Determining a safety stock level for the waiting list of the KNO department at the UMC Utrecht

Author:	Lucas van Haandel	(S2628449)
Supervisor:	S. Rachuba	
UMC Utrecht supervisor:	A. Glerum	

Acknowledgements

I want to thank everyone at the capacity management team at the UMC Utrecht for the continuous support during my stay in with them.

Management summary

The objective of this research is to find out how to calculate the necessary safety stock a surgical specialty should have on the waiting list to prevent stockouts in the operating room. The main goal of this research is to improve the resource capacity planning of operating rooms in hospitals at a tactical level. This is done by answering the research question:

What is the necessary safety stock in hours of work for the KNO department at UMC Utrecht to ensure a prespecified OR utilization using the MSS for June 2024?

This question is answered by first exploring literature on the evaluation methods of Master surgery schedules (MSS), and seeing what methods can be adapted to calculate a safety stock. Then the problem situation at the UMC Utrecht is analysed to give context to the scope of the models , and to gather the relevant data necessary for the model inputs. Then we build the most promising models, on which we base our conclusions.

The evaluation techniques that seemed the most promising are a Markov model and discrete event simulation. The Markov model was applied by modelling the waiting list as a Markov chain in a macro-enabled Excel file, and analysing the waiting list once a steady state was reached. The discrete event simulation was applied by modelling a simplification of the admissions planning process, and analysing the waiting list once the simulation had been completed.

The most important finding is that a Markov model is the best out of the two methods to analyse surgical waiting lists. Verification and validation of a discrete event simulation is very difficult. There are a lot of cases where there is no data or no formal strategy, so it becomes more difficult to make sure the model resembles reality. The Markov model does not have this issue, because it only uses the transition probabilities, so the only necessary input data, apart from the MSS, are the patient arrival distribution and the surgery duration distribution. The Markov model is the only model that was found that could be validated.

The recommended safety stock level in hours of work for the KNO department at the UMC Utrecht depends on the desired expected OR utilization, called the service level, and is given in FiFigure 1. The necessary safety stock level and the prespecified service level have an exponential relationship, with the necessary safety stock level steeply increasing when the service level exceeds 90%.

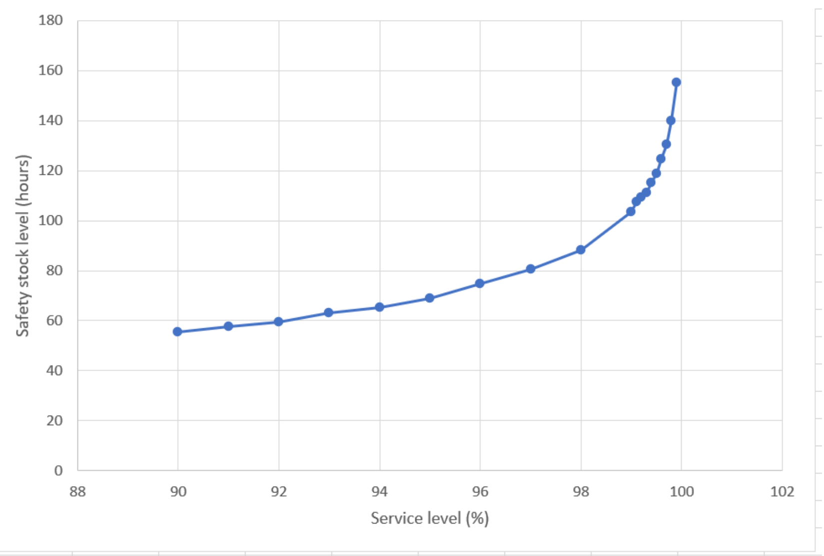


Figure 1 The necessary safety stock level to achieve a prespecified service level

The next step for the UMC Utrecht is to calculate the safety stock level for all their surgical departments, so they can use the data to improve their tactical planning decisions. This safety stock level is also useful when a hospital wants to implement a dynamic master surgery schedule. The safety stock can then be used to monitor whether a surgical specialty could use more or less OR hours, and if a surgical specialty can be expected to utilize the OR enough using a proposed new MSS.

Contents

Table of figures

Operating room planning

It is estimated that over one third of health expenditures can be attributed to waste (Oecd, 2017).

About 40 percent of a hospital's expenses come from their operating theatre (OT) (Marrin et al., 1997). It is a general objective that the OT gets used as optimally as possible while upholding a good quality of care. A substantial sub objective is to schedule operations as optimally as possible. Because operating room (OR) planning is a complex problem with many constraints (patients, staff, materials, post operation bed availability, etc.), an optimal schedule is impossible to make. Instead, hospitals often split OR planning into a strategic, tactical, and operational component. This was expanded upon by (Hans et al., 2012) Into a framework for healthcare planning and control, as shown in Figure 2.

On a strategic level, Structural decisions, like policies and company strategy are decided. For example, hospitals decide how many of a type of surgery they plan to do in a year, and how much time they expect each surgery to take.

The operational level involves short-term decisions that are meant to execute the healthcare delivery process. It involves both an offline – in advance- and an online – reactive- part. An example of offline operational planning is surgery scheduling. An example of online operational planning is handling emergency arrivals, or other unforeseen complications that arise.

Tactical planning involves actions that are in between strategic and operational planning in scope and planning horizon. Because it involves a larger planning horizon than operational planning, tactical planning decisions rely more on trends and patterns than operational planning. Decisions are also more flexible, as they are made further in advance. For example, in tactical planning surgical schedules are made that allocate surgery time in blocks to surgical specialties, without specifying what surgeries will be performed. In operational planning, these blocks are filled in with surgeries.

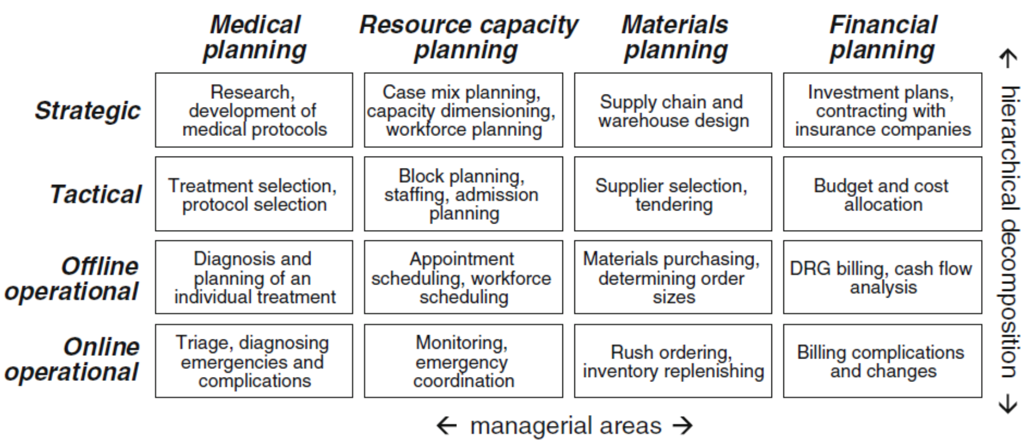


Figure 2 A framework for healthcare planning and control (Hans et al., 2012)

Planning at UMC Utrecht

The UMC Utrecht is exploring to improve the current Tactical planning process by introducing a dynamic master surgery schedule (MSS), where most of the OR time is assigned to specific specialties, but some of the OR blocks are kept unassigned. These blocks can then later be assigned to the specialty that needs it the most. This introduces a trade-off between stability and flexibility where leaving more slots unassigned leads to the slots being allocated to the specialties that need it most, but there is less time to schedule patients and doctors.

Research shows that introducing even a small amount of flexibility into the MSS leads to great improvements in OT performance (Oliveira & Marques, 2021). The downside of introducing flexibility is that it is known much later what surgery will take place where. This is important to stakeholders such as surgeons and surgical staff.

Deciding what part of the MSS should be stable is a strategic decision that would take too long to research in this project, and is part of a multi year project the UMC Utrecht is working on to change tactical planning.
Deciding when a specialty should receive an unassigned OR block is mostly answered by rules in place through quality of care purposes. The patients should not wait for so long that their health deteriorates while waiting for surgery. So when planners see these waiting times tick up, they should receive extra operating sessions.
Deciding when a specialty does not need an unassigned OR block is the problem that this project focuses on. This decision depends on many factors. Some of these are: the number of surgeries the specialty is budgeted to fulfil, the number of surgery hours on the waiting list, the expected change to the specialties waiting list considering the current schedule, and the UMC Utrecht's strategic position.
Because these decisions depend on many factors and have an impact on the surgery scheduling at a tactical level, they impact many different stakeholders at the hospital. The tactical planners have to come to a consensus that the decision being made is the best one for all the stakeholders involved. This research project assists in this decision making process by presenting a safety stock level for the patient waiting list length. This safety stock is the lowest necessary number of hours of work needed on the waiting list to ensure there is enough work to service the given OR hours. This tells the tactical planners if their current waiting list is large enough for their OR hours, or if they can take away OR hours and still ensure timely surgeries. When the safety stock level for the surgical specialties is known, the tactical planners can base their planning decisions on data, which is necessary when a consensus needs to be reached in a timely manner.

Problem definition

The main management problem is:

Operating rooms are not assigned to surgical specialties optimally.

This problem Has many causes, part of whom are given in the problem cluster in Figure 3. The MPSM method is used to find the best fitting core problem (Heerkens & Van Winden, 2011.). This problem cluster only focuses on causes in tactical planning, because different causes fall outside the scope of this research.

Suboptimal tactical planning is caused by either a suboptimal MSS, or by not adjusting the MSS to the current situation. A suboptimal MSS can have many causes. Generally, it is either caused by using a suboptimal modelling technique, or by using the same MSS for too long, and not changing it for seasonality or other dynamic changes, such as surgeon availability or holidays. The MSS does not get adjusted to the current situation properly for two reasons. Either the MSS does not have enough flexibility and cannot change, or the proposed changes in the MSS made during the tactical planning meeting were not the optimal decisions.
Suboptimal decision making during the tactical planning meeting also has many causes, the most valuable of which is *'Safety stock for the waiting lists is unknown'.* This is the chosen core problem for this research.

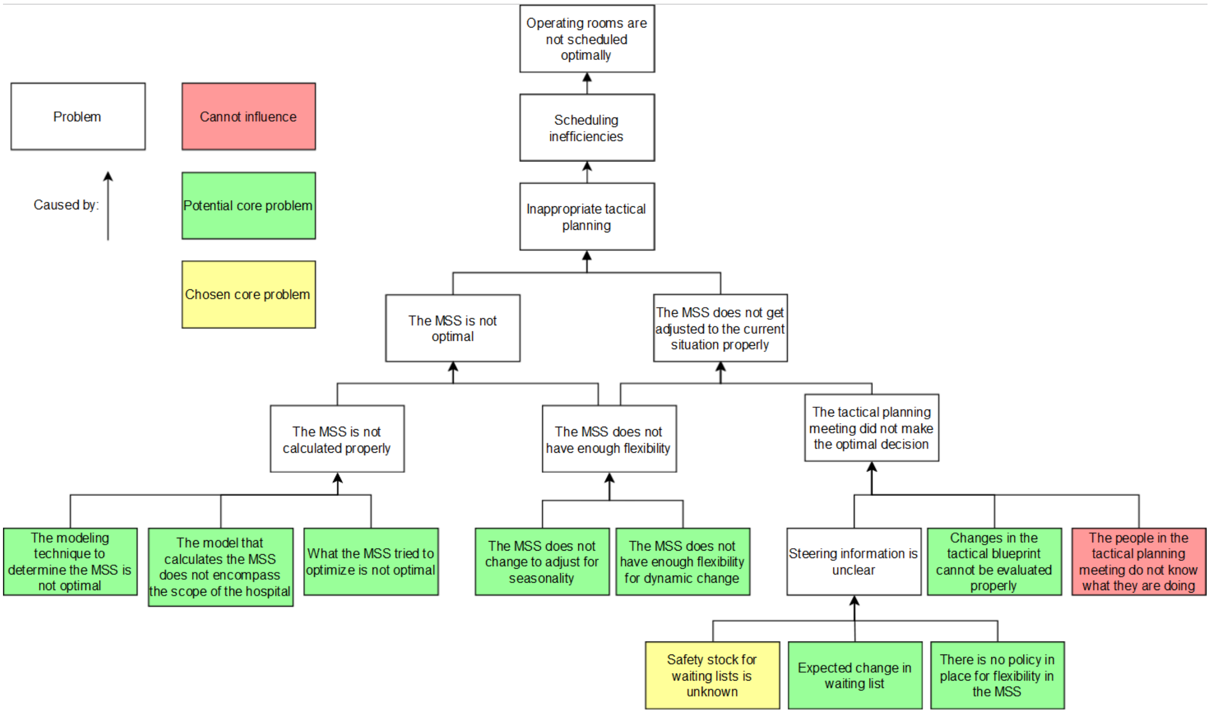


Figure 3 Problem cluster for operating room scheduling inefficiencies caused by tactical planning mistakes.

This problem is both relevant to UMC Utrecht and unexplored in literature. To make sure the problem fits in the scope of a bachelor's assignment, this project focuses on calculating safety stock levels for the patient waiting list for the KNO department at UMC Utrecht. The result from this project can be used to make informed planning decisions based on patient waiting lists.

Global problem solving approach

Research aim

The goal for this research is to calculate safety stock levels for patient waiting lists for surgery specialties based on key performance indicators (KPI's). The results from this project can be used to support decision making for flexible surgery scheduling.

The knowledge question associated with this research aim is:

What is the necessary safety stock in hours of work for the KNO department at UMC Utrecht to ensure a prespecified OR utilization using the MSS for June 2024?

Research questions

The problem is broken down into 4 stages, given in Figure 4. First we have to understand the current situation. There is no clear idea on what service levels are desired, nor do we know the characteristics of patients undergoing surgery. The second step is to review literature, both on safety stock and on techniques to evaluate MSSs. This will show the possible methods that exist to calculate waiting list lengths for surgical specialties, and what needs to be added to calculate a safety stock level. The available model methods that seem to fit the problem context best are adapted to the context of safety stock in the solution design, where different scenarios will be tested. After that key insights and recommendations will be given.

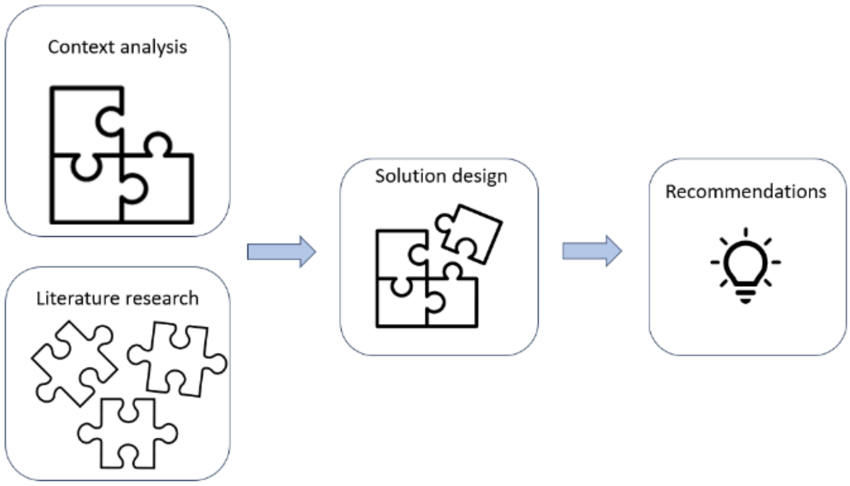


Figure 4 Graphical view of the solution approach

The literature review will be researched through a narrative literature review for each research question. The research questions for the literature research are:

1. *What techniques to evaluate the impact of an MSS exist?*
2. *How is safety stock calculated, and how does this translate to a hospital setting?*

The goal for the literature review is to describe theories, models, and frameworks developed in past studies. A narrative literature review is used (King & He, 2005). We choose this method because the number of papers with different models is limited, and most models are adaptations from one another. The literature review will create a toolbox with possible methods to model tactical planning and to calculate safety stock. This implies that this project is limited by the models currently available in literature, but the models can be adapted to fit the research goal. Research questions 1 and 2 are answered in the chapter ‘Theory’.

The context analysis will be researched through semi-structured interviews with employees at UMC Utrecht that are involved in the tactical planning process, and by using data from UMC

Utrecht. The analysis will deliver an analysis and statistical distributions for these main patient characteristics. The research questions for the context analysis are:

3. *What are the main patient characteristics?*
    a. *What are the statistical patient arrivals?*
    b. *What are the statistical surgery durations?*
4. *Wat strategy is used to plan patients in OR blocks at the KNO department at the UMC Utrecht?*

The context analysis is given in the chapter 'System description'. The research questions for the context analysis are both descriptive questions. They are all meant to elaborate on the current planning policy and situation at UMC Utrecht. These semi-structured interviews focus both on answering the specific questions, while also allowing the interviewees to expand and delve into aspects that they consider important, but are not explicitly mentioned. Semi-structured interviews allow for the collection of both quantitative and qualitative (questions 3 and 4, respectively) data (Dicicco-Bloom & Crabtree, 2006). Research questions 3 and 4 are answered in the chapter 'System description'.

The research questions regarding the solution design and recommendations are answered by estimating what type of method to evaluate the MSS works best for the situation, building this method, and testing scenarios and configurations. The research questions for the solution design and recommendations are:

5. *What models work best to estimate the necessary safety stock levels for surgical departments?*
6. *What is the relationship between OR performance and safety stock levels?*
7. *What are the practical insights gained from these models?*

The research questions for the conclusions and recommendations answer the original problem statement for the project. The answer to question 5 validates the information provided in question 6. This question gives the insight necessary to answer question 7. To answer question 6, experiments need to be conducted. Question 5 is answered by evaluating the quality of the models designed during the project. Research questions 5, 6, and 7 are answered in the conclusion.

Relevance and scope

There are no papers on cases where safety stock was calculated for operating rooms. Only on theoretical models, as part of a simulation-optimization approach, expected waiting list lengths have been calculated (Razali et al., 2022). These waiting list lengths are not calculated using very robust methods, as the actual values are not important for optimization. It is only important to see whether the expected waiting list length goes up or down when making changes in the MSS. The specific methods used to calculate these waiting list levels are only vaguely explained and not reproducible (Abedini et al., 2017; Kumar et al., 2018; Oliveira et al., 2022). This project bridges the gap between studies that focus on theoretical models and using these models in a practical setting, while adapting the models to more accurately reflect the real world. This makes the results given by the models accurate enough to be used in a real world setting.

The information given in this project is relevant to UMC Utrecht, and helps them improve their tactical planning process. Results given by this project help improve decisions made during tactical planning meetings, where changes are made to the MSS based on the current situation in the hospital.

To keep the project within the scope of a bachelor's thesis, this project will focus on the relationship between waiting list length and operating room utilization.

Theory

The goal for the theory chapter is to answer research questions 1 and 2:

1. *What techniques to evaluate the impact of an MSS exist?*
2. *How is safety stock calculated, and how does this translate to a hospital setting?*

First we gather what modelling techniques have been used to evaluate MSSs. Then we see how safety stock can be calculated in the context of the OR at a hospital. Finally we determine what techniques could be adapted to calculate the safety stock level for an OR.

One of the biggest improvements in recent years to tactical planning in hospitals is introducing flexibility to the MSS. In dynamic MSS planning, operating room time for surgical specialties is adjusted depending on changes in staff availability and changes in the demand pattern (Oliveira & Marques, 2021).

MSSs are evaluated to understand their expected quality, or their effect on performance indicators (Razali et al., 2022). Literature often benchmarks their own planning technique against established modelling methods. For example, (van der Sande, 2023) used the data from (Adan et al., 2009) to compare results. (Dellaert et al., 2016) used Markov chains to model and evaluate the waiting list length, and (Pulido, 2014) used Monte Carlo simulation for scenario reduction. Simulation is also often used to evaluate schedules (Zhu et al., 2019).

Discrete-event simulation is also often used to test and evaluate master surgery schedules in a stochastic environment. (Kumar et al., 2018), (Britt, 2016) and (Abedini et al., 2017) used discrete-event simulation to evaluate and test their optimization model. (Bovim et al., 2020) and (Oliveira et al., 2022b) use the expected waiting time these simulations to make changes in their master surgery schedule, as part of a simulation-optimization approach.

One way to make dynamic decisions more informed is by introducing safety stocks for surgical specialties. Safety stock are the extra resources that a company will keep on hand to reduce the probability of a stockout in case of variability in demand, lead times, or forecast inaccuracies. The more accurate the forecast, the less safety stock that is required, because safety stock is the buffer to counterbalance forecast variability (Monk & Wagner, 2008).
Safety stock can also prevent stockouts in case of uncertain yield rates from variability in production processes (Hung & Chang, 1999).
Safety stock is not meant to eliminate all stockouts, just the majority of them. The amount of time where safety stock prevents a stockout is called the service level. A high service level will mean higher safety stocks and costs, but fewer stockouts.

In a hospital setting, the safety stock should prevent the OR from operating without there being any work to be done. The safety stock level is therefore the minimum number of hours of work necessary on the waiting list to achieve a prespecified OR utilization with a certainty described by the service level. The forecast variability comes from both the uncertainty of patient arrivals, and the uncertainty in surgery times. When there are not enough hours of work on the waiting list to fill the OR schedule the prespecified amount, The hospital has a stockout. The OR utilization and the service level can be determined by the UMC Utrecht depending on their own goals. Choosing a higher OR utilization or necessary service level leads to a higher necessary safety stock level, but fewer stockouts compared to choosing a lower OR utilization and service level.

To calculate the necessary safety stock for the KNO department at the UMC Utrecht, techniques that evaluate the waiting list will be adapted to evaluate whether a certain amount of work on the waiting list is enough to achieve a prespecified service level. The modelling techniques that have been used to evaluate the waiting list are from (Dellaert et al., 2016), who use a Markov model to evaluate the waiting list, and (Oliveira et al., 2022), who use discrete event simulation to evaluate waiting time, although they were not able to validate their model. Both these techniques will be explored in the solution design. The other techniques mentioned focus on something other than waiting list lengths, and will not be used in the solution design.

System description

The system description describes the system that the real world problem resides in. Where the real world is unknown, assumptions are made. First the objective of the model is given, to give context on the scope of the description of the problem situation (Robinson & Macmillan, 2014). The system description describes the situation that the Markov model and the discrete event simulation are based on, and answers research questions 3 and 4:

3. *What are the main patient characteristics?*
    c. *What are the statistical patient arrivals?*
    d. *What are the statistical surgery durations?*
4. *Wat strategy is used to plan patients in OR blocks at the KNO department at the UMC Utrecht?*

Modelling objectives

The modelling objectives help inform the content of the model. This chapter focuses on the scope of the model, with the aim to clarify the breadth of the system that is to be modelled.

Specific objective

Determine the minimum amount of surgery time necessary on the waiting list at the KNO department at UMC Utrecht to achieve a prespecified service level for any OR utilization level for the MSS of May2024.

General objectives

General project objectives:

- The model should be as simple as possible
- The model should be as user friendly as possible
- The model should be as adaptable as possible, so the model might be used for different surgical departments than KNO at UMC Utrecht, or different surgical departments at different hospitals.

Model inputs and outputs

The inputs are the experimental factors that we use to achieve the modelling objectives. (Robinson & Macmillan, 2014). These factors are:

- A distribution for patient arrival rates.
- A distribution for patient surgery durations.
- A distribution for patient urgency types.
- The MSS.
- A desired utility level.
- A desired service level.

The outputs of the model are the statistics that show whether the modelling objectives were met (Robinson & Macmillan, 2014). The outputs for this model are:

- The service level, the percentage of days that the utilization threshold is reached.

- The operating room utilization, the percentage of OR time that is used.
- The percentage of patients that were operated on before their deadline.
- The necessary safety stock to achieve the service level objective.

Problem situation

The problem situation outlines the aspects of the real world that are of interest in the model design. The problem situation is used to make decisions on the scope and level of detail of the conceptual model.

The KNO (ear, nose, and throat) department at UMC Utrecht has the longest waiting list for surgery at the hospital. People get added to the waiting list when they are scheduled for surgery by their doctor, and they get removed from the waiting list when they are planned in for surgery. If they need to receive surgery again, or need to be planned again, they are added to the waiting list again.

The patients get scheduled for surgery by the KNO admissions coordinator. The admissions coordinator is in charge of scheduling patients for surgery during the OR time the KNO department has been given in the MSS. The OR time given to the KNO department can change monthly from the tactical planning process, where the hospital tries to balance all the available OR time in the hospital with the needs for every surgical department. The given OR time can also change every year because of the strategic planning process, where the hospital recalibrates their plans, priorities, and budgets.

When a patient has been scheduled, they get a label in the planning software indicating that they have been scheduled. After their surgery, patients are removed from the waiting list and leave the system. If the patient is in need of surgery again, they are added to the waiting list again.

Planning strategy

When scheduling patients, the admissions coordinator has to balance many interests. These include:

- The patients interests, making sure that the patient is available during their surgery (not on holiday for example), and that the patient receives surgery on time.
- The doctors interests, making sure that the patient is scheduled on a day when their doctor is also scheduled to perform surgery.
- The OR's interests. They prefer to finish with a shorter surgery, so it can be cancelled when they are risking overtime.
- The organizations interests: making sure that the OR capacity is utilized adequately.
- Other interests: When an anaesthetist is assigned to multiple operating rooms, the surgery can only start when the anaesthetist has done their job. This means that if 2 surgeries start at the same time, one surgery might have to wait until the anaesthetist is finished with the first patient to get to the second, which delays surgery time.

The admissions officer usually gets the MSS at least 6 weeks ahead, so their planning horizon is at least 6 weeks. The expected surgery time and the patients urgency are given by the patient's doctor. This means that a more experienced surgeon might give a lower expected surgery time for a routine surgery than a less experienced surgeon.

When selecting patients to schedule, the admissions coordinator works with a FIFO (first in first out) approach. Patients who have been waiting the longest get planning priority. The exception to this rule is that patients with a higher urgency label get priority first, and patients with a lower urgency label get priority second. During office hours, the admissions coordinator keeps constant watch over the waiting list, and tries to schedule patients as soon as they enter the waiting list.

When scheduling patients far before their date of surgery, the admissions officer has to keep some space in the schedule in case semi- urgent patients with a high urgency show up. In the case that someone with high urgency arrives on the waiting list, and there is no space for them in the schedule, the admissions coordinator might replan a patient with a lower urgency from the schedule to make room for this high urgency patient. In general, everything is provisional, and semi-urgent cases can always take your place. If a patients place is taken, the admissions officer does not place them in the back of the waiting list, but tries to plan them in as fast as possible, because the patient might have waited in the waiting room for the OR the entire day.

When the waiting list gets long, as is the case with the KNO department, the admissions scheduler has to make a decision for how many patients with a lower urgency (over 3 months) to schedule, and how much room to leave open for patients with a higher urgency (under 3 months). This decision is left up to the admissions coordinator.

If there are not enough patients on the waiting list to fill up the given OR time, the KNO department tries to shop for surgeries at divisions with similar specialties. For KNO this is only the 'kaak' (Maxillofacial) department. Different divisions cannot make effective use of the KNO's OR time, because their requirements in surgical equipment or surgical staff might be different. In general, giving back unused OR time to the OR division is difficult, because no division can get their staff ready to use the operating room within a short timeframe.

The goal for the admissions coordinator is to fill up all the given surgery time with surgeries, while keeping the interests of all the parties in mind, and keeping the patients waiting time as short as possible. There is no simple 'best' approach for admissions planning at UMC Utrecht. It is a skill and an art, that you get better at over time. Admissions planning is difficult at UMC Utrecht, because it is a tertiary hospital. This means that every incoming patient has different complexities and needs that must be taken into account, and the variety in surgery durations and surgery types is high.

Assumptions

- It is assumed that the arrival of patients is a memoryless process. This means that patients arrive to the waiting list independent of each other.
- It is assumed that the surgery time that was planned in by the doctor is the amount of time the surgery actually took.
- It is assumed that there are always the necessary OR personnel available during the given OR hours.

- It is assumed that OR-time allocated to the KNO department cannot be shared with other specialties.

Because there is no given method of admission's planning, it is assumed for the conceptual model that any planning strategy that either improves OR utilization, the percentage of patients planned within their deadline, or both of these factors, when compared to not using this strategy, is a valid planning strategy to add to the conceptual model. The goal for the finished model is to get these KPI's as high as possible, because the assumption is that an actual admissions coordinator can always plan better than a computer model that makes assumptions and simplifications.

Data

The data is obtained from the KNO department at UMC Utrecht. Where there was no available data, the best guess of the KNO admissions officer is used.

Patient type distribution

Within the given data there is no distinction between different patient types. The KNO admissions officer's best guess is that the patient distribution is: 0% semi-urgent, 25% deadline within 3 months, 75% deadline over 3 months. In the simulation, 25% of patients will be assumed to have a normal urgency, and 75% of patients will be assumed to be 'not urgent'.

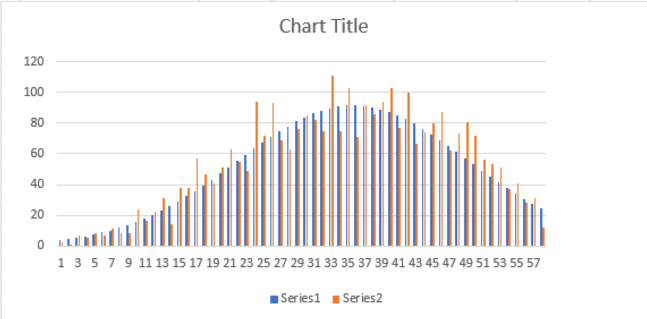
Patient interarrival times

Within the given data there is no indication when patients arrive to the system. Only when they leave. It is assumed that The patients that left the system arrived independent from each other. The dataset contains all the departures from surgery from the KNO department at UMC Utrecht starting in 2023. We only look at data starting in 2023, because before 2023 the KNO department had a different schedule, because of the covid pandemic. The average interarrival time to the KNO department at the UMC Utrecht since the start of 2023 has been 11:06:40. We use this average as the mean interarrival time in a Poisson distribution.

Surgery times

Surgery times are usually assumed to be lognormally distributed (Marques & Captivo, 2017), therefore, the data for realized surgeries at the KNO department is used to find a fitting lognormal distribution. The dataset contains the realized surgery times from the KNO department at UMC Utrecht since the start of 2019. Surgery times shorter than 30 minutes and longer than 240 minutes are removed from the dataset, because the data seems mostly faulty. There are over 20 surgeries that claim to have taken over 1500 minutes, for example. Figure 5 Shows the best fitting normal distribution for the natural log of the data's realized surgery times. Both with 30 and 58 bins, we are not able to reject the null hypothesis using the chi squared test. However, because literature suggests that surgery times are lognormally distributed, and cannot find a probability function that fits the data better, this statistical distribution is used in the model. The best fit with 30 bins is chosen, because using 58 bins (the square root of the number of data points) gives peaks in the data from bins having an inconsistent size. Using 30 bins eliminates this problem: the data is between 30 and 240 minutes, meaning that every bin has a size of 7. The excel solver is used to find a mean and standard deviation that fit the dataset the best. This distribution is used as input for the Markov model. The simulation model generates a sample surgery duration from the lognormal distribution associated with this normal distribution. If the surgery duration is not between 30 and 240 minutes, the sample is regenerated. If the surgery duration is between these values the sample is accepted. We do this because it fits the data we have more accurately. We do not do this for the Markov model because we cannot write this into mathematical form.

sample mean	4,626538		
sampel stdev	0,489015		
sample amount	3142		
nr of bins	58		
minimum	3,401197		
maximum	5,476464		
bin width	0,03578		
degrees of freedom	56		
chi squared sum	132,3638		
critical value	74,46832		
critical value (30 bins)	41,33714	sum(30 bins)	75,18035331



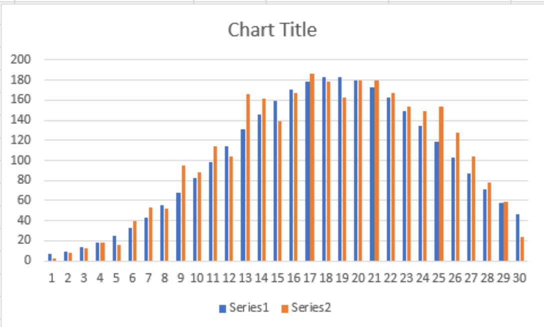


Figure 5 The best fitting distribution to the surgery time data.

MSS

The MSS for June 2024 for the KNO department at UMC Utrecht is used as an input for the schedule for the model. The OR is assumed to be opened between 8:00:00 and 16:00:00.

Markov model

The mathematical model evaluates the MSS by modelling the waiting list as a Markov chain. Once the Markov chain has reached a steady state, the expected OR utility is calculated. This method is introduced by (Dellaert et al., 2016), and practically explained by (van der Sande, 2023). This section expands on the established theory by adding uncertainty of surgery lengths into the model, which greatly improves the model's practical usability.

Notation	
Parameters	
A	Safety stock level that is being tested.
S	Service level to be achieved.
T	Length of the master surgery schedule (MSS), in days.
Z_t	The amount of time the OR is open on day t.
Variables	
t	Day index, integers in the range $[0, \infty >$.
y	The number of patients arriving on a day. Integers in the range $[0, \infty >$
$Y(y, t)$	The probability of y patients arriving on day t. A large y has an incremental smaller probability.
$q_t^{\mathrm{BP}}, q_t^{\mathrm{AP}}$	The number of patients in the queue before and after planning on day t, respectively. BP stands for Before Planning AP stands for After Planning Both q_t^{BP} and q_t^{AP} are integers in the range $[0, A]$.
$Q_t^{\mathrm{BP}}(q_t^{\mathrm{BP}}), Q_t^{\mathrm{AP}}(q_t^{\mathrm{AP}})$.	The probability that the number of patients in the queue before and after planning on day t equals q_t^{BP} and q_t^{AP}, respectively.
x	The number of surgeries that can be performed on a day. This is an integer in the range $[0, \infty >$.
$X(x, q_t^{BP})$	The probability of having enough time to perform exactly x surgeries, given that there were q_t^{BP} people on the waiting list before planning, and $x = q_t^{BP} - q_t^{AP}$.
$U(t)$	Expected OR utilization on day t.
ε	The difference between the average expected waiting list length for some MSS cycle e, and the average expected waiting list length for MSS cycle $e - 1$

Queue length calculation

The number of patients that are in the queue after planning on day t equals the number of people in the queue before planning, minus the number of people that can be planned:

$$q_t^{AP} = q_t^{BP} - min(x, q_t^{BP})$$

The number of patients on the waiting list before the surgery session on day t is the number of people on the waiting list after planning the day before, plus the number of new arrivals:

$$q_t^{BP} = q_{t-1}^{AP} + min(y, A - q_{t-1}^{AP})$$

The number of surgeries

The number of surgeries that can be performed in a day depends on the amount of time z_t the OR is open, and the length of the surgeries. Because the length of the surgeries is stochastic there exists some probability that on day t there is enough time to perform x surgeries.

The probability that there is enough time to perform at least x surgeries during the given OR time Z_t is the probability that the sum of the surgery times of x people exceeds Z_t.
Because patient surgery times can best be described by a lognormal distribution(Marques et al., 2019), we can rewrite the surgery time distribution to a normal distribution to make use of the normal distributions additive property. The probability that there is enough time to perform at least x surgeries then becomes:

$$P\left(X > ln\left(\frac{Z_t}{x}\right)\right), \, where\, X \sim Normal(\mu, \sigma)$$

with μ being the mean of the natural log of the surgery time distribution, and σ being the standard deviation of the natural log of the surgery time distribution.

Let $X(x, q_t^{BP})$ be the probability of having enough time to perform exactly x surgeries, when $x = q_t^{BP} - q_t^{AP}$. This probability is dependent on q_t^{BP} because there are 5 cases:

1. If $q_t^{BP} > x, and\ x > 0$: $X(x, q_t^{BP})$ is the probability that we have enough time to perform at least x surgeries, minus the probability that we have enough time to perform at least $x + 1$ surgeries:
$$X(x, q_t^{BP}) = P\left(X > ln\left(\frac{Z_t}{x}\right)\right) - P\left(X > ln\left(\frac{Z_t}{x+1}\right)\right), \, where\, X \sim Normal(\mu, \sigma)$$
2. If $q_t^{BP} > x, and\ x = 0$: $X(x, q_t^{BP})$ is the probability that we have enough time to perform at least 0 surgeries, minus the probability that we have enough time to perform at least 1 surgery:
$$X(x, q_t^{BP}) = 1 - P(X > ln(Z_t)), \, where\, X \sim Normal(\mu, \sigma)$$
3. If $q_t^{BP} = x, and\ x > 0$: It is impossible to perform more than q_t^{BP} surgeries, so $X(x, q_t^{BP})$ is the probability that we have enough time to perform at least x surgeries:
$$X(x, q_t^{BP}) = P\left(X > ln\left(\frac{Z_t}{x}\right)\right), \, where\, X \sim Normal(\mu, \sigma)$$
4. If $q_t^{BP} = x, and\ x = 0$: It is impossible to fill up your OR time when there is nobody on the waiting list:
$$X(x, q_t^{BP}) = 0$$

5. If $q_t^{BP} < x$: It is impossible to perform surgery on more people than there exist on the waiting list:

$$X(x, q_t^{BP}) = 0$$

Calculating the steady state

If we want to calculate $Q_t^{BP}(q_t^{BP})$, note that for each integer k in the interval $[0, q_t^{BP}]$, $Q_{t-1}^{AP}(k)$ and $Y(q_t^{BP} - k, t)$ contribute to the probability of having q_t^{BP} with probability $Q_{t-1}^{AP}(k) \cdot Y(q_t^{BP} - k, t)$, hence:

$$Q_t^{BP}(q_t^{BP}) = \sum_{k=0}^{q_t^{BP}} Q_{t-1}^{AP}(k) \cdot Y(q_t^{BP} - k, t).$$

However, because y has no upper bound and q_t^{BP} has upper bound A, we have to consider the case $q_t^{BP} = A$ separate: In this case all $y > A - q_{t-1}^{AP}$ contributes to $Q_t^{BP}(A), so$ the $Y(q_t^{BP} - k, t)$ factor is not only $Y(A - k, t)$ with $y = A - k$ but with $\sum_{y=A-k}^{\infty} Y(y, t)$.

In conclusion:

$$Q_t^{BP}(q_t^{BP}) = \begin{cases} \sum_{k=0}^{q_t^{BP}} Q_{t-1}^{AP}(k) \cdot Y(q_t^{BP} - k, t), & when\ 0 \leq q_t^{BP} < A \\ \sum_{k=0}^{q_t^{BP}} Q_{t-1}^{AP}(k) \cdot \sum_{y=A-k}^{\infty} Y(y, t), & when\ q_t^{BP} = A \\ 0, & when\ q_t^{BP} > A \end{cases}$$

Similar to this derivation, $Q_t^{AP}(q_t^{AP})$ can be expressed as follows:

$$Q_t^{AP}(q_t^{AP}) = \begin{cases} \sum_{k=0}^{A-q_t^{AP}} Q_t^{BP}(q_t^{AP} + k) * \mathrm{X}(k, q_t^{AP} + k), & when\ 0 < q_t^{AP} \leq A \\ \sum_{k=0}^{A} Q_t^{BP}(k) \cdot \sum_{x=k}^{\infty} X(x, k), & when\ q_t^{AP} = 0 \\ 0, & when\ q_t^{AP} > A \end{cases}$$

We use the power method (Bolch, 1998) until the values of $Q_t^{\mathrm{BP}}(q_t^{\mathrm{BP}})$ and $Q_t^{\mathrm{AP}}(q_t^{\mathrm{AP}})$ reach their steady state. Obtaining the steady state probabilities allows for the calculation of the expected OR utilization.

Utilization

The utilization $U(t)$ is the probability that there are enough people on the waiting list to service all the surgery time. It is therefore also 1 – the probability that there are not enough people on the waiting list before planning to utilize all the given OR time. This is given by the formula:

$$U(t) = 1 - \sum_{q_t^{BP}=0}^{A} \begin{cases} Q_t^{BP}(q_t^{BP}), & when\ q_t^{BP} = 0 \\ Q_t^{BP}(q_t^{BP}) \cdot \sum_{l=q_t^{BP}+1}^{\infty} X(x, q_t^{BP}), & when\ 0 < q_t^{BP} \le A \end{cases}$$

Tool

The tool is made in a macro-enabled Excel document. The VBA code can be found in Appendix 3: VBA code. This chapter explains how the safety stock level is calculated, what is used as an input in the tool, and what the tool shows as an output. Figure 6 shows a screenshot of the dashboard for illustration.

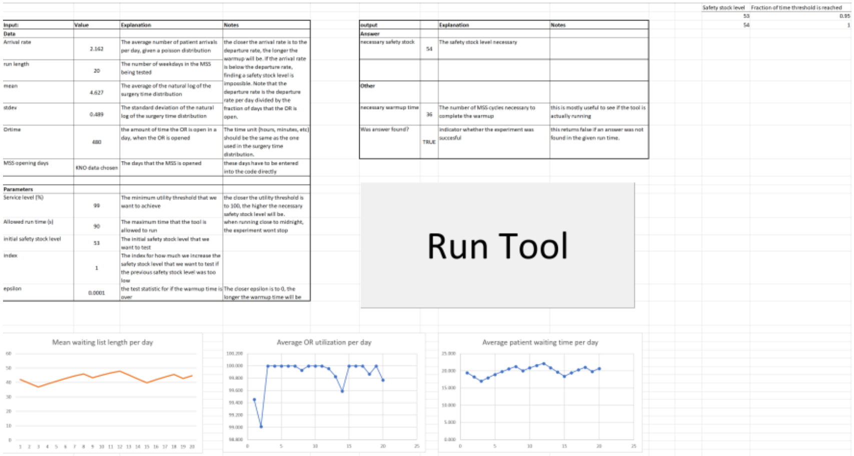


Figure 6 A screenshot from the Markov chain tool, showing the dashboard.

Determining the safety stock level

We let ε be the difference between the average expected waiting list length for some MSS cycle e, and the average expected waiting list length for MSS cycle $e - 1$. If ε is smaller than our desired input value, we assume that the steady state has be reached. If ε is not smaller than the desired input value, the steady state has not yet been reached and we calculate the transitions in the Markov chain for another MSS cycle (T days), after which we evaluate ε again. This is the Power method, as described by (Bolch, 1998).

We let S be the necessary service level we want to achieve. To calculate the necessary safety stock we test whether, after reaching a steady state, $U(t) \geq \ S$ for the entire MSS cycle ($T\ days$), given safety stock A. If this is true, A is a sufficient safety stock level, if this is not true, we try again with $A = A + I$, where I is an index for how much we increase the safety stock level after not reaching the service level with safety stock level A. If an appropriate safety stock level is found within the given run time, the tool returns this safety stock level with additional information on waiting list lengths and waiting time. Figure 6 shows what the input and output data look like in the computer model. Note that the safety stock level that the tool recommends is a number of people on the waiting list. If you want to how many hours of buffer stock you need, the safety stock level should be multiplied by the average surgery time.

Input

The input data is the data from the surgery department that is being tested. This includes surgery days, a patient arrival rate, and a surgery time distribution. The input parameters can be chosen by whoever is evaluating their surgical department. These parameters have an impact on the service level the tool is testing, the time it takes to get a steady state with the power method, and the safety stock levels that are being tested. These parameters have a large impact in the run time for an experiment.

Output

The output data shows the safety stock level that was found to be sufficient, and the amount of warmup cycles it took to get the calculation. Additional graphs are also given to give context on the average waiting list length and patient waiting time.

Simulation model

The simulation model chapter shows the process of creating the discrete event simulation model, following the steps as recommended by (Robinson & Macmillan, 2014). Figure 7 shows the artefacts of conceptual modelling. This chapter covers the conceptual model, model design, and the computer model. The system description has already been given in a previous chapter.

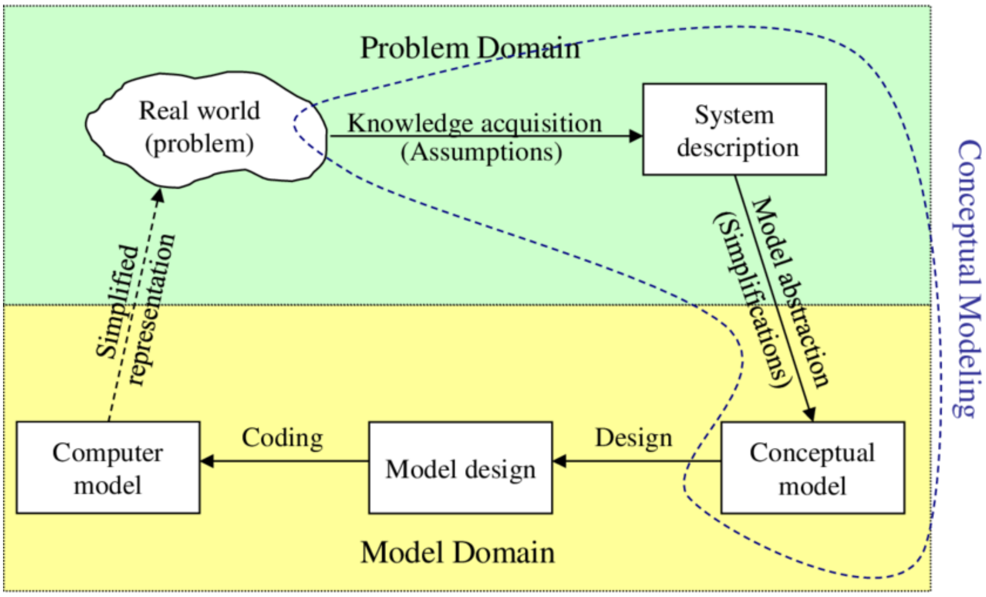


Figure 7 The artefacts of conceptual modeling (Robinson, 2011).

The conceptual model

The conceptual model uses the system description as an input for the breadth of the conceptual model, and clarifies the level of detail the model goes into, and any simplifications that are made for the sake of simplicity (Robinson & Macmillan, 2014). Choices for the model scope and level of detail are based on the given modelling objectives, and explained in this chapter.

Scope

The scope highlights parts of the system description and addresses whether they are included in the conceptual model or not.

Component	**Include/exclude**	**Justification**
Entities:		
Patients	Include	These are the entities that flow through the model
Activities:		
Postoperative care	Exclude	Has some impact on the planning process in a practical setting, but this impact is outweighed by the complexities adding the postoperative care to the model would bring, because the impact on the planning process is difficult to measure.
diagnostics	Exclude	Simplification: patients arrive not through a

		diagnostic process, but through poisson arrivals based on past arrival data.
planning	Include	Experimental factor, required for the patients to flow through the model
Surgery	Include	Including surgery makes the model easier to read for outsiders and thus improves ease of use.
Queues:		
Waiting before planning	Include	Required for safety stock level calculations
Waiting after planning	Include	Required when patients need to be replanned
Waiting on surgery day	Exclude	This has no impact on experimental factors
Resources:		
Surgeons	Exclude	Collecting data on what surgeons are linked to what surgery is too time consuming, and the impact of surgeon specific factors is expected to be small
Admissions planner	Exclude	The planning is included, the planner themself is not
Other OR personnel	Exclude	Assumption: There are always enough OR personnel available during given OR hours.
Nursing staff	Exclude	Postoperative and preoperative care are not being modeled.
OR schedule	Include	This is the schedule that the planner uses to schedule patients, and is therefore essential to achieve the modeling objectives.

Level of detail

The level of detail highlights details of components that are included in the conceptual model and addresses whether they are included in the conceptual model or not. If there are details that are not addressed in this chapter, they can be assumed to be excluded in the conceptual model.

Component	**Detail**	**Include/exclude**	**Justification**
Patients	Quantity: 1 entity represents 1 patient	Include	Necessary to achieve the modeling objectives
	Arrival pattern: patients are referred by a doctor	exclude	Simplification: patients arrive according to a poisson distribution based on realized patient arrival data.
	Different types of patients	Include	Simplification: Patients are assumed to be part of one of three groups: Semi-urgent, normal, and not urgent. Each of these groups has a different surgery deadline. Patients are divided into each group based on expected arrival rates to the KNO department. Patient types are pooled by deadline instead of e.g. illness because it is the main overarching patient characteristic

			that can be extrapolated from the data.
	Attributes: surgery time	include	Simplification: Patients are given an expected surgery time based on a lognormal distribution based on realized surgery times.
	Attributes: urgency level	include	Simplification: Patients are given a specific deadline based on expected patient urgency distribution
Planning	Patient availability	Exclude	Assumption: There is no data for patient availability. Patients are expected to be available for surgery whenever they are scheduled
	Doctor's availability	Exclude	Connecting data for surgeries to doctors is too time consuming. It is added as a feature in the model design, but this feature is turned off.
	OR interests	Exclude	The complexity and modeling time this adds to the model does not weigh with the accuracy it would add to the model
	Organization's interests	Include	Making adequate use of the OR capacity is the specific objective of this model.
	Other interests	exclude	The complexity and modeling time this adds to the model does not weigh with the accuracy it would add to the model
	Planning horizon	Include	Simplification: The planning horizon is assumed to always be 6 weeks ahead.
	Discipline: first in first out with priority exceptions	Include	Simplification: Patients are planned based on their deadline. A faster deadline means higher priority. This balances patient arrival times being scheduled based on FIFO principles, and higher urgency (faster deadlines) getting priority. This ensures that patients are planned within their deadline, which is a high priority in the planning process.
	Operating hours:	Include	Simplification: The model attempts to plan all patients on the waiting list at 9:00:00 every day, and attempts to plan patients that arrive to the waiting list if they arrive between 9:00:00 and 17:00:00.
	Leaving space for semi-urgent patients	Include	This is an experimental factor, called 'LeaveUrgentSlotsOpenPercent'
	Among of low urgency	Include	This is an experimental factor, called

	patients to plan		'LeaveSlotsOpenPercent'
	Shopping at other divisions to fill OR time	Exclude	Including this adds unnecessary complexity to the model and defeats the purpose of achieving the modeling objectives
	Specific planning strategy	Include	A specific planning strategy is necessary to maximize the model outputs, but does not formally exist at UMC Utrecht. It is elaborated on in the planning process paragraph.
	Making space in the OR for semi-urgent patients	Include	Replanning patients to make space for semi-urgent patients has great impact on the model outputs, adds model accuracy, and helps the model plan in patients that it might have not been able to, which can clog the model and reduce accuracy.
Surgery	Operating time	Include	The activity 'surgery' only exists to show patients existing in the operating room, which helps in model validation and model explanation. Therefore, only the operating time is included in the conceptual model. Any other details like the doctor being in the room or surgery specific tasks being completed are excluded.
Waiting before planning	Capacity: safety stock level	Include	Experimental factor.
Waiting after planning	Capacity: unlimited	Include	No limit to the number of people that can be waiting for their surgery after they were planned.
OR Schedule	Level of detail: slot size: 5 minutes	Exclude	Simplification: To balance model running time and level of detail, The OR schedule is split in 15 minute timeslots that can be given to patients.
	Schedule: follows the MSS every month	Exclude	Simplification: Because the model objective is to test the necessary safety stock of one MSS cycle, the OR schedule repeats one given MSS cycle every month.

Planning process

The model includes a planning process, but there is no given protocol for admissions planning at UMC Utrecht. This chapter proposes different planning strategies, and the planning strategy that fits the problem the best is chosen as the preferred planning strategy for the model.

Although there are general principle and ideas for an admissions planner to follow, there is no specific and protocolized planning protocol to follow, especially for what timeslot a patient should be planned into. It is reasonable to assume that an (experienced) admissions planner is always better at planning surgeries in a way that all stakeholders interests are met, than a computer model that makes assumptions and simplifications. Under this assumption, we can argue that a model is closer to reality, and therefore better, if the output values that the model achieves are higher. Note that this assumption and argument only relates to the planning strategy used by the admissions planner.

Choosing what patient to plan

Instead of simply planning FIFO, We plan based on deadline. This is the same as FIFO, but patients with higher urgency are automatically put higher to the waiting list.

Additionally, patients can change urgency levels, based on the time until their expiration date. This means that even when semi-urgent patients don't arrive to the system, they can turn into semi-urgent patients when they exist in the system for long enough. Patients change urgency levels depending on how close to their deadline they are. This does not have a big impact on the planning process, as patients are mainly planned based on FIFO principles, but when the strategies of planning patients with an urgent deadline are different from the planning strategies of patients with a non-urgent deadline, the planning process can be different.

Order of importance when choosing a surgery day

When a patient is chosen to be planned in for surgery, the model looks for suitable timeslots to perform surgery in for the next 30 days. Every timeslot has an associated quality, and a spreadquality. The quality indicates how well the surgery fits in the schedule. The spreadquality indicates how well spread out over the complete planning horizon the planning would be if the suggested timeslot is chosen. The suggested timeslots are sorted by quality of fit first, spreadquality second, and daynumber third. Quality is always sorted in descending order. Spreadquality and daynumber can be sorted in either ascending or descending order, depending on the chosen planning strategy.

Finding the best timeslot for a day based on its quality.

Timeslots are given a value based on the quality of the timeslot. The quality is based on the number of timeslots left over at the start or at the end of the surgery. Because most surgeries are 90 minutes or more, leaving 6 or more timeslots available before or after a surgery gives the highest quality. Specific numbers for the quality calculation are given in Figure 30 in Appendix 1: Logic Flows. Figure 8 Shows how the chosen slot quality calculation impacts how slots are ranked by the model. If a surgery fits the available OR time perfectly it is given the highest quality score. If scheduling a surgery leaves 6 or more 15 minute timeslots after surgery, it is given the second highest quality score. If scheduling a surgery leaves less than 6 15 minute timeslots after surgery, The quality is higher the more slots are left over. The quality is lowest if there are less than 6 slots left before and after surgery. The reason why having more slots left over after a planned surgery gives a higher quality level, is because the chance is higher that another patient will arrive that can be scheduled in the leftover time when there are more slots left over.

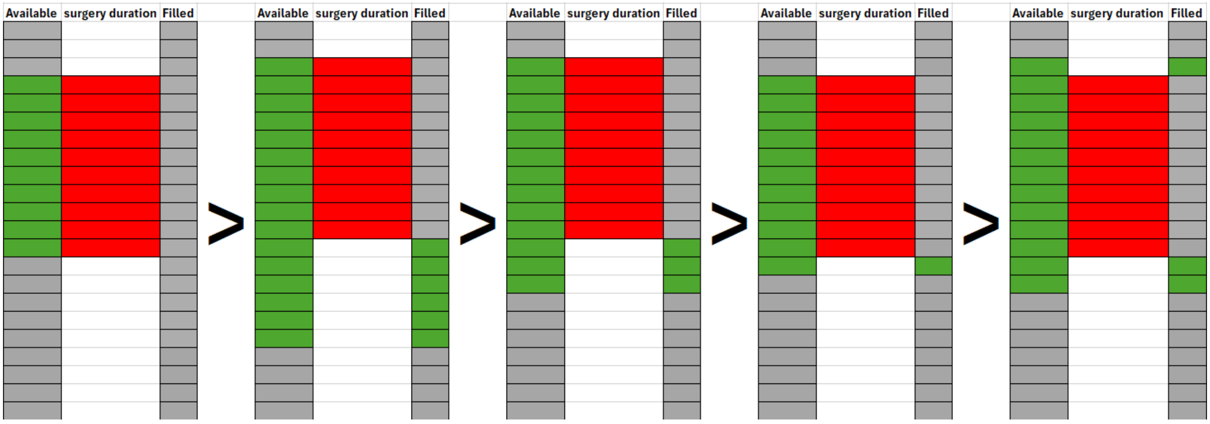


Figure 8 Visual explanation for the slot quality calculation.

When planning less than one week ahead, non urgent patients can be planned according to the LeaveUrgentSlotsOpen benchmark, instead of LeaveSlotsOpen. If this is done, there will be less space in the schedule for semi-urgent patients, but the amount of space in the schedule for normal patients will remain the same. This is hypothesized to be useful when a lot of normal patients are expected to arrive, but not a lot of semi-urgent patients.

LeaveSlotsOpen and LeaveUrgentSlotsOpen

Because there is no information on the amount of space to leave open for urgent patients, and the amount of space to give to non urgent patients, they are turned into experimental factors within the conceptual model. Their optimal values are determined in experiments in a later chapter.

Model design

The model design shows the constructs and logic of the computer model in terms of the software being used (Fishwick, 1995). This chapter gives a process flow diagram for patients, Logic flow diagrams for model processes, and this chapter explains the modelled planning process in detail. Additionally, this chapter goes into the data necessary to run the model. Figure 9 gives a screenshot of the model, to give a high level illustration for how the flows work together. Additionally, the logic flows are shown and explained in Appendix 1: Logic Flows, and the code in all the methods can be found in Appendix 5: plant simulation code.

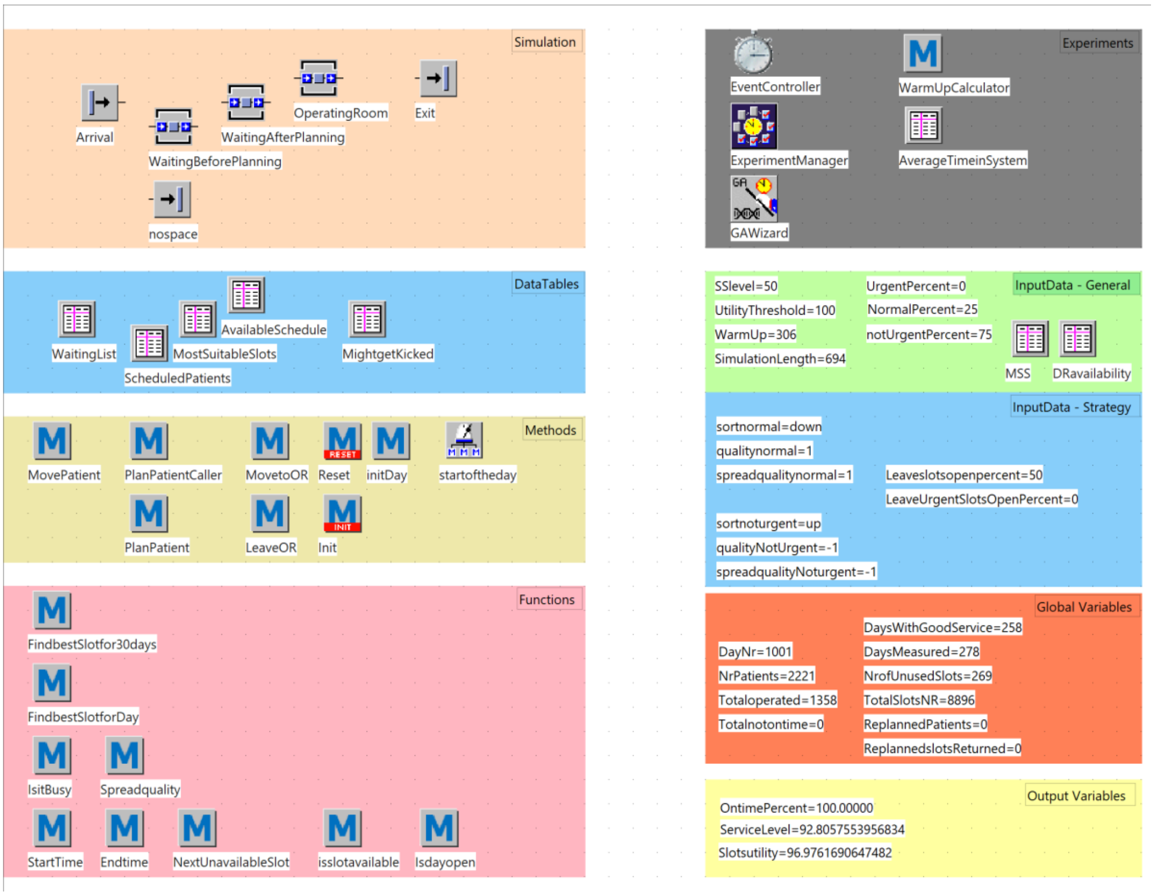


Figure 9. A screenshot of the model.

Process flow

The process flow of patients in the model is highlighted in Figure 10. Patients arrive in the system, and go to either the waiting before planning queue, or they leave the system if the queue is at capacity. When a patient is scheduled for surgery they go to the waiting after planning queue, after which they go to the operating room, followed by leaving the system. One exception is when a patient gets removed from the schedule to make room for a more urgent patient, in which case the patient gets removed from the waiting after planning queue and added to the waiting before planning queue. The process flow can be found in the model in Figure 9 in the orange box named ‘simulation’.

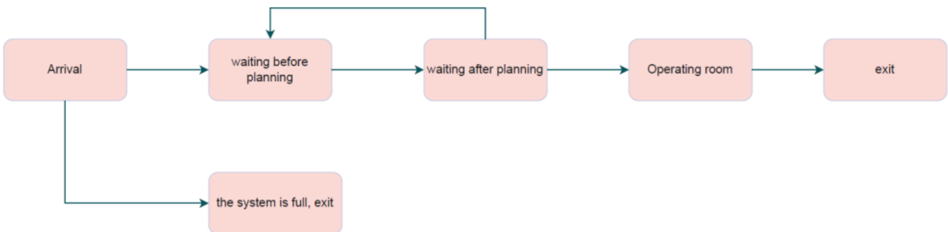


Figure 10, The process flow for patients in the model.

Validity and verification

First, we determine the warmup time, number of replications, and the runtime per replication using methods recommended by (Robinson & Macmillan, 2014). Then the model is verified and validated.

Warmup, replications, runtime

Additional information for the warmup time and number of replications calculations can be found in Appendix 2: Warmup time and number of replications

Warmup

The warmup time for the simulation is calculated according to the marginal standard error rule (MSER) as described in (Robinson & Macmillan, 2014). Because the number of replications is expected to be more than 1, the MSER is applied to an average of multiple replications. The input data used to calculate the warmup time is the data given in the chapter data. The strategic input choices are given in Figure 11. The chosen number of replications is 20, with a simulation length of 1000 days per replication. These numbers are chosen to ensure that the warmup time falls within the replication length, while not letting the computation time get too long.

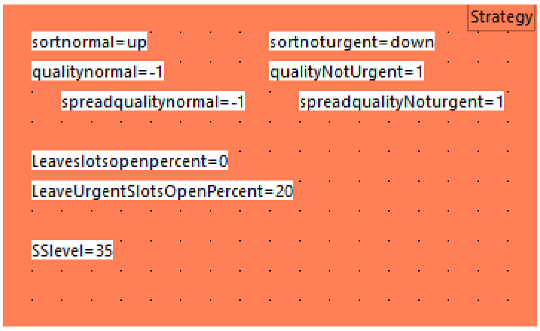


Figure 11 Screenshot of the input data used in the warmup time calculations.

The output values given in the conceptual model cannot be used in the warmup time calculations, because their variability is very high. Daily operating room utilization is 0 when the OR is closed, and 100 when all timeslots are filled. A different output value needs to be decided on that shows that the model is running without an initialization bias. This value is the daily average waiting time in the waiting room. When the daily average waiting time in the waiting rooms becomes stable, it shows that the waiting room contents are not affected by initialization bias. Every day the average waiting time of all the patients in the waiting rooms is calculated.

Figure 12 shows a table with the outcome of the MSER calculation, showing that the output value becomes stable after about 300 days. The chosen warmup time is 306 days.

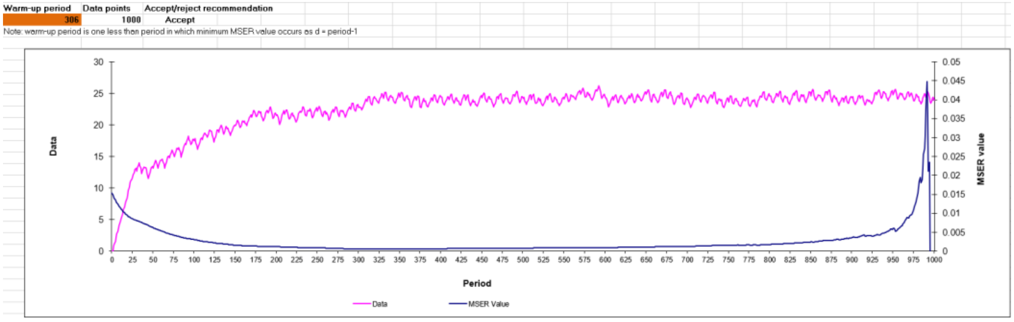


Figure 12 Warmup time calculated using the MSER method.

Number of replications

The number of replications for the simulation is calculated using the confidence interval method as described by (Robinson & Macmillan, 2014). The data used in the calculations is the same data that was used to calculate the warmup time, with the first 306 days removed, to account for the initialization bias. The run length for this data is therefore 1000-306 = 694 days. Figure 13 shows the cumulative mean time in the system, with 95% confidence intervals. The chosen number of replications is 10, because the figure clearly shows that increasing the number of replications barely decreases the size of the confidence intervals.

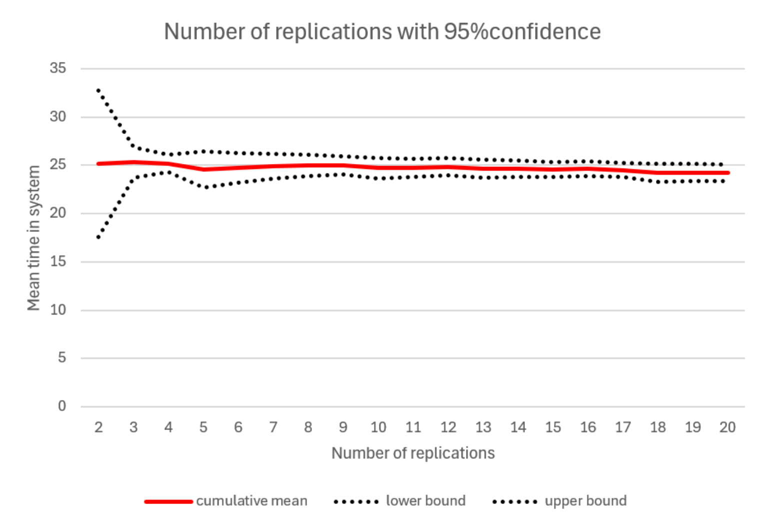


Figure 13 The cumulative mean time in the system with 95% confidence intervals.

Run length

There exists no method to calculate the necessary run length, when the warmup time and number of replications have already been calculated. A rule of thumb is to make sure that the run length is at least 10 times the warm up length, to make sure the initialization bias is properly gone. This is not feasible considering the model's speed. Figure 13 shows that after 306 days we can be reasonably confident that the initialization bias is removed. Because the number of replications was calculated using a run length of 694 days, and the confidence interval after 10 replications is reasonably narrow, this will be the chosen run length.

Verification and validation

Verification in discrete event simulation models is notoriously difficult (Robinson & Macmillan, 2014). To help the reader, every method in the simulation model is outfitted with its function, where it is called from, and when it is called, to improve understandability. The code for all the methods can also be found in Appendix 5: plant simulation code.

The problem situation was created from conversations with both UMC Utrecht's KNO admissions planner, and the UMC Utrecht supervisor. The choices made in the conceptual model have been looked at and agreed to by the UMC Utrecht supervisor.

The model design is validated by testing certain cases that give predictable outcomes. The model outcome is compared to the expected outcome. For these cases, certain design choices might be changed to allow for testing.

Cases:

Case	Expected outcome	Model outcome
Arrival rate is 0	Utilization = 0 Notontimepercent = 0	Utilization = 0 Notontimepercent = 0
1 person arrives per day at 7:59:59, with an operating time of 8 hours. The OR is open for 8 hours every day. We plan the patient as early as possible with no 'slots left open' strategy.	Utilization = 100% Notontimepercent = 0 Waiting list max contents: 1	Utilization = 100% Notontimepercent = 0 Waiting list max contents: 1
2 persons arrives per day at 7:59:59, with an operating time of 4 hours. The OR is open for 8 hours every day. We plan the patients as early as possible with no 'slots left open' strategy.	Utilization = 100% Notontimepercent = 0 Waiting list max contents: 2	Utilization = 100% Notontimepercent = 0 Waiting list max contents: 2
1 persons arrives per day at 7:59:59, with an operating time of 4 hours. The OR is open for 8 hours every day. We plan the patients as early as possible with no 'slots left open' strategy.	Utilization = 50% Notontimepercent = 0 Waiting list max contents: 1	Utilization = 50% Notontimepercent = 0 Waiting list max contents: 1
1 persons arrives every 3 days at 7:59:59, with an operating time of 8 hours. The OR is open for 8 hours every day. We plan the patients as early as possible with no 'slots left open' strategy.	Utilization = 33.33% Notontimepercent = 0 Waiting list max contents: 1	Utilization = 33.27% Notontimepercent = 0 Waiting list max contents: 1 Explanation: because the number of simulation days is not divisible by 3 the utilization is not exactly one third.

Experiments

First the planning strategy that gives the best results is chosen. The Boolean strategy inputs will be chosen by running experiments with 4 cases. After that, the GAWizard tool built into plant simulation is used to find the best values for the variable inputs, given the best performing planning strategy.

With the chosen inputs, a sensitivity analysis will be given for the patient spread and the safety stock level, showing the impact these have on the output of the model.

Finally, different cases will be entered into the model.

Strategy

Spreading patients evenly or unevenly, and planning early or late

Patients, depending on their urgency level, can be planned either as early or as late as possible, as long as the patient is planned within both the patient's deadline, and the planning horizon. It can also be decided to spread patients either as evenly or as unevenly as possible over all the days. Planning patients evenly means that the model tries to fill all days equally, while planning patients unevenly leads to the model first filling up an entire day before moving on to the next. It is assumed that planning semi-urgent patients as early as possible is always preferred. Spread quality also does not matter for semi-urgent patients. In total there are 16 different planning strategies. Figure 14 shows the strategies and the related input values for all 16 experiments.

	Strategy					Input values			
		Patient type				normal		not urgent	
Experiment		**semi urgent**	**normal**	**not urgent**		**quality**	**spread**	**quality**	**spread**
	plan	early	early	early		**sortnormal: up**		**sortnoturgent: up**	
1	**spread**	X	even	even		-1	1	-1	1
2		X	even	uneven		-1	1	-1	-1
3		X	uneven	even		-1	-1	-1	1
4		X	uneven	uneven		-1	-1	-1	-1
	plan	early	early	late		**sortnormal: up**		**sortnoturgent: down**	
5	**spread**	X	even	even		-1	1	1	-1
6		X	even	uneven		-1	1	1	1
7		X	uneven	even		-1	-1	1	-1
8		X	uneven	uneven		-1	-1	1	1
	plan	early	late	early		**sortnormal: down**		**sortnoturgent: up**	
9	**spread**	X	even	even		1	-1	-1	1
10		X	even	uneven		1	-1	-1	-1
11		X	uneven	even		1	1	-1	1
12		X	uneven	uneven		1	1	-1	-1
	plan	early	late	late		**sortnormal: down**		**sortnoturgent: down**	
13	**spread**	X	even	even		1	-1	1	-1
14		X	even	uneven		1	-1	1	1
15		X	uneven	even		1	1	1	-1
16		X	uneven	uneven		1	1	1	1

Figure 14 The input values for all 16 experiments.

Figure 15 Shows the impact that the 16 available planning strategies have on the output variables, when given 5 cases. All $16 \cdot 5 = 80$ experiments use the same random number seeds. The cases have the same given safety stock level of 50, but use a different planning strategy, based on different values of leaveslotsopenpercent and leaveurgentslotsopenpercent. This shows how the binary decisions behave when given different planning strategies. The 5 cases are:

Leaveslotsopenpercent	leaveurgentslotsopenpercent
20	0
50	0
20	20
50	20
25	0

The results of all 80 individual experiments is shown in Appendix 4: binary strategy results.

Experiment number	Service level	Slotsutility	OntimePercent
Exp 01	249.712	397.589	498.103
Exp 02	453.453	480.545	498.694
Exp 03	327.05	427.114	498.682
Exp 04	480.899	498.584	498.684
Exp 05	229.676	389.739	498.245
Exp 06	435.252	469.542	498.79
Exp 07	315.576	418.939	498.482
Exp 08	388.129	421.681	498.746
Exp 09	242.986	392.837	498.161
Exp 10	428.129	458.431	498.715
Exp 11	324.712	420.687	498.683
Exp 12	468.885	494.007	498.688
Exp 13	223.381	385.758	498.163
Exp 14	406.655	439.119	498.925
Exp 15	306.007	409.843	498.635
Exp 16	370.072	386.885	498.973

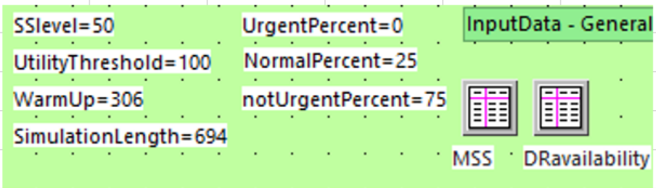


Figure 15 The sum of the results of 16 different planning strategies tested for 5 different cases, with their associated input data. The colours in the columns are formatted so that the highest number is the most saturated, and the lowest number is the least saturated.

Experiment 4 scores the best when looking at the average level of the KPI outputs, and will be used for the variable inputs and the sensitivity analysis.

Variable inputs

The optimal values for leaveslotsopenpercent and leaveurgentslotsopenpercent are found by using the GAWizard built into Tecnomatix Plant Simulation. The optimization parameter is the service level. The chosen generation size is 5, and the chosen number of generations is 20, with 10 observations per individual per generation. These values are chosen because it keeps the running time of the optimization within reasonable bounds. The running time of the optimization ended up being 7:23:18. The optimal values for leaveslotsopenpercent and leaveurgentslotsopenpercent is 20 and 0 respectively. Figure 16 Shows the input values and a part of the report in Plant Simulation. The evolution of the fitness value graph shows that the optimal solution did not change after generation 6, so we assume that the solution that was found is optimal.

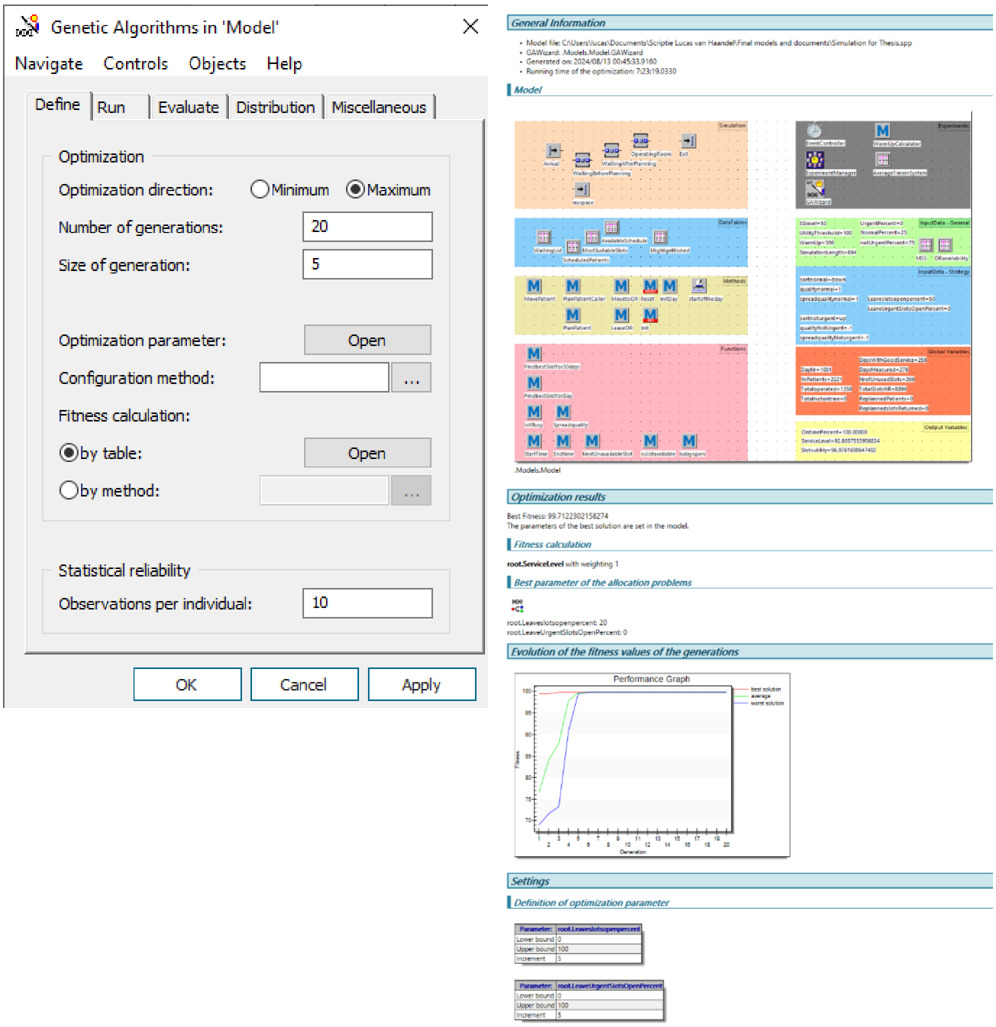


Figure 16 A screenshot of the input values of the genetic algorithm and a screenshot of part of the auto generated report.

Sensitivity analysis

A sensitivity analysis for 3 input variables is given. The non-variable input values are the ones determined in the previous chapter. No sensitivity analysis for the utility threshold is given, because the utility threshold has no impact on the planning process, only on the evaluation of the planning process. Because each experiment uses the same seed values, the utility is the same for each day in each experiment and the graph is flat.

Leaveslotsopenpercent

leaveslotsopenpercent							
root.Leaveslotsopenpercent	root.LeaveUrgentSlotsOpenPercent	root.sslevel	root.utilitythreshold		root.Slotsutility	root.ServiceLevel	root.OntimePercent
0	0	50	100		97.13916	42.1223	98.4906
5	0	50	100		99.34915	87.30216	99.31323
10	0	50	100		99.9112	98.66906	99.87828
15	0	50	100		99.98314	99.82014	100
20	0	50	100		99.96403	99.71223	100
25	0	50	100		99.95391	99.64029	100
30	0	50	100		99.92918	99.42446	100
35	0	50	100		99.89433	99.20863	100
40	0	50	100		99.89883	99.17266	100
45	0	50	100		99.76169	98.09353	100
50	0	50	100		99.70886	97.55396	100
55	0	50	100		99.47504	95.82734	100
60	0	50	100		99.36826	95.07194	100
65	0	50	100		98.81969	90.79137	100
70	0	50	100		98.2464	85.93525	100
75	0	50	100		97.30216	79.82014	100
80	0	50	100		94.85836	63.05755	100
85	0	50	100		90.24168	41.8705	100
90	0	50	100		76.37028	21.07914	100
95	0	50	100		49.39299	1.402878	100
100	0	50	100		29.79654	8.417266	100

Figure 17 The chosen input data for all 4 variables, and their associated outcomes.

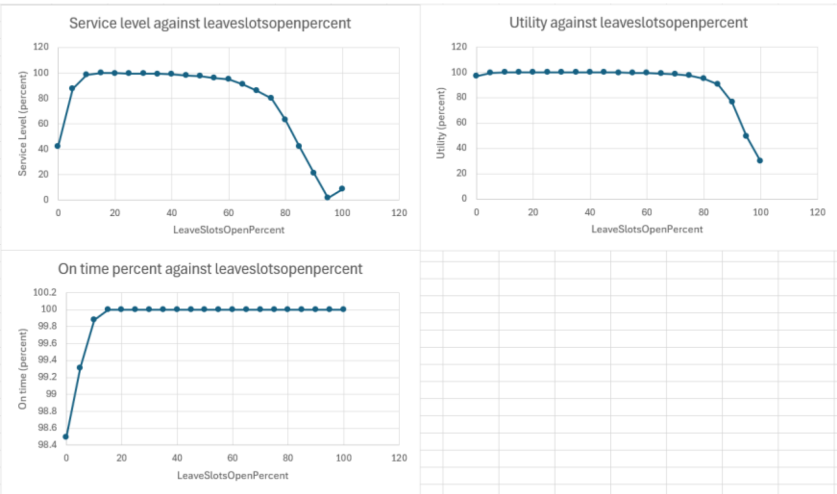


Figure 18 Graphs showing the relationship between LeaveSlotsOpenPercent and each KPI.

Figure 17 shows the input values chosen for the sensitivity analysis of leaveslotsopenpercent, and their associated output values. Figure 18 shows the relationship between leaveslotsopenpercent and each output value graphically. The figure shows that between a leaveslotsopenpercent value of 15 and 40 the service level remains relatively stable. As leaveslotsopenpercent increases over 40 and decreases under 15, the service level starts slowly dropping. After it increases over 60 the service level dives to 0, with a small bump when reaching 100. The utility also continuously dips at the same time, without the small bump. The small bump in the service level can be explained by the model only scheduling urgent patients at a leaveslotsopenpercent level of 100, while there were still some non urgent patients being

planned before. This means that the overall number of patients operated on is lower, but the number of days when the OR was opened and it was filled completely was relatively higher, given that the patient arrivals for both experiments were the same.
Figure 18 shows that using a leaveslotsopenpercent values of at least 15 is necessary to plan all the patients on time. It is also clear that using a leaveslotsopenpercent value of over 60 has detrimental effects on the service level.

Leaveurgentslotsopenpercent

leaveurgentslotsopenpercent						
root.Leaveslotsopenpercent	root.LeaveUrgentSlotsOpenPercent	root.sslevel	root.utilitythreshold	root.Slotsutility	root.ServiceLevel	root.OntimePercent
100	0	50	100	29.7965	8.41727	100
100	5	50	100	29.7965	8.41727	100
100	10	50	100	29.7965	8.41727	100
100	15	50	100	29.7965	8.41727	100
100	20	50	100	29.7965	8.41727	100
100	25	50	100	29.7965	8.41727	100
100	30	50	100	29.7965	8.41727	100
100	35	50	100	29.7965	8.41727	100
100	40	50	100	29.7965	8.41727	100
100	45	50	100	29.7965	8.41727	100
100	50	50	100	29.7965	8.41727	100
100	55	50	100	29.7965	8.41727	100
100	60	50	100	29.7965	8.41727	100
100	65	50	100	29.7965	8.41727	100
100	70	50	100	29.7965	8.41727	100
100	75	50	100	29.9326	8.38129	100
100	80	50	100	29.8033	7.8777	100
100	85	50	100	29.4177	5.97122	100
100	90	50	100	29.7077	3.1295	100
100	95	50	100	29.656	0	99.981
100	100	50	100	21.3208	0.57554	95.9309

Figure 19 The chosen inputdata for all 4 variables, and their associated outcomes.

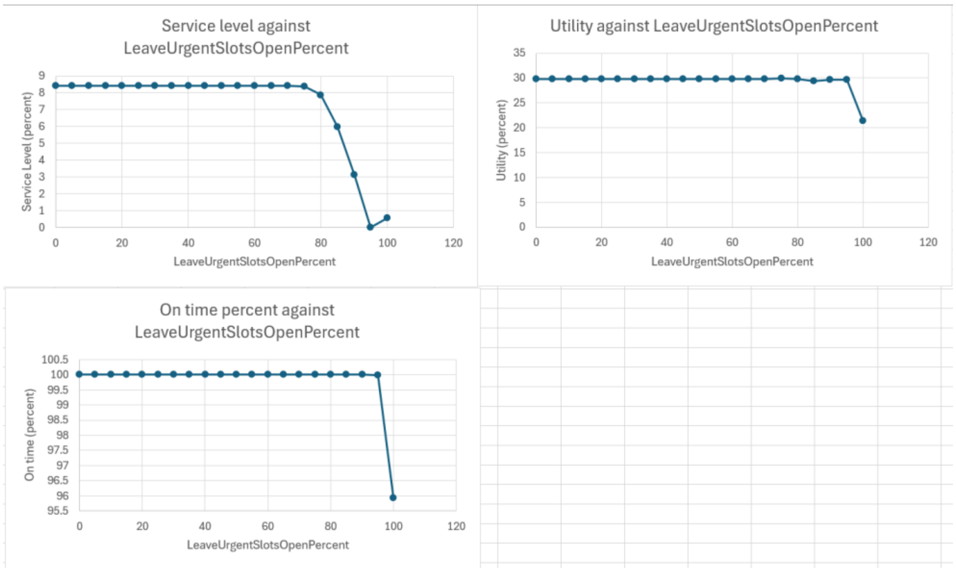


Figure 20 Graphs showing the relationship between LeaveUrgentSlotsOpenPercent and each KPI.

Figure 20 shows that using a leaveurgentslotsopenpercent values of over 75 has detrimental effects on the service level. There is also a slight bump in the service level when leaveurgentslotsopenpercent is at 100, which can be explained in the same way as the bump for leaveslotsopenpercent. Note that the utilization and service level are low, because a leaveslotsopenpercent value of 100 has to be chosen to perform this sensitivity analysis, because leaveurgentslotsopenpercent can not be higher than leaveslotsopenpercent, and we want to analyze the entire range of leaveurgentslotsopenpercent.

SSlevel

sslevel							
root.Leaveslotsopenpercent	root.LeaveUrgentSlotsOpenPercent	root.sslevel	root.utilitythreshold		root.Slotsutility	root.ServiceLevel	root.OntimePercent
20	0	0	100		0	0	0
20	0	5	100		99.049	94.1367	100
20	0	10	100		99.9562	99.6763	100
20	0	15	100		99.9595	99.7122	100
20	0	20	100		99.9674	99.7122	100
20	0	25	100		99.9618	99.6763	100
20	0	30	100		99.9472	99.6043	100
20	0	35	100		99.9865	99.8561	100
20	0	40	100		99.9505	99.6403	100
20	0	45	100		99.9888	99.8921	100
20	0	50	100		99.964	99.7122	100
20	0	55	100		99.9517	99.6043	100
20	0	60	100		99.9663	99.7482	100
20	0	65	100		99.955	99.6043	100
20	0	70	100		99.9618	99.6403	100
20	0	75	100		99.9618	99.7122	100
20	0	80	100		99.9809	99.8201	100
20	0	85	100		99.9652	99.7482	100
20	0	90	100		99.9809	99.8201	100
20	0	95	100		99.9674	99.7482	100
20	0	100	100		99.955	99.6043	100

Figure 21 The chosen inputdata for all 4 variables, and their associated outcomes.

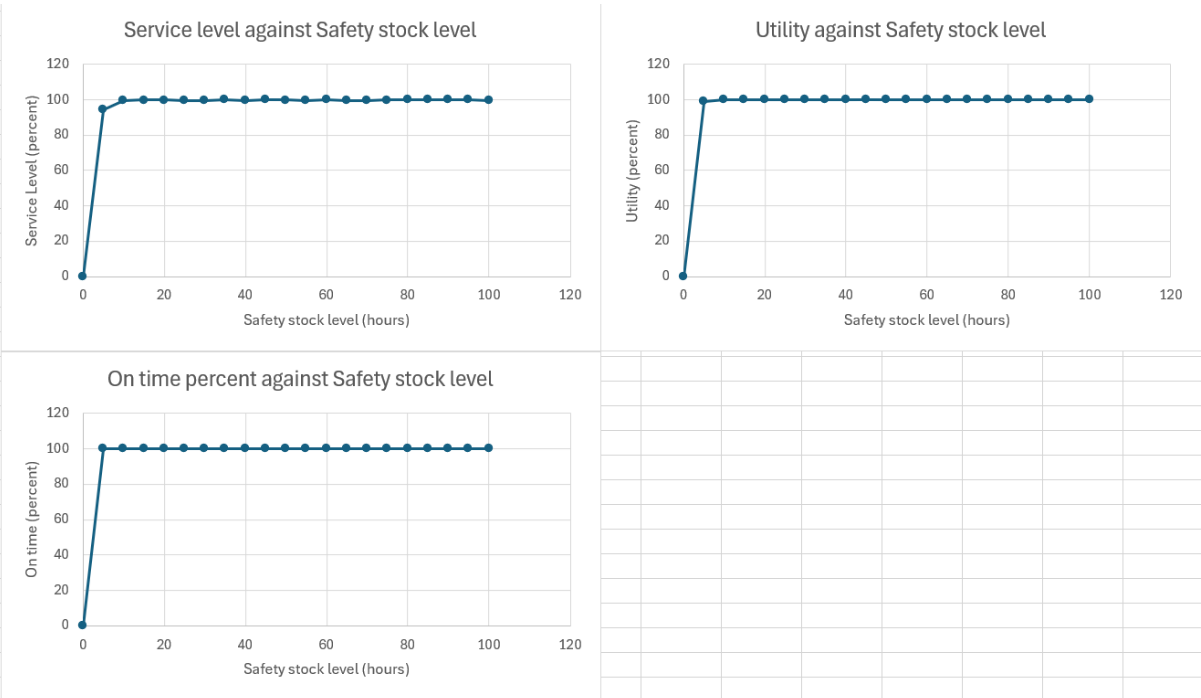


Figure 22 Graphs showing the relationship between the safety stock level and each KPI.

Figure 22 shows that using a safety stock level lower than 10 hours affects the KPI's negatively, but any safety stock level over 10 hours does not improve the KPI levels in any significant way. This means that having at least 10 hours of surgeries on the waiting list is enough to plan every patient on time, and ensure the OR is always occupied. Increasing the safety stock level beyond 10 hours has no impact on the KPI's, because the amount of hours on the waiting list never exceeds 10 hours.

Case: KNO department at UMC Utrecht

To determine the necessary safety stock level at the KNO department at UMC Utrecht, the data for patient arrivals, surgery duration, and the MSS are entered into the simulation, and the Markov model.

Simulation

Because the data for the KNO department was used to perform the sensitivity analysis, there is no reason to perform these experiments again. The results from the sensitivity analysis are used.

Markov

The data for the KNO department will be entered into the Markov model to analyse the relationship between the desired service level and the necessary safety stock level. The chosen epsilon level is 0.000001. Note that the tool gives the safety stock level as a number of patients, and not the amount of hours of surgeries. Figure 23 shows the relationship between the desired service level and the safety stock level. The safety stock level is expressed in hours by multiplying the number of people required by the average surgery time. The graph shows that as the desired service level approaches 100%, the necessary safety stock increases exponentially.

Service level	Safety stock (people)	Safety stock (hours)
90	29	56
91	30	58
92	31	59
93	33	63
94	34	65
95	36	69
96	39	75
97	42	81
98	46	88
99	54	104
99.1	56	107
99.2	57	109
99.3	58	111
99.4	60	115
99.5	62	119
99.6	65	125
99.7	68	130
99.8	73	140
99.9	81	155

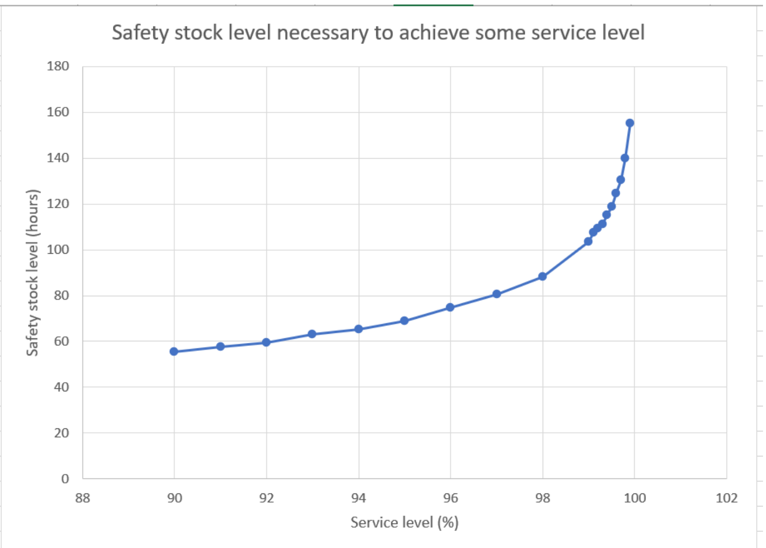


Figure 23 The necessary safety stock level to achieve some desired service level.

Discussion

Data

There are multiple flaws in the data that must be discussed. First, the patient arrival rate. The data used to calculate the arrival rate is the KNO department's departure rate. The number of patient departures was divided by the number of weekdays since the start date in the data set. From this we devised an average number of patients arriving per weekday. In reality, patients arrive on the waiting list when their doctor decides that they require surgery. This means that the patient arrival rate is 0 when the doctors are not diagnosing patients, but doing something else, like performing surgery for example. This detail cannot be captured by using data from the patient departures.

Second, the patient departure rate. The Markov model uses a lognormal distribution to calculate the probability of there being enough time to perform some number of surgeries. This lognormal distribution has range $[0, \infty >$. In reality however, the probability of there being enough time to perform e.g. 50 surgeries in one day is 0, because this simply is not realistic. Although this probability is low in the lognormal distribution, it is not 0. However, the probability of having enough time to perform a large number of surgeries is low enough that the difference between the lognormal distribution and reality is insignificant. This does mean that for any safety stock level the associated service level is *at least* the service level given by the tool, but the service level could also be higher.

Third, the patient deadline distribution. There is no available data to make a probability distribution for the length of a patient's deadline, so we had to use the admission's planner's best guess. We cannot prove any confidence for how accurate this guess is.

Markov

The tool is $\Omega(N^3)$ in Big O notation. This is because the number of calculations performed per day goes up exponentially when the safety stock level that we test goes up, and the necessary warm up length goes up too. There are features built into the tool to reduce runtime and stop an experiment when the runtime gets too long, but the mathematical model would need to be changed fundamentally if we want to solve this problem. This means that the model runs fine when testing lower safety stock levels, but testing a safety stock level of 80 already requires over a minute. When using the tool however, often a safety stock level below 80 will be enough to achieve the desired service level.

Simulation

Admissions planning at UMC Utrecht does not follow a strict protocol, because the admissions planner has to juggle many interests from surgeons, patients, other OR staff etc. These interests are not expressed in data. There is no data for a patient's availability for surgery for example. It is also difficult to verify what parts of the planning process were modelled accurately, and what parts were not. This makes it difficult to turn admissions planning into a simulation model whose resulting KPI's can be trusted to reflect reality. That is the case for this simulation model. In this case the result from the simulation model is 90% lower than the result from the Markov model, while it is not clear where that difference comes from. This stems from how difficult it is to verify and validate the simulation model, especially considering how abstract the current planning process at the UMC Utrecht is. We cannot verify that the planning process in the model reliably mimics the planning process followed by the admissions planners. Because of these reliability issues we cannot use the simulation model to recommend a safety stock level.

This does not mean that the simulation model is useless. The simulation model reflects reality to a level where it can give insight into what planning strategy would probably work the best for a prespecified patient mix. Even though the KPI levels resulting from the simulation cannot be trusted to be accurate enough to reflect reality, we can see if and how much different planning strategies change the KPI levels. Although we cannot say that the KPI levels from some strategy in the simulation can be expected in reality, we can expect that if we test multiple strategies for admissions planning in the simulation, the one that results in the highest KPI levels will likely also work the best in reality.

Future research

A safety stock level alone is not enough to make an informed decision on whether changing an OR schedule is a good decision. Another important factor is the waiting list's growth factor. If a waiting list is smaller than its recommended safety stock level, but the waiting list is growing, changing the schedule might not be a smart idea, because the waiting list can be expected to grow larger than the safety stock level. On the other hand, changing the OR schedule might be a smart decision when the waiting list is larger than the safety stock level, but the waiting list is actively shrinking. Future research could explore how the growth of a waiting list and its safety stock levels are connected, to be able to make recommendations for when an OR schedule should be changed. The expected shrinkage or growth of a waiting list can be calculated mathematically by adapting the Markov tool created in this research. A screenshot for this tool and its accompanying code are given in Appendix 6: Expected shrinkage and growth tool.

Conclusion

First we will answer research questions 5, 6, and 7. After these are answered the answer to the main research question is given.

5. *What models work best to estimate the necessary safety stock levels for surgical departments?*

While at first both discrete event simulation and a Markov model seem promising, the amount of data required to make a reliable discrete event simulation is almost impossible in practice. In addition, verification and validation for the simulation model is difficult. A discrete event simulation is not a feasible method to calculate the necessary safety stock level for a surgical specialty, at least when the simulation mimics an admissions planner.
A Markov model is a feasible method to calculate the safety stock level for surgical departments, given that there is reliable data for patient arrivals and surgery durations. In larger systems the necessary computation time might get too long, but this is unlikely.

6. *What is the relationship between OR performance and safety stock levels?*

We can conclude that there is an exponential relationship between OR performance and safety stock level, as shown by the Markov model. This is to be expected, as the relationship between safety stock and service level is usually exponential (Hung & Chang, 1999). The exact values for this relationship depend on the patient arrival rate, the surgery length distribution, and the MSS, and are therefore different for every surgical specialty. The relationship between OR performance and safety stock levels for the KNO department at UMC Utrecht is shown in Figure 23.

7. *What are the practical insights gained from these models?*

From the simulation model we have found that it is almost impossible to model the admissions planning process in a way that can be verified and validated. We can also hypothesize that an effective way to approach the planning process is to plan all the patients as early as possible, while focusing on completely filling one OR day before starting to plan patients into the next. From the Markov model we can find a suggested safety stock level using a method that is easily verifiable.

The research main research question is:

What is the necessary safety stock in hours of work for the KNO department at UMC Utrecht to ensure a prespecified OR utilization using the MSS for June 2024?

We recommend the KNO department at UMC Utrecht to keep at least 104 hours of surgeries on the waiting list, because this ensures that we can expect at least 99% of the given OR time to be utilized. Keeping the safety stock higher has diminishing effects, and lowering the safety stock will cause the expected utilization to drop off quickly. For different OR utilizations the necessary safety stock level is given in Figure 23.

Recommendation

The UMC Utrecht is recommended to use the Markov tool to evaluate the necessary safety stock levels for all the surgical departments, so the safety stock levels can be used to inform decisions that need to be made when planning the MSS dynamically in the future. Additionally, when different MSSs are proposed, it is recommended to see whether the current waiting list length is above the recommended safety stock length as given by the Markov model. This allows the admissions planners to see if they can expect to fill their given OR hours with the new MSS. If the admissions planners want to see if their waiting list is large enough to achieve enough OR utilization for the current MSS, without regard for the next MSS, they are recommended to use the tool from Appendix 6: Expected shrinkage and growth tool.

The UMC Utrecht is recommended to improve the admissions planning for the KNO department by following the planning strategy discussed in the conclusion, if they do not expect it to be a problem for patients to be informed about their surgery date only one week in advance.

The UMC Utrecht is recommended to use the Simulation model if they want to know whether a change in admissions planning strategy will actually lead to improvement for any department.

Bibliography

Abedini, A., Li, W., & Ye, H. (2017). An Optimization Model for Operating Room Scheduling to Reduce Blocking Across the Perioperative Process. *Procedia Manufacturing*, *10*, 60–70. https://doi.org/10.1016/J.PROMFG.2017.07.022

Adan, I., Bekkers, J., Dellaert, N., Vissers, J., & Yu, X. (2009). Patient mix optimisation and stochastic resource requirements : a case study in cardiothoracic surgery planning. *Health Care Management Science*, *12*(2), 129–141. https://doi.org/10.1007/S10729-008-9080-9

Bolch, G. (1998). *Enhanced Reader*.

Bovim, T. R., Christiansen, M., Gullhav, A. N., Range, T. M., & Hellemo, L. (2020). Stochastic master surgery scheduling. *European Journal of Operational Research*, *285*(2), 695–711. https://doi.org/10.1016/j.ejor.2020.02.001

Britt, J. (2016). *Stochastic Goal Programming and a Metaheuristic for Scheduling of Operating Rooms*.

Dellaert, N., Cayiroglu, E., & Jeunet, J. (2016). Assessing and controlling the impact of hospital capacity planning on the waiting time. *International Journal of Production Research*, *54*(8), 2203–2214. https://doi.org/10.1080/00207543.2015.1051668

Dicicco-Bloom, B., & Crabtree, B. F. (2006). The qualitative research interview. *Medical Education*, *40*, 314–321. https://doi.org/10.1111/j.1365-2929.2006.02418.x

Fishwick. (1995). *1995_0029*.

Hans, E. W., Van Houdenhoven, M., & Hulshof, P. J. H. (2012). A framework for healthcare planning and control. *International Series in Operations Research and Management Science*, *168*, 303–320. https://doi.org/10.1007/978-1-4614-1734-7_12/FIGURES/2

Heerkens, H., & Van Winden, A. (n.d.). *Solving Managerial Problems Systematically 1 e edition*.

Hung, Y. F., & Chang, C. Bin. (1999). Determining safety stocks for production planning in uncertain manufacturing. *International Journal of Production Economics*, *58*(2), 199–208. https://doi.org/10.1016/S0925-5273(98)00124-8

Ingegneria Gestionale, D., Pulido Martínez, R., García Sánchez, Á., & Brun, A. (n.d.). *In cooperation with Politecnico di Milano Analysing the complexity of the model-based decision making processes within the industrial management context*.

King, W. R., & He, J. (2005). Understanding the Role and Methods of Meta-Analysis in IS Research. *Communications of the Association for Information Systems*, *16*, 665–686. https://doi.org/10.17705/1CAIS.01632

Kumar, A., Costa, A. M., Fackrell, M., & Taylor, P. G. (2018). A sequential stochastic mixed integer programming model for tactical master surgery scheduling. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, *270*(2), 734–746. https://doi.org/10.1016/j.ejor.2018.04.007

Marques, I., & Captivo, M. E. (2017). Different stakeholders' perspectives for a surgical case assignment problem: Deterministic and robust approaches. *European Journal of Operational Research*, *261*, 260–278. https://doi.org/10.1016/j.ejor.2017.01.036

Marques, I., Captivo, M. E., & Barros, N. (2019). Optimizing the master surgery schedule in a private hospital. *Operations Research for Health Care*, *20*, 11–24. https://doi.org/10.1016/j.orhc.2018.11.002

Marrin, C. A. S., Johnson, L. C., Beggs, V. L., & Batalden, P. B. (1997). *Clinical Process Cost Analysis*.

Monk, E., & Wagner, B. (2008). *Concepts in Enterprise Resource Planning*.

Oecd. (2017). *Tackling Wasteful Spending on Health*.

Oliveira, M., & Marques, I. (2021). Facing Dynamic Demand for Surgeries in a Portuguese Case Study. *Springer Proceedings in Mathematics and Statistics*, *374*, 79–94. https://doi.org/10.1007/978-3-030-85476-8_7

Oliveira, M., Visintin, F., Santos, D., & Marques, I. (2022). Flexible master surgery scheduling: combining optimization and simulation in a rolling horizon approach. *Flexible Services and Manufacturing Journal*, *34*(4), 824–858. https://doi.org/10.1007/S10696-021-09422-X/FIGURES/7

Razali, M. K. M., Rahman, A. H. A., Ayob, M., Jarmin, R., Qamar, F., & Kendall, G. (2022). Research Trends in the Optimization of the Master Surgery Scheduling Problem. *IEEE Access*. https://doi.org/10.1109/ACCESS.2022.3202546

Robinson, S. (2011). Choosing the right model: Conceptual modeling for simulation. *Proceedings - Winter Simulation Conference*, 1423–1435. https://doi.org/10.1109/WSC.2011.6147862

Robinson, S., & Macmillan, P. (2014). *The Practice of Model Development and Use Second edition*.

van der Sande, L. (2023). *Solving the Master Surgery Scheduling Problem to improve waiting list management at the cardiothoracic surgery department of the MUMC+*.

Zhu, S., Fan, W., Yang, S., Pei, J., & Pardalos, P. M. (2019). Operating room planning and surgical case scheduling: a review of literature. *Journal of Combinatorial Optimization*, *37*(3), 757–805. https://doi.org/10.1007/S10878-018-0322-6/TABLES/5

Appendix 1: Logic Flows

The logic flows illustrate decisions made and actions performed by the model to move patients around. Every logic flow given in this chapter refers to an entity in the blocks 'methods' or 'functions' in Figure 9. Blue blocks in the logic flow figures are references to different logic flows.

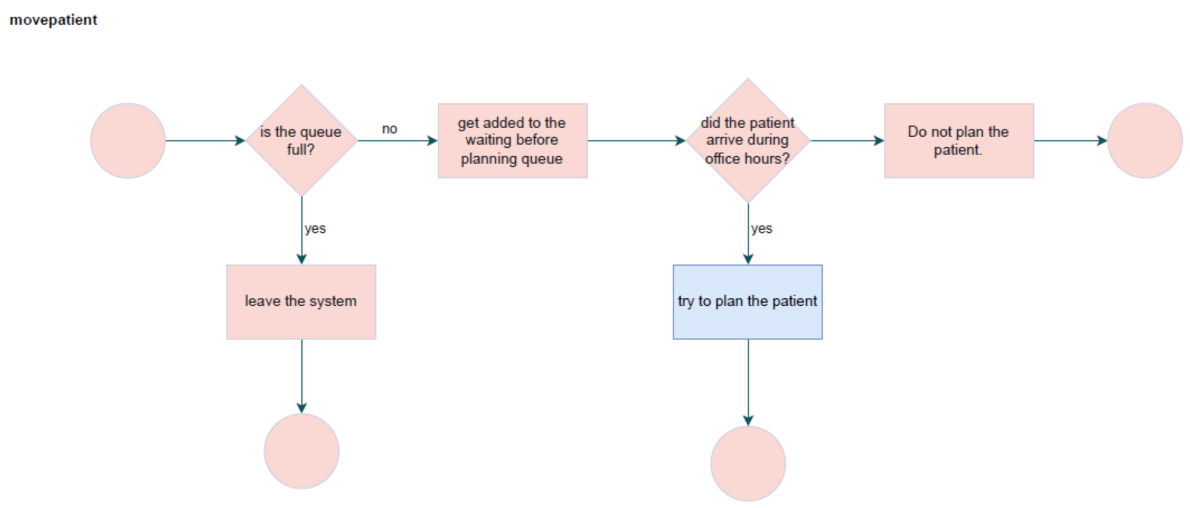


Figure 24 Logic flow 'MovePatient'

Figure 24 Describes the logic flow 'MovePatient'. MovePatient is triggered whenever a patient enters the system. It gives the patient their attribute values. Then it moves the patient to the waiting list if there is space. Otherwise the patient leaves the system. If the patient arrives within office hours, it immediately tries to plan the patient using PlanPatient (Figure 26).

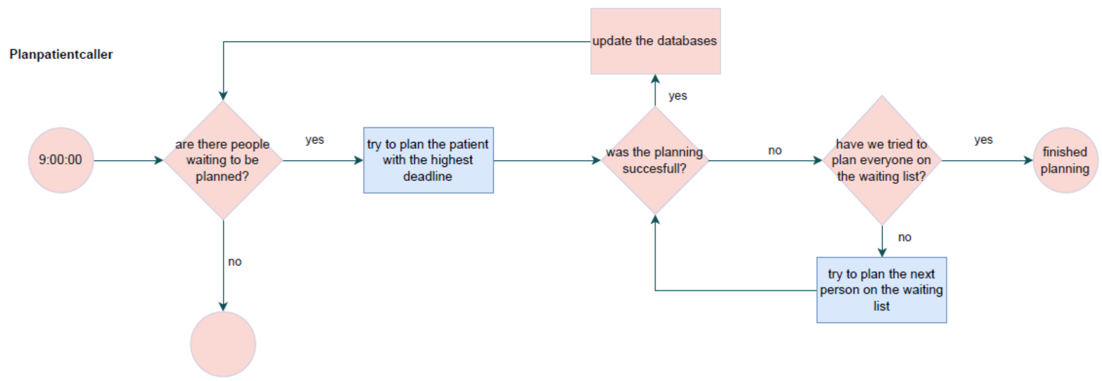


Figure 25 Logic flow 'PlanPatientCaller'

Figure 25 describes the logic flow PlanPatientCaller. It is triggered every day at 9:00:00, and tries to plan every patient in the waiting before planning queue using 'PlanPatient' (Figure 26).

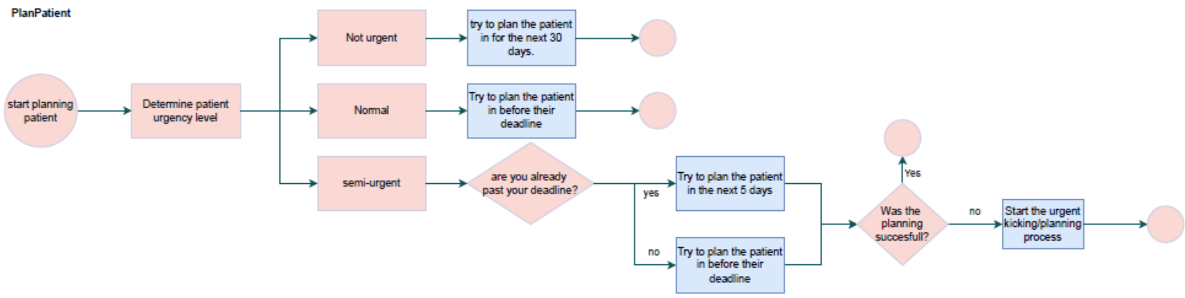


Figure 26 Logic flow PlanPatient

Figure 26 describes the logic flow PlanPatient. It is triggered whenever a patient needs to be planned in. Planpatient determines the patient's urgency based on the time until their deadline expires, and plans them in based on the suitable timeslots found by the method 'FindBestSlotsFor30Days'. PlanPatient returns true or false based on whether the planning was successful. If the patient is semi-urgent and the planning was not successful, planpatient starts the urgent kicking/planning process (Figure 27) to find a suitable person on the schedule to replace.

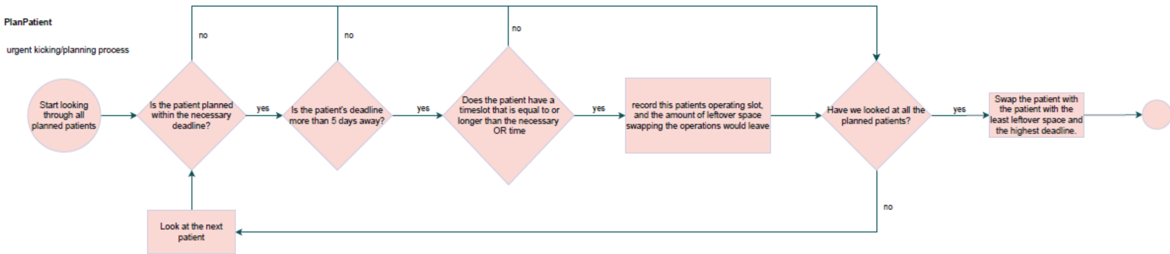


Figure 27 Logic flow urgent kicking/planning process from planpatient

Figure 27 describes the urgent kicking/planning process from planpatient. The process looks at every scheduled patient to check if they were scheduled in the necessary timeframe, and if their own deadline is further away than the patient we are trying to swap in. It also checks if the patient occupies enough OR timeslots to be able to be rescheduled. After that the process chooses the patient whose swap leaves the least unoccupied timeslots first, and who has the furthest away deadline second.

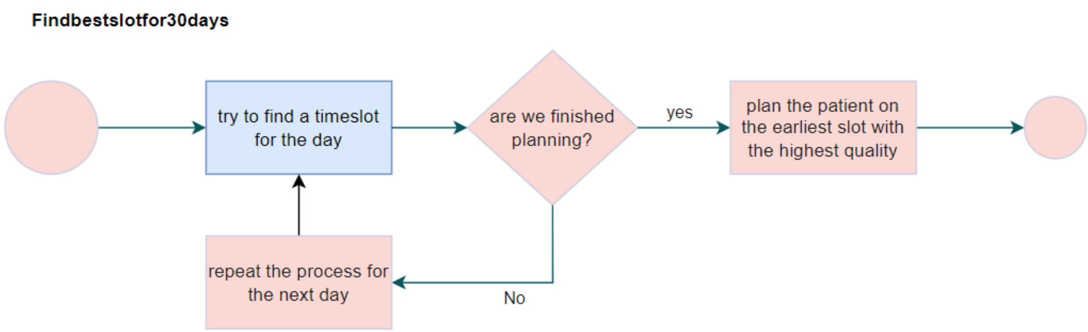


Figure 28 Logic flow FindBestSlotFor30Days

Figure 28 describes the process to find the best slot for 30 days. The process looks at multiple days using the FindBestSlotforDay process (Figure 29).
The days it looks at depend on a patients urgency level. For a semi-urgent patient the process looks from 1 day ahead to the patients deadline, unless the patient's deadline has already expired. Then it looks from 1 to 5 days ahead. For a normal patient the process looks from 1 day to 30 days ahead. For a non urgent patient the process looks from 5 days to 30 days ahead.
The way the patients are planned also depends on their urgency level. All the possible slots are collected and sorted by quality first, spread quality second, and surgery day third. 'Quality' is elaborated on in 'FindBestSlotforDay' (Figure 29) and 'spread quality' is elaborated on in 'spread quality' (Figure 35) The direction these patients are sorted in is based on a patients urgency level.
A semi-urgent patient is planned in the spot with the highest quality first, with the worst spread second, on the earliest day third.
A normal patient is planned in the spot with the highest quality first, with the worst spread second, on the earliest day third.
A non urgent patient is planned in the spot with the highest quality first, with the best spread second, on the latest day third.
Planning semi-urgent and normal patients on during times where they cause the worst spread leaves large holes in the planning that can be used for patients with a high urgency and a high surgery time. This method of planning gives the best resulting output variables that we could find.

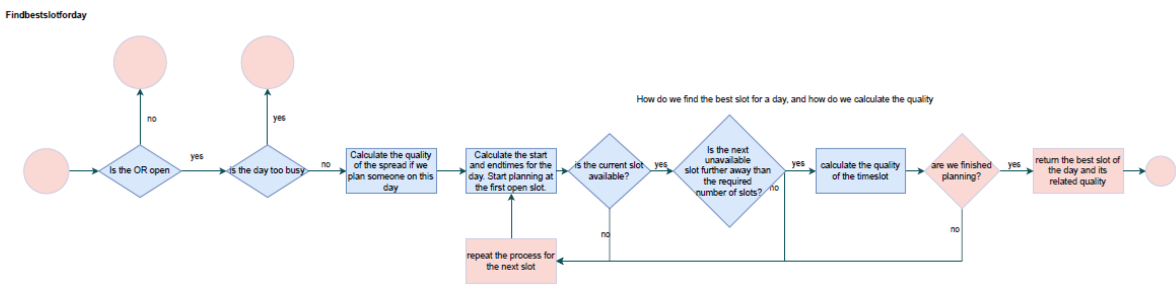


Figure 29 Logic flow FindBestSlotforDay

Figure 29 describes the logic flow for FindBestSlotforDay. When given a day and a patient, it checks whether the OR is open that day using 'isdayopen' (Figure 32) and if it is too busy to schedule this patient using 'IsItBusy' (Figure 34). Then it calculates the quality of the spread that would result if we planned the patient on this day using 'SpreadQuality' (Figure 35). After that the start- and end times for the OR that day are calculated using 'StartTime' and 'EndTime' (Figure 33). The process loops from the starttime to the endtime and checks whether each slot is available using 'IsSlotAvailable' (Figure 31). If the slot is available it checks whether we have enough future slots available to schedule our patient using 'NextUnavailableSlot'. If this comes back true, FindBestSlotForDay calculates the quality of the timeslot (Figure 30). After looping through every timeslot the timeslot with the highest quality is returned.

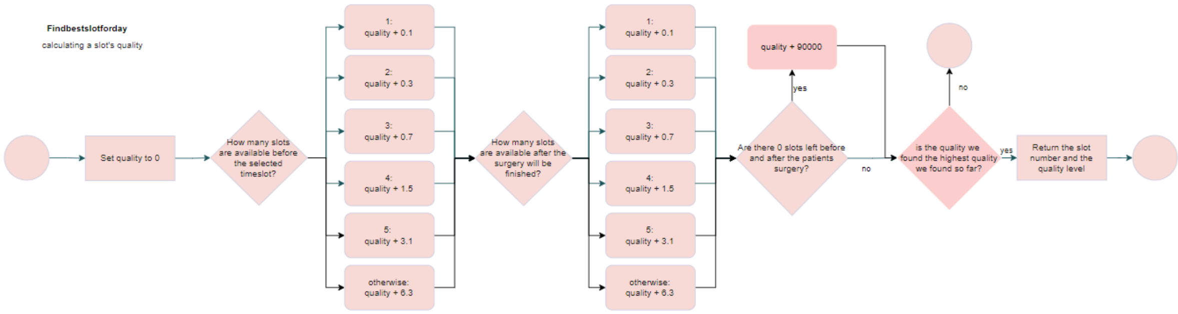


Figure 30 The part of FindBestSlotforDay's logic flow that calculates a slot's quality

Figure 30 Describes how a slot's quality is calculated. The quality of every slot is given a value based on how many slots are left open before and after a proposed surgery is scheduled. If a surgery fits perfectly, its value is put very high.

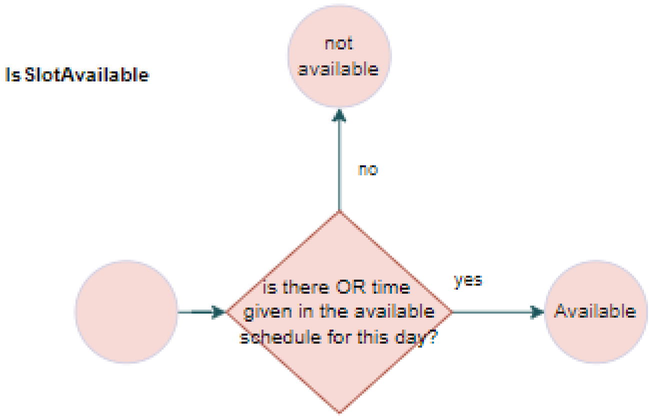


Figure 31 Logic flow IsSlotAvailable

Figure 31 describes logic flow IsSlotAvailable. This process checks if there is OR time available on this day, and returns true or false.

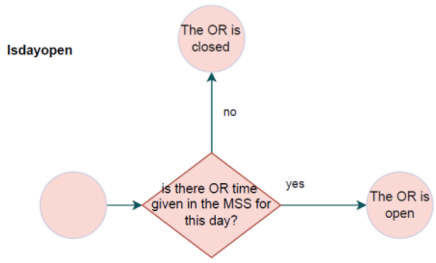


Figure 32 Logic flow IsDayOpen

Figure 32 describes logic flow IsDayOpen. This process checks if there was OR time given on this day, and returns true or false.

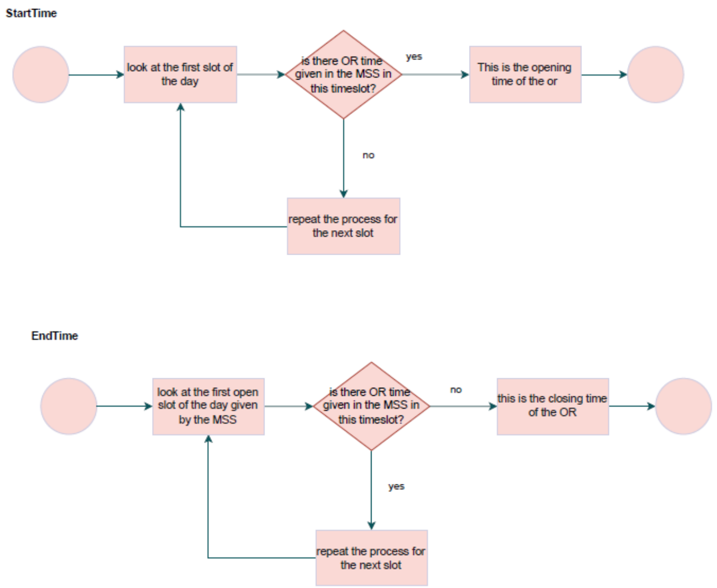


Figure 33 Logic flows StartTime and EndTime

Figure 33 describes logic flows StartTime and EndTime. These logic flows look at what the opening and closing times are for the OR on a given day that the OR is open.

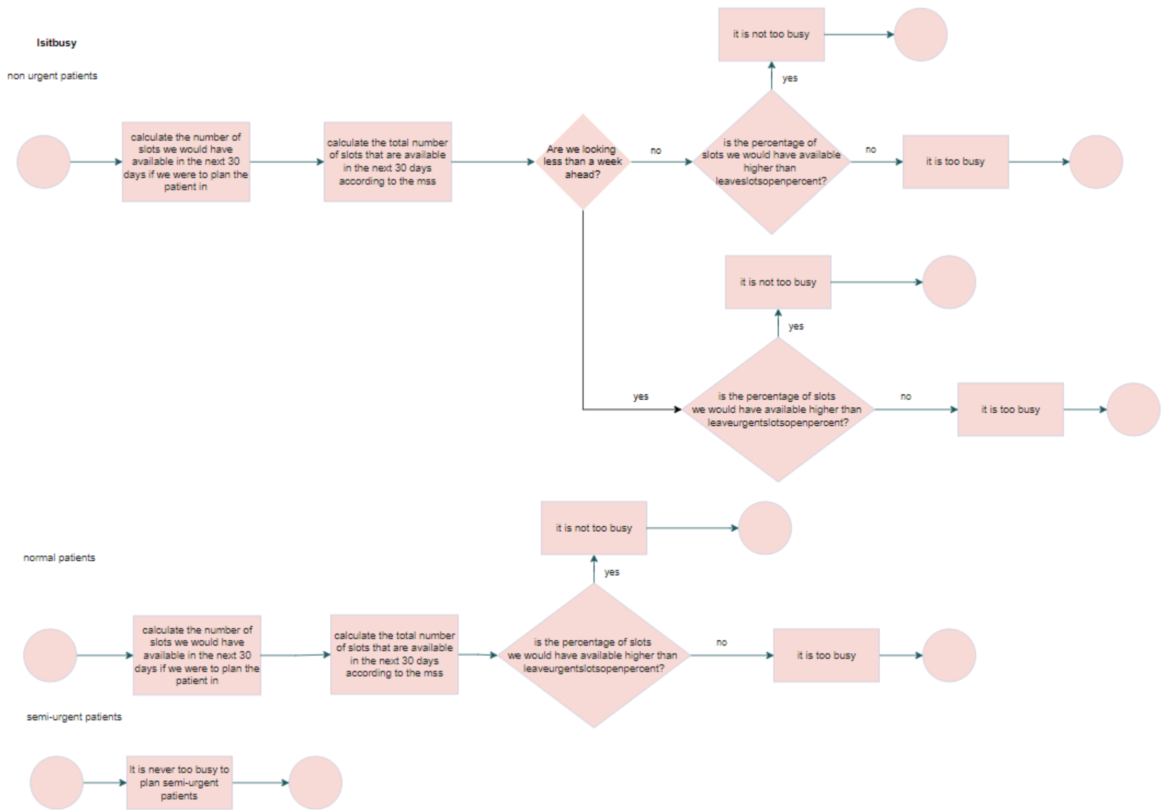


Figure 34 Logic flows for IsitBusy, based on a patients urgency.

Figure 34 describes the logic flows for IsitBusy. These processes check whether it is too busy to plan a patient based on their urgency, and the experimental factors LeaveSlotsOpenPercent and LeaveUrgentSlotsOpenPercent. It is never too busy to plan a semi-urgent patient.

It is too busy to plan a normal patient if planning the normal patient means we exceed the percentage of urgent slots that we are supposed to leave open.
It is too busy to plan a non urgent patient one week in advance if planning the patient means we exceed the percentage of urgent slots that we are supposed to leave open, and it is too busy to plan a non urgent patient up to 6 weeks in advance if planning the patient means we exceed the percentage of non urgent slots that we are supposed to leave open.

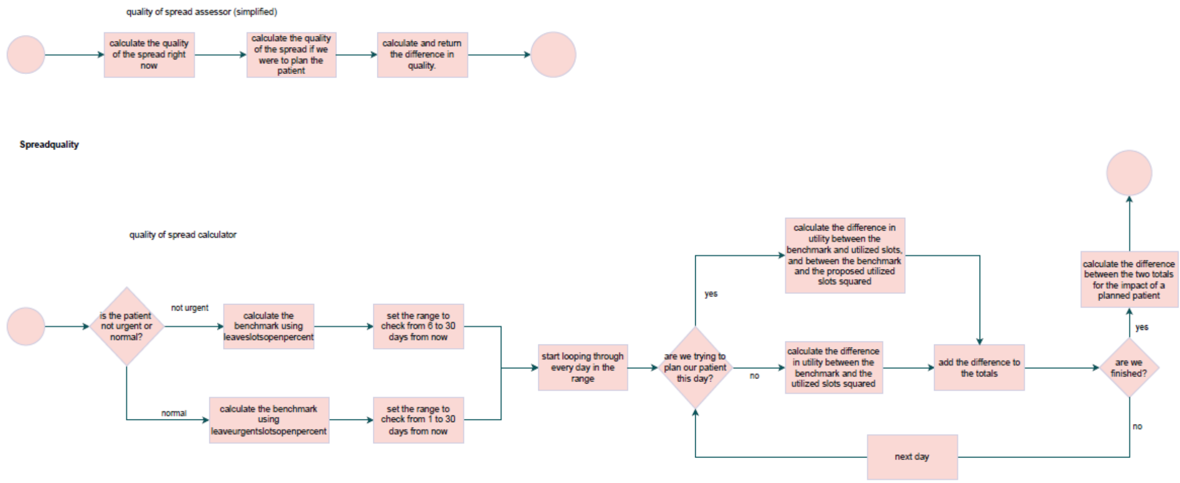


Figure 35 Logic flow for SpreadQuality

Figure 35 describes the logic flow for SpreadQuality. The spreadquality calculates the quality of the spread when planning normal or non urgent patients. SpreadQuality does not matter when planning semi-urgent patients. SpreadQuality calculates the difference in quality between the patient spread before and after we hypothetically plan in a patient on a day. The quality is calculated for the next 30 days by adding the difference between the benchmark utilization and the actual utilization squared for every day. SpreadQuality returns the quality of planning a patient on a given day.

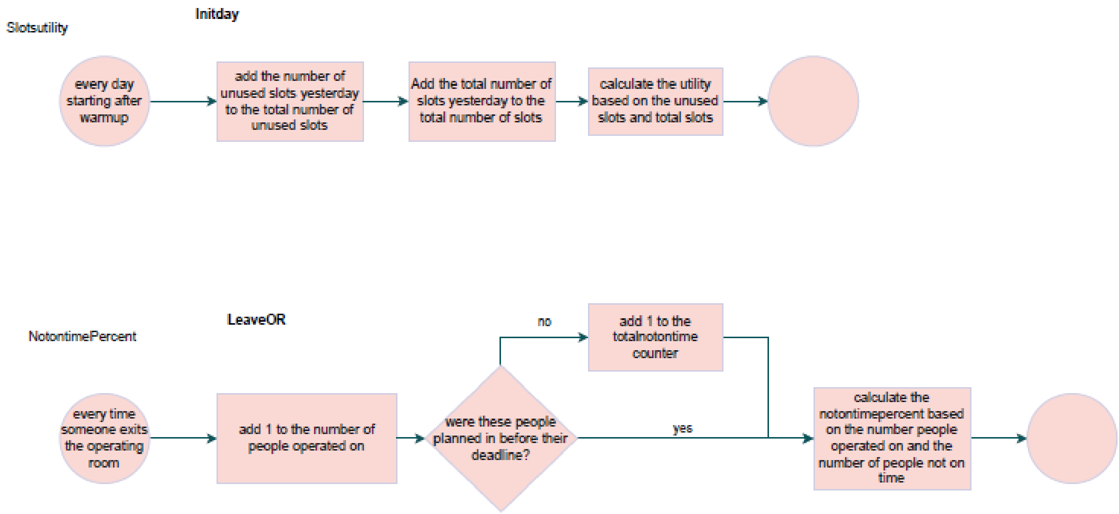


Figure 36 Logic flows for Initday and LeaveOr

Figure 36 Describes the part of logic flows InitDay and LeaveOr that calculate the output variables SlotsUtility and NotOnTimePercent. SlotsUtility is calculated by adding the total available slots yesterday to the total, and adding the total number of unused slots yesterday to the total. The total slot utilization is 100- the percentage of the total number of unused slots.

The NotontimePercent is calculated whenever someone leaves the operating room. It checks whether the person that was just operated on was operated on on time. NotontimePercent is the percentage of people who were operated on too late.

Appendix 2: Warmup time and number of replications

Period	Data	d	Var	MSER(d)
1	0.30	0	15.25001	0.015235
2	0.83	1	14.77573	0.014776
3	1.10	2	14.32303	0.014337
4	1.55	3	13.87987	0.013908
5	2.18	4	13.4541	0.013495
6	2.71	5	13.05282	0.013105
7	3.10	6	12.67138	0.012735
8	3.58	7	12.30359	0.012378
9	4.07	8	11.953	0.012037
10	4.66	9	11.61964	0.011713
11	5.06	10	11.30646	0.011409
12	5.60	11	11.00649	0.011118
13	6.01	12	10.72408	0.010843

Figure 37 A small sample of the calculations done to calculate the warmup time

Base Data									Replications										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.209	0.333	0.333	0.333	0.228	0.259	0.217	0.333	0.333	0.289	0.333	0.333	0.172	0.333	0.271	0.333	0.333	0.333	0.309	0.333
0.874	0.621	0.613	0.606	1.228	0.809	0.716	0.707	0.776	0.865	1.333	0.672	0.587	0.719	0.901	1.333	0.59	1.333	0.637	0.753
0.535	1.218	1.613	1.127	1.688	0.817	0.95	1.472	1.271	0.56	0.357	0.586	1.587	1.477	1.218	1.104	1.15	0.658	1.365	1.192
1.051	2.042	2.073	1.565	1.661	1.255	1.617	1.74	1.952	1.104	0.697	0.697	1.881	1.963	1.73	1.105	1.747	1.086	2.061	1.955
1.712	2.541	3.073	1.912	2.218	1.824	2.404	2.421	2.211	1.843	1.242	1.244	2.375	2.963	2.73	1.919	2.747	1.19	2.651	2.4
2.27	2.684	3.33	2.912	3.218	2.824	3.137	2.993	2.785	2.321	1.564	1.889	2.691	3.158	2.588	2.728	3.072	1.338	3.651	3.095
2.807	3.27	3.24	3.373	3.612	3.35	3.465	3.419	3.494	2.824	1.472	2.721	3.188	3.508	3.128	3.456	2.863	1.744	3.885	3.113
3.333	3.849	4.24	3.91	3.421	3.364	4.465	3.914	4.328	3.488	1.95	3.035	3.468	3.927	3.809	3.863	2.869	2.146	4.345	3.865
3.909	3.897	3.985	4.345	3.548	4.07	5.465	4.699	4.338	3.786	2.95	3.692	4.209	4.729	4.809	4.571	3.62	2.691	4.651	3.479
3.823	4.418	4.697	5.284	3.979	3.585	5.773	5.488	5.041	4.598	3.328	4.375	4.913	5.473	5.366	4.525	4.584	3.225	6.225	4.438
4.012	5.418	5.697	5.86	4.729	3.321	5.838	5.614	5.536	4.732	4.134	4.938	5.576	5.146	5.942	5.083	4.675	3.618	6.526	4.811
4.703	6.004	5.922	6.017	5.729	3.358	6.541	5.995	6.365	5.412	4.448	5.938	6.043	4.981	6.719	5.428	5.232	4.392	7.526	5.184
5.046	6.581	6.152	6.061	5.46	3.384	7.476	6.207	7.173	5.891	5.232	5.918	6.075	5.472	7.719	5.983	5.948	4.96	7.788	5.77
5.537	6.538	6.477	7.652	6.282	2.602	7.427	6.491	7.771	6.545	5.691	6.668	6.92	6.42	8.12	6.983	7.431	5.648	8.039	6.169
5.898	7.335	7.028	6.915	7.379	3.253	7.819	6.977	8.559	6.445	6.455	7.521	7.265	6.906	8.297	7.204	7.74	6.648	7.519	6.406
6.493	7.709	7.368	6.173	7.643	3.898	8.511	7.977	9.336	6.699	7.455	8.431	8.095	7.535	9.043	7.491	8.419	6.505	8.011	6.83
6.872	8.11	7.53	7.173	8.302	3.976	8.083	8.977	10.08	7.55	7.603	8.516	8.25	8.535	8.558	7.627	8.766	6.999	8.478	7.29
7.656	8.92	7.593	7.339	8.919	4.423	8.565	9.977	10.58	8.239	6.802	8.913	8.928	8.897	9.11	8.417	9.766	6.857	8.937	7.754
8.425	9.514	7.763	7.698	8.889	5.661	8.712	9.865	11.08	8.902	7.402	9.617	9.511	9.686	9.589	8.955	9.996	7.633	9.937	8.009
8.97	10.1	8.257	9.327	9.226	5.597	10.05	10.26	11.82	9.228	7.808	10.33	10.68	10.46	11.03	9.263	10.89	8.412	10.83	9.074
9.725	10.47	9.096	9.963	9.603	6.293	10.99	11.21	12.06	9.987	8.621	9.853	11.48	10.59	12.24	10.03	11.52	8.714	10.62	9.827
10.73	11.38	10.04	10.29	10.29	6.972	11.51	12.3	12.28	11.34	9.014	10.43	11.59	11.37	13.03	11.24	11.43	9.482	10.81	10.73
10.79	12.2	10.53	11.29	11.29	6.724	11.37	12.13	12.1	11.06	9.811	11.15	12.33	11.97	13.13	10.9	12.05	10.17	11.04	11.4
11.48	11.92	10.82	11.91	11.59	6.959	10.24	12.02	11.63	11.84	10.81	11.34	11.82	11.63	13.11	11.47	14.47	11.08	10.82	11.57

Figure 38 A small sample of the input data for the warmup calculations

The warmup time is calculated using the tool provided in (Robinson & Macmillan, 2014). Figure 37 and Figure 38 fhow some of the inputdata and calculations made in the tool. Figure 27 shows the calculations done to determine the number of replications.

replication	mean	cumulative mean	stdev	Standard error	t value	lower bound	upper bound
1	24.58681204	24.58681204					
2	25.77569445	25.18125325	0.84	0.594441205	12.706	17.62816159	32.7343449
3	25.56689104	25.30979918	0.63	0.366484431	4.3027	23.73294394	26.8866544
4	24.88756091	25.20423961	0.56	0.279818229	3.1824	24.31373312	26.0947461
5	21.96659351	24.55671039	1.53	0.682841882	2.7764	22.66083739	26.4525834
6	25.77107458	24.75910442	1.45	0.59313745	2.5706	23.23439607	26.2838128
7	25.72431541	24.8969917	1.38	0.519910775	2.4469	23.62481587	26.1691675
8	25.83245127	25.01392415	1.32	0.465192011	2.3646	23.91391984	26.1139285
9	24.83208262	24.99371954	1.23	0.41075801	2.306	24.04650987	25.9409292
10	22.14234632	24.70858221	1.47	0.465060218	2.2622	23.65654291	25.7606215
11	25.27323005	24.75991384	1.41	0.423783108	2.2281	23.81566623	25.7041614
12	25.86603938	24.85209097	1.38	0.397689232	2.201	23.97678287	25.7273991
13	22.29766592	24.65559673	1.5	0.415252821	2.1788	23.75083856	25.5603549
14	24.7413614	24.66172278	1.44	0.384498213	2.1604	23.83106489	25.4923807
15	23.71203717	24.5984104	1.41	0.363504446	2.1448	23.81877091	25.3780499
16	25.61638757	24.66203398	1.38	0.34592847	2.1314	23.9247049	25.3993631
17	22.24413995	24.51980492	1.46	0.354707202	2.1199	23.76785924	25.2717506
18	19.09165164	24.21824085	1.91	0.450309226	2.1098	23.26817143	25.1683103
19	25.04644153	24.26183036	1.87	0.428174439	2.1009	23.36226924	25.1613915
20	23.51463977	24.22447083	1.82	0.407916351	2.093	23.37069209	25.0782496

Figure 39 Calculations for the number of replications

Appendix 3: VBA code

```
Option Explicit
```

```
'global variables - input

    Dim L As Double
    Dim UtilityThreshold As Double
    Dim NrOfdays As Integer
    Dim RunLength As Single
    Dim mean As Double
    Dim stdev As Double
    Dim ORtime As Double
    Dim index As Integer
    Dim epsilon As Double



'global variables - useful during calculation
    Dim listlength As Integer
    Dim AP() As Double
    Dim BP() As Double

Sub ToolExecute()
Dim CurrentServiceLevel As Double
Dim i As Integer
Dim j As Integer
Dim d As Integer
Dim safetystock As Integer
Dim nextRow As Long
Dim WeightedAverage As Double
Dim total As Double
Dim probability As Double
Dim percentile As Integer
Dim startTime, elapsedTime As Single
Dim WasanswerFound As Boolean


With ThisWorkbook.Sheets("Dashboard")
    .Range("L2:M" & .Rows.Count).ClearContents
End With

With ThisWorkbook.Sheets("CalculationData")
    .Range("A2:P" & .Rows.Count).ClearContents
End With

With ThisWorkbook.Sheets("dashboard")
    UtilityThreshold = .Cells(13, "B").value
    L = .Cells(5, "B").value
    NrOfdays = .Cells(6, "B").value
    RunLength = .Cells(14, "B").value
    mean = .Cells(7, "B").value
    stdev = .Cells(8, "B").value
    ORtime = .Cells(9, "B").value
    safetystock = .Cells(15, "B").value - .Cells(16, "B").value
    index = .Cells(16, "B").value
    epsilon = .Cells(17, "B").value
    .Range("G5").ClearContents
    .Range("G8").ClearContents
```

```
    .Range("G9").ClearContents
End With

CurrentServiceLevel = 0

startTime = Timer
WasanswerFound = True

Do While CurrentServiceLevel < 1

    elapsedTime = Timer - startTime

    'If elapsed time is greater than allowed, exit the loop
    If elapsedTime > RunLength Then
        WasanswerFound = False
        Exit Do
    End If
    safetystock = safetystock + index
    CurrentServiceLevel = MarkovChains(safetystock)
    With ThisWorkbook.Sheets("dashboard")
        nextRow = .Cells(.Rows.Count, "L").End(xlUp).Row + 1
        .Cells(nextRow, "L").value = safetystock
        .Cells(nextRow, "M").value = CurrentServiceLevel
        If CurrentServiceLevel = 1 Then
            .Cells(5, "G").value = safetystock
        End If
    End With
Loop
With ThisWorkbook.Sheets("dashboard")
    .Cells(9, "G").value = WasanswerFound
End With

For d = 1 To NrOfdays
        WeightedAverage = 0
        probability = 100
        For i = 0 To listlength
            WeightedAverage = WeightedAverage + (i * BP(d, i))
            If MSS(d) = True Then
                    probability = probability - 100 * (BP(d, i) *
utilityprobability(i))
            End If
        Next i

          With ThisWorkbook.Sheets("CalculationData")
        ' Find the next empty row in column A

        nextRow = .Cells(.Rows.Count, "A").End(xlUp).Row + 1

        ' Paste values into columAP A and B in the next empty row
        .Cells(nextRow, "A").value = d
        .Cells(nextRow, "B").value = WeightedAverage
        .Cells(nextRow, "C").value = probability
        .Cells(nextRow, "D").value = WeightedAverage / L
        End With
Next d
```

```
With ThisWorkbook.Sheets("CalculationData")
    For i = 0 To listlength
        .Cells(i + 2, "F").value = i
        .Cells(i + 2, "G").value = BP(NrOfdays, i) 'add the chances
that there are i people on the waitlist at the end just for extra
information
    Next i
End With

  ' Turn all the values in the array to 0 to be safe
    For i = 0 To NrOfdays
        For j = 0 To listlength
                AP(i, j) = 0
                BP(i, j) = 0
        Next j
    Next i

' Update the chart sizes
    Dim lastRow As Long
    Dim startRow As Long
    Dim rangeAddress1 As String
    Dim rangeAddress2 As String
    Dim rangeAddress3 As String


    ' Calculate the last row for the first range
    lastRow = NrOfdays + 1
    rangeAddress1 = "A2:B" & lastRow

    ' Update Chart 1
    With ActiveSheet.ChartObjects("Chart 4")
        .Activate
        Application.CutCopyMode = False
        ActiveChart.SetSourceData
Source:=Sheets("CalculationData").Range(rangeAddress1)
        ActiveChart.FullSeriesCollection(1).IsFiltered = True
        ActiveChart.FullSeriesCollection(2).IsFiltered = False
    End With

    ' Calculate the address for the second range

    startRow = 1
    rangeAddress2 = "A" & startRow & ":A" & lastRow & ",C" &
startRow & ":C" & lastRow

    ' Update Chart 2
    With ActiveSheet.ChartObjects("Chart 5")
        .Activate
        Application.CutCopyMode = False
        ActiveChart.SetSourceData
Source:=Sheets("CalculationData").Range(rangeAddress2)
    End With

    ' Calculate the address for the third range
```

```
    rangeAddress3 = "A" & startRow & ":A" & lastRow & ",D" &
startRow & ":D" & lastRow

    With ActiveSheet.ChartObjects("Chart 6")
        .Activate
        Application.CutCopyMode = False
        ActiveChart.SetSourceData
Source:=Sheets("CalculationData").Range(rangeAddress3)
    End With
End Sub

Function MarkovChains(safetystock As Integer) As Double

    Dim i As Integer
    Dim j As Integer
    Dim d As Integer
    Dim Servicelevel As Double
    Dim totalservicelevel As Integer
    Dim WeightedAverageNew, WeightedAverageOld As Double
    Dim Convergence As Double
    Dim loopnr As Integer



    ' Initialize the variables with some values (if needed)
    listlength = safetystock
    WeightedAverageOld = 0
    loopnr = 0
    Convergence = epsilon + 1

    ReDim AP(0 To NrOfdays, 0 To listlength)
    ReDim BP(0 To NrOfdays, 0 To listlength)

    Do While Convergence > epsilon


    ' Turn all the values in the array to 0 to be safe
    If loopnr = 0 Then
        For i = 0 To NrOfdays
            For j = 0 To listlength
                If i = 0 And j = 0 Then 'at the start, the chance of
no people on the waitlist is 1
                    AP(i, j) = 1
                    BP(i, j) = 0
                Else
                    AP(i, j) = 0 'the rest is just cleaning up the
array for safety
                    BP(i, j) = 0
                End If
            Next j
        Next i
    Else
        For i = 0 To NrOfdays
            For j = 0 To listlength
                If i = 0 Then 'at the start, the probability after
planning on day 0 is the
```

```
                    AP(i, j) = AP(NrOfdays, j)
                    BP(i, j) = 0
                Else
                    AP(i, j) = 0 'the rest is just cleaning up the
array for safety
                    BP(i, j) = 0
                End If
            Next j
        Next i
    End If

    For d = 1 To NrOfdays

        'add the chances of i people existing on the waiting list
before planning. Based on the chance that i-j people arrive, given
the probability of yesterdays waiting list length being i
        For i = 0 To listlength
            If i <> listlength Then
                For j = 0 To i
                    BP(d, i) = BP(d, i) + AP(d - 1, j) *
WorksheetFunction.Poisson_Dist((i - j), L, False)
                Next j
            Else
                For j = 0 To i

                        BP(d, i) = BP(d, i) + AP(d - 1, j) *
CumProbability((i - j), L)

                Next j
            End If

        Next i

        'add the chances of people existing after planning, based on
whether the OR is open today
        If MSS(d) = True Then


            For i = 0 To listlength
                For j = 0 To listlength - i
                    'the probability that there are i people after
planning, is the probability that there were i+j people before
planning* the probability that j-i people were planned

                    AP(d, i) = AP(d, i) + BP(d, i + j) *
NrOfSurgeries(j, i)
                Next j

            Next i
        Else
            For i = 0 To listlength
                AP(d, i) = BP(d, i)
            Next i
        End If
```

```
    Next d


        WeightedAverageNew = 0
        For i = 0 To listlength
                WeightedAverageNew = WeightedAverageNew + (i * BP(1,
i))
        Next i

        Convergence = WeightedAverageNew - WeightedAverageOld
        WeightedAverageOld = WeightedAverageNew
        With ThisWorkbook.Sheets("Dashboard")
            .Cells(8, "G").value = loopnr
        End With

        loopnr = loopnr + 1


    Loop

    totalservicelevel = 0

    For d = 1 To NrOfdays
        Servicelevel = 100

        If MSS(d) = True Then

            For i = 0 To listlength
                    Servicelevel = Servicelevel - 100 * (BP(d, i) *
utilityprobability(i)) 'the probability that our expected lost
utilization is not below our service level
            Next i

            If Servicelevel > UtilityThreshold Then
            totalservicelevel = totalservicelevel + 1
            End If
        Else
            totalservicelevel = totalservicelevel + 1
        End If

    Next d

    MarkovChains = totalservicelevel / NrOfdays


End Function

Function MSS(day As Integer) As Boolean

    Select Case day
        Case 1, 2, 8, 12, 13, 14, 18, 20
            MSS = True
        Case Else
            MSS = False
    End Select
```

```
End Function

Function CumProbability(x As Integer, L As Double) As Double

If x = 0 Then
        CumProbability = 1
Else
        CumProbability = 1 - WorksheetFunction.Poisson_Dist(x - 1,
L, True)
End If

End Function

Function NrOfSurgeries(vectorsize As Integer, state As Integer) As
Double

'The NrOFSurgeries is the probability that the amount of OR minutes
on the waiting list is enough to perform vectorsize surgeries
'The Nrofsurgeries is thus the probability of at least vectorsize
surgeries - the probability of at least vectorsize+1 surgeries

Dim probability As Double

If state = 0 Then
    If vectorsize = 0 Then
        probability = 1
    Else
        probability =
WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime /
vectorsize), mean, stdev, True)
    End If
ElseIf state = listlength Then
        probability = 1 -
WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime /
(vectorsize + 1)), mean, stdev, True)
Else
    If vectorsize = 0 Then
        probability = 1 -
WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime /
(vectorsize + 1)), mean, stdev, True)
    Else
        probability = 1 -
WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime /
(vectorsize + 1)), mean, stdev, True) - (1 -
WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime /
vectorsize), mean, stdev, True))
    End If
End If


NrOfSurgeries = probability

End Function

Function utilityprobability(state As Integer) As Double
```

```
'the utilityprobability is the probability that the amount of OR
minutes is at least the ORtime, given the number of people on the
waiting list(state)
'the utilityprobability is the probability that with state number of
surgeries, we still dont have enough ORtime to fill the entire
schedule
'the utilityprobability is therefore the probability that state
surgeries take less than the ortime.
'the probability that x surgeries take less than the ortime is the
probability that one surgery takes less than ln(ortime/x)
'if our state number of surgeries is 0, the probability of not
having enough ORtime is 1
Dim LNOR As Double

If state = 0 Then
    utilityprobability = 1
Else

    LNOR = WorksheetFunction.Ln(ORtime / state)
    utilityprobability = WorksheetFunction.Norm_Dist(LNOR, mean,
stdev, True)
End If

End Function
```

Appendix 4: binary strategy results

Experiment	root.Slotsutility	root.ServiceLevel	root.OntimePercent
Exp 01	95.1922212	81.7985612	99.6942045
Exp 02	99.9494155	99.6043165	99.9929478
Exp 03	98.3565647	92.7338129	99.9277264
Exp 04	99.9640288	99.7122302	100
Exp 05	93.426259	76.618705	99.7048448
Exp 06	99.8257644	98.6330935	100
Exp 07	97.0334982	88.7769784	99.8973354
Exp 08	96.9593076	94.2805755	99.9927641
Exp 09	93.9433453	78.8848921	99.7967509
Exp 10	99.8639838	99.1726619	99.9929279
Exp 11	97.1043165	89.7841727	99.9116435
Exp 12	99.9392986	99.4604317	100
Exp 13	92.2661871	73.4532374	99.7626242
Exp 14	99.1029676	97.2661871	100
Exp 15	95.3394784	85.8992806	99.9398602
Exp 16	90.1821043	88.6690647	99.9844048
Exp 17	58.169964	12.3021583	100
Exp 18	90.6441097	83.1294964	100
Exp 19	67.9395234	34.5683453	100
Exp 20	99.7088579	97.5539568	100
Exp 21	57.4348022	11.2230216	100
Exp 22	85.6148831	76.7266187	100
Exp 23	66.5040468	35.8633094	100
Exp 24	66.975045	60.5755396	100
Exp 25	57.3977068	12.5539568	100
Exp 26	79.8066547	71.4748201	100
Exp 27	66.6097122	37.4820144	100
Exp 28	97.4359263	91.294964	100
Exp 29	56.975045	12.0863309	100
Exp 30	72.533723	65.7194245	100
Exp 31	63.9725719	34.3884892	100
Exp 32	58.7904676	57.5899281	100
Exp 33	94.63129496	70.68345324	98.57087749
Exp 34	99.35926259	87.98561151	98.70135589
Exp 35	96.90647482	79.02877698	98.76883381
Exp 36	99.24797662	86.43884892	98.68404756
Exp 37	92.24595324	64.42446043	98.65798196
Exp 38	98.94222122	85.82733813	98.78956289
Exp 39	95.04608813	72.98561151	98.59993222
Exp 40	96.82104317	82.87769784	98.75372407
Exp 41	94.16816547	70.03597122	98.53705967
Exp 42	99.29518885	87.48201439	98.72231156
Exp 43	96.33093525	77.15827338	98.78639759
Exp 44	99.29294065	87.62589928	98.68808531
Exp 45	92.23133993	63.95683453	98.55262573
Exp 46	97.57419065	84.0647482	98.92505678
Exp 47	94.43570144	72.26618705	98.70267583
Exp 48	94.5267536	82.51798561	99.0137039
Exp 49	58.16996403	12.30215827	100
Exp 50	90.64410971	83.1294964	100
Exp 51	67.93952338	34.56834532	100
Exp 52	99.70885791	97.55395683	100
Exp 53	57.43480216	11.22302158	100
Exp 54	85.61488309	76.72661871	100
Exp 55	66.50404676	35.86330935	100
Exp 56	66.97504496	60.57553957	100
Exp 57	57.39770683	12.55395683	100
Exp 58	79.80665468	71.47482014	100
Exp 59	66.60971223	37.48201439	100
Exp 60	97.43592626	91.29496403	100
Exp 61	56.97504496	12.08633094	100
Exp 62	72.53372302	65.71942446	100
Exp 63	63.97257194	34.38848921	100
Exp 64	58.79046763	57.58992806	100
Exp 65	91.42535971	72.62589928	99.838011
Exp 66	99.94829137	99.60431655	100
Exp 67	95.97234712	86.15107914	99.98528866
Exp 68	99.95391187	99.64028777	100
Exp 69	89.19739209	66.18705036	99.88230952
Exp 70	99.54473921	97.3381295	100
Exp 71	93.85116906	82.08633094	99.98499625
Exp 72	93.95008993	89.82014388	100
Exp 73	89.93030576	68.95683453	99.82767651
Exp 74	99.65827338	98.52517986	100
Exp 75	94.03214928	82.8057554	99.98491118
Exp 76	99.90332734	99.20863309	100
Exp 77	87.31002698	61.79856115	99.84768182
Exp 78	97.37410072	93.88489209	100
Exp 79	92.12230216	79.0647482	99.99231951
Exp 80	84.59532374	83.70503597	99.97510155

Appendix 5: plant simulation code

```
-- .Models.Model.MovePatient
-- Function: gives the patient their attribute values. Moves the
patient to the waiting list if there is space. Otherwise the patient
leaves the system.
-- called by: whenever a patient arrives to the system
-- author: Lucas van Haandel
-- date: 2-07-2024

@.arrivaltime := eventController.simTime

var urgencynumber:real := z_uniform(38497,0,1)

if urgencynumber >= 0 and urgencynumber <= urgentpercent/100
    @.urgency := 1
```

```
elseif urgencynumber > urgentpercent/100 and urgencynumber <=
(urgentpercent/100 + normalpercent/100)
    @.urgency := 4
else
    @.urgency := 7
end


@.doctor := ceil(z_uniform(467,0,2))


if @.urgency = 1
    @.deadline:= 5*86400
elseif @.urgency = 2
    @.deadline:= 10*86400
elseif @.urgency = 3
    @.deadline := 20*86400
elseif @.urgency = 4
    @.deadline := 30*86400
else
    @.deadline := @.urgency*15*86400
end

--@.surgerytime:= 60*1*60

@.surgerytime := 240*61
while @.surgerytime > 240*60 or @.surgerytime < 30*60
    @.surgerytime := exp(z_normal(5,4.31,0.649))*60
end

Nrpatients += 1
@.PatientNr := nrpatients

if @.patientnr = 0
    debug
end


var queuelength: integer
var Queuetime: time:= 0
var j: integer
queuelength:= .models.model.waitingbeforePlanning.contentsList.ydim

for j := 1 to queuelength
    queuetime += waitinglist[2,j]
next


if queuetime + @.surgerytime > SSlevel*3600
        @.move(nospace)
else
    Waitinglist.appendrow(@.PatientNr, @.urgency ,@.surgerytime,
@.arrivaltime, @.arrivaltime + @.deadline, @.category)
    waitinglist.sort(4,"up")
    @.move(waitingbeforePlanning)
```

```
    if @.arrivaltime mod 86400 > 8*3600 and @.arrivaltime mod 86400
<= 16*3600
        planpatient(@)
    end

end


-- .Models.Model.PlanPatientCaller
--function: try to plan every patient on the waiting list.
--called by: called daily by initday
--author: Lucas van Haandel
--date: 2-07-2024
var waspatientplanned: boolean:= false

for var j:= 1 to waitinglist.ydim
    var patient: object
    var Patientnumber : integer
    var patientname: string


    patientnumber := waitinglist[0,j]
    patientname:= ".userobjects.patient:"+ patientnumber
    patient := patientname

    waspatientplanned:= planpatient(patient)

    if waspatientplanned = true
        exitloop
    end

next

if waspatientplanned = true
    planpatientcaller
end


-- .Models.Model.WarmUpCalculator
--called every day by initday to measure the average current dwell
time in the waiting lists.



var totaltimeinsystem: time:= 0
        var waitingbeforedimension: integer :=
.models.model.waitingbeforePlanning.contentslist.ydim
        var patient : object
        var contentlist: table

.models.model.waitingbeforePlanning.contentslist(contentlist)
        for var i := 1 to waitingbeforedimension
            patient := contentlist [1,i]
            totaltimeinsystem+= eventController.simtime -
patient.arrivaltime
```

```
        next
        contentlist.delete
        var waitingafterdimension: integer :=
.models.model.waitingafterplanning.contentslist.ydim
        .models.model.waitingafterPlanning.contentslist(contentlist)
        for var j := 1 to waitingafterdimension
            patient := contentlist[1,j]
            totaltimeinsystem+= eventController.simtime -
patient.arrivaltime
        next

        var avgtimeinsystem: real
        avgtimeinsystem := (totaltimeinsystem
/(waitingbeforedimension + waitingafterdimension))/86400
        var runNr: integer := experimentManager.CurrRunNo
        averageTimeinSystem[runNr, daynr]:= avgtimeinsystem


-- .Models.Model.PlanPatient
-- Function: plans a patient. Returns true or false based on whether
planning was successful
-- called by: planpatientcaller, movepatient(if the patient arrives
within working hours)
-- author: Lucas van Haandel
-- date: 16-07-2024
param patient: object
-> boolean

var slotsRequired:integer := ceil(patient.surgeryTime /900)

-- Find the most suitable slots for the next 30 days
var patientdeadline:integer:=
ceil((patient.arrivaltime+patient.deadline -
eventController.simtime)/86400)
var patienturgency: integer

if patientdeadline > 30
    patienturgency := 3 -- not urgent
elseif patientdeadline >5 and patientdeadline <= 30
    patienturgency := 2 -- normal
elseif patientdeadline <= 5
    patienturgency := 1
end



findbestslotFor30Days(patient, patienturgency)

if patienturgency = 1
    mostsuitableslots.sort(3,2,"up" )
elseif patienturgency = 2
    mostsuitableslots.sort(3,4,2,sortnormal  )
elseif patienturgency = 3
    mostsuitableslots.sort(3,4,2,sortnoturgent )
end
```

```
var ontime:boolean


-- Handle the chosen slot (e.g., update the schedule database, move
the patient from WaitingBeforePlanning to WaitingAfterPlanning)
if mostsuitableslots.ydim /= 0 then

    patient.move(WaitingAfterPlanning)



scheduledPatients.appendRow(patient.patientnr,mostSuitableSlots[1,
1], mostSuitableSlots[2, 1], slotsRequired, patient.deadline,
ontime, patient.arrivaltime)

    var ydim: integer:= scheduledpatients.ydim
    var surgerylength: integer := scheduledPatients[3,ydim]

    var surgerystart, surgerystop, currentvalue: integer
    surgerystart := scheduledPatients[1, ydim]
    surgerystop := scheduledpatients[1,ydim] + surgerylength - 1

    for var i:= surgerystart to surgerystop
        currentvalue :=
availableschedule[scheduledPatients[2,ydim],i]
        availableschedule[scheduledPatients[2,ydim],i] :=
currentvalue - 1
    next



    var i: integer := waitinglist.getrowno(patient.patientnr)
    if patient.arrivaltime + patient.deadline -
((mostsuitableslots[2,1]-1) * 86400) > 0
        ontime := true
    else
        ontime:= false
    end
    waitinglist.cutRow(i)

    var timetoOr: time:=
Scheduledpatients[2,scheduledpatients.ydim]*86400 +
(Scheduledpatients[1,scheduledpatients.ydim]-1)*900 -
eventcontroller.simtime

    &MovetoOR.methcall(timetoOR, patient)
    result:= true

    scheduledpatients[5, scheduledpatients.ydim] := ontime
    --remove the patient from the waiting list

elseif mostSuitableSlots.ydim = 0 and ceil((patient.arrivaltime +
patient.deadline)/86400) - daynr <= 5
```

```
        mightgetkicked.delete
        var urgentdeadline:integer := ceil((patient.arrivaltime +
patient.deadline)/86400)
        if urgentdeadline <daynr
            urgentdeadline := 4 + daynr
        end



        for var j := 1 to scheduledPatients.ydim
            if ceil((scheduledPatients[ 4,j]+
scheduledPatients[6,j])/86400)- daynr > 5  and
scheduledpatients[2,j]+1 - daynr >= 0 and scheduledpatients[2,j] <=
urgentdeadline and slotsrequired <= scheduledpatients[3,j]


                var urgentPatientnumber : integer
                var urgentpatientname: string
                var urgentpatientnamename: object

                urgentpatientnumber :=scheduledpatients[0,j]
                urgentpatientname:= ".userobjects.patient:"+
urgentpatientnumber
                urgentpatientnamename:= urgentpatientname

                if urgentpatientnamename /= void
                    if urgentpatientnamename.location =
.models.model.waitingafterplanning

mightgetkicked.appendrow(scheduledpatients[0,j],scheduledpatients[4,
j] + urgentpatientnamename.arrivaltime, -1*(scheduledpatients[3,j]-
slotsrequired))
                    end
                end
            end
        next
        mightgetkicked.sort(3,2,"down")



        if mightgetkicked.ydim /= 0

            var cutUrgentPatientnumber : integer :=
mightgetkicked[1,1]
            var cutUrgentpatientstring: string :=
".userobjects.patient:"+ cutUrgentpatientnumber
            var cutUrgentpatient: object := cutUrgentpatientstring
            var cutUrgentpatientrowno :=
scheduledpatients.getrowno(cutUrgentpatientnumber)
            --add new person to scheduledpatients

            scheduledpatients.appendrow(patient.patientnr,
Scheduledpatients[1,cutUrgentpatientrowno],Scheduledpatients[2,cutur
gentpatientrowno], slotsRequired, patient.deadline, ontime,
patient.arrivaltime)
```

```
            var funkyUrgenttimetoOr: time:=
Scheduledpatients[2,scheduledpatients.ydim]*86400 +
(Scheduledpatients[1,scheduledpatients.ydim]-1)*900 -
eventcontroller.simtime

            if patient.arrivaltime + patient.deadline -
((Scheduledpatients[2,cuturgentpatientrowno]-1) * 86400) > 0
                ontime := true

            else
                ontime:= false


            end

            -- add 1's back to the availableschedule if we have
slotsleftover
            var slotsleftover:= mightgetkicked[3,1]*-1


            if slotsleftover /= 0
                if slotsleftover <0
                    debug
                end

                for var k:= 1 to slotsleftover

availableschedule[scheduledpatients[2,scheduledpatients.ydim],
scheduledpatients[1,scheduledpatients.ydim] +
scheduledpatients[3,scheduledpatients.ydim]-1+ k]+= 1
                next
            end


            scheduledpatients[5, scheduledpatients.ydim] := ontime
            &MovetoOR.methcall(funkyUrgenttimetoOR, patient)
            scheduledpatients.cutrow(cutUrgentpatientrowno)

waitinglist.cutrow(waitinglist.getrowno(patient.patientnr))
            patient.move(WaitingAfterPlanning)
            cutUrgentpatient.move(waitingbeforeplanning)
            waitinglist.appendRow(cutUrgentpatientnumber ,
cutUrgentpatient.urgency, cutUrgentpatient.surgerytime,
cutUrgentpatient.arrivaltime,
cutUrgentpatient.arrivaltime+cutUrgentpatient.deadline)

            waitinglist.sort(4,"up")
            if daynr > warmup+1 and daynr <= warmup +
simulationlength+1
                replannedpatients += 1
                replannedslotsReturned += slotsleftover
            end
            result:= true
        end

end
```

```
-- .Models.Model.FindbestSlotforDay
-- Function: finds the best slot for a patient within MSS opening
and closing times for a specific day. Returns the best slotsnumber,
the associated quality, and the quality of the planning spread that
planning the patient in the slot would provide.
-- called by: findbestslotfor30days
-- author: Lucas van Haandel
-- date: 16-07-2024

param patient: object, currentday: integer, endday: integer,
wasitbusy: boolean
-> list[real] -- best slot for that day, integer; the quality of
fit, Real; the quality of spread, real; wasitbusy, boolean
result.create
var isdayopen:boolean:= isdayopen(currentday)
var isittoobusy: boolean

--determine the patients urgency
var patientdeadline:integer:= ceil((
patient.arrivaltime+patient.deadline - eventController.simtime
)/86400)
var patienturgency: integer
var slotsRequired:integer := ceil(patient.surgeryTime /900)
if patientdeadline > 30
    patienturgency := 3 -- not urgent
elseif patientdeadline >5 and patientdeadline <= 30
    patienturgency := 2 -- normal
elseif patientdeadline <= 5
    patienturgency := 1
end



-- the part between this and the next comment exist to improve
performance and reduce the number of times the method isittoobusy is
called. References to wasitbusy in FindbestSlotfor30days or this
method are for the same reason.
-- explanation: if it was too busy to plan a patient on day x-1, it
would also be too busy to plan a patient on day x. Therefore today
'isittoobusy' will be the same as it was yesterday. The variable
wasitbusy exists to pass today's business status to the next
planning day.
-- exceptions: on day 1, we still need to calculate whether it is
too busy. When the benchmark for if it is too busy changes we also
need to recalculate whether it is too busy.
if  patienturgency = 3 and currentday = daynr
    isittoobusy := isitbusy(patient, endday, patienturgency,
currentday)
elseif patienturgency = 3 and currentday-daynr = 5
    isittoobusy := isitbusy(patient, endday, patienturgency,
currentday)
elseif patienturgency = 3 and currentday-daynr /= 5 and currentday
/= daynr
```

```
    isittoobusy:= wasitbusy
elseif patienturgency = 2 and currentday = daynr
    isittoobusy := isitbusy(patient, endday, patienturgency,
currentday)
elseif patienturgency = 2 and currentday /= daynr
    isittoobusy := wasitbusy
elseif patienturgency = 1
    isittoobusy := false
end




-- the results list does not want to pass booleans, so we convert it
to a different datatype. It is converted back in
findbestslotfor30days.
var isittoobusyasreal: real
if isittoobusy = true
    isittoobusyasreal:= 1
elseif isittoobusy = false
    isittoobusyasreal:= 0
end
result[4] := isittoobusyasreal
-- end of the performance improvement part

if isdayopen = true and isittoobusy = false
    result[3] := spreadquality(patienturgency, slotsrequired,
currentday)
  var openingTime:integer := startTime(currentday) -- first cell of
mss where cell /= 0
  var closingTime:integer := EndTime(currentday, openingtime) --
first cell of mss where after the starttime cell = 0




    for var SlotNr := openingtime to closingtime
        var quality: real := 0
        var isavailable: boolean := isslotavailable(slotnr,
currentday) -- see if the slot is available

        if isavailable = true -- if it is available, check if there
are enough slots available
            var nextUnavailable := NextUnavailableSlot(currentday,
SlotNr, closingtime)

            -- Calculate the number of slots until the next
operation, or closing time
            var NumSlotsAvailable:integer := min(nextunavailable -
slotNr, closingtime - slotNr )

            -- Calculate the number of slots required for the
surgery


            -- Check if there are enough slots available until the
```

```
next slot or closing time
            if numslotsAvailable >= slotsRequired


                var slotsleftafter :=   numslotsavailable -
slotsrequired

                if slotsleftafter = 1
                    quality += 0.1
                elseif slotsleftafter =  2
                    quality += 0.3
                elseif slotsleftafter =  3
                    quality += 0.7
                elseif slotsleftafter =  4
                    quality += 1.5
                elseif slotsleftafter =  5
                    quality += 3.1
                else
                    quality += 6.3
                end
                var slotsleftbefore: integer
                for var i := 1 to 3
                    if isslotavailable(slotNr - i, currentday) =
true
                        slotsleftbefore += 1
                    end
                next
                if slotsleftbefore = 1
                    quality += 0.1
                elseif slotsleftbefore = 2
                    quality += 0.3
                elseif slotsleftbefore = 3
                    quality += 0.7
                elseif slotsleftbefore = 4
                    quality += 1.5
                elseif slotsleftbefore = 5
                    quality += 3.1
                else
                    quality +=6.3
                end

                --if the slot fits perfectly in the given schedule,
the quality should be the highest it can possibly be. Higher than
keeping some slots before or after.
                if slotsleftbefore = 0 and slotsleftafter = 0
                    quality := 9001
                end



                if quality > result[2]
                    result[1] := slotNr
                    result[2] := quality
                end
            end
        end
```

```
    next
end


-- .Models.Model.FindbestSlotfor30days
-- Function: looks for the best slots to schedule a patient for the
next 30 days. Fills these slots with associated qualities into the
'mostsuitableSlots' data table.
-- called by: planpatient
-- author: Lucas van Haandel
-- date: 16-07-2024


param patient: object, patienturgency: integer
mostSuitableSlots.delete

var endday:integer
var currentday: integer
var bestSlotandQuality: list
var bestslot: integer
var quality: real
var spreadquality: real
var wasitbusyasreal: real:=0
var wasitbusy:boolean:= false



var patientdeadline:integer:=
ceil((patient.arrivaltime+patient.deadline -
eventController.simtime)/86400)



if patientdeadline < 1
    patientdeadline := 5
end

if patienturgency = 3
 endday:= daynr + 29
for var dayOffset := 0 to 29

     currentDay := daynr + dayOffset

    if  dravailability[patient.doctor, (((currentday-1) mod 20)+1)]
= true

        bestslotandquality := findbestslotforday(patient,
currentday, endday, wasitbusy)
        bestslot := bestslotandquality[1]
        quality := bestslotandquality[2]
        spreadquality:= bestslotandquality[3]
        wasitbusyasreal:= bestslotandquality [4]
        if wasitbusyasreal = 1
            wasitbusy:= true
        elseif wasitbusyasreal = 0
```

```
            wasitbusy:= false
        else
            debug
        end
        if bestSlot /= 0
            mostSuitableSlots.appendrow(bestSlot, currentday,
qualitynoturgent*quality, spreadqualitynoturgent*spreadquality)
        end
    end
next

elseif patienturgency = 2

    endday := daynr + patientdeadline - 1
for var dayOffset := 0 to patientdeadline-1

     currentDay := daynr + dayOffset

    if  dravailability[patient.doctor, (((currentday-1) mod 20)+1)]
= true

        bestslotandquality := findbestslotforday(patient,
currentday, endday, wasitbusy)
        bestslot := bestslotandquality[1]
        quality := bestslotandquality[2]
        spreadquality:= bestslotandquality[3]
        wasitbusyasreal:= bestslotandquality [4]
        if wasitbusyasreal = 1
            wasitbusy:= true
        elseif wasitbusyasreal = 0
            wasitbusy:= false
        else
            debug
        end
        if bestSlot /= 0
            mostSuitableSlots.appendrow(bestSlot, currentday,
qualitynormal*quality, spreadqualitynormal*spreadquality)
        end
    end
next

elseif patienturgency = 1
    endday := daynr + patientdeadline - 1
    for var dayOffset := 0 to patientdeadline - 1
        currentDay := daynr + dayOffset
        bestslotandquality := findbestslotforday(patient,
currentday, endday, wasitbusy)
        bestslot := bestslotandquality[1]
        quality := bestslotandquality[2]
        spreadquality:= bestslotandquality[3]
        wasitbusyasreal:= bestslotandquality [4]
        if wasitbusyasreal = 1
            wasitbusy:= true
        elseif wasitbusyasreal = 0
            wasitbusy:= false
        else
```

```
            debug
        end
        if bestSlot /= 0
            mostSuitableSlots.appendrow(bestSlot, currentday, -
1*quality, spreadquality) --note that the spreadquality is always 0
for semi-urgent patients, because it does not matter.
        end
    next

end


-- .Models.Model.IsitBusy
-- Function: given a patient's urgency, calculates whether it is too
busy to plan a patient on a certain day.
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 16-07-2024

param patient : object, endday: integer, patienturgency:integer,
currentday:integer
-> boolean

var isittoobusy:boolean


--determine the total nr of slots, and the available slots
var totalslots: integer
var availableslots: integer

var startday: integer:= daynr -1




for var i:= startday to endday
    var mssday:= ((i-1) mod 20)+1
    for var j:= 1 to mss.ydim
        totalslots+= MSS[mssday,j]
        availableslots+= availableschedule[i,j]
    next
next
availableslots -= ceil(patient.surgerytime/900)
if totalslots = 0
    result:= true
    return
end






if patienturgency = 3
/*  if currentday-daynr<=4
```

```
        if availableslots/totalslots >=
leaveurgentslotsopenpercent/100
            isittoobusy:= false
        else
            isittoobusy :=true
        end
    else*/
        if availableslots/totalslots >= leaveslotsopenpercent/100
            isittoobusy:= false
        else
            isittoobusy :=true
        end
    --end
elseif patienturgency = 2
    if availableslots/totalslots >= leaveurgentslotsopenpercent/100
        isittoobusy:= false
    else
        isittoobusy :=true
    end
elseif patienturgency = 1
    isittoobusy := false
else
    debug -- it is never too busy to plan a semi-urgent patient, so
this method should not be called for semi-urgent patients
end

result:= isittoobusy


-- .Models.Model.Spreadquality
-- Function: based on the patient's urgency, slots required, and the
current day that we are trying to plan the patient in, calculate how
much the quality of the spread improves if we were to actually plan
the patient on this day.
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 16-07-2024

param patienturgency, surgerylength, operationday: integer
-> real


--var benchmark: real
var oldspread, newspread: real
var slotspreadquality: real


oldspread := 0
newspread := 0

if patienturgency = 1--semi-urgent patients should always be planned
as soon as possible, no point in calculating.
    result:= 0
    return
end
```

```
var slotsfilled: integer:= 0
var availableslots: integer:= 0
var totalslots: integer:= 0
var mssday:= ((operationday-1) mod 20)+1

for var j:= 1 to mss.ydim

    totalslots+= MSS[mssday,j]
    availableslots+= availableschedule[operationday,j]

next

slotsfilled:= totalslots-availableslots


oldspread := pow((slotsfilled/totalslots)*100, 2)
newspread := pow(((slotsfilled+surgerylength)/totalslots)*100, 2)

slotspreadquality := newspread- oldspread

result:= slotspreadquality -- the lower this number, the more even
the slot spread quality.
return


-- .Models.Model.StartTime
-- Function: checks at what time the operating room opens based on
the MSS
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 2-07-2024

Param Currentday: integer
-> integer
var modexperiment := currentday - 1
var MssDay := ((modexperiment mod 20) +1)
for var i := 1 to MSS.yDim
    if MSS[MssDay, i] /= 0 then
        result := i
        return
    end
next


-- .Models.Model.Endtime
-- Function: checks when the operating room closes, based on the MSS
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 2-07-2024

Param Currentday, starttime: integer
-> integer
 var modexperiment := currentday - 1
```

```
var MssDay := ((modexperiment  mod 20) +1)
if starttime = 0
    result:= 0
    return
end
for var i := starttime to mss.ydim
    if MSS[MssDay, i] = 0 then
        result := i
        return
    end
next


-- .Models.Model.NextUnavailableSlot
-- Function: checks when the next unavailable slot is based on some
current available slot.
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 2-07-2024

Param currentday, currentslot, closingtime: integer
-> integer -- the next unavailable slot after you had some available
slots
var nextslot := currentslot +1
for var i := nextslot to closingtime
    if availableschedule[currentday, i] = 0 and
availableschedule[currentday,i-1] /= 0
        result := i
        return

    end

next


-- .Models.Model.isslotavailable
-- Function: checks if a timeslot is available. returns true or
false.
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 2-07-2024

param slot, daynr: integer
-> boolean

if Availableschedule[daynr, slot] = 0
    result := false
else
    result := true
end


-- .Models.Model.Isdayopen
-- Function: checks if the operating room is opened on a day.
Returns true or false
```

```
-- called by: findbestslotforday
-- author: Lucas van Haandel
-- date: 2-07-2024
Param Currentday: integer
-> boolean
var modexperiment := currentday - 1
var MssDay := ((modexperiment mod 20) +1)
for var i := 1 to MSS.yDim
    if MSS[MssDay, i] /= 0 then
        result := true
        return
    end
next
result:= false


-- .Models.Model.MovetoOR
-- Function: moves a patient to the operating room.
-- called by: methcall from planpatient
-- author: Lucas van Haandel
-- date: 2-07-2024

param patient: object


if patient /=void and scheduledpatients.getrowno(patient.patientnr)
/= -1
    var patientrowno :=
scheduledpatients.getrowno(patient.patientnr)
    if patient.location = .models.model.waitingafterplanning and
scheduledpatients[2, patientrowno]*86400 +
(scheduledpatients[1,patientrowno]-1)*900 = eventcontroller.simtime
    patient.move(operatingroom)
    &Leaveor.methcall(patient.surgerytime, patient)
    end
end


-- .Models.Model.LeaveOR
-- Function: takes a patient out of the operating room. Updates some
KPI's
-- called by: methcall from movetoOR
-- author: Lucas van Haandel
-- date: 2-07-2024

param patient: object

patient.move(exit)
if daynr > warmup+1 and daynr <= warmup + simulationlength+1
    totaloperated += 1
    if
scheduledpatients[5,scheduledpatients.getrowNo(patient.patientnr)] =
false
        totalnotontime += 1
    end
```

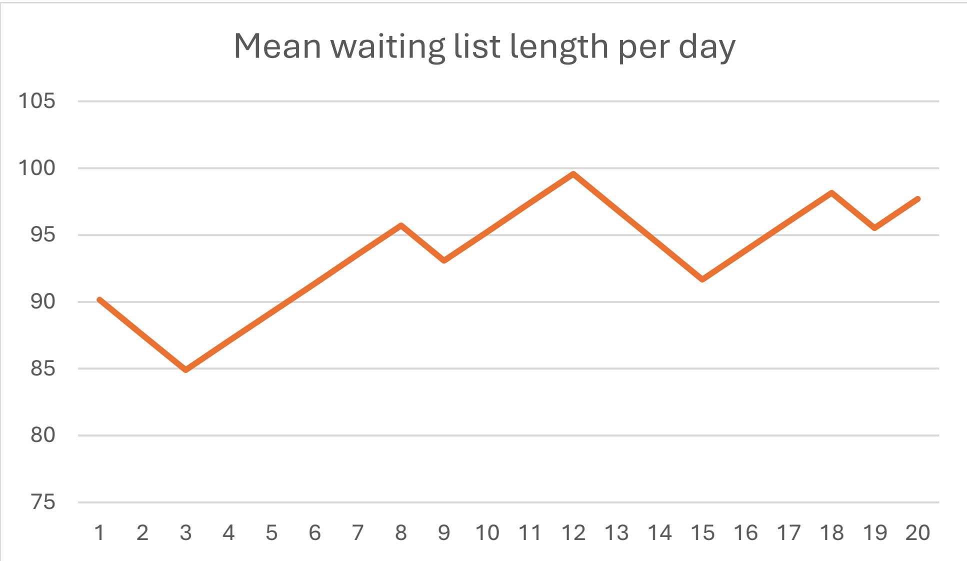
```
end


-- .Models.Model.Init
-- Function: initialises data tables based on the input data.
calculates the total number of slots available during the
simulation.
-- called by: start of simulation
-- author: Lucas van Haandel
-- date: 2-07-2024

eventController.end := (warmup+simulationlength+1)*86400

noturgentpercent := 100 - urgentpercent-normalPercent
-- Clear the Schedule table before copying new data
availableSchedule.delete
var enddate: integer
enddate:=ceil(eventController.end/86400)+30
daynr += 1
-- Loop through the next 30 days
for  var dayNr := 1 to enddate

    var dayIndex := ((dayNr-1) mod 20) + 1  -- Calculate the column
index based on the day, cycling every 20 days

    -- Assuming MSS and Schedule are tables and we need to copy data
from MSS to Schedule
    for var i := 1 to 96   -- Loop through the rows in MSS
        var rowData := MSS[dayIndex, i] -- Get the data from the
specific column in MSS
        availableSchedule[daynr, i]:= rowdata

        if rowdata/= 0
            availableschedule.setbackgroundcolorcolumn(daynr, 3)
        end

/*      if daynr <= enddate -30 and daynr > 30
            totalslotsNR += rowdata
        end*/

    next

    if dayindex = 21
        debug
    end
next


-- .Models.Model.initDay
-- Function: calculates some KPI's and calls the planpatientcaller
every day, which tries to plan all the patients.
-- called by: called at 9:00:00 by generator startoftheday
-- author: Lucas van Haandel
-- date: 2-07-2024
```

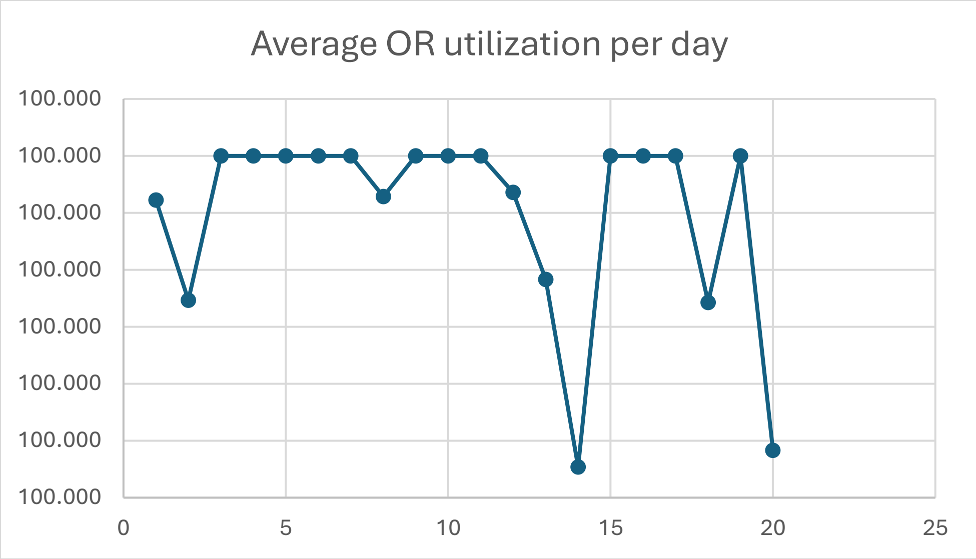
```
if eventcontroller.simtime >= 86400
    daynr += 1
end

if daynr > warmup+1 and daynr <= warmup + simulationlength+1
    var unusedslotstoday := 0
    var totalslotstoday := 0

    if totaloperated /= 0
        ontimepercent := (1-(totalnotontime/totaloperated)) *100
    end

    for var i := 1 to 96
        nrofunusedSlots += availableschedule[daynr-1,i]
        unusedslotstoday += availableschedule[daynr-1,i]
        totalslotsNR += MSS[(((daynr-1)-1)mod 20)+1,i]
        totalslotstoday += MSS[(((daynr-1)-1)mod 20)+1,i]
    next

    if totalslotstoday /= 0
        slotsutility := (1-(nrofUnusedSlots/totalSlotsNR))*100
        daysmeasured += 1

        if (1-(unusedslotstoday/totalslotstoday))*100 >=
utilityThreshold
            daysWithGoodService += 1
        end
    end

    if daysmeasured /= 0
        serviceLevel:= (daysWithGoodService/daysmeasured)*100
    end
end



--warmuptimecalculator


    planpatientcaller


-- .Models.Model.Reset
-- Function: resets all necessary variables and data tables
-- called by: reset
-- author: Lucas van Haandel
-- date: 2-07-2024

deletemovables
NrPatients := 0
Waitinglist.delete({0,1}..{*,*})
DayNr := 0
nrofUnusedSlots := 0
```

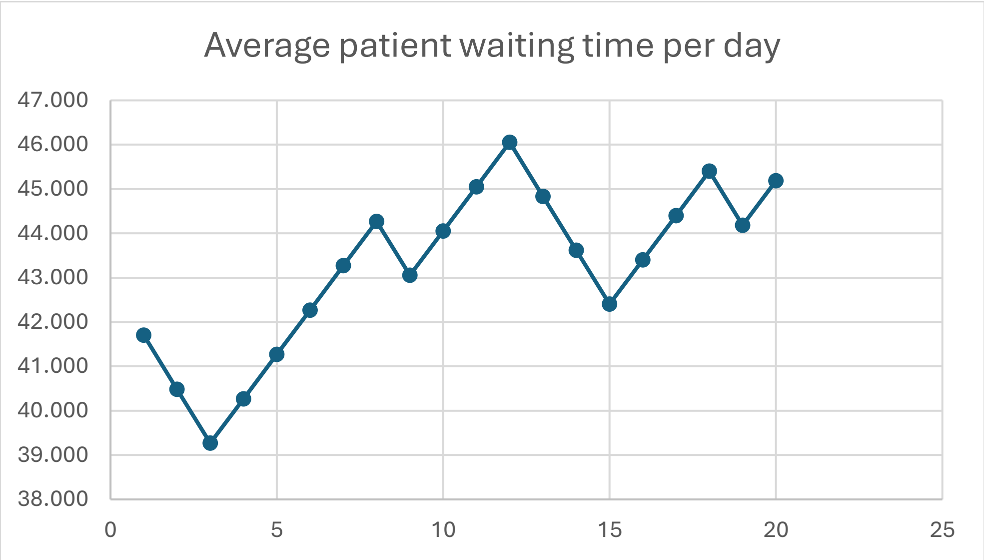
```
totalSlotsNR := 0
totalnotontime:= 0
replannedslotsReturned := 0
totaloperated := 0
slotsutility := 0
ontimepercent := 0
replannedPatients := 0
serviceLevel := 0
daysmeasured := 0
dayswithGoodService := 0
availableschedule.setbackgroundcolorcolumn({0,0}..{*,*},
makeRGBValue(255,255,255))
availableschedule.delete({1,1}..{*,*})

mightgetKicked.delete({1,1}..{*,*})
scheduledPatients.delete({0,1}..{*,*})
mostsuitableSlots.delete({1,1}..{*,*})
--warmupSlotsUtility.delete({1,1}..{*,*})

--warmupslotsutility[experimentManager.
```

Appendix 6: Expected shrinkage and growth tool

The expected shrinkage and growth tool shows how a waiting list changes during an MSS cycle, and how this change impacts the expected utilization and average patient waiting time. The code is adapted from the original Markov tool, however instead of first calculating a steady state, we simply input how many people we have on the waiting list at the start, and use that as an input for the Markov chain. At the end of the cycle we calculate how the waiting list behaved during the cycle.








```
Sub ToolExecute()

Dim CurrentServiceLevel As Double

Dim i As Integer

Dim j As Integer

Dim d As Integer

Dim safetystock As Integer

Dim nextRow As Long

Dim WeightedAverage As Double

Dim total As Double

Dim probability As Double

Dim percentile As Integer

Dim startTime, elapsedTime As Single
```

```
Dim WasanswerFound As Boolean



With ThisWorkbook.Sheets("Dashboard")
    .Range("L2:M" & .Rows.Count).ClearContents
End With

With ThisWorkbook.Sheets("CalculationData")
    .Range("A2:P" & .Rows.Count).ClearContents
End With

With ThisWorkbook.Sheets("dashboard")
    UtilityThreshold = .Cells(13, "B").value
    L = .Cells(5, "B").value
    NrOfdays = .Cells(6, "B").value
    RunLength = .Cells(14, "B").value
    mean = .Cells(7, "B").value
    stdev = .Cells(8, "B").value
    ORtime = .Cells(9, "B").value
    safetystock = .Cells(15, "B").value
    index = .Cells(16, "B").value
    epsilon = .Cells(17, "B").value
    .Range("G5").ClearContents
    .Range("G8").ClearContents
    .Range("G9").ClearContents
End With

CurrentServiceLevel = 0

startTime = Timer
WasanswerFound = True

Do While CurrentServiceLevel < 1

    elapsedTime = Timer - startTime
```

```
    'If elapsed time is greater than allowed, exit the loop
    If elapsedTime > RunLength Then
        WasanswerFound = False
        Exit Do
    End If

    CurrentServiceLevel = MarkovChains(safetystock)
    With ThisWorkbook.Sheets("dashboard")
        nextRow = .Cells(.Rows.Count, "L").End(xlUp).Row + 1
        .Cells(nextRow, "L").value = safetystock
        .Cells(nextRow, "M").value = CurrentServiceLevel
        If CurrentServiceLevel = 1 Then
            .Cells(5, "G").value = safetystock
        End If
    End With
    safetystock = safetystock + epsilon
Loop
With ThisWorkbook.Sheets("dashboard")
    .Cells(9, "G").value = WasanswerFound
End With

For d = 1 To NrOfdays
        WeightedAverage = 0
        probability = 100
        For i = 0 To listlength
            WeightedAverage = WeightedAverage + (i * BP(d, i))
            If MSS(d) = True Then
                    probability = probability - 100 * (BP(d, i) *
utilityprobability(i))
            End If
        Next i

          With ThisWorkbook.Sheets("CalculationData")
        ' Find the next empty row in column A
```

```
        nextRow = .Cells(.Rows.Count, "A").End(xlUp).Row + 1

        ' Paste values into columAP A and B in the next empty row
        .Cells(nextRow, "A").value = d
        .Cells(nextRow, "B").value = WeightedAverage
        .Cells(nextRow, "C").value = probability
        .Cells(nextRow, "D").value = WeightedAverage / L
        End With
Next d

With ThisWorkbook.Sheets("CalculationData")
    For i = 0 To listlength
        .Cells(i + 2, "F").value = i
        .Cells(i + 2, "G").value = BP(NrOfdays, i) 'add the chances that there are
i people on the waitlist at the end just for extra information
    Next i
End With

  ' Turn all the values in the array to 0 to be safe
    For i = 0 To NrOfdays
        For j = 0 To listlength
                AP(i, j) = 0
                BP(i, j) = 0
        Next j
    Next i

' Update the chart sizes
    Dim lastRow As Long
    Dim startRow As Long
    Dim rangeAddress1 As String
    Dim rangeAddress2 As String
    Dim rangeAddress3 As String


    ' Calculate the last row for the first range
    lastRow = NrOfdays + 1
```

```
    rangeAddress1 = "A2:B" & lastRow


    ' Update Chart 1

    With ActiveSheet.ChartObjects("Chart 4")

        .Activate

        Application.CutCopyMode = False

        ActiveChart.SetSourceData
Source:=Sheets("CalculationData").Range(rangeAddress1)

        ActiveChart.FullSeriesCollection(1).IsFiltered = True

        ActiveChart.FullSeriesCollection(2).IsFiltered = False

    End With


    ' Calculate the address for the second range


    startRow = 1

    rangeAddress2 = "A" & startRow & ":A" & lastRow & ",C" & startRow & ":C" &
lastRow


    ' Update Chart 2

    With ActiveSheet.ChartObjects("Chart 5")

        .Activate

        Application.CutCopyMode = False

        ActiveChart.SetSourceData
Source:=Sheets("CalculationData").Range(rangeAddress2)

    End With


    ' Calculate the address for the third range


    rangeAddress3 = "A" & startRow & ":A" & lastRow & ",D" & startRow & ":D" &
lastRow


    With ActiveSheet.ChartObjects("Chart 6")

        .Activate

        Application.CutCopyMode = False

        ActiveChart.SetSourceData
Source:=Sheets("CalculationData").Range(rangeAddress3)

    End With
```

```
End Sub

Function MarkovChains(safetystock As Integer) As Double

    Dim i As Integer
    Dim j As Integer
    Dim d As Integer
    Dim ServiceLevel As Double
    Dim totalservicelevel As Integer
    Dim WeightedAverageNew, WeightedAverageOld As Double
    Dim Convergence As Double
    Dim loopnr As Integer



    ' Initialize the variables with some values (if needed)
    listlength = safetystock + index
'    WeightedAverageOld = 0
'    loopnr = 0
'    Convergence = epsilon + 1

    ReDim AP(0 To NrOfdays, 0 To listlength)
    ReDim BP(0 To NrOfdays, 0 To listlength)

'    Do While Convergence > epsilon


    ' Turn all the values in the array to 0 to be safe
'    If loopnr = 0 Then
        For i = 0 To NrOfdays
            For j = 0 To listlength
                If i = 0 And j = safetystock Then 'at the start, the chance of one
number of people on the waitlist is 1
                    AP(i, j) = 1
                    BP(i, j) = 0
                Else
```

```
                    AP(i, j) = 0 'the rest is just cleaning up the array for safety
                    BP(i, j) = 0
                End If
            Next j
        Next i
'    Else
'        For i = 0 To NrOfdays
'            For j = 0 To listlength
'                If i = 0 Then 'at the start, the probability after planning on day
0 is the
'                    AP(i, j) = AP(NrOfdays, j)
'                    BP(i, j) = 0
'                Else
'                    AP(i, j) = 0 'the rest is just cleaning up the array for
safety
'                    BP(i, j) = 0
'                End If
'            Next j
'        Next i
'    End If

    For d = 1 To NrOfdays

        'add the chances of i people existing on the waiting list before planning.
Based on the chance that i-j people arrive, given the probability of yesterdays
waiting list length being i
        For i = 0 To listlength
            If i <> listlength Then
                For j = 0 To i
                    BP(d, i) = BP(d, i) + AP(d - 1, j) *
WorksheetFunction.Poisson_Dist((i - j), L, False)
                Next j
            Else
                For j = 0 To i

                        BP(d, i) = BP(d, i) + AP(d - 1, j) * CumProbability((i -
j), L)
```

```
                Next j
            End If

        Next i

        'add the chances of people existing after planning, based on whether the OR is open today
        If MSS(d) = True Then


            For i = 0 To listlength
                For j = 0 To listlength - i
                    'the probability that there are i people after planning, is the probability that there were i+j people before planning* the probability that j-i people were planned

                    AP(d, i) = AP(d, i) + BP(d, i + j) * NrOfSurgeries(j, i)
                Next j

            Next i
        Else
            For i = 0 To listlength
                AP(d, i) = BP(d, i)
            Next i
        End If



    Next d


'        WeightedAverageNew = 0
'        For i = 0 To listlength
'                WeightedAverageNew = WeightedAverageNew + (i * BP(1, i))
'        Next i
'
```

```
'        Convergence = WeightedAverageNew - WeightedAverageOld
'        WeightedAverageOld = WeightedAverageNew
'        With ThisWorkbook.Sheets("Dashboard")
'            .Cells(8, "G").value = loopnr
'        End With
'
'        loopnr = loopnr + 1

'
'    Loop

    totalservicelevel = 0

    For d = 1 To NrOfdays
        ServiceLevel = 100

        If MSS(d) = True Then

            For i = 0 To listlength
                    ServiceLevel = ServiceLevel - 100 * (BP(d, i) *
utilityprobability(i)) 'the probability that our expected lost utilization is not
below our service level
            Next i

            If ServiceLevel > UtilityThreshold Then
            totalservicelevel = totalservicelevel + 1
            End If
        Else
            totalservicelevel = totalservicelevel + 1
        End If

    Next d

    MarkovChains = totalservicelevel / NrOfdays
```

```
End Function

Function MSS(day As Integer) As Boolean

    Select Case day
        Case 1, 2, 8, 12, 13, 14, 18, 20
            MSS = True
        Case Else
            MSS = False
    End Select

End Function

Function CumProbability(x As Integer, L As Double) As Double

If x = 0 Then
        CumProbability = 1
Else
        CumProbability = 1 - WorksheetFunction.Poisson_Dist(x - 1, L, True)
End If

End Function

Function NrOfSurgeries(vectorsize As Integer, state As Integer) As Double

'The NrOFSurgeries is the probability that the amount of OR minutes on the waiting list is enough to perform vectorsize surgeries
'The Nrofsurgeries is thus the probability of at least vectorsize surgeries - the probability of at least vectorsize+1 surgeries

Dim probability As Double

If state = 0 Then
    If vectorsize = 0 Then
        probability = 1
    Else
```

```
        probability = WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime / vectorsize), mean, stdev, True)

    End If

ElseIf state = listlength Then

        probability = 1 - WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime / (vectorsize + 1)), mean, stdev, True)

Else

    If vectorsize = 0 Then

        probability = 1 - WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime / (vectorsize + 1)), mean, stdev, True)

    Else

        probability = 1 - WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime / (vectorsize + 1)), mean, stdev, True) - (1 - WorksheetFunction.Norm_Dist(WorksheetFunction.Ln(ORtime / vectorsize), mean, stdev, True))

    End If

End If



NrOfSurgeries = probability


End Function


Function utilityprobability(state As Integer) As Double

'the utilityprobability is the probability that the amount of OR minutes is at least the ORtime, given the number of people on the waiting list(state)

'the utilityprobability is the probability that with state number of surgeries, we still dont have enough ORtime to fill the entire schedule

'the utilityprobability is therefore the probability that state surgeries take less than the ortime.

'the probability that x surgeries take less than the ortime is the probability that one surgery takes less than ln(ortime/x)

'if our state number of surgeries is 0, the probability of not having enough ORtime is 1

Dim LNOR As Double


If state = 0 Then

    utilityprobability = 1

Else
```

```
    LNOR = WorksheetFunction.Ln(ORtime / state)

    utilityprobability = WorksheetFunction.Norm_Dist(LNOR, mean, stdev, True)

End If


End Function
```