Maastricht University

Simulation and waiting lines

Course notes

Coordinator: Christof Defryn

Version of April 6, 2022

These lecture notes have been created for Bsc students International Business, Business Engineering and Business Analytics at the School of Business and Economics, Maastricht University (The Netherlands). The content is based on the references listed at the end of the document.

Instructions. These course notes contain a to-the-point and complete introduction of the theory that falls within the scope of the course as well as numerous illustrative examples. Before coming to the first tutorial meeting during which the topic will be discussed, please carefully read the course notes and solve the exercises in the grey boxes. During the tutorial meeting, we will address questions that result from your reading and discuss the solutions to the prepared exercises together.

Contents

Learning outcomes

After this chapter, you should be able to:

- explain the importance of waiting line theory in operations management.
- describe the characteristics of waiting lines.
- distinguish the single-server and a multiserver model, and interpret the assumptions of these models.
- calculate the operating characteristics of a single- and multiserver model given the formulas introduced in this chapter.
- discuss the working of a simple simulation model.
- perform a discrete event simulation by hand.
- perform a basic discrete event simulation by means of a simulation software package.
- interpret the results of the output of a simulation software package.
- adopt a critical opinion on the results of waiting line analysis and discuss the limitations of the chosen methods.

PART I

WAITING LINE THEORY

1 Introduction to waiting line theory

Example 1.1. The mission of Walt Disney theme parks is to *make people happy*, and the folks at Disney World in Orlando are masters at doing that. They realize that waiting in lines at attractions does not add to the enjoyment of their customers. They also realize that customers waiting in lines are not generating the revenue that they would if they visited restaurants and souvenir shops. Hence, they have several reasons for wanting to reduce waiting times.

Lately they have been using a reservation system called *FastPass* at some attractions that allow customers to reserve visit times instead of having to wait in line. This is a win-win solution: customers are happier because they do not have to wait in line, and the park's potential for additional revenue is increased.

Waiting lines abound in all sorts of service systems while they are typically considered non-valued-added occurrences. In lean systems, waiting is one of the seven wastes. For customers, having to wait for service can range from being acceptable (usually short waits), to being annoying (longer waits), to being a matter of life and death (e.g., in emergencies). For businesses, the costs of waiting come from lower productivity and competitive disadvantage. For society, the costs are wasted resources (e.g., fuel consumption of cars stuck in traffic) and reduced quality of life. Hence, it is important for system designers and managers of existing service systems to fully appreciate the impact of waiting lines.

Designers must weigh the cost of providing a given level of service capacity against the potential (implicit) cost of having customers wait for service. This planning and analysis of service capacity frequently lends itself to queuing theory, which is a mathematical approach to the analysis of waiting lines. Queuing theory is directly applicable to a wide range of service operations, including call centers, banks, post offices, restaurants, theme parks, telecommunications systems, and traffic management.

Many people are surprised to learn that waiting lines tend to form even though a system is basically underloaded. For example, a fast-food restaurant may have the capacity to handle an average of 200 orders per hour and yet experience waiting lines even though the average number of orders is only 150 per hour. The key word is **average**. In reality, customers arrive at random intervals rather than at evenly spaced intervals, and some orders take longer to fill than others. In other words, both arrivals and service times exhibit a high degree of variability. Because services cannot be performed ahead of time and stored until needed, the system at times becomes temporarily overloaded, giving rise to waiting lines. However, at other times, the system is idle because there are no customers. It follows that in systems where variability is minimal or non-existent (e.g., because arrivals can be scheduled and service time is constant), waiting lines do not ordinarily form.

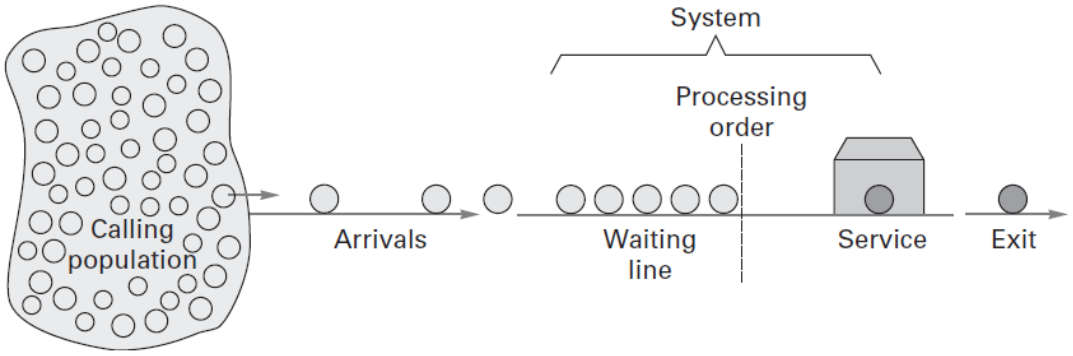


Figure 1: *Visualization of a general queuing system.*

2 Characteristics of waiting lines

A general queuing system is visualized in Figure 1. Depending on the specific characteristics of the system, there are numerous queuing models from which an analyst can choose. Naturally, much of the success of the analysis will depend on choosing an appropriate model. The main characteristics are *population source*, *number of resources (servers)*, *arrival & service patterns*, and *Queue discipline*. We will discuss each characteristic in more detail in what follows.

2.1 Population Source

The approach to use in analyzing a queuing problem depends on whether the potential number of customers is limited. There are two possibilities: *infinite-source* and *finite-source populations*.

In an **infinite-source situation**, the potential number of customers greatly exceeds system capacity. Infinite-source situations exist whenever service is unrestricted. Examples are supermarkets, drugstores, banks, restaurants, theaters, amusement centers, and toll bridges. Theoretically, large numbers of customers from the *calling population* can request service at any time.

When the potential number of customers is limited, a **finite-source situation** exists. An example is the repairman responsible for a certain number of machines in a company. The potential number of machines that might need repairs at any one time cannot exceed the number of machines assigned to the repairer. Similarly, an operator may be responsible for loading and unloading a bank of four machines, a nurse may be responsible for answering patient calls for a 10-bed ward, a secretary may be responsible for taking dictation from three executives, and a company shop may perform repairs as needed on the firm's 20 trucks.

2.2 Number of servers

The capacity of queuing systems is a function of the capacity of each server and the number of servers being used. It is generally assumed that each channel can handle one customer at a time. Systems can be either *single- or multiple-channel*. (A group of servers working together as a team, such as a surgical team, is treated as a single-channel system.) Note that a server is often also referred to as a *resource* or a *channel*.

Examples of **single-server systems** are small grocery stores with one checkout counter, some theaters, single-bay car washes, and drive-in banks with one teller. **Multiserver systems** (those with more than one server) are commonly found in banks, at airline ticket counters, at auto service centers, and at gas stations.

A related distinction is the number of steps or **phases** in a queuing system. For example, at theme parks, people go from one attraction to another. Each attraction constitutes a separate phase where queues can (and usually do) form.

An non-exhaustive overview of different waiting line systems for varying number of servers and phases is given in Figure 2.

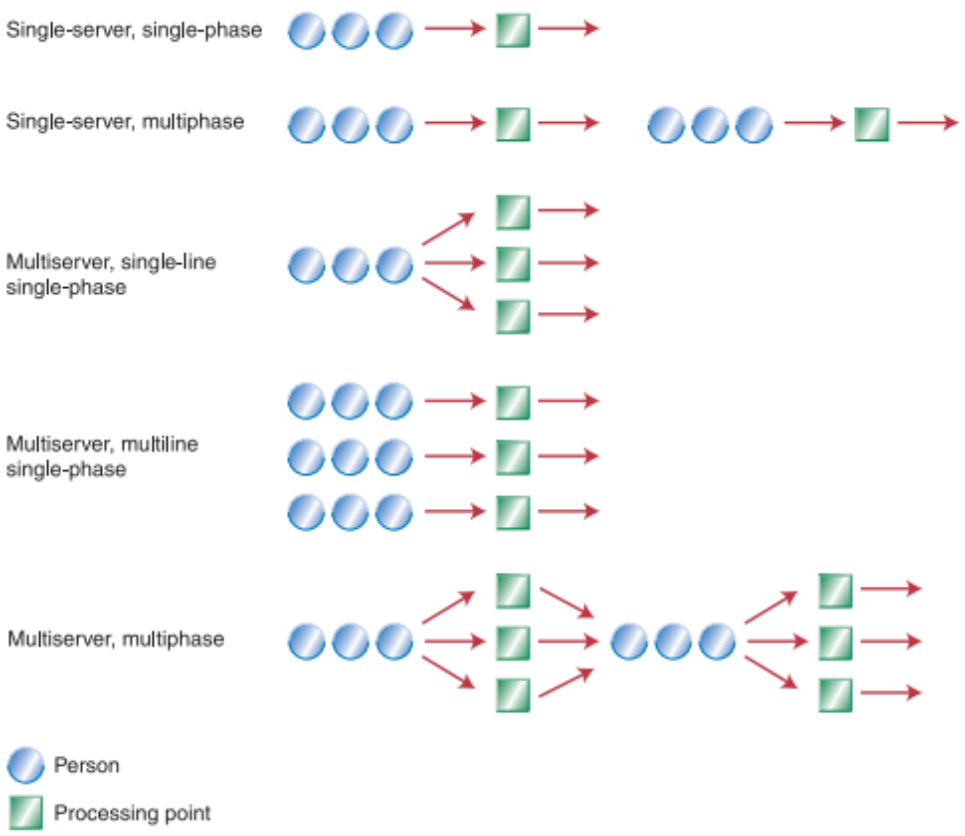


Figure 2: *Overview of different types of waiting line systems.*

2.3 Arrival and Service Patterns

Waiting lines are a direct result of arrival and service variability. They occur because random, highly variable arrival and service patterns cause systems to be temporarily overloaded. In many instances, the variabilities can be described by theoretical distributions. In fact, the most commonly used models assume that arrival and service rates can be described by a *Poisson distribution* or, equivalently, that the interarrival time and service time can be described by a *negative exponential distribution*. Waiting lines are most likely to occur when arrivals are bunched or when service times are particularly lengthy, and they are very likely to occur when both factors are present.

2.4 Queue discipline

The queue discipline refers to the **order in which customers are processed**. FIFO (first-in-first-out) is the most common one. There is first-come service at banks, stores, theaters, restaurants, four-way stop signs, registration lines, and so on.

The opposite of FIFO is LIFO (last-in-first out), where the entity that was added to the queue the latest is processed first. LIFO systems are found when loading and unloading trucks, or when entities are put on a pile (e.g., when washing dishes). The LIFO system is also very common in computer memory management, where the queue is referred to as the stack.

Examples of systems that do not serve on a FIFO or LIFO basis include hospital emergency rooms, rush orders in a factory, and mainframe computer processing of jobs. In these and similar situations, customers do not all represent the same waiting costs; those with the highest costs (e.g., the most seriously ill) are processed first, even though other customers may have arrived earlier.

3 Waiting line models

Different combinations of waiting line characteristics give rise to a big variety in waiting line models. In this section, we will explore the two most common models in detail.

3.1 Single-server waiting line model

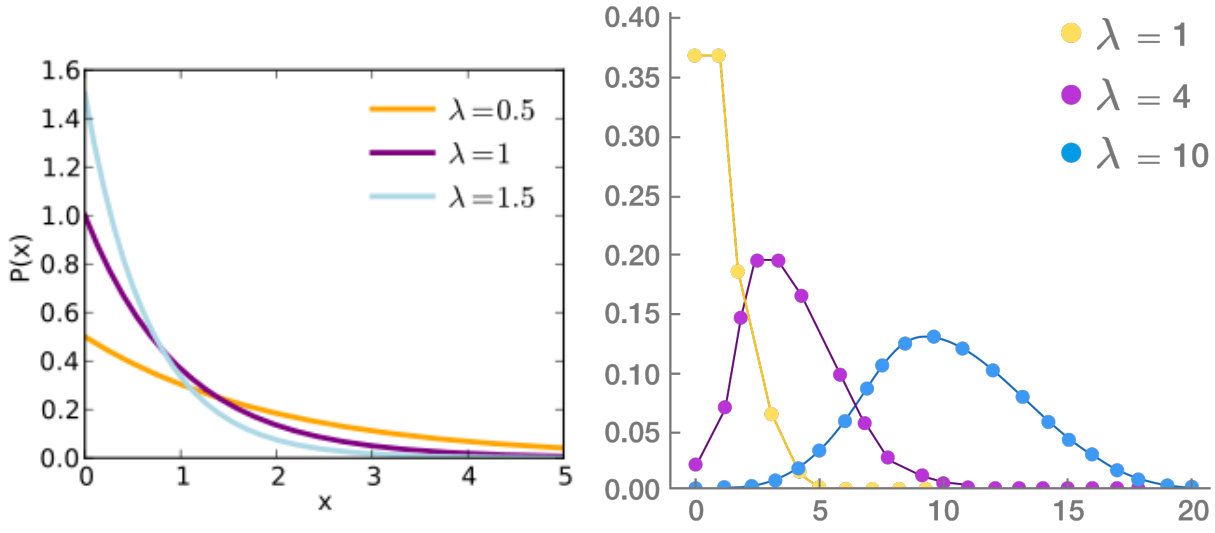
The easiest waiting line model involves a **single-server, single-line, single-phase system**. The following assumptions are made when we model this environment:

1. The customers are **patient** (no balking, reneging, or jockeying) and come from a population that can be considered infinite.
2. Customer **arrivals** follow a Poisson process with a mean arrival rate of λ (lambda). This means that the time between successive customer arrivals follows an exponential distribution with an average of $1/\lambda$.
3. The customer **service rate** is described by a Poisson distribution with a mean service rate of μ (mu). This means that the service time for one customer follows an exponential distribution with an average of $1/\mu$.
4. The waiting line priority rule (**queue discipline**) used is first-come, first-served.

The service rate must be greater than the arrival rate, that is, $\mu > \lambda$. If $\mu \leq \lambda$, the waiting line would eventually grow infinitely large. Before using the formulas, check to be sure that $\mu > \lambda$.

Exponential distribution. The exponential distribution is a continuous probability density function, defined within the range $[0, \infty]$. It is typically used as the probability distribution of the *time between events* in a Poisson process.

Poisson distribution. The Poisson distribution is a discrete probability density function. It can, therefore, only take the value of positive integers. It is typically used to measure the probability of a given *number of events* occurring in a fixed interval of time or space if these events occur with a known constant mean rate and independently of the time since the last event.



(a) Exponential distribution. *(b) Poisson distribution.*

Figure 3: *Visualisation of the Exponential and Poisson distribution.*

A visual representation of the Exponential and Poisson distribution are given in Figure 3. Notice that the exponential and poisson distributions are not symmetric and there is a larger spread of the possible outcomes in the upper regions of the distribution. This has some implications, namely that the median of the exponential distribution is less than the mean. In other words, it is more likely to draw a value that is lower than the average. However, you have a chance of drawing very high numbers compared to the average value.

Using these assumptions, we can calculate the operating characteristics of a waiting line system using the following formulas.

Table 1: *Formulas for the single-server, single-line, single-phase waiting line model.*

λ	Mean arrival rate of customers (average number of customers arriving per unit of time).
μ	Mean service rate (average number of customers that can be served per unit of time).
$\rho = \frac{\lambda}{\mu}$	The average utilization of the system.
$L = \frac{\lambda}{\mu - \lambda}$	The average number of customers in the system (in queue + in service).
$L_Q = \rho L$	The average number of customers waiting in line.
$W = \frac{1}{\mu - \lambda}$	The average time spent in the system, including service.
$W_Q = \rho W$	The average time spent waiting in line.
$P_n = (1 - \rho)\rho^n$	The probability that n customers are in the service system at a given time.

Example 3.1. The computer lab at State University has a help desk to assist students working on computer spreadsheet assignments. The students patiently form a single line in front of the desk to wait for help. Students are served based on a first-come, first-served priority rule.

On average, 15 students per hour arrive at the help desk. Student arrivals are best described using a Poisson distribution. The help desk server can help an average of 20 students per hour, with the service rate being described by an exponential distribution. Calculate the following operating characteristics of the service system.

1. The average utilization of the help desk server
2. The average number of students in the system
3. The average number of students waiting in line
4. The average time a student spends in the system

5. The average time a student spends waiting in line
6. The probability of having more than 4 students in the system

Before You Begin:
The key to solving queuing problems is to identify the mean arrival rate of customers and the mean service rate. In this case, on average, 15 customers arrive each hour. On average, the consultant can serve 20 customers per hour. Once you have established these values, you merely plug them into the appropriate formula.

Solution:

1. Average utilization:
$$\rho = \frac{\lambda}{\mu} = \frac{15}{20} = 0.75, \text{ or } 75\%.$$
2. Average number of students in the system:
$$L = \frac{\lambda}{\mu - \lambda} = \frac{15}{20 - 15} = 3 \text{ students.}$$
3. Average number of students waiting in line:
$$L_Q = \rho L = 0.75 \times 3 = 2.25 \text{ students.}$$
4. Average time a student spent in the system:
$$W = \frac{1}{\mu - \lambda} = \frac{1}{20 - 15} = 0.2 \text{ hours, or 12 minutes.}$$
5. Average time a student spent waiting in line:
$$W_Q = \rho W = 0.75 \times 0.2 = 0.15 \text{ hours, or 9 minutes.}$$
6. The probability that there are more than four students in the system equals one minus the probability that there are four or fewer students in the system. We use the following formula:
$$\begin{aligned} P &= 1 - \sum_{n=0}^{4} P_n = 1 - \sum_{n=0}^{4} (1 - \rho)\rho^n \\ &= 1 - 0.25 \times (1 + 0.75 + 0.75^2 + 0.75^3 + 0.75^4) \\ &= 1 - 0.7626 = 0.2374 \end{aligned}$$
There is a 23.74% chance of having more than four students in the system.

Exercise 3.1. The local division of MOTOR VEHICLES (MV) is concerned with its waiting line system. Currently, the MV uses a single-server, single-line, single-phase system when processing license renewals. Based on historical evidence, the average number of customers arriving per hour is 9 and is described by a Poisson distribution. The service rate is 12 customers per hour with the service times following an exponential distribution. The customers are patient and come from an infinite population. The manager of the MV would like you to calculate the operational characteristics of the waiting line system.

1. What is the average system utilization?
2. What is the average number of customers in the system?
3. What is the average number of customers waiting in line?
4. What is the average time a customer spends in the system?
5. What is the average time a customer spends waiting in line?

3.2 Multiserver waiting line model

In the single-line, multiserver, single-phase model, customers form a single line and are served by the first server available. The model assumes that there are s identical servers and the service time distribution for each server is the same as for the single-server model (i.e., exponential distribution with an average of $1/\mu$). Also, all other assumptions from the single-server model (patient entities, arrivals according to Poisson process and FIFO queue discipline) should hold again. Using these assumptions, we can describe the operating characteristics with the following formulas provided in Table 2.

The total service rate must be greater than the arrival rate, that is, $s\mu > \lambda$. If $s\mu \leq \lambda$, the waiting line would eventually grow infinitely large. Before using the formulas, check to be sure that $s\mu > \lambda$.

Example 3.2. State University has decided to increase the number of computer assignments in its curriculum and is concerned about the impact on the help desk. Instead of a single person working at the help desk, the university is considering a plan to have three identical service providers. It expects that students will arrive at a rate of 45 per hour, according to a Poisson distribution. The service rate for each of the three servers is 18 students per hour, with exponential service times. Calculate the following operating characteristics of the service system:

Table 2: *Formulas for the multiserver, single-line, single-phase waiting line model.*

s	The number of servers in the system.
λ	Mean arrival rate of customers (average number of customers arriving per unit of time).
μ	Mean service rate (average number of customers that can be served per unit of time).
$\rho = \frac{\lambda}{s\mu}$	The average utilization of the system.
$P_0 = \left[\sum_{n=0}^{s-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^s}{s!}\left(\frac{1}{1-\rho}\right) \right]^{-1}$	The probability that no customers are in the system.
$L_Q = \frac{P_0(\lambda/\mu)^s \rho}{s!(1-\rho)^2}$	The average number of customers waiting in line.
$W_Q = \frac{L_Q}{\lambda}$	The average time spent waiting in line.
$W = W_Q + \frac{1}{\mu}$	The average time spent waiting in the system, including service.
$L = \lambda W$	The average number of customers in the system (in queue + in service).
$P_n = \begin{cases} \frac{(\lambda/\mu)^n}{n!} P_0 & \text{for } n \leq s \\ \frac{(\lambda/\mu)^n}{s! s^{n-s}} P_0 & \text{for } n > s \end{cases}$	The probability that n customers are in the service system at a given time.

1. The average utilization of the help desk
2. The probability that there are no students in the system
3. The average number of students waiting in line
4. The average time a student spends waiting in line
5. The average time a student spends in the system
6. The average number of students in the system

Solution:

1. Average utilization:

$$\rho = \frac{\lambda}{s\mu} = \frac{45}{3 \times 18} = 0.833, \text{or } 83.3\%.$$

2. The probability that there are no students in the system:

$$\begin{aligned} P_0 &= \left[\sum_{n=0}^{s-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^s}{s!} \left(\frac{1}{1-\rho}\right) \right]^{-1} \\ &= \left[\frac{(45/18)^0}{0!} + \frac{(45/18)^1}{1!} + \frac{(45/18)^2}{2!} + \left(\frac{(45/18)^3}{3!} \left(\frac{1}{1-0.833}\right) \right) \right]^{-1} \\ &= \frac{1}{22.215} = 0.045, \text{or } 4.5\% \text{ of having no students in the system.} \end{aligned}$$

3. The average number of students waiting in line:

$$L_Q = \frac{P_0(\lambda/\mu)^s \rho}{s!(1-\rho)^2} = \frac{0.045(45/18)^3 \times 0.833}{3! \times (1-0.833)^2} = \frac{0.5857}{0.1673} = 3.5 \text{ students.}$$

4. The average time a student spends waiting in line:

$$W_Q = \frac{L_Q}{\lambda} = \frac{3.5}{45} = 0.078 \text{ hours, or } 4.68 \text{ minutes.}$$

5. The average time a student spends in the system:

$$W = W_Q + \frac{1}{\mu} = 0.078 + \frac{1}{18} = 0.134 \text{ hours, or } 8.04 \text{ minutes.}$$

6. The average number of students in the system:

$$L = \lambda W = 45 \times 0.134 = 6.03 \text{ students.}$$

Exercise 3.2. The county has decided to consolidate several of its DMV facilities into a larger, centrally located facility. The DMV manager wants you to calculate the operational characteristics of a multiserver, single-phase waiting line system. The arrival rate is expected to be 72 customers per hour and follows a Poisson distribution. The number of identical servers is seven. Each server will be able to serve an average of 12 customers per hour. The service times are described by an exponential distribution. Your job is to calculate the following:

1. The average system utilization.
2. The probability of no customers in the system.
3. The average number of customers waiting in line.
4. The average time a customer waits in line.
5. The average time a customer spends in the system.

4 Managing of waiting lines

Although it is unlikely that you calculate performance measures for the lines you wait in on a day-to-day basis, you should now be aware of the potential for mathematical analysis of these systems. More importantly, management has a tool by which it can evaluate system performance and make decisions as to how to improve the performance while weighing performance against the costs to achieve that performance.

Waiting line models are important to a company because they directly affect customer service perception and the costs of providing a service. Several functional areas are affected by waiting line decisions.

- **Accounting** is concerned with the cost of the waiting line system used. If system average utilization is low, that suggests the waiting line design is inefficient and too expensive. Poor system design can result in overstaffing or unnecessary capital acquisitions in an effort to improve customer service.
- **Marketing** is concerned about response time for customers—how long customers must wait in line before being served and how long it takes to be served. Quick service or response can be a competitive advantage. Long waits suggest a lack of concern by the organization or can be linked to a perception of poor service quality.
- **Purchasing** must be sure to buy capital equipment capable of achieving the proposed service rate. Operations uses waiting line theory to estimate queues or waiting times at different processing points, to allow for a better estimate of lead time and improve due-date delivery promising.

- **Operations** is also affected by the system design. When single-phase systems are used, operators must have greater skills. The organization needs to hire employees with higher skill levels or provide training to upgrade the workforce.

After calculating the operating characteristics for a waiting line system, sometimes you need to change the system to alter its performance. Let us look at the type of changes you can make to the different elements of the waiting line system.

- **Customer arrival rates.** You can try to change arrival rates in a number of ways. For example, you can provide discounts or run special promotions during the nonpeak hours to attract customers.
- **Number and type of service facilities.** You can either increase or decrease the number of server facilities. For example, a grocery store can easily change the number of cashiers open for business (up to the number of registers available). The store increases the number of cashiers open when lines are too long. Another approach is to dedicate specific servers for specific transactions. One example would be to limit the number of items that can be processed at a particular cashier (ten items or less) or to limit a cashier to cash-only transactions. Still another possibility is to install self-service checkout systems.
- **Changing the number of phases.** You can use a multiphase system where servers specialize in a portion of the total service rather than needing to know the entire service provided. Since a server has fewer tasks to learn, the individual server proficiency should improve. This goes back to the concept of division of labor.
- **Server efficiency.** You can improve server efficiency through process improvements or dedication of additional resources. For example, cashier accuracy and speed are improved through the use of scanners. Service speed can also be increased by dedicating additional resources. For example, if a grocery bagger is added at each cashier station, service speed will be improved and customers will flow through the system more quickly.
- **Changing the priority rule.** The priority rule determines who should be served next. There are priority rules other than first-come, first-served. If you want to change priority rules, consider the impact on those customers who will wait longer.
- **Changing the number of lines.** Changing to a single-line model from a multiline model is most appropriate when the company is concerned about fairness for its customers. A single line ensures that customers do not jockey in an attempt to gain an advantage over another customer. Multiline models easily accommodate specialty servers (express lanes).

Exercise 4.1. In a bookshop in the city center there is only one cashier because the other two are off with influenza. Normally the cashier is able to cash up an average of 70 customers per hour with a service rate being described by an exponential distribution. On average there are only 55 customers per hour in the bookshop who want to buy a book. Customer arrivals to the counter are described using a Poisson distribution. Next to cashing up customers the cashier also has to unpack deliveries. We assume that the customers are patient while waiting in the queue.

Answer the following questions.

1. What is the amount of time between two consecutive customers wanting to buy a book, on average?
2. Compute how much time the cashier can spend on unpacking deliveries on average.
3. On average, how many people are waiting in the queue? How many people are present in the whole system?
4. A marketing campaign showed that if a customer has to wait more than 5 minutes before being served, he will not come back to the shop in the future. Should the bookshop be worried about that?
5. What is the probability that a customer will not have to wait to buy his book? We assume that the cashier serves a customer as soon as he arrives to the counter even if he was unpacking deliveries.
6. Give two pieces of advice to the bookshop in order to improve the efficiency of the waiting lines and explain them briefly.

PART II

SIMULATION

5 Introduction to simulation

Simulation refers to a broad collection of methods and applications to mimic the operation of a real-world process or system over time. It is typically done on a computer using appropriate software, and can be applied across many fields, industries and applications.

5.1 What is a system?

To model a system, it is necessary to understand the concept of a system and the system boundary. A **system** is defined as a group of objects that are joined together in some regular interaction or interdependence towards the accomplishment of some purpose. This maybe sounds rather vague. We therefore provide you with a few examples:

- A manufacturing plant with machines, people, transport devices, conveyor belts, and storage space.
- An airport with departing passengers checking in, going through security, going to the departure gate, and boarding; departing flights contending for push-back tugs and runway slots; arriving flights contending for runways, gates, and arrival crew; arriving passengers moving to baggage claim and waiting for their bags; and the baggage-handling system dealing with delays, security issues, and equipment failure.
- An emergency facility in a hospital, including personnel, rooms, equipment, supplies, and patient transport.
- A fast-food restaurant with different types of staff, customers, and equipment.
- A theme park with rides, stores, restaurants, workers, guests, and parking lots.

People often study a system to measure its performance, improve its operation, or design it if it does not exist yet. Managers or controllers of a system might also like to have a readily available aid for day-to-day operations, like help in deciding what to do in a factory if an important machine goes down.

A system is often affected by changes occurring outside the system. Such changes are said to occur in the system environment. When modelling systems, it is therefore necessary to decide on the **system boundary**, i.e. the boundary between the system and its environment. This decision may depend on the purpose of the study.

5.2 The need for a model

It might be possible to experiment with the actual physical system. For example:

- A supermarket manager might try different policies for inventory control and checkout-personnel assignment to see what combinations seem to be most profitable and provide the best service.
- An airline could test the expanded use of automated check-in kiosks (with employees urging passengers to use them) to see if this speeds check-in.

This approach certainly has its advantages. If you can experiment directly with the system and know that nothing else about it will change significantly, then you're unquestionably looking at the right thing and you do not need to worry about whether a model for the system faithfully mimics it for your purposes.

In many cases, however, it's just too difficult, costly, or downright impossible to do physical studies on the system itself.

- Obviously, you can't experiment with alternative layouts of a factory if it's not yet built.
- Even in an existing factory, it might be costly to change to an experimental layout that might not work out in the end.
- Fiddling around with emergency room staffing in a hospital clearly won't do.
- You do not want to experiment with different emergency policies by actually creating a fire in your facility.

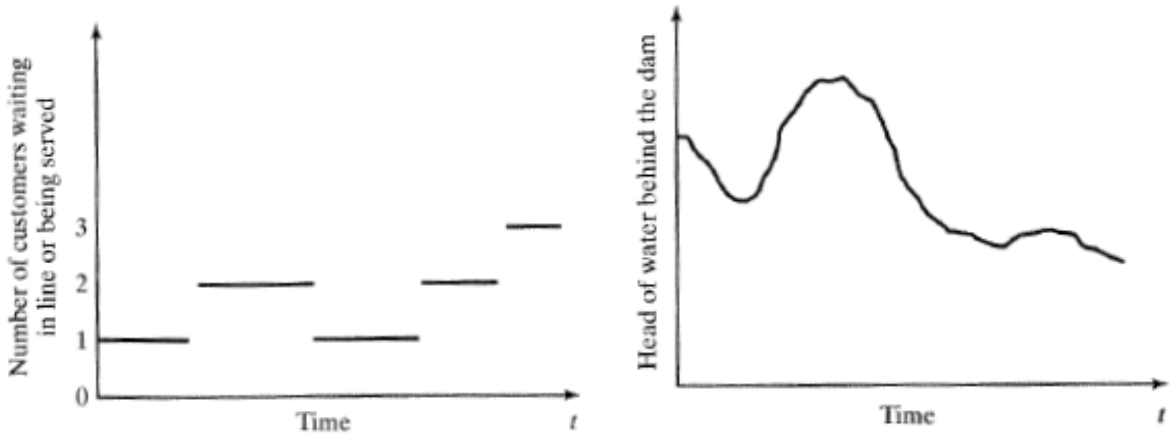
In these situations, you might build a **model** to serve as a stand-in for studying the system and ask pertinent questions about what *would* happen in the system *if* you did this or that, or *if* some situations beyond your control were to develop. Nobody gets hurt, and your freedom to try wide-ranging ideas with the model could uncover attractive alternatives that you might not have been able to try with the real system.

However, you have to build models carefully and with enough detail so that what you learn about the model will never be different from what you would have learned about the system by playing with it directly. This is called **model validity**, and we will come back to this later.

5.3 Discrete and continuous systems

Systems can be categorized as discrete or continuous. A *discrete system* is one in which the state variable(s) change only at a discrete set of points in time. The bank is an example of a discrete system. The number of customers in the bank (our state variable), changes only when a customer arrives or when the service provided to the customer is completed. Figure 4a shows how the number of customers changes only at discrete points in time.

A *continuous system* is one in which the state variable(s) change continuously over time. An example is the head of water behind a dam. During and for some time after a rain storm, water flows into the lake behind the dam. Water is drawn from the dam for flood control an to make electricity. Evaporation also decreases the water level. Figure 4b shows the state variable changes for this continuous system.



(a) Discrete system *(b) Continuous system*

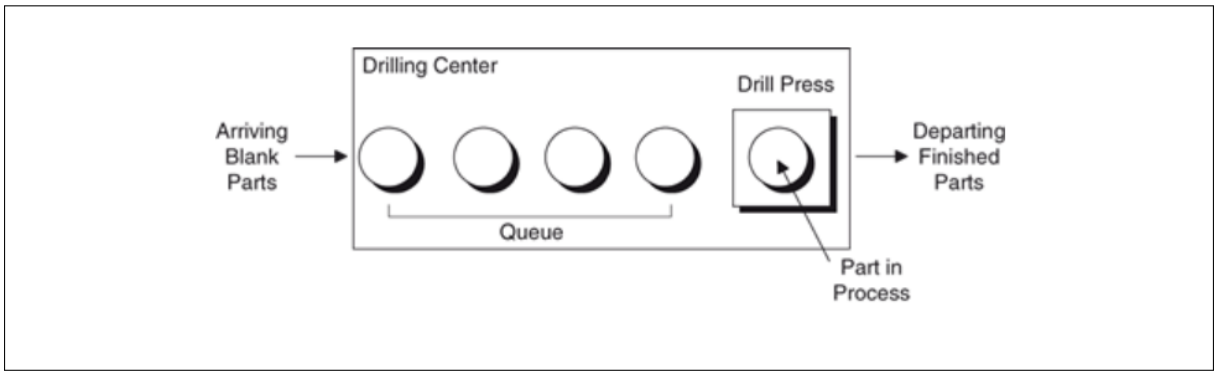
Figure 4: *Visualisation of the state variable in a discrete and continuous system.*

Focus of this chapter = Discrete-event simulation.

6 Fundamentals of discrete-event simulation

To illustrated the main concepts within a discrete-event simulation model, we will make use of the following single-server, single-phase example: the drilling center.

Blank **parts** arrive to a drilling center, are processed by a single drill press, and then leave (see the illustration below). If a part arrives and finds the drill press idle, its processing at the drill press starts right away; otherwise, it waits in a First-In First-Out (FIFO), that is, first-come first-served queue.



6.1 Entities

Most simulations involve players called *entities* that move around, change status, affect and are affected by other entities and the state of the system, and affect the output performance measures. Entities are the dynamic objects in the simulation. They usually are created, move around for a while, and then are disposed of as they leave.

Figuring out what the entities are is the first thing you need to do in modelling a system.

> The entities for our drill press the parts to be processed. They are created when they arrive, move through the queue (if necessary), are served by the drill press, and are then disposed of as they leave. Even though there is only one kind of entity in our example, there can be many independent copies (or realizations) of it in existence at a time, just as there can be many different individual parts of this type in the real system at a time.

6.2 Resources

Entities often compete with each other for service from *resources*, such as *personnel*, *equipment*, or *space* in a storage area of limited size. An entity seizes (units of) a resource when available and releases it (or them) when finished. It is better to think of the resource as being given to the entity rather than the entity being assigned to the resource since an entity (like a part) could need simultaneous service from multiple resources (such as a machine and a person).

A resource can represent a group of several individual *servers*, each of which is called a unit of that resource. This is useful to model, for instance, several identical parallel agents at an airline ticketing counter. The number of available units of a resource can be changed during the simulation run to represent agents going on break or opening up their stations if things get busy.

In our example, there is just a single drill press, so this resource has a single unit available at all times.

6.3 Queues

When an entity cannot move on, perhaps because it needs to seize a unit of a resource that is currently tied up by another entity, it needs a place to wait, which is the purpose of a *queue*. Queues can have limited capacities to represent, for instance, limited floor space for a buffer. You have to decide as part of your modeling how to handle an entity arriving at a queue that is already full.

6.4 Attributes

To individualize entities, you attach *attributes*. An attribute is a common characteristic of all entities, but with a specific value that can differ from one entity to another. It is up to you to figure out what attributes your entities need, name them, assign values to them, change them as appropriate, and then use them when it is time (that is all part of modelling).

In our example, the part entities could have attributes, such as *due date*, *priority*, and *color* to indicate these characteristics for each individual entity.

The most important thing to remember about attributes is that their values are tied to specific entities. The same attribute will generally have different values for different entities, just as different parts have different due dates, priorities, and color codes. Think of an attribute as a tag attached to each entity, but what is written on this tag can differ across entities to characterize them individually.

6.5 Events

Now let us turn to how things work when we run our model. Basically, everything is centered around events. An event is **something that happens at an instant of (simulated) time that might change attributes, variables, or one of our performance measures (also denoted as *statistical accumulators*.**

In our example, there are three kinds of events:

- **Arrival**: A new part enters the system.
- **Departure**: A part finishes its service at the drill press and leaves the system.
- **The End**: The simulation is stopped at time 20 minutes. (It might seem rather artificial to model this as an event, but it certainly changes things, and this is one way to stop a simulation.)

Remark: Other things happen in our example model, but do not need to be separate events. For instance, parts leave the queue and begin service at the drill press, which changes the system. However, this only happens because of some other entity's departure, which is already an event.

In a discrete-event model, the variables that describe the system do not change between successive events. Most of the work in event-driven simulation involves getting the logic right for what happens with each kind of event.

6.6 Simulation Clock

The current value of time in the simulation is simply held in a variable called the simulation clock. Unlike real time, the simulation clock does not take on all values and flow continuously; rather, it moves from the time of one event to the time of the next event scheduled to happen. Since nothing changes between events, there is no need to waste (real) time looking at (simulated) times that do not matter.

It is important to decide on the underlying *base units* with which time will be measured (e.g., minutes). It doesn't logically matter what the time units are, so pick whatever is most appropriate, familiar, and convenient for your application. You can express input time quantities in different units if it is customary or convenient, like minutes for mean service times but hours for mean machine-up times. However, for arithmetic and calculations, all times must be converted to the base time units if they are not already in it.

6.7 Starting and Stopping

Important, but sometimes overlooked, issues in a simulation are how it will start and stop. It is important to think about this and make assumptions consistent with what you are modelling. These decisions can have a significant effect on your results.

For the drilling center, we start at time 0 minutes with no parts present and the drill press idle. This empty-and-idle assumption would be realistic if the system starts afresh each morning, but might not be so great as a model of the initial situation to simulate an ongoing operation. Additionally, we have decided that the simulation will stop at exactly 40 minutes. If there are any parts present at that time (in service at the drill press or waiting in the queue), they are never finished.

7 System performance measurement

Being the decision makers, we would like to understand how the drilling center is actually performing. Next to just observing the system, as we did in the previous section, we would like to quantify its performance. But how to measure the performance of your system? You can think of multiple performance measures, depending on the application and the insights you want to gain. We present some suggestions below:

- **The total production** during the 40 minutes of operation. (i.e., the number of parts that complete their service at the drill press and leave). Presumably, more is better.

  From a performance standpoint, small is good.

- The **maximum waiting time in queue** of parts that enter service at the drill press during the simulation. This is a worst-case measure, which might be of interest in giving service-level guarantees to customers. Small is good.

- The **time-average number of parts waiting in the queue** (again, not counting any part in service at the drill press). By *time average*, we mean a weighted average of the possible queue lengths (0, 1, 2,...) weighted by the proportion of time during the run that the queue was at that length. Letting $Q(t)$ be the number of parts in the queue at any time instant t, this time-average queue length is the total area under the $Q(t)$ curve, divided by the length of the run, 20. In integralcalculus terms, this is:

  $$\frac{\int_0^{20} Q(t)dt}{20}$$

  Such time-persistent statistics are common in simulation. This one indicates how long the queue is (on average), which might be of interest for allocating floor space.

- The **maximum number of parts that were ever waiting in the queue**. Actually, this might be a better indication of how much floor space is needed than is the time average if you want to be reasonably sure to have room at all times. This is another worst-case measure, and smaller is presumably better.

- The **average and maximum total time in system** of parts that finish being processed on the drill press and leave. Also called **cycle time**, this is the time that elapses between a part's arrival and its departure, so it's the sum of the part's waiting time in queue and its service time at the drill press. This is a kind of turnaround time, so smaller is better.

- The **utilization** of the drill press, defined as the proportion of time it is busy during the simulation.

$$B(t) = \begin{cases} 1 & \textit{if the drill press is busy at time } t \\ 0 & \textit{if the drill press is idle at time } t \end{cases}$$

  The utilization is the area under $B(t)$, divided by the length of the run:

$$\frac{\int_0^{20} B(t)dt}{20}$$

  Resource utilizations are of obvious interest in many simulations, but it's hard to say whether you really want them to be high (close to 1) or low (close to 0). High is good since it indicates little excess capacity, but can also be bad since it might mean a lot of congestion in the form of long queues and slow throughput.

There are usually a lot of possible output performance measures, and it is probably a good idea to observe a lot of things in a simulation since you can always ignore things you have but can never look at things you do not have, plus sometimes you might find a surprise. The only downside is that collecting extraneous data can slow down execution of the simulation.

8 Basic modelling in JaamSim

8.1 Download and run the software

It is time to move to the JaamSim software. If you did not install the software yet, do it now. The software is open source and can be downloaded freely here:

`https://jaamsim.com/downloads.html`

The software is compatible with all Operating Systems (Windows, OSX or Linux). JaamSim is distributed with all dependencies in a single file, which means you just need to download the executable and run it directly (no installation in required). It is possible

that a warning is shown by your operating system. You can assume that the software is safe. On Windows you might have to click 'show more' first to be able to allow the software to run.

Sometimes, Mac users experience difficulties running the software correctly. In this case, the following website might help to resolve the issues.

`https://www.physiome.org/jsim/download/macos.html`

In case the issues remain, I suggest using the Athena Desktop Everywhere for running the Jaamsim software.

8.2 Your first simulation model in JaamSim

In this section, we will lead you through the construction of the drilling center example model from scratch. We would like to encourage you to try all the different steps yourself using the software while reading this section. This will prepare you to build your own models afterwards.

8.2.1 A first look at the JaamSim window

When you open the JaamSim software, you will see the screen depicted in Figure 5.

After launching JaamSim, the following windows will appear:

- **Control Panel.** This window provides a number of run control features.
- **View Window.** This window displays a graphical representation of the model.
- **Model Builder.** This window offers a selection of objects that can be added to the model.
- **Object Selector.** Thiw window lists the objects present in the model.
- **Input Editor.** This window allows for editing of keywords for a selected object; and
- **Output Viewer.** This window displays outputs for a selected object.

For your convenience, change the settings in the upper *Control Panel* as follows: Switch to 2D view and unclick the *"show axis"* and *"show grid"* features.
Also, adjust the base units to a setting that is convenient for your application via the *"units"* menu tab.

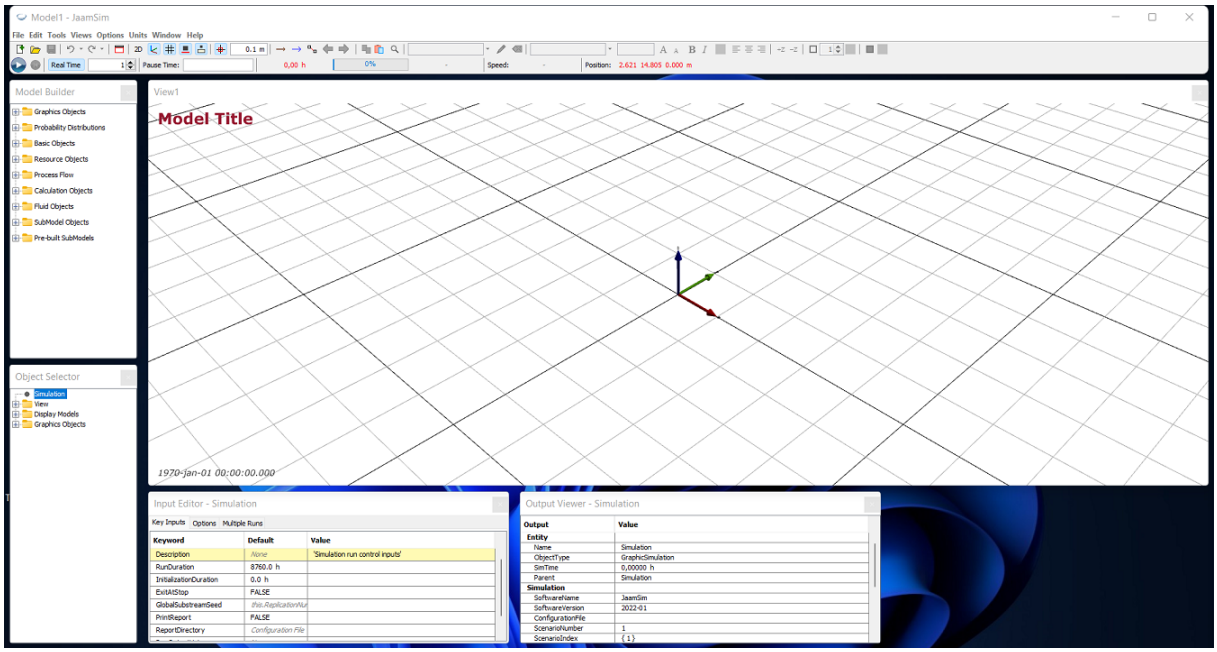


Figure 5: *The* JAAMSIM *start screen.*

8.3 Building a (single-server) drill press model

In the Model Builder, expand the *Process Flow palette* and then drag-and-drop a SimEntity into the View Window. This creates a SimEntity object with a default name (*SimEntity1*) and shape (*Sphere*) and automatically selects it, denoted by green highlighting as below. This object will serve as the prototype for entities that will be processed in the model. In the *Object Selector*, select SimEntity1 and press F2 to rename the object to *part*. Simarly, also create the following objects:

Model Builder Palette	**Object Type**	**Name**
Process Flow	EntityGenerator	Gen
Process Flow	EntityConveyor	GenToDrill
Process Flow	EntityProcessor	Drill
Process Flow	EntityConveyor	DrillToSink
Process Flow	EntitySink	Sink
Process Flow	Queue	DrillQueue
Resource Objects	Resource	ResourceDrill

An object can be moved by selecting it and using *CTRL + Left Click + Drag*.

Next, all objects placed must be set to interact with one another. Connect the objects together by setting the keyword values listed below. To set the values, click on the object, and complete the required fields in the *Input Editor window*. Alternatively, you can set the path of the entities by clicking the *create entity links* button in the control panel.

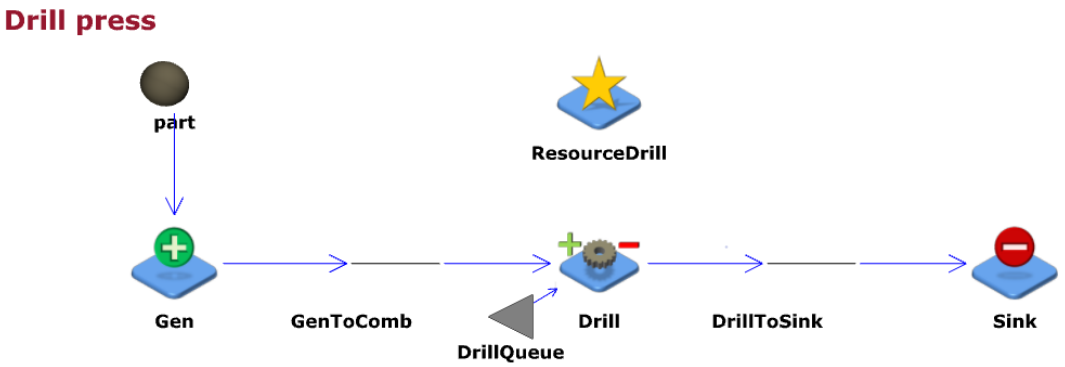


Figure 6: *Our basic* JaamSim *model.*

Object	Keyword	Value
gen	PrototypeEntity	part
gen	NextComponent	genToDrill
genToDrill	NextComponent	queueDrill
drill	WaitQueue	queueDrill
drill	NextComponent	drillToSink
drillToSink	NextComponent	sink

After positioning and connecting the objects, the model should look similar to Figure 6.

Now, use the *Input Editor* to set remaining inputs, i.e. the time at which new parts will enter the model by Gen, and the time that these parts will spent at each stage of the model[1]. We will start with a single-server model, meaning there is only one drill available which is seized by the parts to get processed afterwhich the drill will be released again to be used by a next part.

Object	Keyword	Value
gen	InterArrivalTime	2 s
genToDrill	TravelTime	1 s
drill	ResourceList	ResourceDrill
drill	NumberOfUnits	1.0
drill	ServiceTime	1 s
resourceDrill	Capacity	1
drillToSink	TravelTime	1.5 s

For convenience in the interpretation of the results, we will change the base time units to minutes instead of hours. To do this, click on *units >> TimeUnit >> min.* in the *Control panel.*

Save the model and press the *Play* button to run the simulation.

[1]Make sure to type a space between the value and the time unit.

The Real Time button in the *Control Panel* indicates that the model speed is restricted to the Real Time speed multiplier shown in the text box to the right of the Real Time button. The *Real Time speed multiplier* controls how fast the simulated time elapses in the model. The default Real Time speed multiplier is 1, meaning that each real (wall-clock) second corresponds to one simulated second. It can be changed by entering the desired *Real Time speed multiplier* into the text box, or by pressing the up or down arrow buttons (which double or halve the *Real Time speed multiplier factor* respectively).

To continue, Pause or Reset the model by selecting the corresponding buttons on the *Control Panel*. Pausing the model allows the simulation to resume later starting from the paused time, while resetting the model forces the model to start from time zero.

When running the model at this point, you will see that parts move through the system but no queues arise. Can you explain why this is the case?

Let us now add dynamic variability to the model by including a probability distribution to control the inter-arrival time of the part entities.

Create an exponential distribution object as listed below,

Model Builder Palette	**Object type**	**Name**
Probability Distributions	ExponentialDistribution	genIATDist

and set the keyword values for genIATDist as follows:

Tab	**Keyword**	**Value**
Key Inputs	UnitType	TimeUnit
Key Inputs	MinValue	0 s
Key Inputs	MaxValue	10 s
Key Inputs	Mean	2 s

Finally, update the Gen object to sample the genIATDist distribution for its inter-arrival time:

Tab	**Keyword**	**Value**
Key Inputs	InterArrivalTime	genIATDist

Save the model again and press Play. Observe that the part entities are now generated randomly, and that there are now times when part entities are waiting in DrillQueue to be processed.

8.4 Running the single-server simulation model

The *Simulation object* is used to store inputs that define basic parameters of the model, such as run duration. The Simulation object is created automatically and can be found in the *Object Selector* window.

After clicking the *Simulation object*, you can adjust the simulation parameters in the *Input Editor*. We will initialize the values as follows:

Tab	**Keyword**	**Value**
Key Inputs	RunDuration	10 min
Key Inputs	PrintReport	TRUE

Now, set the *Real Time speed multiplier* to a large value, e.g., 1000, save the model and press *play*. The simulation will stop automatically as soon as the run duration of 10 minutes is reached.

Once the simulation is finished, we can retrieve the .rep file in the same folder as from where we are launching the JaamSim application. You can open this file with any text editor (such as wordpad, notepad, notepad++[2],...). It contains a full overview of all statistics you might be interested in with respect to your simulation model. We here summarize the main results:

[2]I use and recommend this editor as it has a nice look and is easy to use. The editor can be downloaded freely via `https://notepad-plus-plus.org/`. Notepad++ is only available on Windows, however.

Object	Statistical accumulator	Value	Interpretation
gen	NumberAdded	304.0	In total, 304 parts entered the system.
sink	NumberAdded	302.0	From the 304 parts that entered, 302 left the system again within a 10-minute time interval. This means that 2 parts were still in the process at the end of the simulation. These parts could be on a conveyor, in the queue before the drill or in the drill press.
queueDrill	QueueLengthMinimum	0.0	The lowest number of parts in the queue was zero.
queueDrill	QueueLengthMaximum	4.0	At most 4 parts were waiting in queue simultaneously.
queueDrill	AverageQueueTime	0.008 min	The average waiting time in queue was equal to .48 seconds.
queueDrill	QueueLengthTimes[0]	8.22 min	From the 10 minutes that we simulated, there was no queue during 8.22 minutes.
queueDrill	QueueLengthTimes[1]	1.21 min	From the 10 minutes that we simulated, there was exactly 1 part waiting in queue for 1.21 minutes.
resourceDrill	UnitsSeized	303.0	303 of the 304 parts that entered the system seized a resource (i.e., started the process on the drill press).
resourceDrill	UnitsInUseAverage	0.505	The average utilization of the drill (denoted by ρ in the part on waiting line theory).

8.5 Multi-server drill press model extension

To extend our model to a multi-server variant, we need to alter only a few parameter values thanks to the fact that we already made use of the *EntityProcessor* and *Resource object*. Please change the capacities in the current model as follows (we assume 2 drills in parallel now):

Object	Tab	Keyword	Value
drill	Key Inputs	Capacity	2
resourceDrill	Key Inputs	Capacity	2

After running the model again, we obtain the following results:

Object	Statistical accumulator	Value	Interpretation
gen	NumberAdded	304.0	In total, 304 parts entered the system. *We did not change the generation process, so still the same amount of parts enter the system.*
sink	NumberAdded	302.0	From the 304 parts that entered, 302 left the system again within a 10-minute time interval. This means that 2 parts were still in the process at the end of the simulation. These parts could be on a conveyor, in the queue before the drill or in the drill press.
queueDrill	QueueLengthMinimum	0.0	The lowest number of parts in the queue was zero.
queueDrill	QueueLengthMaximum	1.0	At most 1 part was waiting in queue simultaneously. *This has decreased thanks to the additional server (drill) available.*
queueDrill	AverageQueueTime	5.06E-4 min	*The average queue time dropped to almost 0 minutes. Most of the parts did not have to wait, and if there was a queue it vanished very quickly.*
queueDrill	QueueLengthTimes[0]	9.85 min	From the 10 minutes that we simulated, there was no queue during 9.85 minutes.
queueDrill	QueueLengthTimes[1]	0.15 min	From the 10 minutes that we simulated, there was exactly 1 part waiting in queue for 9.2 seconds.
resourceDrill	UnitsSeized	303.0	303 of the 304 parts that entered the system seized a resource (i.e., started the process on the drill press).
resourceDrill	UnitsInUseTimes[0]	5.9 min	5.9 minutes, there was no resource active (i.e., the drill was empty).
resourceDrill	UnitsInUseTimes[1]	3.15 min	During 3.15 minutes, exactly one of the drills was working.
ResourceDrill	UnitsInUseTimes[2]	0.95 min	During 0.95 minutes, both drills were working.

8.6 Running multiple simulation runs

Recall that the simulation model that we run makes use of *random* numbers, sampled from a probability distribution. More specifically, the inter-arrival times for the parts entering the system are sampled from an exponential distribution.

It is quite likely that the simulation results, as presented in the previous Sections, will vary with different input values sampled from these probability distributions. In other words, if the inter-arrival times would be different (although still sampled from the same probability distribution), and therefore parts arrive at different moments in time, one might also expect a different waiting time in the system.

Because of the above-mentioned reason, it is not a good idea to rely entirely on the one simulation run that we just executed in order to evaluate the performance of your system.

In this Section, we will increase the number of runs in simulation model, to get a better view on the real performance of the drill press.

To allow for multiple runs, click on *simulation* in the *Object selector* and alter the value in the *Input editor* as follows:

Tab	**Keyword**	**Value**	**Explanation**
Multiple Runs	numberOfReplications	10	Put this value equal to the number of runs that you JaamSim to run for you.

Now, run the simulation model again. You will see that the simulation will start over again 10 times. Looking into the report, once the simulation has finished, you will now see detailed results per simulation run.

Exercise 8.1. Implement the scenario depicted in Exercise 3.1 in JaamSim. Assume a working day of 8 hours. You should not model breaks of the servers, so you can assume that all servers are available during the full day. Run the simulation for 20 days (runs) and compare the results obtained from the software with the values found in Exercise 3.1.

Exercise 8.2. Implement the scenario depicted in Exercise 3.2 in JaamSim. Assume a working day of 8 hours. You should not model breaks of the servers, so you can assume that all servers are available during the full day. Run the simulation for 20 days (runs) and compare the results obtained from the software with the values found in Exercise 3.2.

Exercise 8.3. Implement the scenario depicted in Exercise 4.1 in JaamSim. Assume a working day of 8 hours. You should not model breaks of the servers, so you can assume that all servers are available during the full day. Run the simulation for 20 days (runs) and compare the results obtained from the software with the values found in Exercise 4.1.

A Overview of the most important formulas

Single server model.

$$\begin{aligned}
\rho &= \frac{\lambda}{\mu} \\
L &= \frac{\lambda}{\mu - \lambda} \\
L_Q &= \rho L \\
W &= \frac{1}{\mu - \lambda} \\
W_Q &= \rho W \\
P_n &= (1-\rho)\rho^n
\end{aligned}$$

Multi-server model.

$$\begin{aligned}
\rho &= \frac{\lambda}{s\mu} \\
P_0 &= \left[\sum_{n=0}^{s-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^s}{s!} \left(\frac{1}{1-\rho} \right) \right]^{-1} \\
L_Q &= \frac{P_0 (\lambda/\mu)^s \rho}{s!(1-\rho)^2} \\
W_Q &= \frac{L_Q}{\lambda} \\
W &= W_Q + \frac{1}{\mu} \\
P_n &= \begin{cases} \frac{(\lambda/\mu)^n}{n!} P_0 & \text{for } n \leq s \\ \frac{(\lambda/\mu)^n}{s! s^{n-s}} P_0 & \text{for } n > s \end{cases} \\
L &= \lambda W
\end{aligned}$$