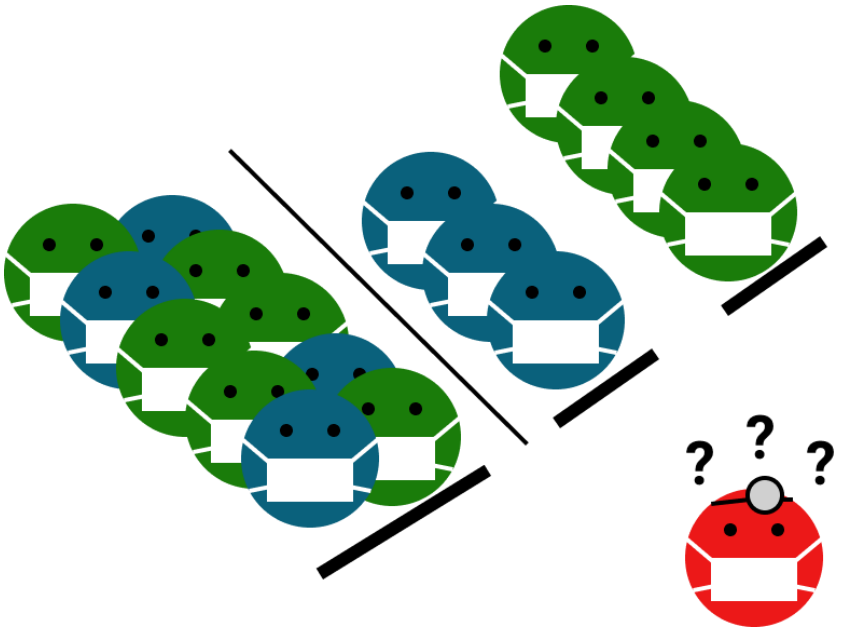
Developing a Web-based Tool for Teaching the Effects of Pooling on Queueing Systems

by Philipp Ungrund (s2121778)



University of Twente

Industrial Engineering and Management
Faculty of Behavioural, Management and Social Sciences (BMS)

AUGUST 2021

Supervisors

Prof. Dr. ir. E.W. Hans
UT & CHOIR
Ir. Rob Vromans
RHYTHM B.V.
Dr. ir. A.G. LEEFTINK
UT & CHOIR

UNIVERSITY OF TWENTE. rhythm

Management Summary

Motivation

The practical experience of consulting healthcare professionals with the goal of improving (logistical) healthcare performance shows there are many reasons for the currently poor performance. One aspect that stands out and that seems comparatively easy to improve upon are the troubles of healthcare professionals to properly and intuitively understand various concepts from operations management as well as operations research. Thus, the idea to develop a digital educational tool that incorporates game aspects. This tool is supposed to create and support a teaching process that actively engages the learner, with requirements on achieving a more intuitive understanding, good flexibility and high accessibility by realization in a web environment. This also comes with the possibility of applying the tool in general education besides consulting healthcare professionals. In light of scope and feasibility we decided to focus the educational tool on queueing theory and the effects of pooling queues. The learning objectives are hereby on understanding that pooled and unpooled queueing systems behave differently and pooled systems in many cases are beneficial in terms of logistical performance. But also, that there are situations in which pooling is not beneficial at all and even diminishes performance.

Approach

We first research the literature about pooling effects on queueing systems and work out the theoretical foundation needed behind the educational tool. We also research educational tools itself and the closely related topic of serious games. Two already existing educational tools for similar use cases are also discussed and analyzed in detail. This research ultimately culminates in the decision for the application of a conceptual modeling framework for serious games based on simulation, which is then iteratively applied in the development of the educational tool. The conceptualized tool is then implemented with the help of the programming language R, the R framework Shiny, and the web programming language JavaScript paired with a library called D3. The framework Shiny hereby offers the functionality to create web applications without pronounced web development skills and D3 provides methods for data-driven visualizations. Consequently, the R source code is responsible for handling the underlying simulation model and general interface, and JavaScript with D3 for the interface's output and visualizations.

Results

The finished and running educational tool can be found at:

---

https://philippun.shinyapps.io/queueing-ed-tool/

In the navigation bar at the top the tool provides a tab with the main application and a tab titled *About* giving background and context information. The main application has a sidebar on the left for custom

scenario settings and the output and visualizations in the center and right. The system's status is represented for the player by an unpooled and pooled system in direct comparison that each have two types of patients and two doctors that can serve any patient in the pooled system and only one in the unpooled. The representation is supported with graphs plotting the historic performance of both systems, allowing for better comparison. The game aspects are achieved through the involvement of a game operator, i.e. the user applying the tool in teaching or consulting, by setting the application into different scenarios that represent the varying learning purposes and challenging the player to assess the systems performance. After the player comes to an assessment the theoretical performance of each system layout in terms of patient's expected waiting time can be displayed by clicking the buttons labeled 'Show Expected Waiting Times'.

Conclusion and Outlook

The developed educational tool complies with the specified objectives from a theoretical point of view. From the practical point of view, however, and the achievement of the learning objectives the tool needs to be validated by experimentation and real-world application, which is out of the scope of this project. Nevertheless, should this validation lead to a negative result, the tool offers various possibilities for further improvements and/or adaptations. One that might be of special interest is the automation of the game operator's function to help make the tool (potentially) much more valuable as a stand-alone application.

Contents

1 Introduction

This first section clarifies the Why, What, and How to this project by discussing each of them after the other. First the motivation, that provides the context and identifies the underlying core problem. Then the objectives, that define the specific goal in terms of the scope and result in the research questions that need to be answered. And finally, the approach that lies down the path to the goal and presents the tools utilized on the way.

1.1 Motivation

1.1.1 Context and Current Situation

The context in which this project was conceived are the activities and work of the research group CHOIR and the company Rhythm. CHOIR is an acronym for "Center for Healthcare Operations Improvement & Research" and was founded at the University of Twente in 2003 by mathematicians and industrial engineers. Their self-imposed objective is "to help healthcare institutions improve the efficiency, quality of care and service, and quality of labor through redesigning and optimizing their processes". To achieve this they design and apply models and insights from the field of operations research and operations management to healthcare environments. By 2014 their work became so involved with the practical application and consultation of hospitals that they decided to establish a spin-off company. The result of this is Rhythm, which was created as a joint venture with the international software company ORTEC. In 2021, Rhythm has grown to about 20 employees and maintains its close connection to CHOIR. Both organizations current focus is mainly on hospitals and Rhythm advises multiple hospitals throughout the Netherlands, as well as a few more across Europe at the time.

CHOIR and Rhythm describe the state of individual hospitals, and healthcare institutions in general, as a compartmentalized system that is not properly capable of integral planning. Each compartment, or 'silo' as they often refer to it, is considered as a separate entity and only responsible for its own success. Silos can be of different sizes and complexities, but often appear along the boundaries of departments or wards, as would be expected. Some consequences that result from this are that certain silos are considered as more important than others, like for example the surgery department. In the case of the surgery department, hospital management infers this importance from the high costs that accumulate here and decides on this basis to give it planning priority with the intention to have as high a utilization as possible. Forcing the other silos to come to terms with it and accept a subordinate role. Besides that, however, the situation presents itself diffuse and highly complex with each hospital or institution developing its own dynamics.

1.1.2 Problem Introduction and Core Problem Identification

The problems arising from the current situation in hospitals are manifold and complex. They reach from doctors being badly utilized or patients experiencing long waiting times, to missing or hoarding supplies and nurses having to work overtime. Figure 1 depicts some of these as part of a problem cluster

in the second last column. What most of them have in common though, at least from an operations management point of view, is the outcome of poor (logistical) performance and the resulting dissatisfaction of patients. This dissatisfaction often not only affects the patients themselves but the employees as well. It can be seen as the overall action problem that this project tries to improve upon. Underlying all this in a lot of cases is the inefficiency of planning and the systems that build the foundations for the planning. This is certainly not the only cause, but we believe it is one of the main ones and for the purpose of this project the most relevant. To give an example, one could think of other causes like low motivation, but these are often even more complex psychological issues, that might, once again, stem from planning inefficiencies and are hard to classify.

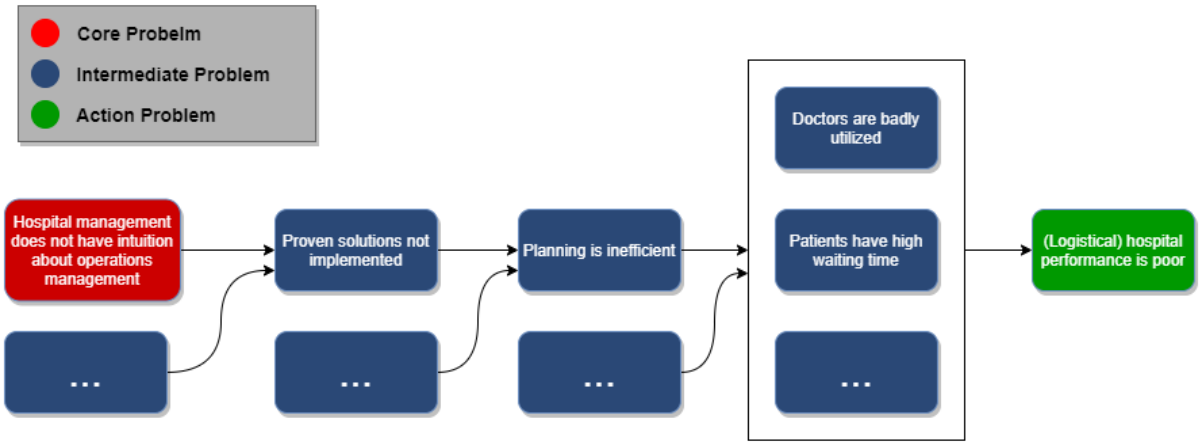


Figure 1 - Problem Cluster

Considering all this, and the importance of healthcare to people and society, it is quite urging, maybe even frustrating, to know that a lot of proven solutions to tackle the planning inefficiencies already exist in theory. They are just not implemented (yet). Again, many reasons for this can be thought of and there is likely not a single proven solution that is not implemented due to only one specific cause. CHOIR and Rhythm, though, have identified one major problem that is a prerequisite for the solution implementation. They say, that from their practical experience with working in hospitals, they know that much of their time is not actually spent implementing the solutions, but explaining to healthcare professionals insights from operations managements that are at the core of the solutions. The knowledge of these insights needs to be gained first to convince and make sure the solutions are used permanently. Healthcare professionals and management most often have a strong background in medical education but only rudimentary training in operations management. So, in this way, the reality of hospital management not having (enough) knowledge and intuition about operations management is the core problem to this project that needs to be solved.

1.2 Objectives

1.2.1 Goal and Scope

The problem description and identification show that the overall goal should be to find a better way of educating healthcare professionals about operations management, as the current approach is hardly sufficient. The term healthcare professionals is deliberately chosen to include a wide ranging field of people like for example students that will work in healthcare in the future as well. In anticipation of this we decided to develop an educational tool in a digital form. The idea for the tool includes the usability in a web environment and aspects of serious gaming. The web environment, on the one hand, should ensure high availability and flexibility, so that people can use it from everywhere and across different platforms. The need to install or download anything is also eliminated then and a spontaneous application of the tool in lectures or similar becomes possible. Serious gaming, on the other hand, is the idea of using computer games in the context of education by using elements of entertainment, like levels and high scores, to raise curiosity and engagement. During the last decades an entire sector for serious gaming has developed that ranges from flight simulators to math games for preschool children. We believe that the tool solution also opens up the possibility of having a general and wider impact on healthcare management. A good example for this are the Excel tools (see Section 2.2.1) that were developed by Richard S. Steyn with a similar purpose about 20 years ago. They are still used widely today and though by now they seem dated, judging by current standards, they clearly show the overall potential and need of these solutions.

Considering the scope and time limitations (10 weeks) that this project imposes, we decided to focus the educational tool on one specific insight from operations management. Later on, this tool can then potentially be combined with educational tools from projects having the same goal but focus on another insight, that are undertaken by fellow students in parallel to this project. The specific insight, with which this project is concerned, is the pooling effects that can occur in queueing systems that are present in many healthcare settings. Think for example the access queue to the surgery department. An additional restriction imposed on this project is the still ongoing COVID-19 pandemic, preventing access to hospitals and other healthcare institutions, which might especially make evaluation of the final tool more difficult and less extensive. This should again underline this projects focus on the conceptualization and implementation of the tool itself. Nevertheless, the intended deliverables of this project include next to the final tool itself, a users guide or manual for the tool, as well as answers to some knowledge problems that necessarily need to be answered before conceptualizing a tool in order to be successful. The knowledge problems are discussed in the next section.

1.2.2 Research Questions

There are two main knowledge problems that come up for the development of this specific educational tool. The first one is about operations management and the particular insight chosen. So, in essence, it is about queuing systems in general and the effects of pooling queues. These effects vary for different circumstances and scenarios. The second problem that arises is about educational tools and serious

gaming from a healthcare perspective. It is not entirely clear what tools are already out there and what properties make them effective or whether they even are effective. Stated in form of a question the two knowledge problems are:

1. What are pooling effects in healthcare queueing systems?
2. What makes an educational tool in the context of healthcare and queueing effective?

Following from these main questions are several sub-questions that constitute the overall answer. For the first question two problems are fundamental: What is queueing theory in the context of healthcare? And: What is pooling in these queueing systems? Only after finding the answer to these, the sub-question of which queues are relevant to healthcare settings can be answered. Which then leads to: How does pooling affect these queues? Answering these four sub-questions should be sufficient to answer the first main question. For the second main question the situation is similar, though the order of which the sub-questions are discussed is less important. It needs to be known what makes a good educational tool in general. Next to that, it is helpful to know which educational tools already exist for healthcare logistics and, separately, queueing in general. These can then be assessed in terms of how effective they are.

1.3 Approach

1.3.1 Steps

The overall approach to this project is divided into four basic stages.

1. Problem Description with Plan
2. Knowledge Gathering
3. Tool Development
4. Tool Evaluation and Report Finalization

The first stage includes the motivation and problem description with the setting of the objectives. This is addressed with the project plan and mostly finalized with this problem solving approach. After that the main research questions with the sub-questions need to be answered by means of exploratory and descriptive research. As this stage serves as the foundation for the educational tool, it is especially important to ensure the validity and integrity of the answers. Existing literature should be reviewed and experts in the respective fields consulted. Some of the questions can also be answered by revisiting books and study material from earlier in the Bachelor program. The results and gathered knowledge should be documented in parallel, with Sections 2 and 3 serving exactly this purpose. The following stage 3 is the core phase with the actual solution development; in this case the educational tool. The tool needs to be conceptualized and designed first, after which it can be implemented, tested and debugged. Implementing conceptualized improvements to the tool can also be part of this stage. The

conceptualization is addressed in Section 4 and the implementation in Section 5. Finalizing the project is the evaluation of the final tool that Section 6 discusses.

1.3.2 Toolbox

The toolbox includes the techniques and frameworks for the educational tool development. Because the tool is supposed to be running in a web environment, a programming language that supports web development is required. The most widely used one is certainly JavaScript with the combination of HTML and CSS. But due to its stark differences to the educational background of this project, we decided against using it. Instead, we decided for the programming language R with the extension of the framework Shiny. R is a general purpose programming language that is most widely used in statistical and data science applications. It also has good support for simulation and is regarded as easy to learn for programming newcomers. The framework Shiny enables the development of web applications with R and supports the integration of JavaScript, HTML and CSS if wanted or needed. With R Shiny the actual calculations run on the backend, or the server-side in other words, and the browser on the frontend side only operates as an interface to the application. A downside of this is that a computer acting as the server is always needed. An upside, however, is that the implemented models and calculations can stay hidden to the user if required. Knowledge of R and R Shiny is considered as a prerequisite to this project and not part of the knowledge gathering stage or any other stage.

Additional to the use of R with R Shiny is the application of R Studio, which is an Integrated Development Environment (IDE) and programming editor, specifically designed for R. R Studio is free, the most widespread IDE for R and supports Shiny out of the box, making it ideal for this project. As an IDE it also supports version control with git, that will help make development later on more convenient and failsafe. R Studio is also developed and maintained by the same people as R Shiny. Further discussion of these implementation means is presented in Section 5.1.

2 Educational Tools

2.1 Introduction to (digital) serious games

There is no single definition of serious games. The term ‘serious game’ appears to date back to a definition in a 1970 book written by Clark C. Abt that “presents simulations and games to improve education, both in and outside of the classroom”, with game examples running on “mainframe computers” but also “pen-and-paper based”. (Djaouti et al., 2011) The general idea of serious games can even be traced back thousands of years to e.g. Plato, who “philosophized that reinforcing certain behaviors exhibited in play would reinforce those behaviors as an adult”. (Wilkinson, 2016)

According to Susi et al. (2007) serious games nowadays “usually refer to games used for training, advertising, simulation, or education that are designed to run on personal computers or video game consoles”. Other definitions state “any piece of software that merges a non-entertaining purpose

(serious) with a video game structure (game)" (Djaouti et al., 2011) and "Games that do not have entertainment, enjoyment or fun as their primary purpose" (Michael, 2006).

Because of this long history it is no wonder that the application areas of serious games are vast, including military, government, education, corporate, and healthcare settings, that are ever-growing and expanding into new areas and contexts. (Susi et al., 2007) With "serious games [having transitioned] from exploratory, or fringe, experiments, to an increasingly legitimized medium for education, healthcare, and social change". (Wilkinson, 2016)

To get a better understanding of serious games and their application the consideration of a serious game taxonomy is helpful. It can provide an overview of the entire field and its current state, but, ultimately, might help with decision making in conceptualizing a new serious game. An appropriate taxonomy for this case should be relatively recent to reflect an up-to-date state. In addition, it should either capture the entirety of the field of serious games or be concerned with serious games of the specific application area of interest and not those areas that are irrelevant to the project. A verified and validated taxonomy, with (example) applications by others, should naturally be preferred and valued higher.

In this case, there are many classifications for serious games, but not so many taxonomies, which are usually more comprehensive and exhaustive. Additionally, the taxonomies are often only concerned with particular application areas that are mostly less relevant here. Nevertheless, we found one particular taxonomy published by F. Laamarti et al. (2014) which we will discuss here.

The taxonomy is organized into five distinct classifications. 'Application area', 'Activity', 'Modality', 'Interaction style', and 'Environment'. Each one is provided with several options that a specific serious game can correspond with (cf. Table A). It is possible that multiple options of one classification category can correspond with one serious game.

Table A - Serious Game Taxonomy

Application Area	**Activity**	**Modality**	**Interaction Style**	**Environment**
Education	Physical exertion	Visual	Keyboard/mouse	Social presence
Well-being	Physiological	Auditory	Movement tracking	Mixed reality
Training	Mental	Haptic	Tangible interfaces	Virtual environment
Advertisement		Smell	Brain interface	2D/3D
Interpersonal communication		Others	Eye gaze	Location awareness
Health care			Joystick	Mobility
Others			Others	Online

The *Application Area* is similar to the market in which the serious game is sold or applied and also tries to capture the most common application fields by referring less important ones to an option called 'others'. The *Activity* criterion refers to the "function performed by the player as a response and/or input to the game". Here, the taxonomy only lists three options: physical exertion, physiological, and mental. Wherein physical exertion differentiates itself from physiological by the purpose of exerting the body in comparison to bodily recovery or recreation. *Modality* describes the way information is transmitted from the computer to the player. In this sense it provides the player with feedback and is all about the output. The *Interaction Style* can be seen as the opposite to the modality in the way that it describes how the player transmits information to the computer. It is therefore all about the input. Finally, the *Environment* class describes the environment in which the game is played. Besides the physical environment, it is also about the virtual environment.

Considering the scope, context and background discussed in the introduction section, the educational tool, i.e. serious game, this paper is concerned with can be easily classified according to the taxonomy. Here, the *application area* is mostly about education, specifically in the operations management domain. But in some sense 'health care' area applies as well, since it is the context in which the particular education is intended for. The *activity* can be classified as *mental* since the body does not play a role in playing the game. *Modality* is the decision to display the tool online, and consequently on a screen, as *visual*. Similarly, *interaction style* is defined by the decision to run the tool on conventional computers as keyboard/mouse. And lastly, *environment* is twofold, with a 2D interface in an online environment.

2.2 Educational Tools in (Healthcare) Logistics

Finding and probing examples of already existing educational tools for the problem at hand shows what has already been done and might help in finding inspiration and ideas. But it can also highlight the shortcoming and limitations of the already available tools.

This section presents two examples of educational tools and explains the general idea and purpose behind them. It further discusses their overall user interface, shows example scenarios in regards to their (presumed) learning goal and classifies them in terms of the previously introduced taxonomy. Both examples presented are related to the concepts this paper is concerned with and their scope and level of detail is also comparatively similar. Because of that they might be better described as dashboards with a learning goal and aspects of serious gaming, instead of fully-fledged serious games. Nevertheless, they can easily be 'played', or at least probed, with or without the assistance of an operator or instructor.

2.2.1 Example 1

The first example tool was developed in the mid 2000s by Richard Steyn.[1] It shows how variations in weekly demand and capacity of an outpatient clinic affect the total numbers of patients waiting and

[1] The tool can be found at http://www.steyn.org.uk/

the lost capacity, called waste, that results from having more capacity than demand in a week. It is relevant here, because it considers demand and capacity in a hospital context with patients queueing, which is closely related to the learning objective of this paper. It is also a well-known tool in this area of research with many teachers using it to this day.

The tool was created with Excel and its designated programming language VBA, but unfortunately the details and inner workings are protected and therefore not accessible. However, it is likely realized with a simple Monte Carlo simulation and outputs that are the direct consequential results from generating random numbers. When opened in Excel, the model is initially empty and therefore the outputs blank. Each time the settings are changed the model has to be run again by pressing 'F9'. In terms of the classification it is the same as the educational tool this paper creates.

The entire interface is deliberately kept simple and only provides few options for input changes (cf. Figure 1). The designated area for inputs is at the top and the main outputs are comprised of a graph below. Additional statistics are shown on the right hand side. Some text fields are annotated with explanations, which can be viewed by hovering over the small red triangles in the upper right corner of the respective fields. Every time the model is run, the output graph suggests a duration of 204 weeks or four years.

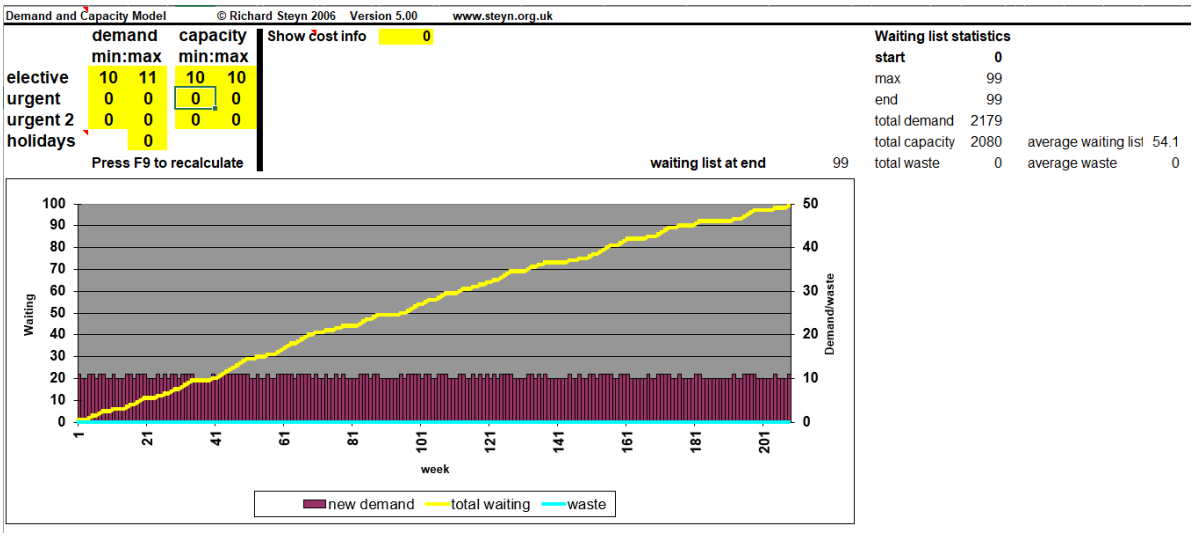


Figure 2 - User Interface Steyn Excel Tool

The input options are divided into two general columns with one for demand and the other for capacity settings. Each of these columns is divided again into one for minimum and the other for a maximum value. According to the annotations the actual input numbers are randomly generated from

a uniform distribution, for which the minimums and maximums are used. There are three different types of demand and capacity: 'elective', 'urgent', and 'urgent 2'. A field for number of holiday weeks, that, if set to a number greater than zero, results in randomly distributed weeks over each year that have no capacity. There is an additional field labeled 'Show cost info', that activates the output of some additional cost calculations to the spreadsheet.

The output-graph plots three variables stacked on top of each other in form of a weekly time series. There is new demand of patients as bars for each week. The number of total waiting patients and waste through unused capacity is represented by a line. Next to the main output as a graph, there are some statistics displayed on the right side about the waiting list and show e.g. the maximum number of patients waiting over the entire run length. Figure 2 depicts a constant growth in the number of patients waiting, because the average demand is higher than the average capacity. Additionally, the demand is always at least as high as the capacity and the displayed results are therefore to be expected.

Figure 3 shows a more interesting scenario that might highlight a learning objective of the tool. In comparison to the first scenario, a low demand for urgent patients is additionally introduced and the capacity for both elective as well as urgent patients is on average higher than the demand. Next to that, four holiday weeks per year are added. This shows that because of the holiday weeks the number of waiting patients is still eventually growing out of hand, as waiting patients accumulated during holiday weeks can not be caught up upon. Thus, only introducing some factors with stronger variability to the capacity can already throw the system out of balance. If variability in capacity and demand were even larger, the effect would be even more pronounced.

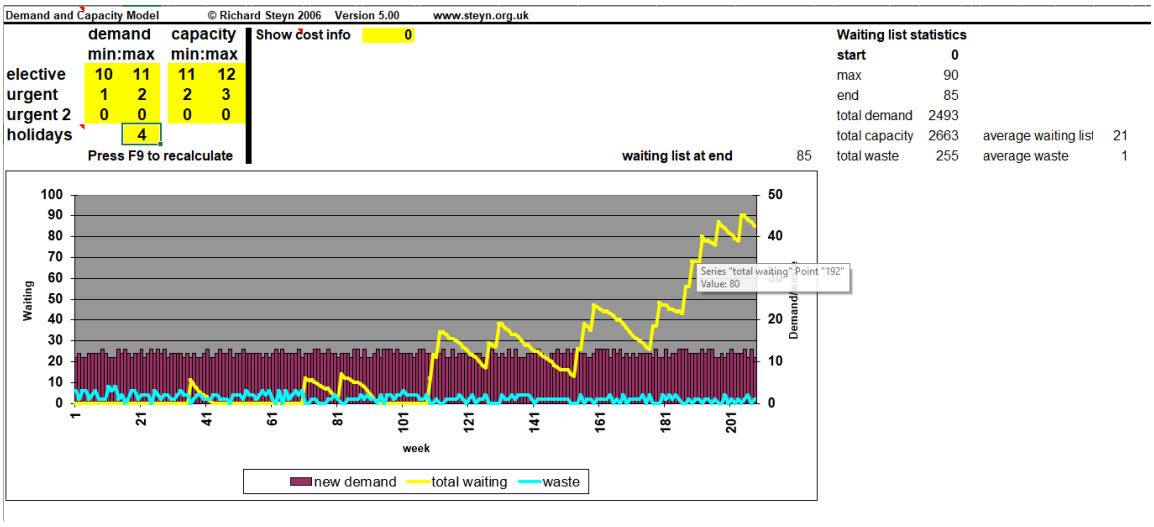


Figure 3 - Steyn Tool showing Scenario with Learning Goal

2.2.2 Example 2

The second example is a tool developed by Gregory Dobson, who still works on and improves it.[2] The tool shows animations of two simple queueing systems besides each other that share the same settings, except for one having separate queues (unpooled) and the other a joint queue (pooled). The learning objective is on showing the advantages of economies of scale, i.e. pooled/centralized systems, by highlighting the systematic benefit of an idle server in the joint case, who can serve all types of arrivals. This example is very relevant to this paper as it is concerned with the same underlying principles for the learning objective. It, however, focuses on showing only the case in which pooling is advantageous. More to that in Section 3.

The tool is largely realized with JavaScript and the manipulation of HTML canvases. From the public repository of the code base it looks like no external or additional libraries were used. However, we did not study the underlying code in detail. The code repository is entirely accessible and published under an open license on GitHub.[3] When the tool is first opened in a browser the program execution has to be started by pressing the 'Play' button. The tool then continuously generates new arrival events that can be influenced in real-time by changing the settings. The idea and function of this specific example is one of the main inspirations to the educational tool this paper develops. In terms of the taxonomy, it can be classified the same way again with an application area of education, a mental activity, visual modality and keyboard/mouse interaction in a 2D online environment.

The user interface is separated into two areas, the output on the left of the screen taking up about two-thirds of the window, and the inputs on the right. (cf. Figure 4) The output area shows a large animation on top that immediately catches the user's attention when started. Below are graphs that seem to update every time a person leaves the animation. The animation itself is made up of two animations, one for separate queues and one for joint queue. Unfortunately, there are no additional annotations or information besides the labels of the settings and output that can be seen in the screenshot. For this reason, it is likely dependent on a game operator that provides context in most situations.

The inputs on the right provide the following main options: 'System utilization', 'Arrival CV', 'Service Rate', 'Service CV', and 'Number of Servers'. Important to note here is that each setting always affects both the separate as well as the joint queue. Then there is also a 'Play' and 'Reset' button, as well as a slider for animation speed. Also important, the bell-shaped button, that activates a functionality for automatic pausing the model in case of an idle server with an empty queue in the unpooled system. This is exactly the special situation in which the joint queue has a systematic advantage. Below these

[2] The tool can be found at https://tiox.org/stable/index.html#eos

[3] And repository: https://github.com/tioxaxis/simulations

main settings, there is another section, labeled ‘Scenarios’, in which settings can be saved and that provides five default scenarios of interest.

The animation itself is rather self-explanatory. People come into the system from the left an go to the server if idle, otherwise they start to queue. On the left there are the separate queues and people only ever go to one specific server. On the right they queue jointly and as soon as a server becomes idle the first person in the queue comes forward. The graphs below plot ‘Average Flow Time’ and ‘Average Throughput’ for both systems in comparison.

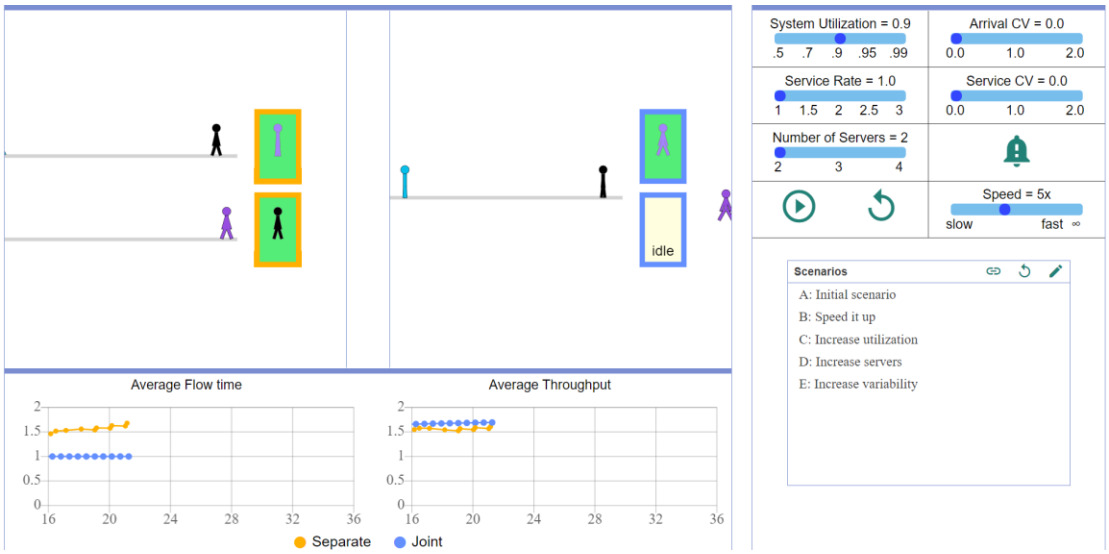


Figure 4 - User Interface Tiox EOS Tool

The case showing the learning objective can easily demonstrated by activating the bell. Figure 5 depicts a snapshot of the system in the special case in which the joint queue exhibits the systematic advantage. Paused right in the moment, the second server in the separate system is idle and no one is waiting in the queue. If this were a joint system the server could have served people from the other queue.

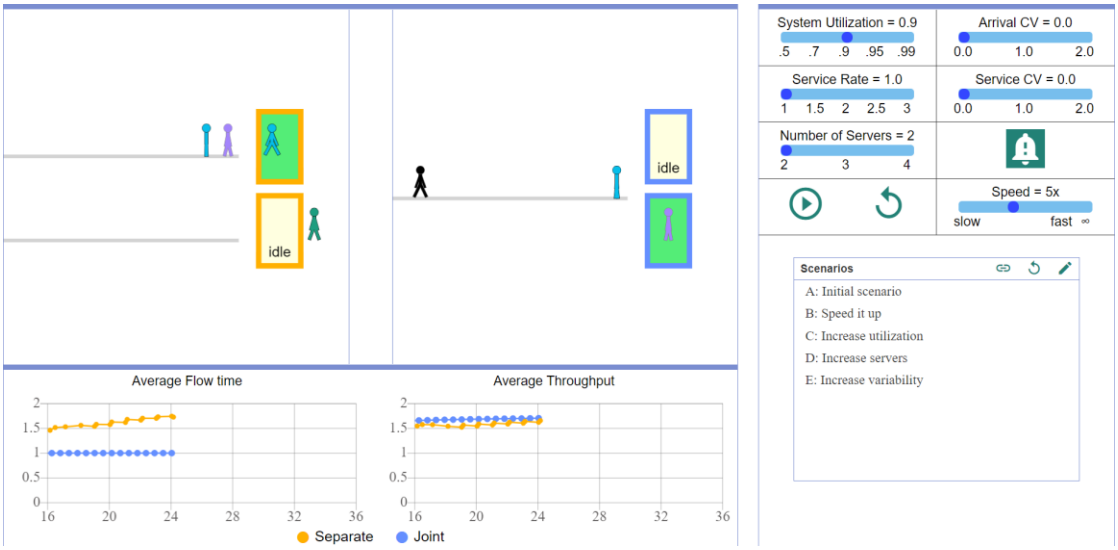


Figure 5 - Tiox Tool showing Scenario with Learning Goal

2.3 "A methodological framework for simulation-based serious gaming"

In the conceptualization of a new educational tool, i.e. a (light) serious game, there are several advantages to using a methodology. For once, it helps to achieve better outcomes by streamlining processes and thoughts. It also makes use of the ideas and research of others, by which costs and time needed can be reduced, too. Most importantly, however, it makes sure the outcomes are reproducible and therefore of higher scientific value. All this is especially true, if the methodology applied has been proven to work before by others.

For the specific situation at hand, the selected methodology should be appropriate and work with the taxonomic classification of the tool. Additionally, it should not be too old as this makes it more likely that the methodology is outdated concerning the current state of technology. And changes to technologies happen fast nowadays. Like just mentioned, it optimally has also already been proven multiple times in practice and can therefore be most reliably found by a systematic literature review.

A suitable methodology we ultimately chose for this project was introduced in a paper titled "Conceptual modeling for simulation-based serious gaming" by van der Zee et al. (2012) It is based on the conceptual modeling framework for simulation by Robinson (2008), who is also a co-author of the new methodology. Next to the Robinson framework, it is also based on the game design process by Greenblat (1988). The fact that the methodology deals with the specific case of games running on a discrete event simulation with the objective of enhancing decision making skill in operations management underscores its suitability, since this is exactly the case this paper deals with. However, as the name suggests, it is only really concerned with the conceptualization and design of a serious game and not the construction, implementation, nor even prototyping.

The chosen serious game methodology is generally built up according to an iterative procedure of five distinct modeling activities:

1. Understanding the learning environment
2. Determine objectives
3. Identify the model outputs
4. Identify the model inputs
5. Determine model content: scope and level of detail

Sub 1. Understanding the learning environment is really about grasping the context and clearly defining the motivation before going into the actual modeling. It entails defining what concept of operations management is at the core of the issue and capturing the requirements and ideas of the commissioning party. The expected knowledge and needs of the future players as well as potential operators need to be considered here, too. Consequently, this first activity really lies the foundation for all following activities and the more thoroughly it is developed the better the final concept can presumably be.

Sub 2. Determine objectives is, as the original authors put it, foremost about identifying "the game's pedagogic purpose(s), i.e., what players should have learned after completing the game". It is divided into two sets of objectives: modeling objectives and general project objectives. Here, modeling objectives refer to the objectives of the underlying simulation model and "can be expressed in terms of players' achievements in mastering their decision making skills". Next to that, the general project objectives, consider the resource usage and nature of the model. Resource usage is strongly linked to the overall scope of the project and the way the game is ultimately intended to be used. The nature of the model defines aspects like how should the model be visualized (e.g. 2D or 3D), how should the operators and players interact with the game (e.g. keyboard/mouse), how quickly should the model respond (e.g. immediately or several seconds later), and also should the model be capable of reusage in other contexts, like in this case for people not involved with healthcare. Note, that the relation to the previously discussed taxonomy can clearly be seen here.

Sub 3. Identify the model outputs distinguishes between two purposes for the model outputs. One is to show players' achievements in terms of the already defined pedagogic purposes. The other is to inform and possibly interpret these achievements. Both can usually be inferred from the modeling objectives but also have to comply with the general project objectives. This activity also includes how these outputs should be represented in general terms. For instance, they could be in a numerical form, graphical, or animated somehow.

Sub 4. Identify the model inputs activity is easily misinterpreted as the inputs and decisions the player has to make. Instead, it is about the inputs an operator and game developers can make to the system, changing it to accommodate alternative configurations for different use cases and scenarios. The

authors mention here, that since in most cases it is relatively easy to come up with numerous different model inputs, it is important to consider each input for its contribution and feasibility. Especially also in regards to the scope.

Sub 5. Determine model content: scope and level of detail. This activity starts with deciding on the model scope, meaning the components that should be included in the game's model. The methodology differentiates between three types of components: agents, flow items, and jobs. The agents are the things in the model that 'make' decisions (direct or indirect) and "represent the infrastructural, non-movable, intelligent elements of an operations system". In this sense, a player of the game can also be an agent. The flow items represent the goods and information that basically flow between the agents and are manipulated according to job definitions, which describe the job component. After deciding on the scope with the components, the model detail is decided upon by setting the components' attributes. For instance, how long a certain job takes, how its duration is distributed or the extent to which it can be varied. Table B presents an overview of all the decisions that need to be undertaken during each activity. For a more detailed discussion and an additional example to how the activities can be applied see van der Zee et al. (2012)

Table B - Activities of Conceptual Modeling Framework (taken from van der Zee et al., 2020)

Activity	Details
1. Understanding the learning environment	• Understand the subject matter, context of use, and likely players/operators, preferably by interviewing clients and subject matter experts • Explore learning needs, given the environment, i.e., student education or professional training • Decide on the appropriateness of a computer-based game format
2. Determine objectives – Modeling objectives	• Identify the game's pedagogic purposes • Express modeling objectives in terms of players' achievements in mastering their decision making skills
– General project objectives	• Establish and assess project requirements on resource use • Clarify the nature of the model and its use with respect to: o Visualization o Player interaction o Responsiveness o Model/component re-use
3. Identify the model outputs	• Check modeling objectives for relevant performance measures, indicating player achievements • Establish model outputs helping to identify potential bottlenecks in systems operations and explain player achievements • Determine format for representing responses
4. Identify the model inputs	• Select quantitative and qualitative data that can be changed, in order to represent alternative system configurations appealing to (alternative) groups of players • Determine range over which model inputs may be varied
5. Determine model content: scope and level of detail	• Determine model scope: o Identify the system boundary o Identify all components in the real system that lie within the model boundary (include player roles) o Assess whether to include components • Determine model detail (attributes) for all components included • Indentify assumptions and simplifications concerning model scope and detail, and assess their impact on model responses

2.3.1 Outline of Conceptual Modeling

The introduced methodology is performed as follows. Parts of the first activity about understanding the learning environment are discussed in the introduction with the motivation and context, but also for whom the tool is intended. The appropriateness of a computer-based serious game is also considered there. The details of the subject matter and learning objective, however, are discussed in Section 3 - Subject Matter of 'Queueing and Pooling in Healthcare'. Activities two to five are treated one after the other in Section 4 - Conceptual Modeling of the Tool. The reason for this division being that the need to research the concept behind the learning objective of the tool is independent from the methodology that is applied and of high importance.

3 Subject Matter of 'Queueing and Pooling in Healthcare'

3.1 Queueing in Hospitals and Healthcare in General

Systems and scenarios in healthcare, or hospitals more specifically, can be viewed and assessed in terms of queueing theory. To give some examples: An entire hospital can be regarded as a queueing system, where the hospital is the resource, or server, and the patients the demand, or arrivals. Zooming in, it can also be viewed as a network of queues, in which each department or ward represents a resource with different types of patients that need to get varying 'jobs' done. But departments, wards, or even single doctors, can also be viewed as a queueing system in isolation and independently. And each of these can, again, be seen as a network of queues. The level of abstraction can hereby be freely chosen, but should be in accordance with the objectives and necessary level of detail.

But what exactly means pooling in this context and especially in terms of queueing theory? Pooling means the combination of two or more separate system components into a joint component. Or, as a similar previously used definition states it, pooling is "the replacement of several ingredients by a functionally equivalent single ingredient" (Mandelbaum & Reiman, 1998). However, pooling is not as straightforward as one might initially think. Pooling can happen along different dimensions and on heavily varying scales. Just like queues in general, it can, for instance, be concerned with two entire hospitals or possibly just two doctors, and everything in between.

According to Mandelbaum and Reiman (1998) it can in general take place in three forms. First, pooling queues (the demand), consider for example a general practitioner that covers for a colleague who is on vacation, both doctors' demands are then pooled and need to be taken care of by one doctor. Second, pooling tasks (the process). A doctor might start handing out new appointments herself after treating the patient, for which the patient had to go to the receptionist before. And finally, pooling servers (the resources), here, a general practitioner might for instance be joined by another doctor that starts treating (serving) the same patients (demand). Often, pooling takes place in multiple forms at once. This project is no different and focuses on the pooling of queues and servers simultaneously.

Pooling is often considered from the perspective of benefits and efficiency improvements. In this case, it is also one reason for why economies of scale can be more efficient. (Vanberkel et al., 2012) As Cattani and Schmidt (2005) exemplarily state: "Pooling of customer demands, along with pooling of the resources used to fill those demands, may yield operational improvements". They clearly assess and consider it from the beneficial perspective. Note the use of the word *may*. It highlights that the benefits are not the entire story and under certain circumstances at least, pooling might also lead to efficiency losses (Mandelbaum & Reiman, 1998; van Dijk & van der Sluis, 2008; Vanberkel et al., 2012).

The advantages and disadvantages of pooling are influenced by many factors and circumstances. Psychological, economical and more factors all play a role. But it also depends on the specific performance measure that is in consideration and some performance measures act opposite of each

other. One that is often chosen as the performance measure for operational improvements is the average waiting time in the queue.

3.2 Effects of Pooling

There are some limitations to pooling that make it disadvantageous, unfeasible or even impossible from the start. For instance, if servers and queues are considered for pooling, "all servers must be able to accommodate all demand" (Vanberkel et al., 2012). This can easily be understood by considering a doctor specialized in cancer treatment and another one in cardiovascular diseases. Pooling these and having each doctor treat any of the patients can not possibly be in the interest of either the patient nor the doctor. Some other factors to consider are, the influences of jockeying if that is a possibility in the scenario in question. Or, whether the systems under consideration are actually independent of each other. If not, the literature and theory out there becomes immediately less relevant and the real behavior of patients might not be as expected. And even, the effect of pooling on personal relationships and the therefrom resulting satisfaction of e.g. patients and doctors. (Rothkopf & Rech, 1987)

There are some scenarios in which pooling is basically always advantageous, at least from the perspective of the plain mathematical model in queueing theory. van Dijk and van der Sluis (2008) state, in the context of call centers, that "in the case of identical calls, or calls for which the mean and variance are more or less equal, pooling is indeed beneficial". Other research in terms of pooling and its benefits found that "pooling always helps in light traffic, because a customer at the pooled system typically enjoys a service rate that is the total capacity of the specialized system. In heavy traffic, pooling effects can go either way". And "with low enough variability pooling is always advantageous". (Mandelbaum & Reiman, 1998) The main intuition to this, that is often given to highlight why pooling is beneficial is as following. Consider two queues with a server each, wherein one server is busy and customers are queueing and the other is idle and nobody is waiting. Then, in a pooled system, the idle server could serve customers from the other queue as well, and, in essence, 'a problem shared is a problem halved'. This is exactly the intuition the second example tool from Section 2.2.2 tries to establish.

In contrast, with different customer types the benefit of pooling is not clear at all, and the opposite might actually be true. van Dijk and van Sluis (2008) analyzed exactly this case by the application of the Pollaczek-Khintchine's (PK)-formula to approximate the performance of pooling two differing M/G/1 queues.

In the example they present to illustrate their point, and that serves as the basis here, they in fact use a deterministic service rate and therefore actually consider two differing M/D/1 queues. They then start by imagining two separate queueing systems with one type of customer and server each. For the arrival (λ) and service (μ) rates of both system they choose to set them at the same traffic load, i.e. utilization, of about 83% but with different magnitudes for the rates. For this difference in magnitude they

determine a factor, called k, which in the example is set to 10. The mean waiting time for each individual system can be easily exactly calculated with the PK-formula:

$$W_G = \left(\frac{1}{2}\right)(1 + c^2)W_E \; with \; c^2 = \sigma^2\mu^2, and \; E \; representing \; standard \; exponential \; case$$

They then use the results to calculate a weighted average mean waiting time for all customers in the two systems by using the ratio of the arrival rate of one type to the overall arrival rate for the weight:

$$p_i = \frac{\lambda_i}{\lambda_1 + \lambda_2} \; for \; i = 1,2$$

In the next step, they imagine the same queueing system in pooled form, in which both types of arrivals join the same queue and each server can serve each customer type. The new overall arrival rate for this system is easily calculated by just adding up the two separate arrival rates. The new (mean) service rate, however, needs to be weighted by the arrival rates again:

$$\bar{\mu} = p_1\mu_1 + p_2\mu_2$$

Finally, they use these two rates in combination with the PK-formula to approximate the mean waiting time in the pooled system, which they consequently show to be reasonably accurate. The only difference being that they substitute the coefficient of variation from the original PK-formula with a mix coefficient that they calculate as following:

$$c^2_{mix} = p_1\left(\frac{\bar{\mu}}{\mu_1}\right)^2 + p_2\left(\frac{\bar{\mu}}{\mu_2}\right)^2 - 1$$

With the end result showing that the average mean waiting time of the pooled system is actually worse than the one from the unpooled system.

4 Conceptual Model of the Tool

Table C gives an overview of the results for each activity from the conceptual modeling of the tool. A more detailed description of each modeling activity follows.

Table C - Overview Results of Conceptual Modeling

Activity	Details
1. Understanding the learning environment	Clients: Consultants to healthcare management personnel (e.g. Rhythm), Teachers of healthcare management, Teachers of operations management topics in general
	Subject matter experts: Specialists in operations management or (more specifically) queueing theory
	Subject matter: Queueing theory with focus on pooling effects
	Players: Healthcare management personnel, students in healthcare management, management students in general
	Operators: Mostly congruent with *clients*, in some situations the players theirselves
	Context of use: Lectures, workshops, consultations, seminars, self-study
	Appropriateness of a computer-based game format: Yes
2. Determine objectives	Pedagogic purposes: 1. Players should understand performance differences in unpooled and pooled queueing systems exist 2. Players should understand that a pooled system is often advantageous 3. Players should understand that a under certain conditions a pooled system might not be advantageous
	Modeling objectives: Needs to ensure that players can evaluate a specific pooled system state in terms of its performance compared to the equivalent system in unpooled form by a patient's expected waiting time in the queue
	Project requirements: player availability, scope and time available for realization
	Model nature: - Visualization: animation showing a representative pooled and unpooled queueing system, graphs reflecting the development of waiting conditions in the representative animation - Player interaction: Mainly through game operator, in some situations by using the operators menus Responsiveness: Online and immediate Model/component re-use: Yes
3. Identify the model outputs	Performance measure (player achievements): (Approximate) expected waiting time in queue
	Explanatory measures: Arrival and service rate settings of two types of patients, animated representative system and graphs based on it
	Format: 2D animation, graphs, numerical performance
4. Identify the model inputs	Inputs: Arrival and service rates for two types of patients
5. Determine model content: scope and level of detail	Scope: see Table D
	Detail: see Figure 6

4.1 Modeling Objectives

From the motivation and understanding of the learning environment a pedagogic purpose can be formulated. First, players should understand that there are performance differences in an unpooled and pooled system. Second, players should understand that a pooled system is often advantageous. And third, players should understand that a pooled system might not be advantageous under certain

circumstances. On this basis, the modeling objectives in terms of players' achievements in mastering their decision making skills can be determined. With the result that the simulation model needs to ensure that players can evaluate a specific pooled system's state in terms of its performance compared to the equivalent system in unpooled form. For this, the key performance indicator being a patient's expected waiting time in the queue.

The project requirements on resource use are mainly limited by the scope and time of the project as well as the non-existing budget. With the available time and knowledge restricting the complexity of the tool and the budget constraining most parts to be build from scratch involving the use of free development means. On the other hand, as presumed players are to a large extent healthcare managers and professionals, who have limited time available to learn from the tool, the tools complexity and extent is limited anyway. For these reasons, and the fact that it is only intended for education, a simple model seems reasonable as long as it captures the learning objective(s).

As a consequence, the nature of the model is determined with a **visualization** that shows the system in unpooled and pooled configuration to allow for direct comparisons by the player. This includes a 2D animation of the queueing system's current status that reflects the overall system conditions in a representative manner. In addition to this are graphs that show the historic development of the waiting performances of the representative animation. The **player interaction** with the tool is, in light of the project scope, mainly through a game operator who sets it into different scenarios, challenging the player to evaluate the system. Nevertheless, players can of course set the tool to different scenarios themselves, for instance after a workshop to repeat and strengthen their understanding of the learning goal. Potentially, as a continuation to the project, most or all of the operator's involvement could be automated.

As for **responsiveness**, the tool should be accessible online to facilitate constant access by both operators and players, which is also one of the project objectives. Immediate reaction of visualizations and output when settings are changed for different scenarios is desirable to better understand the cause-and-effect relations. This implies real-time behavior like the second example in Section 2.2.2 exhibits. The simple nature of the model at hand and its broad application field also allows to mind for **model re-use** in other fields besides healthcare. Adaptations for different use-cases should therefore be made as easy as possible. This can include the programmatic allowance for extensions.

4.2 Model Outputs

Determining the model outputs starts with deriving the relevant performance measure that indicates the player's achievements from the model objectives. Here, the player's achievements and feedback on player's accurateness in evaluating the system is provided by the key performance indicator of patient's expected waiting time in the queue. As the basis for this serves van Dijk and van der Sluis (2008) analyzation of whether to pool or not that is in more detail in Section 3.2. These are supported by the explanatory outputs. Included are the arrival and service settings of two types of patients itself and,

additionally, the visualization of a representative system status as well as the historical measurements of the number of patients in the queue and their average waiting times for the representative system like previously mentioned. The format for the explanatory outputs is a simple 2D animated overview and time-series graph(s). The idea is that patients arrive and queue if a doctor is not available and then join a doctor for an appointment after the doctor becomes idle with the details dependent on the pooled or unpooled system layout . When the appointment finishes the patient leave and disappear. Since all of this changes in real-time, the graphs need to update and re-render accordingly.

4.3 Model Inputs

Note again, that model inputs are linked to operator inputs and not player inputs. The game operator is responsible for setting the tool into different scenarios that represent situations which build up intuition for the learning goals. The main quantitative inputs for this are the arrival and service rates for two types of patients. The possible decisions and attributes for these inputs are again based on van Dijk and van der Sluis (2008) analysis on whether to pool or not to pool in call centers. (cf. Section 3.2) Additional inputs can allow the operator for changes to the representative system visualization like e.g. the number of finished patients, over which the historic measurements in the explanatory graphs are calculated. Next to that, predefined scenarios can be used to set all inputs at once for a specific use case.

4.4 Model Scope and Detail

The system boundary is clearly dependent on what the queueing system under consideration is interpreted. Like previously mentioned, it could be for example a doctor's office, a department within a hospital, or similar. Here, it is interpreted as a doctor's office because most people can easily relate to that. Table D shows the model scope of the queueing education tool in terms of components (cf. Section 2.3 again for explanations on the components). We made a decision for each component to either in- or exclude it in the final model with a justification given next to it.

Table D - Scope of Conceptual Model

Component	In/exclude	Justification
Agents		
Hospital Management	Included	Player's as well as operator's role; sets the state of the system and, especially the player, evluates the system's performance
Doctor's office / hospital department	Included	System that is considered as an example for a queueing system, and that needs to be evaluated
Doctors	Included	The server of the queueing system
Secretary	Excluded	Simplification: Unnecessary detail for learning objective
Nurse	Excluded	Simplification: Unnecessary detail for learning objective
Flow items		
Goods		
Patients	Included	The demand or customers of the queueing system
Data		
Individual waiting time in queue	Included	Explanatory measure and feedback to player
Queue length	Included	Explanatory measure and feedback to player
Doctor idleness	Excluded	Simplification: only indirectly realted to key performance indicator of patient's waiting time
Job definitions		
System layout	Included	Objective that needs to be evluated in comparison to each other
Arrival rate	Included	Key influence on system performance
Service rate	Included	Key influence on system performance
Jobs		
Estimating system performance	Included	Player's main activity; directly related to learning objective
Treating/Diagnosing patients	Included	The service that needs to be undertaken and for which each patient is waiting

The queueing model that results from these decisions and that underlies the final tool can be seen in Figure 6. Several assumptions and simplifications have been made to this model considering the scope and to fulfill the requirements. First, deterministic service times and a very simplified flow layout, to comply with the underlying theory. Second, the arrivals are simplified to a random distribution with no use of scheduling or anything similar, and the number of agents, i.e. servers, are reduced to doctors. Each of these, however, is excluded or simplified on the basis that the tool is exclusively meant for teaching a very specific concept from operations management and the entire system is fictional. These assumptions and simplifications are therefore appropriate and can help in making the concepts easier to grasp.

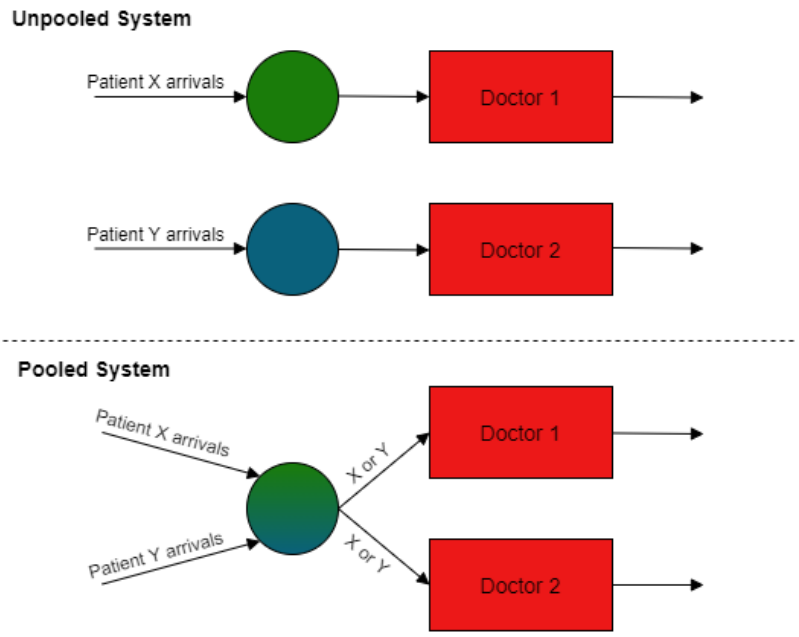


Figure 6 - Queueing Model underlying Tool

5 Implementation of the Tool

5.1 Selection of Implementation Means

The selection of the right implementation means and development environment is substantially important. It needs to be in line with the project objectives and it also makes better results more likely and easier to achieve. As one of the main objectives to the tool under development here is that it is readily online available suggests, the easiest and most straightforward way to explore the options for implementation means is to search the internet for examples. Because serious games are closely related to games with an entertainment purpose, these can be examined, too, as long as they do not need to be installed. Section 2.2 shows two of the examples examined for this project.

Many researchers in operations management are not, or just barely, familiar with web development, but instead have most of their expertise focused on the mathematical models and literature underlying the theory and its insights. This explains why many of the digital tools in this area rely on Excel, some in combination with its designated programming language VBA, as it is one of the most widespread programs used for mathematical applications in general. (cf. example in Section 2.2.1) Additionally, Excel provides instant availability to some extent as files can usually be quickly downloaded or sent and then only need to be opened. However, there are limitations to this approach, that do not only end with people not having Excel installed or no license for it. As soon as moving animations or similar are desired, it is less suited for the task. Next to that, applications with high computing demands show that Excel does not excel in this regard.

On the other side of the spectrum, standard web development with HTML, CSS, and JavaScript with its countless libraries and frameworks for browser side execution provides opportunities that are not greatly inferior to the classical install-and-execute approach. Big downside, however, is the extensive knowledge and time needed for realizing such projects. (cf. example in Section 2.2.2) This affects future reusability and improvability by others, especially those coming from the same domain, as, like previously mentioned, they tend to lack the necessary knowledge. In most cases, this can only be reasonably overcome by multidisciplinary project teams.

In recent years, a third option has more and more presented itself, that offers a new middle way, combining advantages from the Excel and standard web development approach. They make it easier to develop programs for a web environment without knowing much about web development in general and remove the boundaries that Excel is subjected to. This middle way mainly includes the frameworks Dash and Shiny, which are written for the programming languages Python and R, respectively. They can be easily deployed to the internet as well, giving access to every device with a browser, and mostly rely on the knowledge of the programming language. The use of HTML, CSS and JavaScript is only necessary for more advanced applications.

The decision we ultimately made was for R with Shiny. Many examples and applications are already out there and the available documentation is good. In comparison to Dash, it is also easier to get started and runs more stable. Another benefit is, that the programming language R is popular among mathematician and researchers. This of course also includes the people specializing in operations management that we identified as the main game operators. R is commonly used in combination with RStudio, which is an integrated development environment (IDE), and we make no exception here. Since R Shiny does not provide an option for animations out of the box, only visualizations in the form of graphs, we decided to couple it with the JavaScript library D3, that was created for data visualizations by manipulation of scalable vector graphics (SVG). These data visualizations can be programmed in a way that they act like animations. The graphs, intended to complement the animation, can then easily be created with D3 as well. Nevertheless, all of the inputs, as well as the overall tool interface and feedback on player achievements, is still created with Shiny, limiting the D3 use to the output.

5.2 Discrete-event Simulation Model

There are three different approaches to computational simulation: time-slicing, discrete-event simulation, and continuous simulation. Time-slicing is the simplest approach in which the time is advanced by a constant time-step and each time the state of the system is recalculated. Discrete-event simulation differentiates itself from time-slicing approach by using time-steps that are based on the event which is the closest in the future to change the system status instead of constant time-steps. Again, after each time-step the state of the system is recalculated. Continuous simulation is very different from the former two as it usually simulates continuous changes in a system by means of differential equations. (Robinson, 2014) Continuous simulation is therefore unsuitable for the tool at hand. But

time-slicing or discrete-event simulation could potentially both be chosen. For the reason, however, that choosing a good constant time-step can be difficult (Robinson, 2014), the discrete-event simulation approach is chosen.

For realizing a discrete-event simulation programmatically, there are again different approaches. Robinson (2014) lists four of those: three-phase method, event scheduling, activity scanning, and process-based. The latter three are, however, all less appropriate than the three-phase method for various reasons. Even though event scheduling is very efficient, it is quite complex to program and consequently also harder to learn (or recall for that matter) how the final tool functions. Event scheduling, on the other hand, is simple but not very efficient. And the process-based approach seems overly complicated for the few-events problem at hand as it works with setting up various processes for different sequences of activities.

The three-phased simulation approach makes use of two types of events. These are the B and C events, where B is short for *bound* or *booked* and C for *conditional*. B events are those that are scheduled to take place at a specific time and denote the discrete events at which the simulation needs to recalculate its state. This also includes the events that occur at a random point in time by generating and scheduling the next random event before its actual execution. Usually B events are concerned with arrivals or completion of activities. C events on the other hand are conditional on the state changes triggered by the B events and their execution is only determined after a B event. These are usually concerned with the start of an activity. Figure 7 depicts the general flow of the three-phased approach with each type of event corresponding to a phase. Note the box with the caption 'A Phase'. The A here is, however, not referring to an additional type of event but the advancement of the simulation clock to the next B event in time. Overall, the flow diagram shows that the three-phased approach is made up of two general loops, where one loop is part of the body of the other loop. The outer loop ensures that the simulation continues and the simulation clock is advanced until some certain predefined condition applies. For instance, a maximum number of arrivals or days. And the inner loop is responsible for executing all possible C events after a B event. (Robinson, 2014)

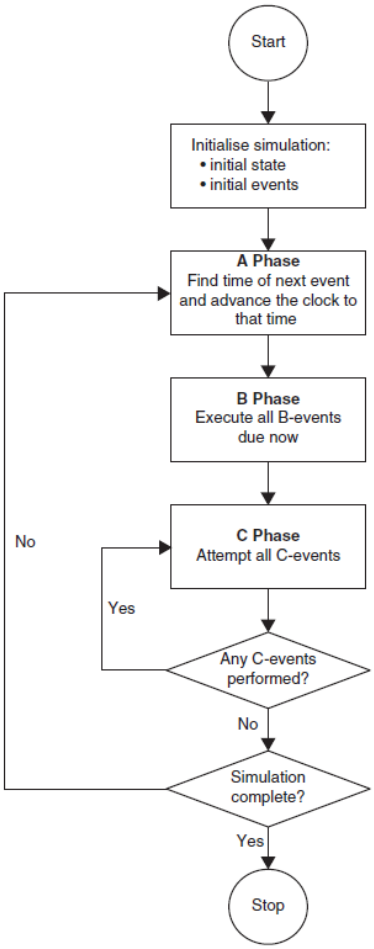


Figure 7 - Flow Diagram Three-phased Approach

The queueing model in Figure 6 helps in identifying all of the events necessary for the simulation. See Table E for the B events. As the pooled and unpooled state share the same arrival event, only two types of arrivals are needed that can have different rates. Important is , that each time an arrival event occurs, the next arrival event of the same type needs to be scheduled.

Table E - B-Events

Event	Type	Change in state	Future events to schedule
B1	Arrival	Patient X (green) arrives and enters waiting queue	B1
B2	Arrival	Patient Y (blue) arrives and enters waiting queue	B2
B3	Finish appointment with doctor 1 - unpooled state	Doctor 1 completes check-up or treatment and outputs patient to world	
B4	Finish appointment with doctor 2 - unpooled state	Doctor 2 completes check-up or treatment and outputs patient to world	
B5	Finish appointment with doctor 1 - pooled state	Doctor 1 completes check-up or treatment and outputs patient to world	
B6	Finish appointment with doctor 2 - pooled state	Doctor 2 completes check-up or treatment and outputs patient to world	

For the C events see Table F. The four conditional events listed here are responsible for scheduling the next 'finish appointment' activity of the B events.

Table F - C-Events

Event	Type	Condition	Change in state	Future events to schedule
C1	Start appointment with doctor 1 - unpooled state	Patient waiting in queue 1 and doctor 1 idle	Doctor 1 calls patient in and starts check-up or treatment	B3
C2	Start appointment with doctor 2 - unpooled state	Patient waiting in queue 2 and doctor 2 idle	Doctor 2 calls patient in and start check-up or treatment	B4
C3	Start appointment with doctor 1 - pooled state	Patient waiting in general queue and doctor 1 idle	Doctor 1 calls patient in and starts check-up or treatment	B5
C4	Start appointment with doctor 2 - pooled state	Patient waiting in general queue and doctor 2 idle	Doctor 2 calls patient in and start check-up or treatment	B6

5.3 User Interface with Visualization

5.3.1 R Shiny

The general interface with the controls and inputs is created with Shiny itself. It makes use of two Shiny methods, that produce a navigation bar at the top of the page, as well as a sidebar for the input and control options. The navigation bar lets the user open up an additional *About* page, giving some context and background information. The sidebar has the controls at the top with a 'Pause/Play' button and an option for choosing multiple animation speeds. Below the controls are the inputs to the

simulation model that the game operator can change to let the tool represent scenarios for the various learning goals. These are the arrival and service rates for both types of patients. Additionally, the sidebar includes a scenario tab that helps in setting all inputs at once.

5.3.2 JavaScript, D3 and SVG

The underlying idea to the explanatory animation of the tool is very similar to the approach of the second example in Section 2.2.2. Patients come into the frame from the left and join their specific queue or a joint queue depending on whether the system is pooled or not. After that they join the doctor and leave the system to the right side after finishing the appointment.

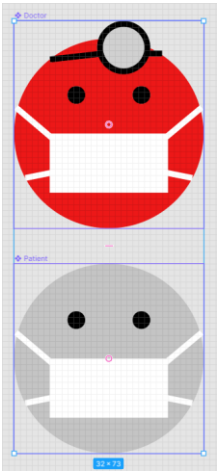


Figure 8 - Doctor and Patient Design for Animation

As JavaScript and D3 produce scalable vector graphics (SVG) for visualizations, it is good practice to come up with a layout that reacts dynamically to its environment parameters like frame height and width. Figure 9 represents the basic idea for this. Additionally, Figure 8 shows the design idea for visualizing a doctor and a patient in the animation.

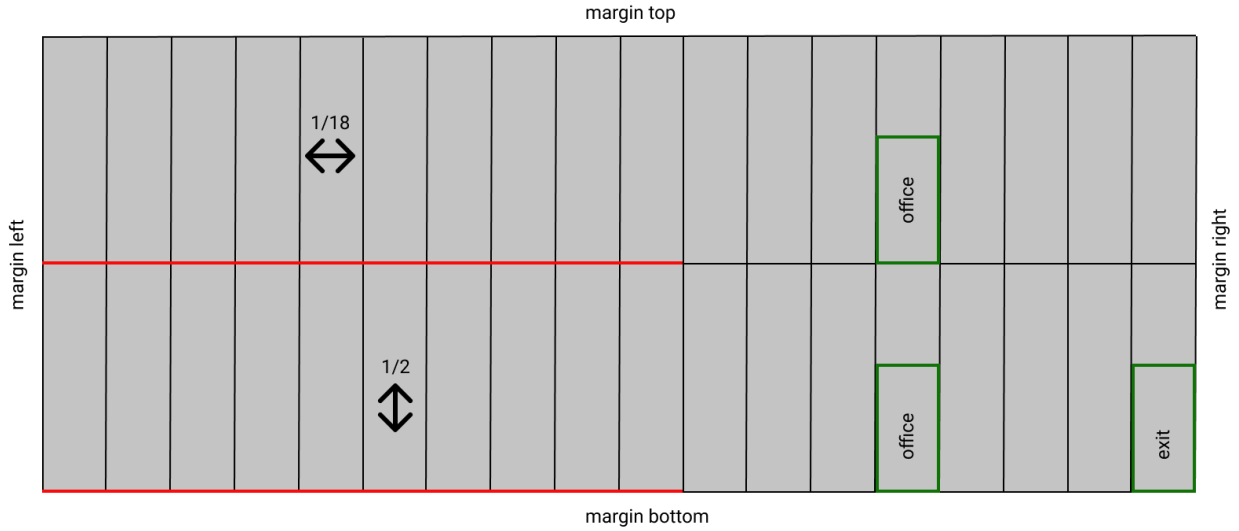


Figure 9 - SVG Layout Animation

5.4 Verification of Implemented Tool

Figure 10 presents the user interface of the completed implemented tool.[4] The navigation bar at the very top of the tool defaults to the game application but also offers the option to open an about page giving some context. In the sidebar on the left all of the defined inputs for the game operator from the conceptual model can be seen. They are clearly divided into sections for two types of patients, a green and blue patient. Some additional settings can be made visible by clicking on the hyperlink at the bottom. Further, some controls are given at the very top of the sidebar and below the user can switch to a second tab that provides predefined scenario settings. The main output is divided into two equally sized columns that show an unpooled system and pooled system, respectively. Each column has a button at the top that can be clicked after the player finishes assessing the system and wants to know the actual theoretical system performance in terms of expected waiting time in the queue. The output from clicking the button is displayed directly below. Next is the animation of a representative system, that, as can be seen when the tool runs, shows the same arrivals to both system layouts. And at the bottom, the graphs with the average measurements of the representative system can be seen.

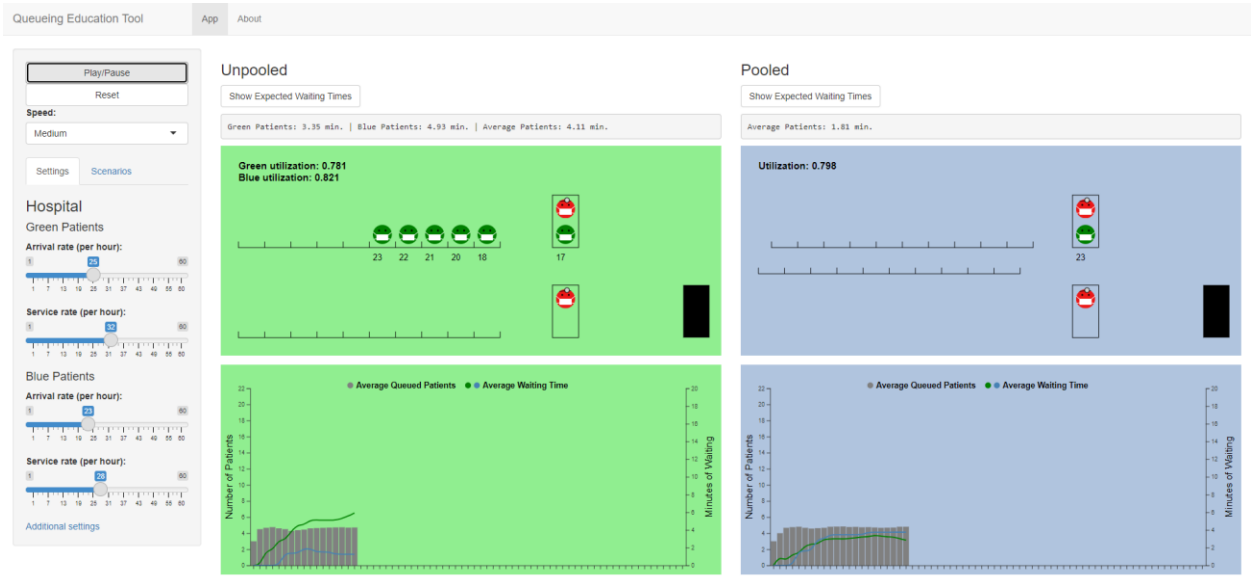


Figure 10 - Screenshot of Tool's User Interface

To verify that the tool works and runs well, it can be set into some extreme situations. First, setting both arrivals to the highest possible value and the service to the lowest, shows the queues filling up quickly until the maximum capacity of the system is achieved. With the downside that patients leave the system only after very long times at the doctor. Next, setting the system to a very high k factor,

[4] The tool can be found at https://philippun.shinyapps.io/queueing-ed-tool/ and the code repository at https://github.com/philippun/queueing-ed-tool

meaning high differences between the arrival and service rates of the two types of patient, shows a high throughput of one patient type and a rather low throughput for the other. Finally, as a last extreme situation, setting the arrivals to very low values and the service to very high values shows the system in mostly empty state.

5.5 Model Validation

Model validation can be done by consulting van Dijk and van der Sluis (2008) analysis about whether to pool or not to pool again. To shows that the tool is in line with their outcomes, two short example settings here shall suffice. Their results show that under a traffic load of 83% for both patient types, but a k-factor of 10, the ratio of mean waiting time in the pooled case to the mean waiting time in the unpooled case is about 135%. This holds true for the performance indicator of the tool. In another case, their results show that at a k-factor of 4 and a traffic load of 90% the ratio is about 75%. This holds true as well.

6 Preliminary Experimentation and Use Cases

6.1 Example Scenarios and Practical Application

This section tries to give three example scenarios for the tool that are interesting and could be used for the tool's practical application in lectures, workshops, etc. All of the examples presented here are oriented on the pedagogic purpose of the educational tool.

1. In this first scenario the precise settings of the arrival and service rate are not that important as it purpose is to understand that performance differences in unpooled and pooled queueing systems exist. For this, the operator can challenge the player to monitor a specific patient through both systems and assess whether the layout matters for the patient. After some time, the player should notice that the patients flow trough the system and the time it takes for the doctor appointment to start varies per layout. The different lengths and compositions of the queues should become apparent to the player.
2. In another scenario oriented on understanding that a pooled system is often advantageous, and sometimes even dramatically, the arrival and service rates of both patient types can be equally set. Important is, however, that the traffic loads for both patient types are smaller than 100%. The player can then with the support of the animation and graphs figure out that the pooled system is beneficial most of the time. This should be apparent by the average number of patients in the queue as well as their average waiting time. When the player finally made up his mind, the theoretical performances can be show by pressing the performance buttons at the top.
3. This last scenario is related to the pedagogic purpose of understanding that under certain conditions pooling might not be advantageous. There are not that many settings of the tool that highlight this and in general a high k-factor is needed for this. One, that achieves this, is

again the example scenario from van Dijk and van der Sluis (2008). For this, the arrival and service rate for one patient can be set at 50/hour and 60/hour, respectively. And the other at 5/hour and 6/hour. This results in a traffic load of about 83% and a k-factor of 10. Here, the expected waiting time for the average patient is lower in the unpooled system, though if just the patients with the lower magnitude-rates are considered, their personal situation worsens.

6.2 Validation of Learning Objective

The validation of the tool's learning objective proves that the final tool actually helps in teaching. In practice, most of the validation of such tools likely happens through trial-and-error application by teachers and their intuition on how successful the tool is. There are also good reasons for doing so, as it is not only cheap but does not take a lot of time either. Especially for a tool that was created for applications that are limited by players available time. However, to ensure that the tool works a more scientific approach needs to be applied. This can for example be done by the use of a questionnaire about the topic. Players could answer questions regarding their intuition of the topic before being confronted with the tool and after that. If the tool were to achieve its learning objective, the level of correctness of players' answers should notably improve. This is, however, out of the scope of this project and is not pursued any further here.

7 Conclusion and Discussion

7.1 Conclusion

In this paper we show that teachers and consultants in the field of operations management and in the context of healthcare environments often times struggle to sufficiently teach and convince healthcare professionals about certain concepts and insights. By assessing this very problem the paper proposes that the development of a digital educational tool, or serious game, can help in improving the situation. Consequently, it presents the realization of such a digital educational tool to support the process of teaching and consulting about one specific insight from operations management. This insight being the effects that can occur on queueing system when pooling. For this, we first consult the literature about educational tools and serious games in general and then discuss the aspects of queueing and pooling that are relevant to the tool. With the help of a conceptual modeling framework for serious games we then design an appropriate tool for the problem at hand, which is then implemented for online access. Finally, the paper shows how the tool can be applied in practice by presenting some example scenarios and use cases.

7.2 Possible Future Improvements and Extensions

During the execution of this project and the included implementation of the tool, several point of improvements and ideas for extensions became apparent. In the following these are listed, though the points made are not exhaustive in any way and every reader is encouraged to think for more and give feedback.

- Right now, there are only two servers (doctors) in the system. The main paper on which this tool is based, however, goes one step further and develops formulas for the case of each server being a group of servers. This functionality can be added to the tool.
- Most, or all, of the activities the game operator has to perform with challenging the player, giving detailed feedback, and answering questions of the player, can potentially be automated by leading the player through a game with an actual report at the end.
- The protocol that is used for communication of system status to the frontend with JavaScript and D3 can be reconstructed in a more efficient manner.
- Alternatively, the entire simulation model could be recreated in JavaScript, removing the bottleneck of constantly sending information to the frontend completely.
- Automated software testing can be added to the code repository. Extensive unit testing would increase code quality dramatically.
- Proper validation of the learning objective with the help of a survey is desirable and could increase the overall quality of the project.

References

Cattani, K., & Schmidt, G. M. (2005). The Pooling Principle. *INFORMS Transactions on Education*, *5*(2), 17–24. https://doi.org/10.1287/ited.5.2.17

Djaouti, D., Alvarez, J., & Jessel, J.-P. (2011). Classifying Serious Games: the G/P/S model. In *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches* (pp. 118–136). https://doi.org/http://doi:10.4018/978-1-60960-495-0.ch006

Greenblat, C. (1988). *Designing games and simulations: An illustrated handbook*. SAGE Publications Inc.

Laamarti, F., Eid, M., & Saddik, A. el. (2014). An Overview of Serious Games. *Int. J. Comput. Games Technol.*, *2014*. https://doi.org/10.1155/2014/358152

Mandelbaum, A., & Reiman, M. I. (1998). On Pooling in Queueing Networks. *Management Science*, *44*(7), 971–981. https://doi.org/10.1287/mnsc.44.7.971

Michael, D. (2006). *Serious games : games that educate, train and inform*. Thomson Course Technology. http://site.ebrary.com/id/10087000

Robinson, S. (2008). Conceptual modelling for simulation Part II: a framework for conceptual modelling. *Journal of the Operational Research Society*, *59*(3), 291–304. https://doi.org/10.1057/palgrave.jors.2602369

Robinson, S. (2014). Inside Simulation Software. In *Simulation - The Practice of Model Development and Use* (Second edi, pp. 21–47).

Rothkopf, M. H., & Rech, P. (1987). Perspectives on Queues : Combining Queues is Not Always Beneficial. *INFORMS Operations Research*, *35*(6), 906–909. https://doi.org/https://doi.org/10.1287/opre.35.6.906

Susi, T., Johannesson, M., & Backlund, P. (2007). Serious Games : An Overview. In *IKI Technical Reports NV - HS-IKI-TR-07-001*. Institutionen för kommunikation och information. http://his.diva-portal.org/smash/get/diva2:2416/FULLTEXT01.pdf

van der Zee, D.-J., Holkenborg, B., & Robinson, S. (2012). Conceptual modeling for simulation-based serious gaming. *Decision Support Systems*, *54*(1), 33–45. https://doi.org/10.1016/j.dss.2012.03.006

van Dijk, N. M., & van der Sluis, E. (2008). To Pool or Not to Pool in Call Centers. *Production and Operations Management*, *17*(3), 296–305. https://doi.org/https://doi.org/10.3401/poms.1080.0029

Vanberkel, P. T., Boucherie, R. J., Hans, E. W., Hurink, J. L., & Litvak, N. (2012). Efficiency evaluation for pooling resources in health care. *OR Spectrum*, *34*(2), 371–390. https://doi.org/10.1007/s00291-010-0228-x

Wilkinson, P. (2016). *A Brief History of Serious Games BT - Entertainment Computing and Serious Games: International GI-Dagstuhl Seminar 15283, Dagstuhl Castle, Germany, July 5-10, 2015, Revised Selected Papers* (R. Dörner, S. Göbel, M. Kickmeier-Rust, M. Masuch, & K. Zweig, Eds.; pp. 17–41). Springer International Publishing. https://doi.org/10.1007/978-3-319-46152-6_2

Appendix

Development Manual

This section describes how the educational tool can be improved and adapted in general. Knowledge of R as well as the very fundamentals of JavaScript, HTML and CSS is assumed. If not, many tutorials and books can be found online covering these exact topics.

First of all, a current version of R needs to be installed on the computer intended for the programming. Here, we recommend the installation of *RStudio* as it is a good IDE and it automatically installs R as well. RStudio also provides good support for running and testing *R Shiny* applications by default. To learn about R Shiny we recommend the official tutorial at https://shiny.rstudio.com/tutorial/. The codebase of the herein developed educational tool can be found in the *GitHub* repository under the following link:

---

https://github.com/philippun/queueing-ed-tool

The code can be best understood by studying the documentation of the repository and commentary of the code itself. We then also recommend to make use of *git* and fork the repository from GitHub. This especially makes programming in a team much more convenient and secure. However, using git is not at all necessary and the codebase can also just be copied onto the local machine. After having access to the code it needs to be ensured that the libraries *Shiny*, *markdown* and *jsonlite* for R are installed within RStudio. It might also be more convenient to install a separate editor for programming with *JavaScript* and *D3* as RStudio heavily focuses on R and the code highlighting of JavaScript is subpar without additional add-ins. We used the lightweight editor *Atom* for this purpose. A good tutorial on the use of D3 can be found at https://www.youtube.com/watch?v=_8V5o2UHG0E by *Curran Kelleher*.