}^m [\alpha_i \cdot (m - i + 1) + \beta \cdot order \cdot S_i \cdot (m - i + 1)] \right) \quad (4.3)$$

Adding the desired objective, the equation Eq.(4.2) will become:

$$\begin{cases} \mu_{srg} \cdot S_i - Obj \leq \alpha_i \\ \mu_{srg} \cdot S_i - Obj \geq \alpha_i \end{cases} \quad \forall i = \overline{1, m} \quad (4.4)$$

The complete MILP model is as follows:

$$\min \left(\sum_{i=1}^m [\alpha_i \cdot (m - i + 1) + \beta \cdot order \cdot S_i \cdot (m - i + 1)] \right) \quad (4.5)$$

Subject to:

$$\begin{aligned} \mu_{srg} \cdot S_i - Obj &\leq \alpha_i & \forall i = \overline{1, m} \\ \mu_{srg} \cdot S_i - Obj &\geq \alpha_i & \forall j = \overline{1, n} \\ \sum_i S_{i,j} &\leq 1 \end{aligned} \quad (4.6)$$

The *size and complexity* of the model is defined as following:

- Number of variables: $m(n + 1)$
- Number of constraints: $2m + n$

An implementation of this model with a schedule result of a random generated list can be found in the following **example**:
 number of patients ($n = 20$), number of days to schedule ($m = 5$), desired *occupancyrate* = 80%, working days of 7 hours and an accepted deviation of maximum 10% of the total day time.

Order Number	Patient		Surgery	Average duration [min]
	Last Name	First Name		
1	Jacobsen	Paulita	Surgery1	100
2	Casio	Les	Surgery5	194
3	Harting	Lois	Surgery3	77
4	Wasielewski	Lorina	Surgery7	184
5	Sill	Teresa	Surgery29	121
6	Pratts	Laurice	Surgery2	81
7	Acebedo	Tamie	Surgery6	97
8	Ganley	Newton	Surgery6	97
9	Zollinger	Shelly	Surgery6	97
10	Walko	Sylvester	Surgery10	85
11	Rosado	Obdulia	Surgery15	150
12	Buonocore	Tamela	Surgery25	153
13	Southern	Kellie	Surgery6	97
14	Raab	Clarissa	Surgery15	150
15	Ballengee	Walton	Surgery6	97
16	Difalco	Destiny	Surgery15	150
17	Gouin	Dana	Surgery4	86
18	Rabago	Ilda	Surgery6	97
19	Carbone	Margareta	Surgery10	83
20	Valliere	Aiko	Surgery29	121

Table 4.1: Random generated list of patients to be scheduled

Desired occupancy rate of 80% from 7 working hours leads to a desired occupancy time of 336 minutes (out of $7[hours] \cdot 60[minutes] = 420[minutes]$). The obtained results for $\beta = 0.333$ are (priority given to occupancy rate closer to the desired rate):

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients	Absolute deviation [min]
1	337	80.2381	4, 12	1
2	335	79.7619	1, 10, 11	1
3	344	81.9048	2, 14	8
4	338	80.4762	3, 6, 7, 19	2
5	339	80.7143	5, 8, 20	3

Table 4.2: Test results for the above patient list using MILP, $\beta = 0.333$

Solution cost: 146.227
 Total time: 2.69 [sec]

The obtained results for $\beta = 2$ (priority given to patient order list) are:

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients	Absolute deviation [min]
1	315	75.0	2, 5	21
2	330	78.5714	1, 3, 12	6
3	337	79.5238	4, 11	2
4	328	78.0952	6, 7, 14	8
5	332	79.0476	8, 10, 16	1

Table 4.3: Test results for the above patient list using MILP, $\beta = 2$

Solution cost: 610
Total time: 0.63 [sec]

4.2.2 Algorithm: MIQCP

The second model is implemented using **MIQCP** algorithm. This model schedules the surgeries by maximizing the occupancy rate of the operating rooms while respecting as much as possible the patients order in the waiting list. This model has the following *variables*:

- n number of patients in the waiting list
- m number of days to schedule
- *order* vector of size n with the preferred order of each patient in the waiting list
- μ_{srg} vector of size n with the average duration of patients' surgeries (for n patients to schedule, this vector will have n average duration values in the same order)
- σ_{srg} vector of size n with standard deviations of patient's surgeries
- μ_{Ct} vector of size n with the average duration of the cleaning time after each surgery
- σ_{Ct} vector of size n with standard deviations of the cleaning time
- X_i vector of size m with total day time in minutes for each day to be scheduled
- S_1, S_2, \dots, S_m binary vectors of size n , that are used for planning and scheduling patients in each day of the current schedule. It is implemented as a matrix of size $m \times n$, each row representing a working day
- C_i the given confidence level percentage of not overcoming the total day time allowed in each OR working day. One percentage for each schedule, available for all the days to be scheduled
- V_{C_i} the value of a *normal variable* for the given minimum confidence level (in percentage)
- β a parameter used to give more importance to one of the two contradictory objectives of this model
- $\alpha \in R_0^m$ a vector with the difference in minutes between the total working time in each day to schedule (X_i) and the average durations of the surgeries scheduled in day i ($size(\alpha) = m$)

The *conditions* in this model are:

- It is mandatory that any patient to be scheduled once and only once in the whole plan. In other words, the sum of every column (from 1 to n, each of size m, to be less or equal with 1). This is implemented in the model as the following:

$$\sum_i S_{i,j} \leq 1 \quad ; \quad \forall j = \overline{1, n} \quad (4.7)$$

- Each element form the vector of absolute deviations is defined as following:

$$X_i - \sum_{j=1}^n (\mu_{srg_j} \cdot S_{i,j}) = \alpha_i \quad ; \quad \forall i = \overline{1, m} \quad (4.8)$$

- The confidance level of not overcoming the total working day time for each day in the schedule has to be grater than the given minimum confidance level.²

$$\frac{X_i - (\mu_{srg} + \mu_{Ct}) \cdot S_i}{\sqrt{(\bar{\sigma}_{srg}^2 + \bar{\sigma}_{Ct}^2) \cdot S_i}} \geq V_{C_i} \quad ; \quad \forall i = \overline{1, m} \quad (4.9)$$

The objective of this model is express with the same two contradictory smaller objectives:

1. obtain the desired occupancy rate
2. maintain the order from the waiting list for the scheduled patients

It's ecuation remains the same as in Eq.(4.3).

The complete MIQCP model is as follows:

$$\min \left(\sum_{i=1}^m [\alpha_i \cdot (m - i + 1) + \beta \cdot order \cdot S_i \cdot (m - i + 1)] \right) \quad (4.10)$$

Subject to:

$$\left\{ \begin{array}{l} X_i - \sum_{j=1}^n (\mu_{srg_j} \cdot S_{i,j}) = \alpha_i \\ \sum_i S_{i,j} \leq 1 \\ K_i \cdot S_i - [A \cdot S_i]^2 \leq X_i^2 \\ X_i - A \cdot S_i \geq 0 \end{array} \right. \quad \begin{array}{l} \forall i = \overline{1, m} \\ \forall j = \overline{1, n} \end{array} \quad (4.11)$$

² \bar{x}^2 is a vector of same dimensions with x where $\bar{x}^2(i) = x(i)^2 = x(i) \cdot x(i)$

Where K , A and B are defined as:

$$\begin{aligned}
 K_i &= V_{C_i}^2 \cdot B + 2 \cdot X_i \cdot A \\
 A &= \mu_{srg} + \mu_{Ct} \\
 B &= \bar{\sigma}_{srg}^2 + \bar{\sigma}_{Ct}^2
 \end{aligned}
 \tag{4.12}$$

The constraint $X_i - A \cdot S_i \geq 0$ ensures that the scheduled working time for each day (where $A \cdot S_i$ is the sum of the average time for each surgery scheduled and their corresponding cleaning time) is lower than the total time available (X_i).

The *size and complexity* of the model is defined as following:

- Number of variables: $m(n + 1)$
- Number of linear constraints: $2m + n$
- Number of quadratic constraints: m

The results obtained for a minimum confidence level of 80%, working days of 7 hours ($X_i = 420$ [min]) and $\beta = 0.333$ (priority given to occupancy rate closer to the desired rate) for the same list of patients, are:

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients
1	334	79.523	3 10
2	331	78.809	5 6 11
3	327	77.857	4 9 19
4	319	75.952	13 16 18
5	294	70.0	0 1
6	291	69.285	7 8 12
7	271	64.523	2 14 17

Table 4.4: Test results for the above patient list using MIQCP, confidence level of 80% and $\beta = 0.333$

Solution cost: 2708.245
 Total time: 34.076 [sec]
 Proposed Occupation rate for MILP: 68.9583[%]

The obtained results for $\beta = 2$ (priority given to patient order list) are:

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients
1	334	79.523	3 10
2	315	75.0	1 4
3	327	77.857	2 6 11
4	316	75.238	5 9 13
5	318	75.714	0 7 19
6	319	75.952	15 16 18
7	291	69.285	8 12 14

Table 4.5: Test results for the above patient list using MIQCP, confidence level of 80% and $\beta = 2$

Solution cost: 3059.0
 Total time: 4.961 [sec]
 Proposed Occupation rate for MILP: 68.9583[%]

This page is intentionally left blank.

Chapter 5

Project Architecture and Implementation

In this chapter we will talk about the software application. This application was built on **4(four)** independent tracks as presented below. Each of the tracks has been built with a constant feedback. It is not yet in full release, but in Beta testing at the university hospital in Zaragoza, SPAIN.

- minimum requirements
- application set-up
- back-end implementation
- planning and scheduling
- database communication
- user interface

5.1 Minimum requirments & Dependencies

Computer hardware and software requirments can be found in below table.

Resource type	Resource name & version				
Operating System	Linux or Windows on 64 bit				
RAM	≥ 4 GB				
Processor	64 bit processor				
Java Environment	JRE 1.8				
CPLEX library	CPLEX 12.6.2				
Database	MySQL server, version 5.7.18 Database connection credentials: <table border="1" style="margin-left: 20px;"> <tr> <td>user</td> <td>rootUser</td> </tr> <tr> <td>password</td> <td>*****</td> </tr> </table>	user	rootUser	password	*****
user	rootUser				
password	*****				

Table 5.1: Computer hardware and software requirments

Observation:

- **CPLEX Licence has to be bought from IBM.**
- **This application has been built using a student licence.**

For installing the database server, please follow the instructions below.

Operating System	Commands
Linux	<ul style="list-style-type: none"> • sudo apt-get install mysql-server • sudo mysql-secure-install
Windows	<ul style="list-style-type: none"> • download MySQL installer (https://dev.mysql.com/downloads/installer/) • install with on-screen instructions

Table 5.2: Instalation commands for MySQL server

Regarding CPLEX instalation, please follow the instruction file that comes with CPLEX Licence.

5.2 Application set-up

Application can run either with double-click on JAR file or run

```
root@host:~# java -jar surgeryPlanning.jar
```

command from Terminal (command-line). The path in Terminal (command-line) must be the location where the JAR file is stored. Although the application will run smoothly using any of the ways presented before, *the scheduling process will not work*.

To use the application at it's fullest, the application must be run from comand-line, adding as an argument, the path for CPLEX "bin" and "lib" files, as shown below (please note that it is an one-line command):

```
root@host:~# java
  -Djava.library.path="/usr/local/cplex/cplex/bin/
  x86-64_linux/:/usr/local/cplex/cplex/lib/x86-64_linux/"
  -jar surgeryPlanning.jar
```

The above path for CPLEX files should be updated with the current path in PC. Even though this is just one line, it's not the most user-friendly way to start an application. This is why, after installing all the dependencies, all the needed files for the application are stored in the same location with all the other programs installed on PC (if the user does not request otherwise). A ".bat" file (for Windows based Systems) and a ".sh" file (for Linux based Systems) had been created to offer an easy and safe way to run the application. These files, shown below, are stored in the same folder with the application and they should have a short-cut (or copy) on desktop.

For Windows users, ".bat" file should look like below:

```
SET pathToJarFile="C:\Users\Diana\Downloads\surgery planning"

SET pathToCplexLibrary="C:\Program
Files\IBM\ILOG\CPLEX_Studio1262\
cplex\bin\x64_win64;C:\Program
Files\IBM\ILOG\CPLEX_Studio1262\ opl\bin\x64_win64"

SET jarFile="surgeryPlanning.jar"

# DO NOT CHANGE AFTER THIS LINE
# -----

cd %pathToJarFile%
java -Djava.library.path=%pathToCplexLibrary% -jar %jarFile%
```

For Linux users, “.sh” file should look like below:

```
pathToJarFile="/home/diana/git_repo/opep/surgeryPlanning/dist"

pathToCplexLibrary="/usr/local/cplex/cplex/bin/x86-64_linux/:
    /usr/local/cplex/cplex/lib/x86-64_linux/"

jarFile="surgeryPlanning.jar"

# DO NOT CHANGE AFTER THIS LINE
# -----

cd $pathToJarFile
java -Djava.library.path=$pathToCplexLibrary -jar $jarFile
```

The application folder should have:

- surgeryPlanning.jar file
- a “lib” folder with the following JAR files: cplex.jar, mysql-connector-java-5.1.42-bin.jar, oplall.jar, swing-layout-1.0.4.jar, swingx-0.9.4.jar
- initial database schema file (.sql file)
- “.bat” or “.sh” file

At this point, using one of the above presented files, user can start the planning and scheduling application like any other application, without having the need to manually write the CPLEX library path or start the application by writing a command in command line interface.

5.3 Platform, Back-end, Dependencies

The development platform used in building this software application is **Java (version 1.8)** on a **64 bit Operating System**. The need of using 64 bit Operating System (OS) when there is the risk that the hospital will have only computers with 32 bit OS, comes because *CPLEX* is only available on 64 bit version (no matter the OS used). The application can be used on x86 OS, but the planning and scheduling will not work. The whole point of this application is to automatize planning and scheduling patients for surgeries so, from now on, we will assume that computer has a x64 OS. This application is built on latest available Java version (1.8) to be easily mentained or updated with new features. For the interface, we used swing package. Also, for cplex and working with the database, we used the available *cplex.jar* file and *mysql-connector-java*, respectively. The *IDE* used in developing this application is *NetBeans 8.1*

The application has **5 different internal packages**. Two of them are for the User interface (one in English and one in Spanish), one for the common classes (like: Patient, Doctor, Surgery or Database Connection/Querry), one for CPLEX definying and computing models and another one for the main files of the application. The UI packages will be explained in more details in section (5.5).

The "mainFiles" package contains the entry-point class of this application, called "OperationPlanningMainClass" which creates a "Welcome window" (Fig.5.1) and makes it visible. At this point, the user will choose the language to use during the next sesion. If the application cannot connect to the database, an error message will be prompted.



Figure 5.1: Welcome window for the surgery planning application

The sesion is initialised with a login form in the language choosen by the user (Fig.5.2). For the following examples, we will use the English version of the interface.

The Spanish version is identical. If the username is not in the database or the password is wrong, an error message will be prompted to inform the user about the login failure. After a successful log in, the application will start with the user's clearance level. Further details about the application can be found in section (5.5).

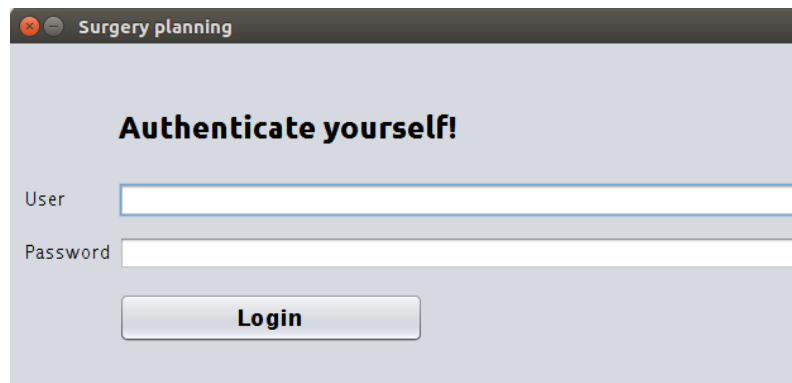


Figure 5.2: English session login window for the surgery planning application

The "commonFiles" package contains all the classes for defining and managing patients, users, doctors, operating rooms, as well as the connection and queries with the database. All these classes are explained below:

- **PatientIdentifiers** contains patientId, lastName, firstName, gender, dateOf-Birth, remarks and medicalHistory. Remarks are strictly connected with the patient and not with any surgery from patient's medical history
- **PatientMedicalHistory** contains vectors of same length for surgeryName, pathologyName, doctorInCharge, doctorTeamLead, admissionDate, scheduledForSurgery, scheduledDate and surgeryCompleted. The fields "scheduledForSurgery" and "surgeryCompleted" are boolean fields that express if the patient was scheduled for surgery (or not) and if the surgery was completed (or not) - only if the patient was scheduled. The "admissionDate" field is used to arrange the patients in the waiting list regarding the number of days the patients are waiting to be scheduled. Oldest admitted patient for surgery that was not yet scheduled, will be the first one in the waiting list.
- **PatientList** is the interface class with the database for managing new or existing patients.
- **MedicIdentifier** contains the last name and the first name of the doctor, the doctor's department, IDnumber and the coordinator's IDnumber
- **MedicalTeams** is the interface class with the database for managing new and existing doctors and medical teams

- **SurgeriesList** is the interface class with the database for managing new and available surgeries that can be performed in the current department
- **ORClass** contains the ID, name and type of the operating room, as well as the booking details like: date, starting and ending hour and teamName
- **OperationRooms** is the interface class with the database for managing the operating theatre and the assigned medical teams to each operating room
- **Users** is the interface class with the database for managing the users. This class contains a private passPhrase to encrypt the passwords. AES is the encryption algorithm used in this application. The password encryption key for each user is a selection of bits from the combination of private passPhrase and user's username. This is the only place where the password is in plain text. For saving a new user or password or checking an existing one (at login), the application will only use the hashed password.

The "**cplexFiles**" package contains the CPLEX object, example files for both algorithms used. It also contains a simpler application solved with CPLEX, for an easy understanding of CPLEX package interface. This application uses only one class from this package. It is called "Opep class", short from Operation Planning of Elective Patients. To use this class and schedule patients, an instance of it needs to be created. Before calling the public method of planning and scheduling patients, the "patientList" needs to be set. If there are less than 3 patients in the waiting list, or no patient at all, the planning and scheduling does not start. Also, if the number of days to schedule is less than 1, the scheduling will not start. In both cases, an error message will be displayed. When a schedule is requested, the *opep_main* method is called, using MILP algorithm. If the Head of Department wants these schedules to be above a certain confidenceLevel, the *dayPercentage* variable can be computed using MIQCP by calling *opep_generateOccupancyRate* method. After the recommended day occupancy percentage is obtained, a more accurate schedule can be requested (for the unscheduled patients).

```
/**
 * The main method to create a schedule. The used algorithm
 * is MILP. The patient waiting list needs to be set before
 * calling this method.
 *
 * @param m number of days to schedule
 * @param dayPercentage percentage of desired occupancy rate
 * @param dayDeviation percentage of accepted deviation
 * @param beta value used in balancing the two contradictory
 * objectives, between [0.333, 3]. Recommended value: 1
 * @param teamId it has to contain "TM" before the ID (e.g.
 * "TM51")
```

```

*
* @return a matrix with 2 rows with the IDs of the scheduled
*   patients and their scheduled surgery index
*/
public Vector<Vector<Integer>> opep_main(int m, int
    dayPercentage, int dayDeviation, double beta, String
    teamId){

    /*
    * Define primary and auxiliary variables.
    * Get available dates for team.
    * Get available surgeries details.
    * Create short version of patient waiting list with only
    *   patientId, surgeryId surgeryAverage.
    */

    try {
        do {
            /** If there are more available days to plan than
            patients, decrement the number of days to plan.*/
            if (solved == false && daysToPlan > 0) {
                daysToPlan--;
            }
            /** If there are no days to plan or less than 3
            patients to schedule, break the loop.*/
            if (daysToPlan == 0 || averageDurations.isEmpty() ||
                averageDurations.size() <= 2) {
                break;
            }

            /** Create model and solve it.*/
            cplex = new IloCplex();
            int m; // m less or equal 7(=planningDays)
            if (planningDays <= daysToPlan) {
                m = planningDays;
            } else {
                m = daysToPlan;
            }
            int n = averageDurations.size();

            IloIntVar[][] S = new IloIntVar[m][n];
            for (int i = 0; i < m; i++) {
                S[i] = cplex.boolVarArray(n);
            }

            IloIntVar[] alfa = new IloIntVar[m]; // alfa vector

```

```

for (int i = 0; i < m; i++) {
    /**tollerance[i] = (int) (dayDeviation *
        totalDayTime[i] / 100)*/
    alfa[i] = cplex.intVar(0, tollerance.get(i));
}

/**
 * Objectives.
 */
double [][] c = new double[m][n];
IloNumExpr cSum = cplex.constant(0);
for (int i = 0; i < m; i++) {//from 0 to number of days
double val = (m - i) * beta;
for (int j = 0; j < n; j++) {//from 0 to number of
    patients
    c[i][j] = (j + 1) * val;
}
cSum = cplex.sum(cSum,
    cplex.prod(alfa[i], cplex.constant(m - i)),
    cplex.scalProd(c[i], S[i]));
}
IloObjective minimizeCost = cplex.addMinimize(cSum);

/**
 * Constrains.
 */
// output vector for occupation per days, using alfa
    vector as boundaries
for (int i = 0; i < m; i++) {
    cplex.addEq(cplex.abs(cplex.diff(cplex.scalProd(
        values, S[i]), effectiveDayTime.get(i))),
        alfa[i]);
}

//constraint: sum on each column is <= 1
for (int j = 0; j < n; j++) {
    IloIntExpr Ssum = cplex.sum(S[0][j], 0);
    for (int i = 1; i < m; i++) {
        Ssum = cplex.sum(Ssum, S[i][j]);
    }
    cplex.addLe(Ssum, 1);
}

/**
 * Solve the model.
 */

```

```

        solved = cplex.solve();
        if (solved) {
            // get alfa values
            // select only first selectPlanningDays(=5)
            // scheduled dates (or less)
            // decrease the number of days remained to
            // schedule
            // get scheduled patients and surgeries per day
        }
        } while (solved || daysToPlan > 0);
    } catch (IloException e) {
        System.err.println("Concert exception caught: " + e);
    } finally {
        if (cplex != null) {
            cplex.end();
        }
    }
    patientsList = new Vector<>();
    if (scheduledPatientsId.isEmpty()) {
        return null;
    }

    /** Create the output matrix form the resulting vectors. */
    return scheduledPatientsAndSurgeries; /** Return the above
        matrix. */
}

/**
 * This method creates a MIQCP schedule for half a year with
 * the specified confidence level. It will consider all the
 * available patients in the database and create an average
 * of the day occupancy percentage obtained.
 *
 * @param confidenceLevel the confidence level of not
 * overcoming the working hours, between {51, ... , 99}
 *
 * @return the recommended day occupancy percentage
 */
public double opep_generateOccupancyRate(int
    confidenceLevel){

    /**
     * Define primary and auxiliary variables.
     * Get available dates for team.
     * Get available surgeries details.
     * Create short version of patient waiting list with only

```

```

    patientId, surgeryId surgeryAverage.
*/
double Vc1 = getNormalizedValueFor(confidenceLevel);
double beta = 1;

int daysToPlan = 130; // 52[weeks/year]/2 =
    26[weeks/halfOfYear] => 130[workingDays/halfOfYear]
int cleaningAverageDuration = 20; // minutes
int cleaningStandardDeviation = 10; // minutes
double averageOccupationRate = 0;

try {
    do {
        /** If there are more available days to plan than
            patients, decrement the number of days to plan.*/
        if (solved == false && daysToPlan > 0) {
            daysToPlan--;
        }
        /** If there are no days to plan or less than 3
            patients to schedule, break the loop.*/
        if (daysToPlan == 0 || averageDurations.isEmpty() ||
            averageDurations.size() <= 2) {
            break;
        }

        /** Create model and solve it.*/
        cplex = new IloCplex();
        int m; // m less or equal 3(=planningDays)
        if (planningDays <= daysToPlan) {
            m = planningDays;
        } else {
            m = daysToPlan;
        }
        int n = averageDurations.size();

        IloIntVar[][] S = new IloIntVar[m][n];
        for (int i = 0; i < m; i++) {
            S[i] = cplex.boolVarArray(n);
        }

        // alfa vector
        IloIntVar[] alfa = cplex.intVarArray(m, 0, dayTime);

        /**
         * Objectives.
         */
    }
}

```

```

double [][] c = new double[m][n];
IloNumExpr cSum = cplex.constant(0);
for (int i = 0; i < m; i++) {//from 0 to number of days
    double val = (m - i) * beta;
    for (int j = 0; j < n; j++) {// 0 to no. of patients
        c[i][j] = (j + 1) * val;
    }
    cSum = cplex.sum(cSum,
        cplex.prod(alfa[i], cplex.constant(m - i)),
        cplex.scalProd(c[i], S[i]));
}
IloObjective minimizeCost = cplex.addMinimize(cSum);

/** Output vector for occupation per days, using alfa
    vector for boundaries */
for (int i = 0; i < m; i++) {
    cplex.addEq(cplex.abs(cplex.diff(cplex.scalProd(
        values, S[i]), effectiveDayTime.get(i))),
        alfa[i]);
}

/**
 * Constrains.
 */
int[] A = new int[n];
double[] B = new double[n];
double[] K = new double[n];
double correctionTime = (Vc1 * Vc1 * 12 * 12) - (Vc1 *
    Vc1 * 10 * 10);

for (int i = 0; i < n; i++) {
    A[i] = values[i] + cleaningAverageDuration;
    B[i] = (standardDeviation[i] * standardDeviation[i]
        + cleaningStandardDeviation *
        cleaningStandardDeviation);
    K[i] = Vc1 * Vc1 * B[i] + 2 * (dayTime + 10) * A[i];
}

for (int i = 0; i < m; i++) {
    cplex.addEq(cplex.diff(dayTime,
        cplex.scalProd(values, S[i])), alfa[i]); // first
        condition
    IloNumExpr quadraticExpr = cplex.scalProd(A, S[i]);
    // (a1*s1 + a2*s2 + ... + an*sn)
    cplex.addLe(cplex.sum(cplex.diff(cplex.scalProd(K,
        S[i]), cplex.prod(quadraticExpr, quadraticExpr)),

```

```

        correctionTime), (dayTime + 10) * (dayTime +
        10)); // third condition
    cplex.addGe(cplex.diff(dayTime, cplex.scalProd(A,
        S[i])), 0); // forth condition
}

//constraint: sum on each column is <= 1
for (int j = 0; j < n; j++) {
    IloIntExpr Ssum = cplex.sum(S[0][j], 0);
    for (int i = 1; i < m; i++) {
        Ssum = cplex.sum(Ssum, S[i][j]);
    }
    cplex.addLe(Ssum, 1);
}

/**
 * Solve the model.
 */
solved = cplex.solve();
if (solved) {
    /** Get alfa values */
    /** Select occupancyRate for first
        selectPlanningDays(=2) scheduled dates (or less)
        */
    /** Decrease the number of days remained to schedule
        */
}
} while (solved || daysToPlan > 0);

/** Create averageOccupancyRate from all occupancyRate
    obtained while scheduling*/
} catch (IloException e) {
    System.err.println("Concert exception caught: " + e);
    averageOccupancyRate = 0;
} finally {
    if (cplex != null) {
        cplex.end();
    }
}

patientsList = new Vector<>();
return averageOccupationRate; /** Return the the obtained
    average occupancy rate. */
}

```

5.4 Database

The application has a local relational database (within the hospital’s intranet) regarding the confidentiality of the data. It’s structure is showed in Fig. 5.3.

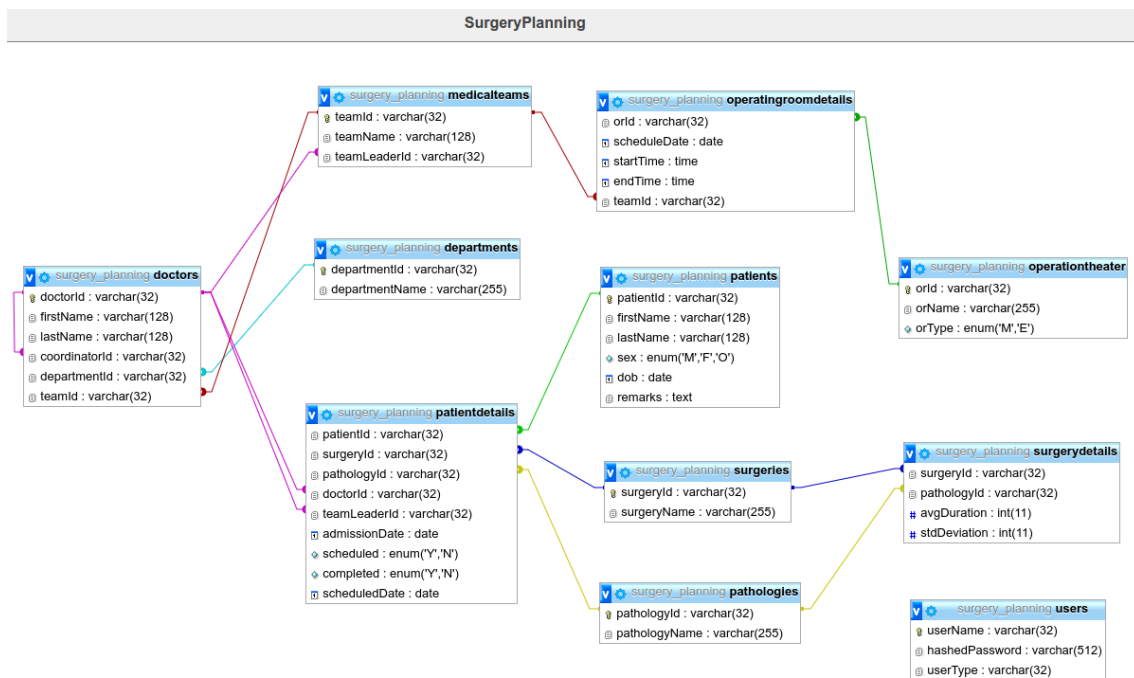


Figure 5.3: Database structure for the surgery planning application

In the lower-right corner, is the "Users" table. It contains the usernames available to login into this application, their hashedPassword and userType. As explained in chapter 2, section 2.5, each user has a specific clearance level. These levels are expressed below.

```

/*
 * UserType enum for establishing user's clearance level.
 */
public enum UserType {
    HEAD_OF_DEPARTMENT, //==0
    COORDINATOR, //==1
    MEDIC, // ==2
    ASSISTANT, //==3
    UNKNOWN_TYPE //==4
}

```


Besides "Users" table, there are other primary tables, as it follows:

- **surgeries** contains id and name for the available surgeries in the department
- **pathologies** contains id and name for the pathologies treated in the department
- **departments** contains id and name of all the departments joined the current medical teams
- **patients** contains id, first name, last name, gender, date of birth and remarks

All the other tables depend on one or more tables described above. These are:

- **doctors** contains doctor Id, first name, last name, coordinator Id, department Id and team Id
- **medicalTeams** contains team Id, team name and coordinator Id
- **patientDetails** contains patient Id, surgery Id, pathology Id, doctor Id, team-Leader Id, admissionDate, scheduled, completed, scheduledDate
- **operatingRoomDetails** contains operatingRoom Id, scheduleDate (=booking date), start time, end time, team Id
- **operationTheatre** contains operatingRoom Id, operatingRoom Name, operatingRoom Type (Morning or Afternoon)

According with the details given in previous section, database files (database-Queries and databaseConnection) are in the "commonFiles" package. *Database-Queries* class contains methods with all the queries for the database. All methods are grouped in seven(7) categories. Methods from the same group have the same beginning, as shown below.

```
/**
 * @abstract Database interface class. Contains all the
 *   needed SQL
 * queries for the entire application.
 *
 * Methods from the same group have the same beginning, as
 * follows:
 *
 * user data manipulation: "u_"
 * teams data manipulation: "tm_"
```

```
* departments data manipulation: "dpt_"
* doctors data manipulation: "d_"
* surgeries data manipulation: "srg_" OBS: pathologies
  included
* operating rooms data manipulation: "or_"
* patients data manipulation: "p_"
*
* @author Diana Botez
*/
public class DatabaseQueries {
    /** Private members. */

    /** User managing methods.*/
    public boolean u_checkIfUserNameExists(String user);
    public boolean u_addNewUser(String user, String
        encryptedPass, Utils.UserType type);
    public boolean u_updateUserType(String user,
        Utils.UserType newType);
    public boolean u_updateUserPassword(String user, String
        encryptedPass);
    public boolean u_deleteUser(String user);
    public Utils.UserType u_getUserTypeForCredentials(String
        user, String encryptedPassword);
    public boolean u_checkUserCredentials(String user, String
        encryptedPassword);

    /** Medical team managing methods.*/
    public boolean tm_checkIfTeamIdExists(String id);
    public boolean tm_checkIfTeamExists(String name);
    public String tm_getTeamId(String name, String leaderId);
    public String tm_getTeamNameByTeamLeaderId(String
        leaderId);
    public Vector<String> tm_getAllTeamNames();
    public String tm_getTeamIdByTeamLeaderId(String leaderId);
    public String tm_getTeamLeaderIdByTeamId(String teamId);
    public String tm_getTeamIdByTeamName(String teamName);
    public boolean tm_setTeamName(String teamId, String
        newName);
    public boolean tm_removeEmptyTeam(String teamId);
    public boolean tm_removeTeamLeaderId(String leaderId);
    public int tm_getNumberOfTeamMembers(String leaderId);
    public Vector<Vector<String>> tm_getAllTeamsDetails();

    /** Medical department managing methods.*/
    public boolean dpt_checkIfDepartmentIdExists(String id);
    public String dpt_getDetaprtmentId(String name);
```

```
/** Doctors managing methods.*/
public boolean d_checkIfDoctorExists(String id);
public boolean d_checkIfDoctorIsTeamLead(String id);
public Vector<Vector<String>> d_getAllTeamLeaders();
public Vector<Vector<String>>
    d_getTeamMembersDetails(String teamId);
public boolean d_moveDoctorIntoOtherTeam(String doctorId,
    String newTeamLeaderId);
public boolean d_changeTeamLeader(String oldTeamLeaderId,
    String newTeamLeaderId);
public Vector<String> d_getDoctorDetailsById(String id);
public Vector<String> d_getDoctorDetailsByFullName(String
    fullName);
public Vector<String>
    d_getTeamLeaderDetailsByTeamId(String id);
public String d_getCoordinatorTeamId(String leaderId);
public boolean d_addNewDoctor(MedicIdentifiers doctor,
    String teamId, String departmentId);
public boolean d_deleteDoctor(String id);

/** Surgeries managing methods.*/
public boolean srg_checkIfSurgeryIdExists(String
    tableName, String id);
public boolean srg_checkIfPathologyIdExists(String
    tableName, String id);
public String srg_getSurgeryId(String name);
public String srg_getPathologyId(String name);
public int srg_getSurgeryAvgDuration(String id);
public int srg_getSurgeryStdDeviation(String id);
public Vector<Vector<String>> srg_getAllSurgeriesDetails();
public boolean srg_addNewSurgery(String surgeryName,
    String pathologyName, int avgD, int stdDev);
public boolean srg_updateSurgery(String surgeryName,
    String pathologyName, int avgD, int stdDev);
public boolean srg_isSurgeryAssignedToPatient(String id);
public boolean srg_deleteSurgery(String surgeryName);

/** Operating rooms managing methods.*/
public boolean or_checkIfOrIdExists(String id);
public boolean or_isOrFree(String id, Date date);
public boolean or_isTeamAvailable(String id, Date date);
public Vector<String> or_getScheduledDatesForOr(String id);
public Vector<String> or_getScheduledDatesForTeam(String
    teamId);
public int or_getNumberOfAvailableDaysAssigned(String id);
```

```
public String or_getTeamIdForScheduledRoom(String roomId,
    String date);
public String or_getRoomNameForScheduledTeam(String
    teamId, String date);
public String or_getOrIdByName(String name);
public String or_getOrId(String name, String orType);
public boolean or_addOrSchedule(String orId, Date date,
    Time start, Time end, String teamId);
public String or_getOrNameById(String id);
public String or_getOrTypeById(String id);
public Vector<String> or_getAllOrNames();
public boolean or_setOrNameById(String id, String newName);
public Vector<String> or_getIdForAllOrs();
public int or_getTotalNumberOfOrs();
public Vector<Vector<String>>
    or_getAvailableDatesToScheduleForTeam( String teamId);
public Vector<Integer>
    or_getTotalTimeForAllAvailableDatesToScheduleForTeam(
    String teamId);
public int or_getTotalTimeForScheduledTeam(String teamId,
    String date);
public boolean or_deleteOr(String orId);
public boolean or_removeBookingDateForRoom(String roomId,
    Date date);
public boolean or_removeRecordsForTeam(String teamId);
public Vector<ORData> or_getRoomScheduledDetails(String
    id);

/** Patients managing methods.*/
public boolean p_checkIfPatientIdExists(String id);
public Vector<String>
    p_getPatientByNameAndBirthdate(String lastName, String
    firstName, String birthDate);
private String p_getPatientId(String lastName, String
    firstName, Date birthDate);
public boolean p_addNewPatient(String lastName, String
    firstName, Utils.PatientGender patientSex, Date
    birthDate, String observations);
public boolean p_updatePatientInfo(String id, String
    lastName, String firstName, Utils.PatientGender
    patientSex, Date birthDate, String observations);
public boolean p_addUpdateSurgeryToPatient(String
    patientId, String srgId, String ptgId, String doctorId,
    String coordinatorId, Date admision, String scheduled,
    Date scheduledDate);
public boolean p_updateSurgeryScheduledStatus(String
```

```
        patientId, String srgId, String ptgId, String doctorId,
        String coordinatorId, String scheduled, Date
        scheduledDate);
    public boolean p_schedulePatient(String patientId, String
        srgId, String scheduledDate);
    public boolean p_deleteSurgeryFromPatientHistory(String
        patientId, String srgId);
    public void
        p_updateDoctorInChargeForAssignedPatients(String oldId,
        String newId);
    public void p_updateTeamLeaderForAssignedPatients(String
        oldId, String newId);
    public Vector<Vector<String>>
        p_getPatientsDetailsForDoctorOrTeam(String id, boolean
        team, boolean includeCompleted);
    public Vector<Vector<String>>
        p_getScheduledPatientsDetailsForTeam(String id, String
        date);
    public Vector<Vector<String>>
        p_getScheduledPatientsDetailsByDate(String date);
    public void p_setSurgeryCompleted(String patientId, String
        surgeryId);
    public PatientIdentifiers p_getPatientDetails(String id,
        boolean includeCompleted);
}
```

5.5 User Interface

The User Interface (UI) is designed and implemented in 6(six) sections. First 2(two) sections are the most used ones ("Medical teams" and "Patients") followed by 2 sections that don't allow full acces to all users ("Surgeries" and "OR timetable"). The fift section ("Schedule") has visible pannels only for Team Leaders (doctors that co-ordonate medical teams). The last section, is the "user" section. This is the section that allows the user to end the current sesion.

"Medical teams" panel, figure 5.4, has three different subpanels. In figure 5.4 firts *tm* subpanels are describing the current *tm* available teams. Each subpanel in the first *tm* panels has a different name and that is medical team's name (e.g. "Superman", "Rainbows" or "coder fault"). These subpannels have all the details about the medical team, from doctors and team coordinator, to operation rooms booked for the team.

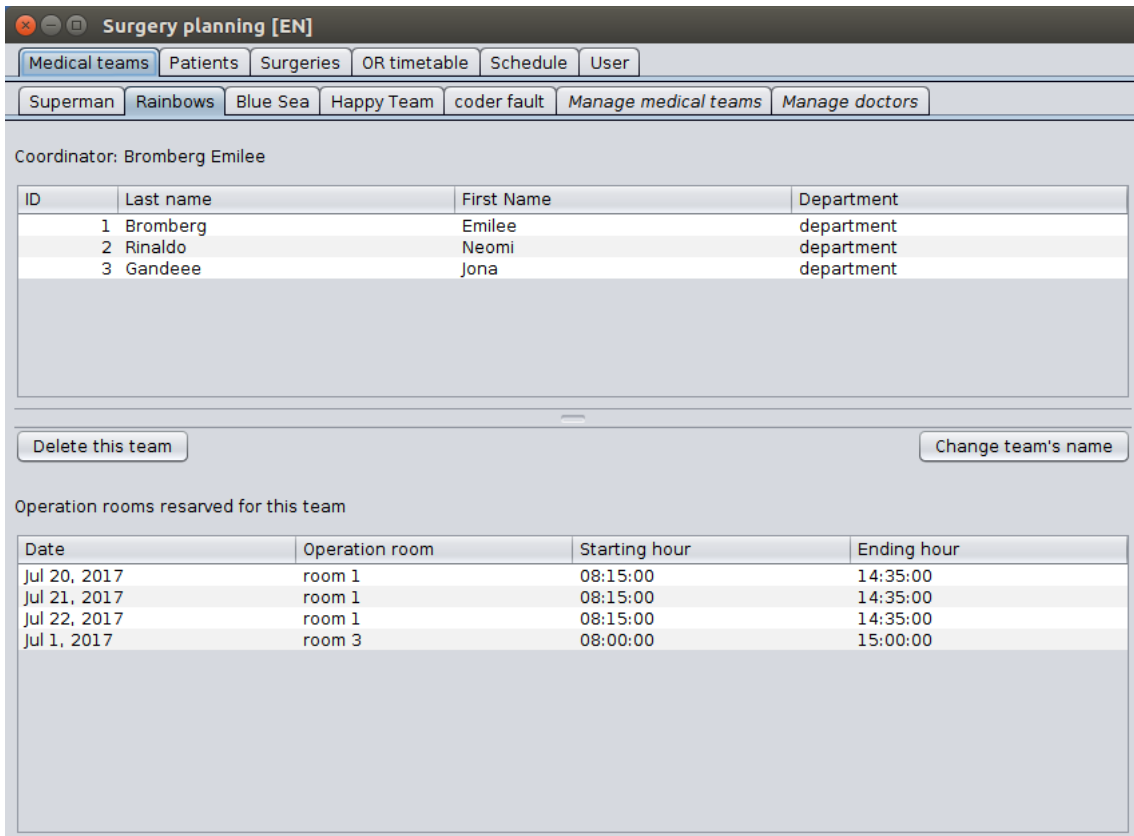


Figure 5.4: Medical teams panel for the surgery planning application

The two buttons availables on team pannels, fig. 5.4, are enabled only for team

leaders. A team leader can change the name of his own team. After the change, any user will be able to see it, but there are no notifications sent about the change. If the team has members, it cannot be deleted (from the database).

”*Manage medical teams*” panel has 2(two) parts. Only one of them is enabled at a time as it is shown in figures 5.5 and 5.6. In figure 5.5, the enabled part allows the user to change a medical team leader with another doctor from the same team, or to add a new doctor to an existing team. If the user wants to add a new medical team, then checking the box ”Add new medical team” is mandatory. Doing so, the second part will be enabled and the first part will be disabled as shown in figure 5.6.

Figure 5.5: Medical teams management for the surgery planning application

If the user wants to change the coordinator of a team with a doctor that is not in the team at that time, an auxiliary step should be done. The user has to move (or add) the (new) doctor to the team. After this step is complete, the user can change the current coordinator of the team with the doctor recently added to team. The user has the ability to manage the medical teams only if it is Head of department. No other user type is allowed to modify the structures of the medical teams and their coordinators.

The screenshot shows a web application window titled "Surgery planning [EN]". At the top, there are several tabs: "Medical teams", "Patients", "Surgeries", "OR timetable", "Schedule", and "User". Below these, there are more tabs for specific teams: "Superman", "Rainbows", "Blue Sea", "Happy Team", "coder fault", "Manage medical teams", and "Manage doctors". The main content area has two buttons at the top: "Change medical lead" and "Add new medic to existing team". Below these is a checked checkbox labeled "Add new medical team". There are four text input fields: "Coordinator's last name", "Coordinator's first name", "Coordinator's department", and "Team name". At the bottom, there are two buttons: "Reset" and "Add new team".

Figure 5.6: Add new medical team in surgery planning application

Even though the medical staff can only be updated by the Head of Department, all users can see the details of any doctor in "Manage doctors" panel.

The screenshot shows the same application window, but with the "Manage doctors" tab selected. At the top, the "Manage medical teams" tab is highlighted. Below it, there is a dropdown menu labeled "Medic" with "Prince Diana" selected. There are seven text input fields: "Personal ID" (26), "Last name" (Prince), "First name" (Diana), "Department" (dpt), "Coordinator's ID" (26), "Coordinator's Last Name" (Prince), and "Coordinator's First Name" (Diana). The "Coordinator's Team Name" field contains "Superman". At the bottom, there are two buttons: "Remove this medic" and "Move medic into other team".

Figure 5.7: Medical staff management for the surgery planning application

From the drop-down box, in figure 5.7, user can select any existing doctor and see its details. The available details about the selected doctor include: coordinator’s details, team name, and doctor’s details. The Head of Department can remove the selected doctor from the database and application if the current doctor is not a team leader. For deleting a team leader, another doctor from the same team, must take the team leader position (see previous panel). Doing so, the doctor is not a team leader anymore and can be removed from database. If there is no other doctor in selected doctor’s team, removing the doctor means leaving a medical team empty (no member at all) and this is not a realistic option. For this case, when the team should no longer be in the system, the Head of Department should move selected doctor (current team leader of a team that is no longer needed) into another team. In the botton-right side, is a button for this step. After this operation, the empty team will be deleted from the system. At this point, the selected doctor is not a team leader anymore and doctor’s team is no longer in the system. So, there are no restrictions anymore in removing selected doctor from the database. This way of removing the medical team or team leader form database has been chosen because, when a doctor is removed from database, all the patients that the doctor is assign to perform a surgery, will be automatically transfered in team leader’s waiting list.

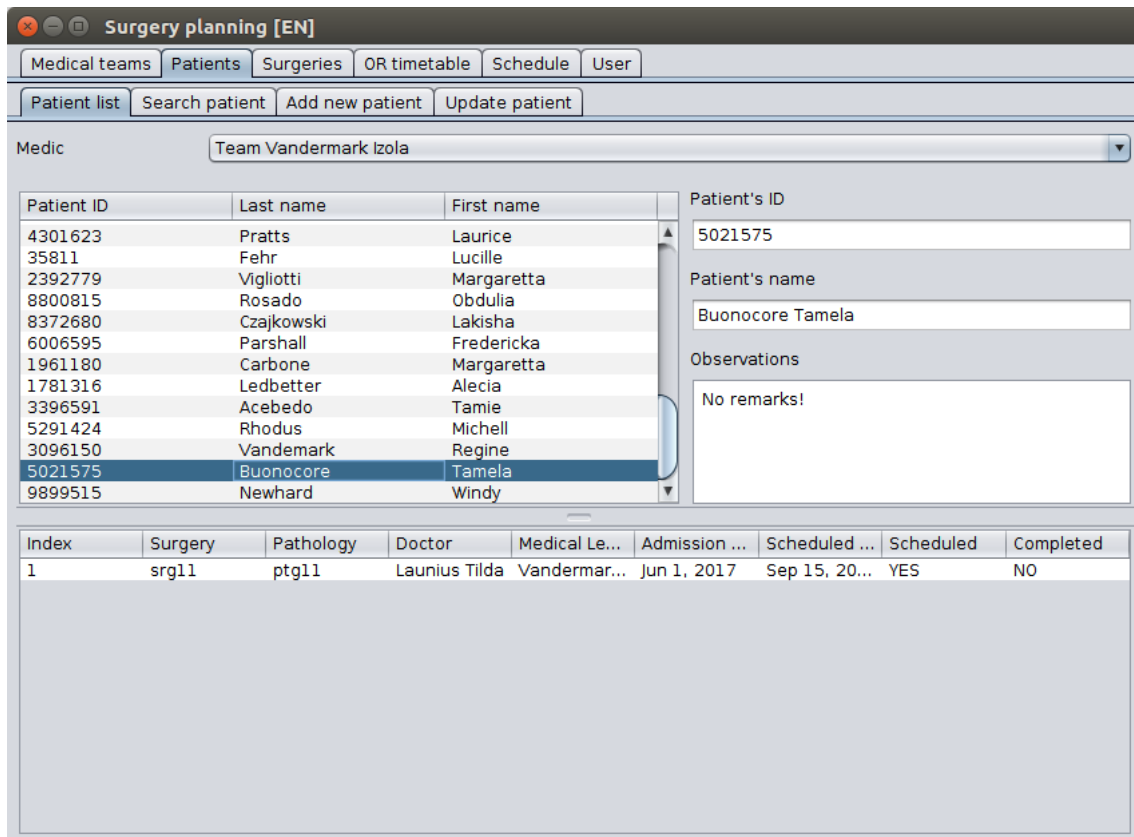


Figure 5.8: Team’s patient list and patient details in surgery planning application

The second main panel in the interface is about **patients**. First sub-panel, shown in figure 5.8, contains team’s patient list or doctor’s patient list, depending on the selection in the drop-down box. After a selection is made, the list of patients will be showed or updated in the left side of the interface.

When a patient from the list is selected, fields in the right side and the bottom table will be updated with selected patient’s details and it’s full medical history, no matter if the surgeries are already completed or just added. In this pannel, all the details about a patient, in a certain waiting list, can be seen. If editing or updating is needed, user must go in *update patient details* panel, update details and save the changes. This panel contains only patients from the waiting lists. All patients that are no longer in a waiting list, cannot be found here. Any patient that has just been admitted in the hospital or department and not yet admitted for a surgery, will not appear in this panel. Also, any patient that has completed surgeries in the past but is not admitted for a surgery at the current time, will not appear in this pannel either. A patient is admitted for a surgery only after different departments atests that the patient is healthy enough for it, (e.g. doctor in charge says that the patient needs a certain surgery, patient is not allergic to used anesthetics) and after the patient accepts it.

To see details about a certain patient, user should use *Search patient* panel, shown in figure 5.9.

Index	Surgery	Pathology	Doctor	Medical Le...	Admission ...	Scheduled ...	Scheduled	Completed
1	srg47	ptg47	Maguire Ka...	Vandermar...	Jun 1, 2017		NO	NO

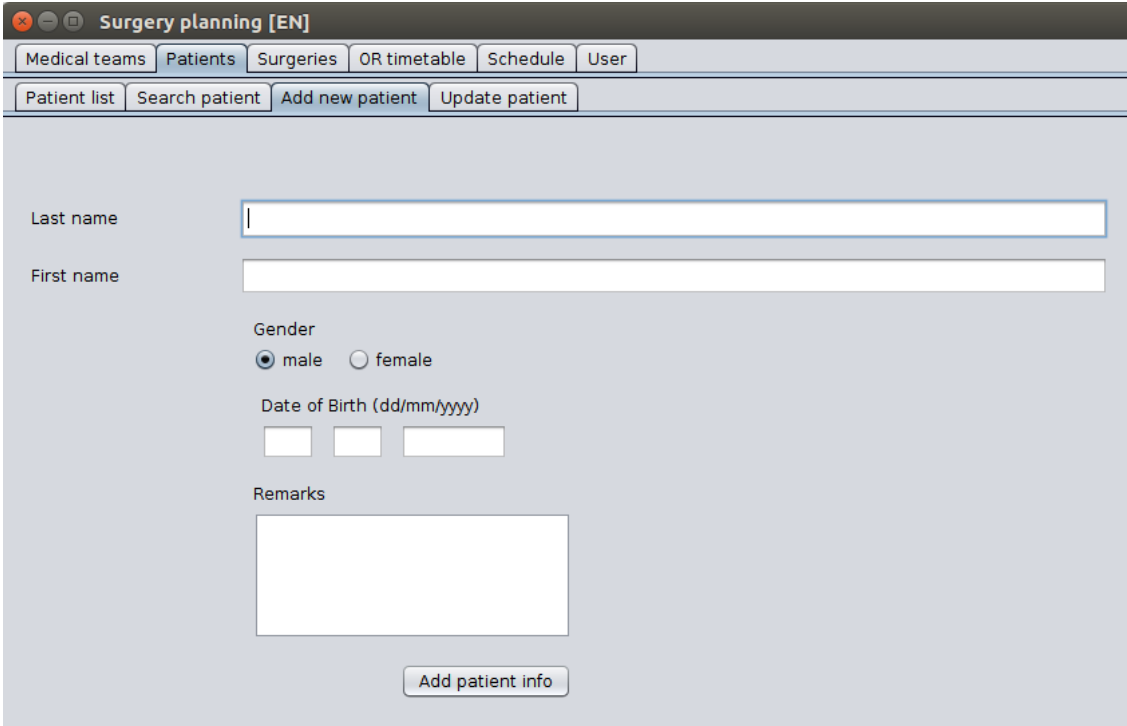
Figure 5.9: Search patient in surgery planning application

The patient can be searched in database using 2 different criteria (mapped on two different buttons):

- **patient Id** the id that had been randomly assigned to the patient in the local database when the patient was added
- **patient name** recommended full name, but using partial name (either last or first) is also acceptable. For more accurate results (in case of patients with same name), birth date should be added for the search

In case there are multiple patients that matches the given criteria, a message box will pop-up and inform the user, giving also the IDs of the patients that match the search criteria. If there are more than 10 patients that match the criteria, no Id will be shown. The user should retake the search operation giving more details. If patient's birth date shows "9999" for the birth year, that means patient's birth date was not set when patient was added in the database. It is recommended to update any uncorresponding details for an accurate result in a following search.

If a patient does not exist in the database, it can be added using *Add new patient* panel, fig 5.10. To make the Search option easier and accurate, all the fields are mandatory. If the adding of a patient is done without previously searching the patient details and if there is a patient in database that matches the input details, an error message will be prompted. The message also includes the ID of the patient that matches the given details.



The screenshot shows a web application window titled "Surgery planning [EN]". The window has a navigation bar with tabs for "Medical teams", "Patients", "Surgeries", "OR timetable", "Schedule", and "User". Below the navigation bar, there are buttons for "Patient list", "Search patient", "Add new patient", and "Update patient". The "Add new patient" form contains the following fields:

- Last name: A text input field.
- First name: A text input field.
- Gender: Radio buttons for "male" (selected) and "female".
- Date of Birth (dd/mm/yyyy): Three separate input fields for day, month, and year.
- Remarks: A large text area.
- At the bottom of the form is a button labeled "Add patient info".

Figure 5.10: Add new patient in surgery planning application

After patient details have been added in the database, a message will inform the user that the adding operation was completed successfully. All patient details should be added correctly from the beginning, but if mistakes are done, they can be corrected in *Update patient* details panel. Like in *Search patient* panel, *Update patient* panel, figure 5.11, also allows the user to search patients by name (and birth date) or by ID. This pannel allows the user to update any personal details about an existing patient.

Figure 5.11: Update patient details or add/update surgery for patient in surgery planning application

In the right side, the user can add or update surgeries for patient. *Add surgery* button will be enabled only if the *Add new surgery* option is checked. In this case, user can add one or more surgeries (only one at a time) for patient using the drop-down boxes. If *Update existing surgery* option is checked, the drop-down boxes will only contain approved surgeries that were not completed (either scheduled or not). At this point, when a surgery is selected, user can change the medical team and/or the doctor in charge or schedule/unschedule the patient. If the doctor in charge cannot complete the surgery, another doctor (from the same team or another one) can be assigned. If the patient was scheduled for surgery but unavailable on the scheduled date, user can unschedule the patient's surgery or reschedule it on another free day. Any changes should be saved using "Save surgery" button. If for some reason, the patient (either scheduled or not) will not have the surgery anymore, user can remove the surgery from patient's medical history.

Everything in "Patients" main panel is available to all users, regardless the clearance level.

The third main panel, ”**Surgeries**” is shown in figure 5.12. This panel contains a list with all surgeries that can be done in the current department/hospital.

Surgery	Pathology	Average duration (min)	Standard deviation (min)
srg1	ptg1	194	56
srg2	ptg6	177	29
srg6	ptg6	97	21
srg3	ptg3	184	26
srg4	ptg4	121	21
srg0	ptg5	115	28
srg5	ptg5	81	21
srg7	ptg7	97	21
srg8	ptg8	97	21
srg9	ptg9	85	35
srg10	ptg10	150	55
srg11	ptg11	153	23
srg12	ptg12	97	21
srg13	ptg13	150	42
srg14	ptg14	97	21
srg15	ptg15	150	42
srg16	ptg16	86	32
srg17	ptg17	97	21
srg18	ptg18	83	22
srg19	ptg19	121	21
srg20	ptg20	103	40
srg21	ptg21	105	24
srg22	ptg22	104	30
srg23	ptg23	110	43
srg24	ptg24	85	35
srg25	ptg25	153	23
srg26	ptg26	145	38
srg27	ptg27	145	38
srg28	ptg28	143	24

Figure 5.12: List of available surgeries and their duration in surgery planning application

Every surgery has assign to it:

- **pathology** name of the usual pathology
- **average duration** the average duration of the surgery [in minutes]
- **standard deviation** the standard deviation of the surgery from the average [in minutes]

Same as in the other pannels, names are not real. The values on the other hand, are computed for each surgery by using the hospital’s history data from the last two years [9]. Buttons are enabled for all users with clearence level grater or equal with ”MEDIC”. They offer the possibility to add new surgeries, update the timings for the existing ones or remove surgeries from database. Removing surgeries can only be done if there is no doctor assign to them and no patient admitted for this surgery and not performed yet.

The forth panel, "OR timetable" contains all the available operating rooms in the operation theatre for non-elective patients. Each sub-panel contains daily bookings for medical teams. These panels are enabled for editing only for the Head of Department. The other types of user can only see the bookings for any operating room.

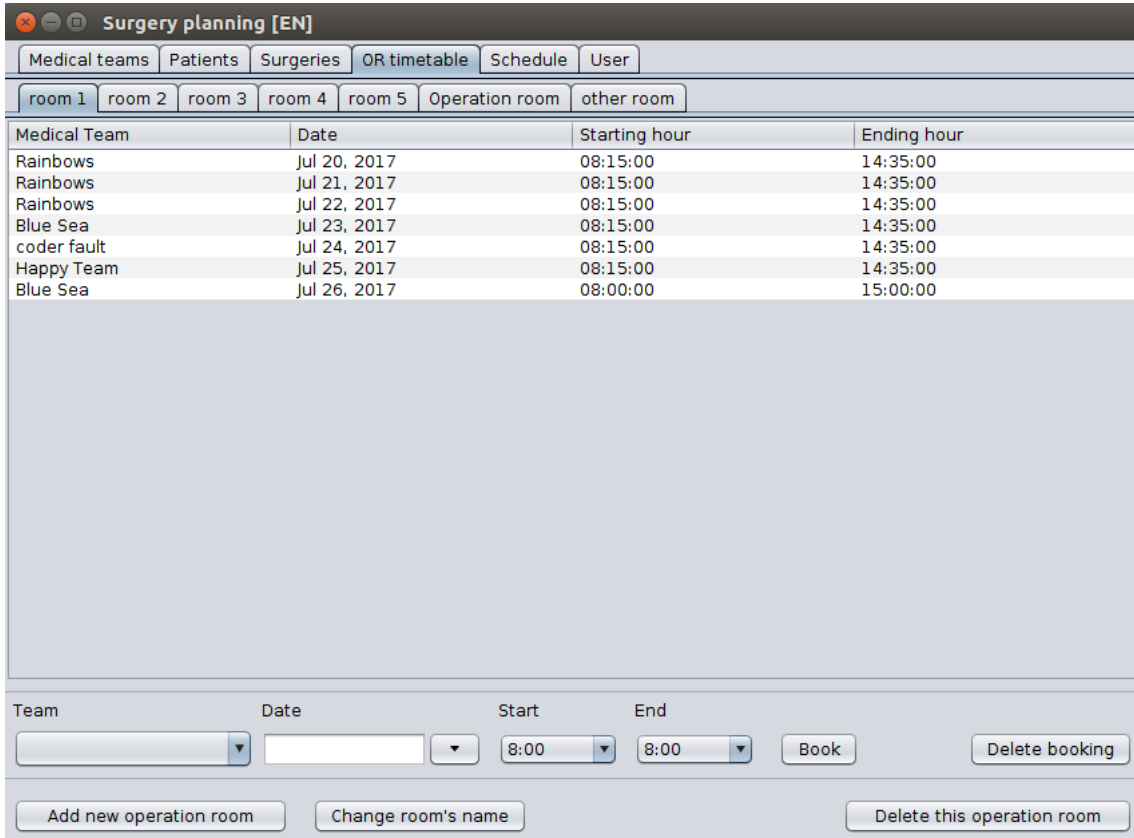


Figure 5.13: Operating room booking details in surgery planning application

Each operating room has a default name that can be changed using the corresponding button. A team can be booked in the selected operating room only if:

- there is no other team booked in the desired date
- team is not booked in the desired date on another operating room

Starting and ending availability time of the operating room can be chosen after selecting the team and date. After all the fields are filled correctly (this includes: ending time greater than starting time), pressing "Book" button will add a new booking in the operating room's timetable. It also informs the user that booking was done successfully. There is no *edit* option for existing bookings. To do so, the wrong booking must be deleted, then create another booking. To *delete* an existing booking, a pop-up window will request for the date of the booking that should be deleted.

It also specifies the accepted date format. After the user files in the date to be deleted, a confirmation message will be prompted. If the date is not valid but in a close format, the application might interpret it as a different date (e.g. 2017-01-32 might be interpreted as 2017-02-01).

There are two types of operation rooms available: Morning and Afternoon. The difference between them is only about the timing. A morning operation room can be available between 8:00 and 17:55, but a afternoon operation room can be available only between 14:00 and 20:55. Regardless the type, an operating room can be deleted only if there is no team booked from the current date forward and if it is not the only one left.

Regarding the main reason of this application, the fifth panel, "**Schedule**", allows to any user with a clearance level of at least "COORDINATOR" to schedule patients, automatically. The sub-panel, "Create schedule" shown in figure 5.14 allows the user to plan and schedule patients for a certain number of days.

The maximum number of days is automatically updated in the corresponding text box when selecting a medical team. For the scheduling program to start, the number of days to schedule has to be at least equals with 1. Another parameter needed for planning and scheduling patients, is a percentage of the desired occupancy rate. From studying the last 2 years of medical data, a good percentage of the occupancy rate is 80%. The rest of 20% is allocated for cleaning time between surgeries and for possible overtime of the scheduled surgeries. This percentage can be changed before starting the scheduling program. It is also necessary a deviation rate, in percentage. This parameter allows the scheduling program to accept or deny possible solutions if the absolute difference between obtained occupancy rate and desired one is greater than it. After the team is selected and all parameters are between boundaries, a schedule can be ordered. The result will appear in the text area below schedule button. If the scheduling algorithm has at least one day with patients planned and schedule, the result message will contain the operating room's name, scheduled date and the IDs of the patients that were scheduled.

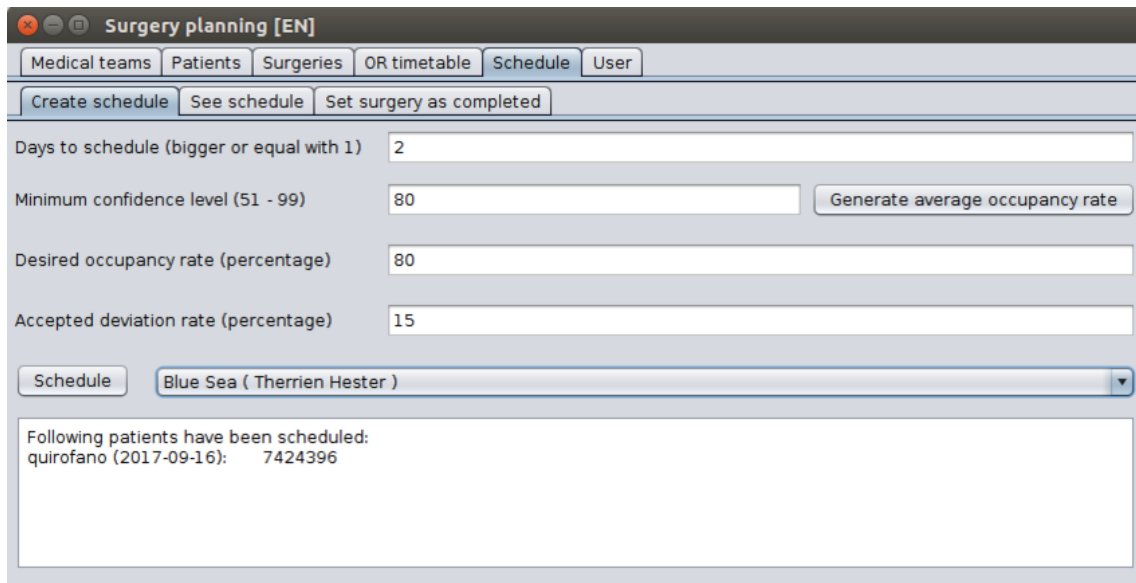


Figure 5.14: Creating a new schedule from a patient list in surgery planning application

To see scheduling details for a certain date, a operation room or medical team must be selected. The calendar will only show the available dates for the selected operating room or medical team. An example can be found in figure 5.15. In the bottom of "See schedule" panel, can be found the total estimated time for selected day, along with a percentage from total day time, also shown below.

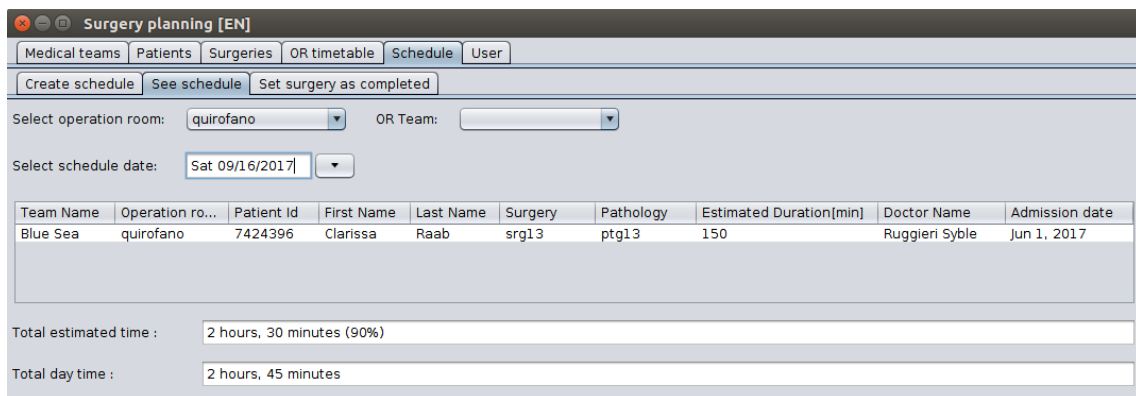


Figure 5.15: See schedule details for a medical team or operating room in surgery planning application

After a surgery is performed, it should be checked as completed. Sub-panel "Set surgery as completed", as shown in figure 5.16, allows user to do so. As a first step, user should select the day in which patient was scheduled. To set a surgery as completed, user has to first select the patient and then press the "Set completed" button. An informing message will be shown before patient's surgery to be set as

completed. At this point, user can either cancel the operation or accept the action of setting it as completed.

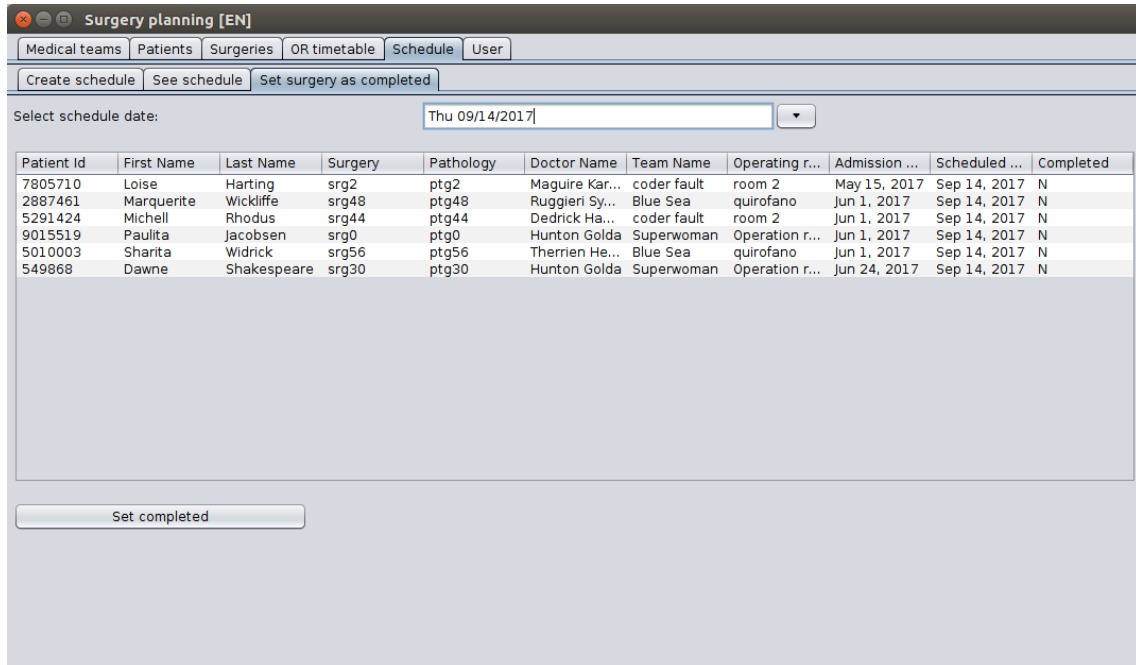


Figure 5.16: Set patient’s surgery as completed in surgery planning application

The last main panel contains only "User" related operations like: Create user (figure 5.17), delete user (figure 5.18), manage user (figure 5.19) and logout (figure 5.20). Only the Head of Department can create a new user or delete an existing one. When creating a new user, the default value for clearance level of the new user, is "MEDIC".

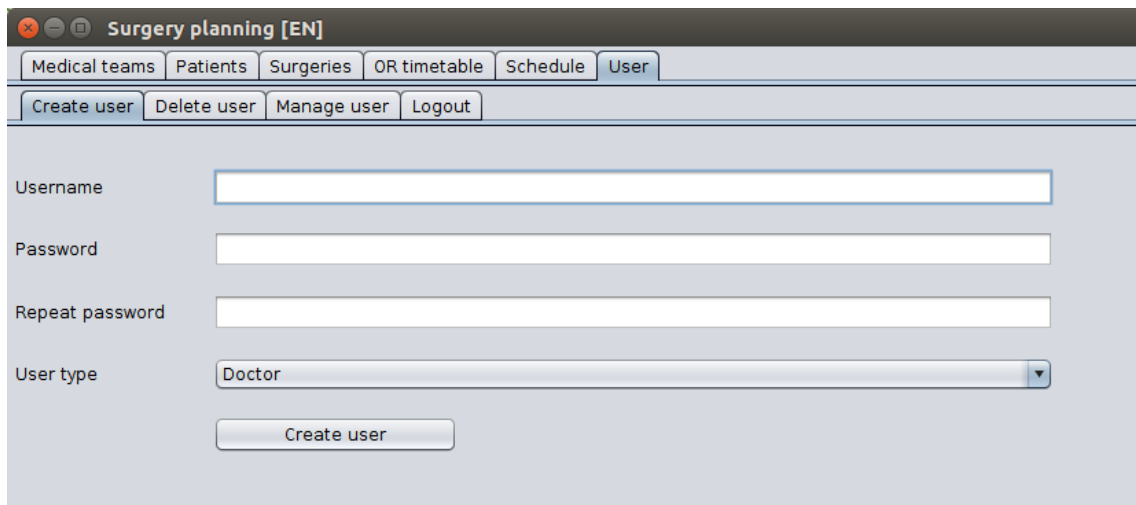


Figure 5.17: Panel for creating a new user in surgery planning application

If an user should no longer have access to the application, Head of Department can delete the user and it's details (password and clearance level) from database. Once the operation is successfully completed, the user that had just been deleted, cannot connect into a following session of the application. If the user does not exist or cannot be deleted, an error message will be prompted.

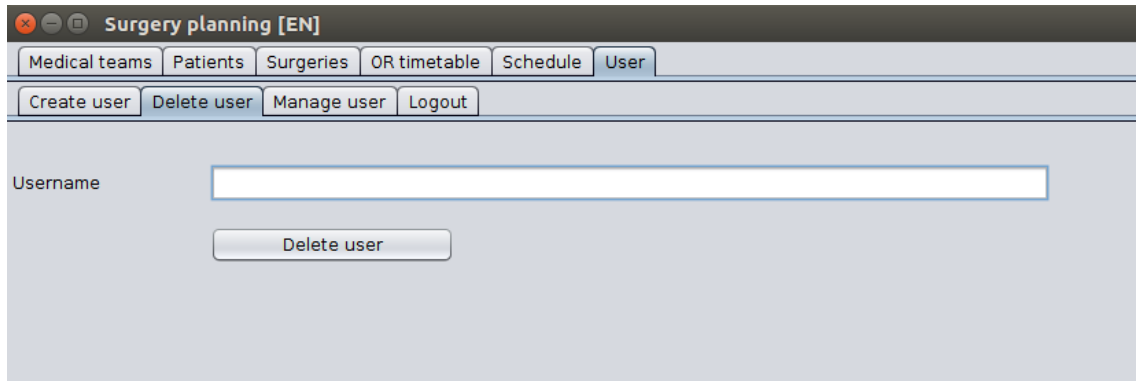


Figure 5.18: Panel for deleting an existing user in surgery planning application

If the user wants to change it's personal details (password), user can do it in panel "Manage user" sub-panel. For changing the password of it's own account, user needs to put the current password along with the new one. The admin user can change password of any other user without needing the current password of that user, but this is not available in the case of changing it's own password. The possibility of changing the password of any other user (if the current user is an admin = Head of Department) is a power that should be used with maximum responsibility.

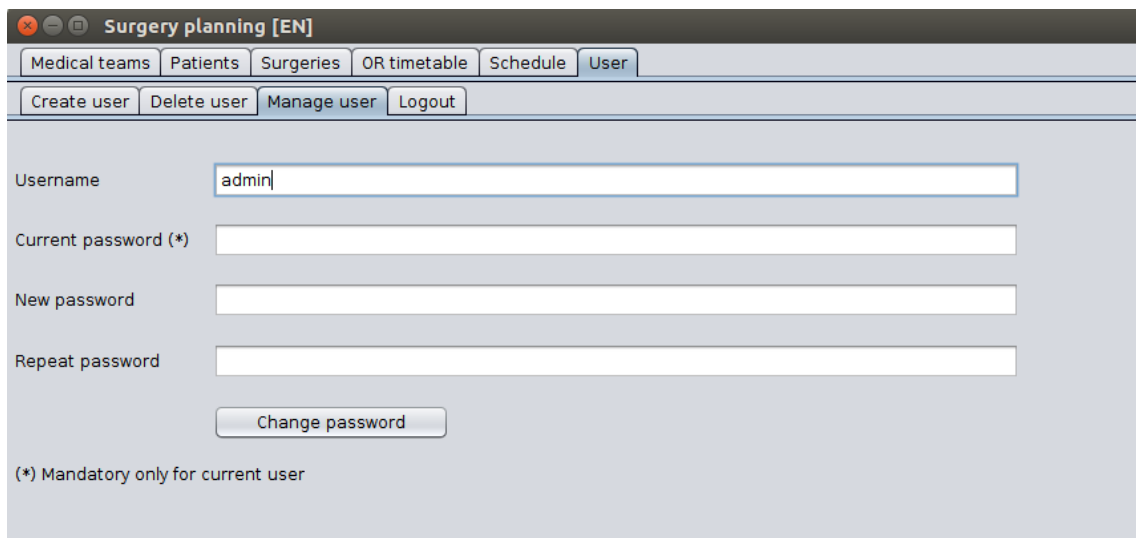


Figure 5.19: Panel for managing users in surgery planning application

The last sub-panel contains only the logout button. Application cannot be closed through regular closing option regarding security reasons. Thus, to close the application, user must logout first.

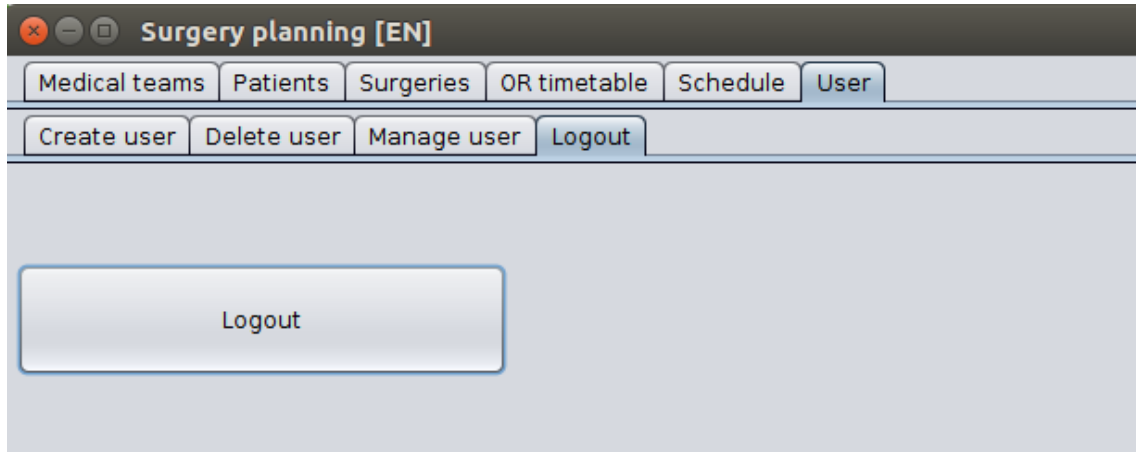


Figure 5.20: Logout panel in surgery planning application

This page is intentionally left blank.

This page is intentionally left blank.

Chapter 6

Simulation Results

This chapter shows some simulation results for the abobe described algorithms. The list of surgeries to be planned and scheduled will be the same for all simulations and it is presented in table 6.1. The values from the table are real, computed from hospital's medical history [9]. For protecting patients' personal data and also keeping the table only with necessary informatons, table contains:

- **ID** the id of the patient that has to be scheduled for surgery
- **avg** the average duration [in minutes] for patient's surgery
- **std** the standard deviation [in minutes] for patient's surgery

The planning and scheduling for the given list of patients will be done with both algorithms, as well as with different values of parameter β .

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
avg	112	110	133	103	97	133	58	150	45	45	81	45	45	104
std	24	23	24	40	21	24	17	55	12	12	21	12	12	30

ID	15	16	17	18	19	20	21	22	23	24	25	26	27	28
avg	88	81	153	103	83	45	97	121	203	184	97	97	97	163
std	23	21	58	40	22	12	21	21	74	26	21	21	21	44

ID	29	30	31	32	33	34	35	36	37	38	39	40	41	42
avg	97	81	97	121	58	121	133	202	97	153	45	121	110	155
std	21	21	21	21	17	21	24	45	21	23	12	21	43	48

ID	43	44	45	46	47	48	49	50	51	52	53	54	55	56
avg	55	83	150	104	85	97	97	97	58	97	97	81	133	95
std	15	22	42	30	35	21	21	21	17	21	21	21	24	21

ID	57	58	59	60	61	62	63	64	65	66	67	68	69	70
avg	45	91	111	145	97	75	145	97	55	45	156	133	81	75
std	12	22	23	27	21	23	38	21	23	12	42	24	21	23

ID	71	72	73	74	75	76	77	78	79	80	81	82	83	84
avg	192	75	126	50	103	137	71	75	90	113	105	63	111	156
std	55	23	29	24	40	25	25	23	19	30	21	17	23	42

ID	85	86	87	88	89	90	91	92	93	94	95	96	97	98
avg	103	153	97	97	155	111	121	121	149	83	184	133	111	94
std	40	23	21	21	48	23	21	21	38	22	26	24	23	20

Table 6.1: Waiting list of patients used for simulations

6.1 MILP

For the above list of patients, we'll show results of 3 MILP simulations, with $\beta = 0.333$ (table 6.2, priority given to operating room occupancy rate), $\beta = 1$ (table 6.3) and $\beta = 2$ (table 6.4, priority given to patient list). The desired occupation rate given as an input is 80%.

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients	Absolute deviation [min]
1	340	80.952	1 2 4	4
2	336	80.0	0 3 21	0
3	336	80.0	5 22	0
4	335	79.761	7 10 13	1
5	337	80.238	16 23	1
6	336	80.0	8 9 18 27	0
7	335	79.761	11 14 35	1
8	336	80.0	12 20 24 25	0
9	337	80.238	15 17 37	1
10	467	111.190	6 31 34 41	131
11	336	80.0	19 26 28 30	0
12	339	80.714	33 36 39	3
13	335	79.761	29 44 45	1
14	336	80.0	32 40 43 46	0
15	336	80.0	38 47 48 49	0
16	336	80.0	50 54 59	0
17	336	80.0	51 52 56 60	0
18	336	80.0	42 55 58 61	0
19	337	80.238	62 70	1
20	334	79.523	53 63 66	2
21	336	80.0	64 65 67 74	0
22	338	80.476	69 72 75	2
23	335	79.761	57 68 73 79	1
24	336	80.0	71 80 83	0
25	337	80.238	85 94	1
26	337	80.238	76 82 88	1
27	336	80.0	78 86 92	0
28	338	80.476	77 81 84 87	2
29	336	80.0	90 91 97	0
30	335	79.761	89 95 99	1

Table 6.2: Simulation results for the above patient list using MILP, $\beta = 0.333$

Solution cost: 1813.31099
Total time: 1880.068 [sec]

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients	Absolute deviation [min]
1	340	80.952	1 2 4	4
2	336	80.0	0 3 21	0
3	336	80.0	5 22	0
4	335	79.761	7 10 13	1
5	337	80.238	16 23	1
6	336	80.0	8 9 18 27	0
7	339	80.714	6 15 17 20	3
8	335	79.761	11 14 35	1
9	336	80.0	12 24 25 26	0
10	339	80.714	28 31 33	3
11	333	79.285	19 30 32 34	3
12	331	78.809	29 36 37	5
13	335	79.761	39 40 45	1
14	335	79.761	41 43 47	1
15	338	80.476	42 44 54	2
16	336	80.0	38 48 49 51	0
17	335	79.761	46 50 52 55	1
18	337	80.238	53 58 59	1
19	337	80.238	62 70	1
20	334	79.523	56 66 67	2
21	338	80.476	57 60 61 69	2
22	336	80.0	63 64 68 74	0
23	338	80.476	71 72 75	2
24	336	80.0	77 80 83	0
25	337	80.238	76 79 85	1
26	335	79.761	65 78 84 86	1
27	335	79.761	73 81 82 89	1
28	337	80.238	88 98	1
29	333	79.285	92 94	3
30	339	80.714	87 90 91	3

Table 6.3: Simulation results for the above patient list using MILP, $\beta = 1$

Solution cost: 5196.9999
Total time: 353.883 [sec]

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients	Absolute deviation [min]
1	325	77.380	0 1 3	11
2	334	79.523	2 4 13	2
3	336	80.0	5 22	0
4	334	79.523	7 23	2
5	337	80.238	6 8 10 16	1
6	335	79.761	9 14 15 21	1
7	336	80.0	11 12 18 27	0
8	333	79.285	17 20 34	3
9	336	80.0	19 24 25 26	0
10	323	76.904	31 35	13
11	331	78.809	28 29 37	5
12	334	79.523	32 33 41	2
13	328	78.095	30 39 40	8
14	330	78.571	36 43 44	6
15	334	79.523	45 47 54	2
16	336	80.0	38 48 49 51	0
17	335	79.761	46 50 52 55	1
18	337	80.238	53 58 59	1
19	338	80.476	42 57 70	2
20	339	80.714	60 62 63	3
21	334	79.523	56 66 67	2
22	337	80.238	61 64 68 72	1
23	335	79.761	65 75 85	1
24	334	79.523	69 74 83	2
25	335	79.761	71 80 88	1
26	337	80.238	73 76 79 84	1
27	335	79.761	77 82 92	1
28	337	80.238	78 81 94	1
29	339	80.714	86 90 91	3
30	341	81.190	87 89 95	5

Table 6.4: Simulation results for the above patient list using MILP, $\beta = 2$

Solution cost: 9895.0

Total time: 66.375 [sec]

As can be seen from the results, a good compromise between the two contradictory objective in a reasonable time (aproximately one minute), can be obtained with $\beta = 1$.

6.2 MIQCP

For the above list of patients, we'll show results of 2 MIQCP simulations, with $\beta = 1$, table 6.5 and $\beta = 2$, table 6.7. The given input confidence level is 80%.

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients
1	331	78.809	1 2 14
2	326	77.619	0 5 10
3	334	79.523	7 23
4	321	76.428	3 4 21
5	316	75.238	16 27
6	335	79.762	34 35
7	333	79.285	18 20 37
8	307	73.095	13 22
9	321	76.428	17 24 31
10	323	76.904	15 33 39
11	310	73.809	6 25 41
12	327	77.857	26 28 54
13	323	76.904	29 30 59
14	315	75.0	40 42 44
15	325	77.380	36 43 62
16	304	72.380	8 32 45 47
17	327	77.857	48 49 67
18	331	78.809	51 52 75
19	328	78.095	57 58 72
20	306	72.857	9 46 53 55
21	340	80.952	66 94
22	312	74.285	11 61 70
23	331	78.809	60 68 85
24	331	78.809	63 79 90
25	319	75.952	50 80 83
26	334	79.523	78 82 95
27	331	78.809	92 98
28	321	76.428	12 88 91
29	306	72.857	19 69 71 89
30	304	72.380	38 56 74 96
31	310	73.809	65 76 86 87
32	307	73.095	64 77 93 97
33	257	61.190	81 84 99

Table 6.5: Simulation results for the above patient list using MIQCP, confidence level of 80% and $\beta = 1$

Solution cost: 6906.6
Total time: 23628613 [sec]
Proposed Occupation rate for MILP: 74.2016[%]

Day number	Occupation time [min]	Occupancy rate[%]	Order number of scheduled patients
1	326	77.619	0 2 10
2	331	78.809	1 5 14
3	334	79.523	7 23
4	321	76.428	3 4 21
5	316	75.238	16 27
6	307	73.095	13 22
7	335	79.761	34 35
8	333	79.285	18 20 37
9	321	76.428	17 24 31
10	323	76.904	15 33 39
11	310	73.809	6 25 41
12	327	77.857	26 28 54
13	323	76.904	29 30 59
14	305	72.619	8 40 44
15	325	77.380	36 43 62
16	304	72.380	9 32 45 47
17	327	77.857	48 49 67
18	322	76.666	42 58 66
19	331	78.809	51 52 75
20	306	72.857	11 46 53 55
21	312	74.285	12 61 70
22	320	76.190	60 63 72
23	334	79.523	57 78 85
24	340	80.952	83 94
25	329	78.333	79 80 82
26	331	78.809	92 98
27	321	76.428	19 88 90
28	329	78.333	69 91 95
29	311	74.047	50 68 71 86
30	304	72.380	38 56 74 89
31	311	74.047	84 87 96
32	307	73.095	64 77 93 97
33	275	65.476	73 76 81 99

Table 6.6: Simulation results for the above patient list using MIQCP, confidence level of 80% and $\beta = 2$

Solution cost: 7012
Total time: 8167.364 [sec]
Proposed Occupation rate for MILP: 74.201[%]

This page is intentionally left blank.

Chapter 7

Conclusions & Recommendations

In conclusion, this thesis was focused on creating an easy-to-use application for surgeons and auxiliary staff (e.g. assistants) that will be used for planning and scheduling non-elective patients for surgeries (non-urgent patients).

The main achievements for a medical departments of using such an application are:

- optimise resource usage in an operation theatre (OT) by keeping the occupation rate of each operating room at a stable given level (in percentage)
- compute an optimal occupancy rate based on a confidence level of not overcoming the total day time
- automatic update of the estimated duration of all available days for OT
- prevent human mistakes (such as multiple allocation of a team, for the same day, in different operating rooms)
- more time for doctors to take care of patients
- keep track of newer and older patients in the waiting lists as well as previous patients

As a further scope of work, this application can be used at hospital level and not just department level. This will allow multi-department teams to be formed, with their own waiting list of patients. Also, using a database at hospital's level, will prevent a patient from having multiple IDs because the patient can be found in different databases from different departments.

This page is intentionally left blank.

Bibliography

- [1] D. Clavel, C. Mahulea, J. Albareda, M. Silva, "Operation planning of Elective Patients with a minimal confidence level of not overcoming the total time", 2017
- [2] E. Demeulemeester, J. Baliën, B. Cardoen, "Operating room planning and scheduling: A literature review", *European journal of operational research*, pp. 921-932, vol. 201, no 3, pp. 2010
- [3] W. E. Spangler, D. P. Strum, L. G. Vargas, J. H. May, "The surgical scheduling problem: Current research and future option", *Production and operations management*, pp. 392-405, 2011
- [4] A. Godhawale, "Introduction to planning and Scheduling", *projectcontrolsonline.com*, London, UK, 2010
- [5] C. D. Spyropoulos, "AI planning and scheduling in the medical hospital environment", *Artificial Intelligence in Medicine*, pp. 101-111, 1 10 2010
- [6] B. Daily, M. Egan, W. S. Sandberg, "Deliberate perioperative systems design improves operating room throughput", 2005
- [7] Jeffrey. A. Hermann, "A history of production scheduling", in *Handbook of production scheduling*, Springer US, 2006, pp. 1-22
- [8] IBM, CPLEX manual, 2014
- [9] D. Clavel, C. Mahulea, J. Albareda, M. Silva, "Operation Planning of Elective Patients in an Orthopedic Surgery Department", 2016

This page is intentionally left blank.