Business Process Simulation

Lecture notes 2II75

W.M.P. van der Aalst and M. Voorhoeve
Dept. of Mathematics and Computer Science,
Technical University Eindhoven,
Postbox 513, 5600 MB, Eindhoven.

Preface

These lecture notes are based on a document (Handboek simulatie) in Dutch from the first author. It has been translated, revised and adapted as course material by the second author.

The examples and exercises are based on the Arena package [5].
A downgraded student version of Arena can be downloaded from the directory //campusmp/software/rockwell. This version suffices to view, modify, create and simulate the models of this course. The Arena example models in the exercises can be found at www.win.tue.nl/~mvoorhoe/sim.

The simulation course is intended for systems engineering students with modeling experience and a basic knowledge of probability theory and statistics. Students are expected to be have some experience in Petri net modeling, as e.g. taught in the course 2V060 (systems modeling 1). A recapitulation of the principles can be found in Appendix A.

Contents

1 Introduction

Suppose you run a discotheque and have problems in deploying staff on Saturday nights. At certain times there is too much staff capacity, however customers complain about long waiting times for getting their coats hung up and ordering drinks. Because you feel you are employing too much staff and yet face the threat of losing customers due to excessive waiting times, you decide to make a thorough investigation. Examples of questions you want answered are:

- What are the average waiting times of customers at the bars and the cloakroom?
- What is the occupation rate of the bar staff?
- Will waiting times be reduced substantially if extra staff is deployed?
- Would it serve a purpose to deploy staff flexibly? (e.g. no longer assigning staff members to one bar only)
- What is the effect of introducing refreshment coupons on average waiting times?
- What are the effects of introducing a 'happy hour' to spread the arrivals of guests?

Simulation
Simulation model

To answer these and similar questions, *simulation* can be used. A model is built that reflects reality and is used to simulate that reality in a computer. In the same way that an architect uses construction drawings to understand a building, a systems analyst may use simulation models to assess a business process.

Reasons for simulation

When is simulation appropriate? Some reasons are:

- Gaining *insight* in an existing or proposed future situation. By charting and simulating a business process, it becomes apparent which parts are critical. These parts can then be examined more closely.
- A real experiment is *too expensive*. Simulation is a cost-effective way to analyze several alternatives. Trial and error is not an option when it comes to hiring extra staff or introducing a refreshment coupon system. You want to make sure in advance whether a certain measure will have the desired effect. Especially when starting up a new business process, simulation can save a lot of money.
- A real experiment is *too dangerous*. Some experiments cannot be carried out in reality. Before a railway company installs a new traffic guidance system, it must assess the safety consequences. It must be noted, however, that simulation alone cannot address safety issues. The safety itself must be addressed by formal analysis techniques, whereas simulation can assist in assessing e.g. the performance. The same holds for other processes where safety is critical (e.g. aviation or nuclear reactors).

Sometimes, rather than using simulation, a mathematical model, also called an analytical model is sufficient. In Operations Research (OR) many models have been developed which can be analyzed without simulation, such as queuing models, optimization models or stochastic models. It is advisable to address a simplified version of the problem at hand by an analytical model and compute its characteristics before conducting a simulation study on the full problem. The analysis gives extra insight from which the simulation study can profit. Also, the analysis results provide a reference for the expected simulation results, so that simulation errors can be avoided as much as possible. If the outcome of the simulation experiments deviates from the computed characteristics, one should find reasons for the difference.
In this course, we shall discuss some simple analysis methods that can often be used to support simulation. More advanced analysis methods, requiring considerable knowledge and effort, can and should be used in certain cases. Strong points of simulation versus analysis:

Advantages of simulation

- Simulation is flexible. Any situation, no matter how complex, can be investigated through simulation.
- Simulation can be used to answer a variety of questions. It is possible to assess e.g. waiting times, occupation rates and fault percentages from one and the same model.
- Simulation is easy to understand. In essence, it is nothing but replaying a modeled situation. In contrast to many analytical models, little specialist knowledge is necessary to understand the model.

Simulation also has some disadvantages.

Disadvantages of simulation

- A simulation study can be very time consuming. Sometimes, very long simulation runs are necessary to achieve reliable results.
- One has to be very careful in interpreting simulation results. Determining the reliability of results can be very treacherous indeed.
- Simulation does not offer proofs. Whatever occurs in a correct simulation model may occur in reality, but the reverse does not hold. Things may happen in reality that have not been witnessed during simulation.

Simulation tools

For the construction of a simulation model we use a *tool* , which ensures that the computer can simulate the situation charted by the model. Tools can be languages or graphical packages.

Simulation languages A *simulation language* is a programming language with special provisions for simulation, such as Simula.

Simulation packages A *simulation package* is a tool with building blocks allowing the assembly of a simulation model. There are special-purpose packages for certain application areas, such as Taylor II, (production) and more general packages like Arena.

The advantage of a simulation language is extreme flexibility, but at the cost of clarity and a considerable programming effort. The use of a simulation package corresponds to graphical modeling with building blocks that need to be parameterized. The more general (and flexible) the package, the more parameters are needed.
Often it is desirable to combine various techniques, such as a Taylor manufacturing model with a language-based scheduling algorithm. By means of *conceptual modeling*, insight is obtained in the various aspects of the system that needs to be modeled. From such a conceptual model, a simulation tool can be selected and used to construct the executable simulation model.

Photo example

A civil servant creates documents for citizens, who are queueing for his help. On average, a new citizen needing help arrives every 5 minutes. A photo must be attached to the document, which is judged by the servant. The judgement takes 1 minute on average. In 40% of the cases, the photo is judged inadequate, upon which the citizen is deferred to a nearby photographer to make a new one. This advice takes on average 1.5 minutes, after which the next citizen is helped. It takes on average 20 minutes to obtain a new photo; when the citizen returns with it, he gets priority for his document and the photo is immediately judged correct. Making the document takes on average 3.5 minutes.

Exercise

The Arena model gemeente.doe (at www.win.tue.nl/~mvoorhoe/sim) contains a model of the photo example sketched above.
You are asked to conduct a simulation to assess the occupation rate of the civil servant. Open the Arena file and press the ▶ (run) button and observe the behavior. When you have seen enough, press the ▶▶ (fast-forward) button. When the simulation is completed, you are looking at the state after 1000 hours of simulation. To find the requested occupation rate, open the Reports tab at the left and then the Resources report. Under the instantaneous utilization Inst Util column, the answer (0.97) can be found. Why is this number incorrect? Is it useful at all to simulate the sketched situation?

2 Conducting a simulation project

Simulation is often used to support strategic decisions. This requires a proper project-oriented approach, consisting of problem analysis, model construction, simulating and interpreting the results as illustrated in Fig 1). User-friendly tools may allow rapid prototyping, but it is dangerous to base important decisions on prototypes.

Phasing

An often used "waterfall" approach that illustrates the *phasing* of a simulation project is depicted in Figure 1. Note that a phase can start before completing its predecessors and that it can be influenced by its successors. We give a short description of every phase.

Problem definition

The simulation project starts with a *problem definition* , describing the goals and fixing the scope of the simulation study. The scope tells what will and what will not be a part of the simulation model. The problem definition should also state the questions to be answered. These questions should be quantifiable and focus on selection rather than optimization. Instead of asking "what is the best bar

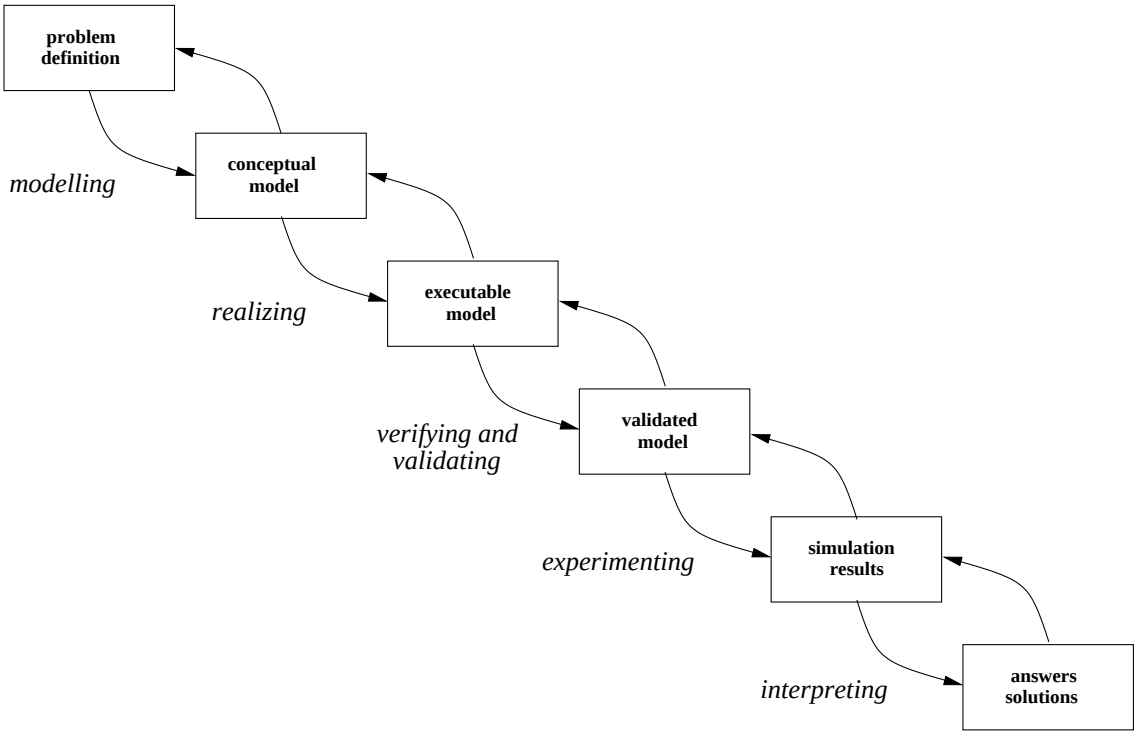

Figure 1: Phases of a simulation study.

schedule to satisfy customers", one should ask "How long do customers have to wait on average with schedule A/B/C?", where A,B,C are scheduling policies given in advance.

During the problem definition phase, one should decide whether a simulation approach is asked for in the problem to be studied. By simplifying the problem, some of its characteristics can be computed directly. These computations can be used as a feedback for the questions to be answered, identifying e.g. bottleneck resources. Also, deviations of the "real" problem from the simplified version should be assessed. This assessment can help to decide between a simulation or mathematical analysis approach. In the sequel, we assume that simulation has been chosen. The initial analysis will help in validating the simulation results.

Modeling Conceptual model

After defining the problem, the next phase is *modeling*. In this phase the *conceptual model* is created. The conceptual model defines classes of *objects* and the *relations* between these objects. In the case of the discotheque the objects to be distinguished are a.o. staff members, bars, cloakroom and customers. The relevant characteristics (properties) of these objects are charted. We distinguish between *qualitative* and *quantitative* characteristics. Examples of qualitative characteristics are queue disciplines (How are customers attended to, e.g. in order of arrival (FIFO - First-In-First-Out) or randomly? (SIRO - Select-In-Random-Order)), sequences (What actions are performed in what order when a customer enters?) and decision rules (When do we open an extra bar?). Quantitative characteristics of objects describe primarily speed (How many customers can be served per minute?) and capacity (How many customers can be served simultaneously?). If we are dealing with

objects of the same class, we specify these characteristics for the entire class (parameterized if necessary). Graphs can be drawn for showing connections between the various objects or object classes. Suitable techniques are situation diagrams, data flow diagrams or simulation-specific flow diagrams.
In section 3, we treat the construction of a conceptual model based on Petri nets and the translation into an Arena executable model. Petri nets are *concurrent*, which is of prime importance in simulation. The models are based on *cases* (called *entities* in Arena), of which there may exist several classes. In the discotheque example, an individual customer is modeled as a case. Each case has a life cycle; it is created (the customer arrives), is involved in various actions (such as ordering drinks) and finally is disposed of (leaves). A case may contain *attributes*, that may be set at creation and altered by the actions. What actions are performed may depend on *decisions*, which may be based on attribute values of the case (or global parameters). Some actions require the presence of *resources* (e.g. a bartender). Some resources may also have a life cycle, as we shall see. Different cases behave *concurrently*: they influence each other's behavior mainly through the claiming and releasing of resources. The conceptual model should be *timed*, in order to measure waiting times of cases and occupation rates of resources. *Verification* of the conceptual model is advisable, for example by deriving place invariants for resources.
The construction of the conceptual model will most likely unveil incomplete and contradictory aspects in the problem definition. Also, the modeling process may bring forth new questions for the simulation study to answer. The problem definition should be adjusted accordingly.

Realization
Executable model

After conceptual modeling, the *realization phase* starts. Here, the conceptual model is mapped onto an *executable model*. The executable model can be simulated on the computer. How to create this model depends strongly on the simulation tool used. Simulation languages require a genuine design and implementation phase. Simulation packages, allow a more straightforward translation of the conceptual model into building blocks, to which the proper quantitative characteristics (e.g. speed) must be added.

Verification

An executable model is not necessarily correct, so it has to be *verified*. Verification of the model is necessary to examine whether the model contains qualitative or quantitative errors, like programming errors or wrong parameter settings. For verification purposes, trial runs can be observed and its results assessed, or a *stress test* can be applied to the model. In the stress test, the model is subjected to extreme situations, e.g. arrival of more customers than can be attended to. In such cases, queues should rapidly grow in the course of time. Some tools support more advanced forms of verification, e.g. proving invariance of resources or absence of deadlock.

Validation

Apart from verification, *validation* of the model is required, comparing the model to the actual system. It is good practice to model the "as-is" system and demonstrating trial runs to experts to elicit comments from them. In addition, the simulation results are compared to observed historical and analysis data. Mismatches should be accounted for and the model adapted if necessary. Once a match is established, adaptations can be made to model the possible future situations.
Verification and validation may lead to adjustments in the simulation model. New

insights may even lead to adjusting the problem definition and/or the conceptual model. A simulation model found to be correct after validation is called a *validated model*.

Validated model

Experimenting

The next phase is conducting *experiments*. These experiments should efficiently obtain reliable results. A simulation experiment is based on a validated model to which the computer adds random samples from the specified probability distributions while the simulation proceeds. Quantitative results are accumulated that are returned upon completion. It is standard practice to replicate a simulation run with new (and different) random samples in order to assess whether their results agree. In many cases, this is achieved by dividing a run into subruns. Important decisions during this stage are the run length and the division into subruns.

Interpreting

The simulation results will have to be *interpreted*, to allow feedback to the problem definition. Reliability intervals will have to be calculated for the various measures gathered during simulation. The statement
customers have to wait on average two minutes before being served
is not acceptable; the correct statement would be e.g.
with 95% reliability, the average waiting time lies between 110 and 130 seconds.
If the subruns are in agreement, the reliability will be high. Also, the quantitative results will have to be interpreted to answer the questions in the problem definition. For each such answer the reliability should be stated as well. If the reliability is too low to allow for any definite answers, new experiments with longer runs are required. These matters should be summarized in a final report containing answers to questions from the problem definition and solution proposals.

Answers and solutions

Figure 1 shows that feedback is possible between phases. In practice, many phases do overlap. Specifically, experimentation and interpretation will often go hand in hand.

Alternatives

What-if analysis

Figure 1 assumes the existence of a single simulation model. Usually, several *alternative situations* are compared to one another. In that case, several simulation models are created and experimented with and the results compared. Often, several possible improvements of an existing situation have to be compared through simulation. We call this a *what-if analysis*. In such a case a model of the current situation is made first. For this model the phases of Figure 1 are followed. The model is then repeatedly adjusted to represent the possible future situations. These adjustments may just concern the executable model (e.g. by changing parameters). In some cases (e.g. when changing control structures), the conceptual model is affected too. In each case, the adjustments should be validated. The different alternatives are experimented with and the results compared to indicate the expected consequences of each alternative.

User

Systems analyst

The people involved in a simulation study have their specific responsibilities. In the first place, there are *users*: the persons confronted with the problem to be investigated. Secondly, there is a *systems analyst* , responsible for writing a clear problem definition. The analyst also creates the conceptual model. Depending on

Programmer

the tools used, the systems analyst can be supported by a *programmer* to realize the simulation model. The number of simulation experiments often dictates who should conduct them. If the experiments have to be conducted regularly, e.g. for supporting tactical decisions, a user seems appropriate. If it concerns a simulation study supporting a one-time strategic decision, the systems analyst or programmer is preferred. For the correct interpretation of the simulation results, it is important that the persons involved have sufficient knowledge of the statistical aspects of simulation.

Animation

The builders of a simulation model (the systems analyst and the programmer) in general differ from the "users" (system experts). Their communication is vital to the simulation project, though. One way of promoting user involvement is by *animation*. Animation is the *graphical* simulation of the modeled situation from a simulation model, e.g. by moving objects that change shape. In this way the simulation can be made to look like reality. Animation is a useful tool for obtaining user validations of a model, but it cannot replace a proper simulation approach when it comes to accepting the conclusions.

Exercise

Open the model gemeente.doe at www.win.tue.nl/~mvoorhoe/sim. The civil servant has been instructed to pass a less harsh judgement on the photos offered. At present, only 25% of the photos are rejected. Modify the model by selecting and opening the photo_ok block and changing the parameter Percent True from 60% to 75%. We want to replicate our simulation; press the Run tab at the top and select Setup ... from the pull-down menu. Under the Replication Parameters tab, change the Number of Replications value to 9 and the Warm-up Period to 50. Press OK and simulate. Upon termination, open the Reports tab at the left, select Category Overview and open the Passport/Queue subdirectory. You find listed that the average waiting time for citizens is 98 minutes, with half width 64 (which means that the probability of an average waiting time between 98-64 and 98+64 minutes is 95%), whereas the subrun averages were between 23 and 277 minutes. How do you interpret these simulation results?

Case study: ferry simulation

We will use an example connected to traffic simulation to illustrate the concepts sketched here. Here follows the problem definition, extracted from interviews and observations.

Ferry example

A small ferry boat carries cars across a river. At each bank, cars can appear needing to cross. When the ferry is at a certain bank and is empty, cars can get on it. There are 10 places for cars at the ferry. If the ferry is full or if all cars at the current bank got on board, the ferry crosses to the other bank, where the cars get off. The cars waiting at the other bank can then enter the ferry, starting a new cycle.
The current ferry needs replacement; it is at the end of its life cycle, there are complaints about long waiting times and traffic is expected to increase. The replacement candidates are a (faster) one with a capacity of 10 places and a (slower) one with a capacity of 15 places. Find out through simulation for the current ferry and both candidate replacements what

the average waiting and throughput times of cars are for the present situation and when traffic increases by 10%.

Parameters Estimate
Data gathering

Before even starting the modeling activity, one must worry about obtaining the necessary parameters. Many decisions are based on an estimate of the number of cases to expect. If the estimate is incorrect, the decision may be wrong. Also, there must be quantitative data e.g. about the duration of actions. If too much guesswork is needed, the simulation (or analysis) effort will be wasted.

Exercise

You are able to use video cameras to obtain data about the ferry system. The installation of cameras may depends on which data should be obtained. Describe the data required for the ferry simulation, their importance and how to obtain them.

3 Problem definition and analysis

In this phase, the scope of the simulation study should be clearly defined and the questions needing an answer should be addressed. For the ferry case, it seems probable that traffic density is not constant during the day. How should we interpret "average waiting time"? Is it acceptable to have longer waiting times during rush hour if this is compensated by shorter ones at other times? How frequent are exceptional situations and how often can they arise? Can and should the simulation study address (some of) these situations? These and similar questions should be answered and agreed upon . For example, it can be decided that the simulations should be carried out based on busy traffic densities that regularly occur (e.g. the average morning rush on workdays).

Defining the scope

In addition to clearing up these matters, a "common sense" analysis is carried out.

Cycle time

In the ferry case one observes that an important concept is the *cycle time* , the time needed to complete a cycle between two successive arrivals at a certain bank. The faster ferry has the shortest cycle time; crossing the river takes less time and fewer cars are unloaded and loaded. If traffic density is low, the average waiting time of a car is half the cycle time. In this case, the faster ferry is preferable: not only the average waiting time, but also the time needed to cross the river is shorter.

Capacity

With higher traffic densities, the probability increases that more cars are waiting than the capacity of the ferry, in which case one or more cycle times are added to the waiting time. This points at another important concept: the *capacity* of the ferry, the maximum number of cars per time unit that can be transported. The capacity of a ferry more or less equals the number of cars it can carry multiplied by its speed. If the speed of the slower ferry is less than 67% of the speed, then the capacity of the faster ferry exceeds that of the slower one. No simulation study is required in this case, since the faster ferry is always better. In the other case, there exists a pivotal queue length q. Cars arriving in a queue of length less than q can expect a better performance from the faster ferry, whereas the slower one will perform better for queues of greater length.

We assume that the slower boat has indeed the greater capacity. From interviews, we gather that capacity matters, since the building up of queues is not uncommon. In this respect, it is important to notice that the waiting time distribution with

the faster ferry is more "spread out". When queues are absent, the faster ferry has shorter waiting times; on the other its lower capacity can lead to the building up of queues with longer waiting times. The *standard deviation* (or its square the *variance* of the waiting time distribution is a measure for its spread. In appendix B, the basic concepts from probability theory and statistics are treated. Probably, the average throughput time is not the best indicator to compare solutions. Users of the ferry will prefer a slightly longer average waiting time if extremes are avoided, which suggests using e.g. the sum of the average and standard deviation for comparison. About 85% of the cases, the time needed to cross will not exceed this amount.

Importance of variance

We thus start our simulation study, taking good notice of the concepts like cycle time and capacity that have crept up during our analysis. The following questions are to be addressed.

- Determine for each bank the current average numbers X_A, X_B of arrivals per time unit during the morning rush (working days between 7:00 and 9:00).
- For each alternative ferry, determine by simulation the average and standard deviation for the throughput time (i.e. the waiting time plus crossing time) of cars at each bank during the morning rush.
- Determine the same characteristics based on a 10% traffic increase (i.e. replacing X_A, X_B by $1.1 * X_A, 1.1 * X_B$ respectively).

4 Modeling

4.1 Conceptual Modeling

Conceptual modeling immediately follows and supports problem definition by clarifying the entities and events that need to be studied. A good way to start the conceptual model is by identifying the object classes. In simulation models, there is a marked distinction between *cases* and *resources*. Cases (called "entities" in Arena) have a life cycle: they are created, are involved in various actions and finally disposed of. Cases can be products to be manufactured, customers requiring certain services or traffic participants (e.g. cars) needing to get from A to B. Resources have a more permanent character; they are temporally needed for certain actions and as such they are seized and released. Examples of resources are machines, employees or road stretches.

Case
Entity
Action
Resource
Seize
Release

The first conceptual model starts by listing the cases, resources and actions and the connections between them. This table is converted into a Petri net. Principles of net modeling suggest a simple initial model that approximates the desired behavior. We first model the "case workflow" is modeled as a state machine net. Places are used to contain the various cases; a place is added for each state of the case. In addition, places are added for resources. Every action is represented by a transition. The action takes a case from one state to a next (maybe the same) state, which is modeled by a "case" consumption and production arc. Likewise the seizing and releasing of resources is modeled by respectively consumption and production arcs. It is good practice to keep models simple initially and then stepwise refine them.

Net modeling

Case workflow
Resource behavior

4.1.1 Ferry example

For the ferry case, our cases are cars, of which there are two kinds: one for either side of the river (banks A and B). To cross, the cars need the ferry, more specifically a free place on the ferry. So a free place is a resource. A car can seize a free place only under specific circumstances, when the ferry is at the car's bank and the cars that just crossed have left. We shall deal with that problem later.

Divide and conquer

Summarizing, we obtain the following diagram, which is converted to the net in Figure 2. Since crossing the river takes considerable time, it is preferred to model getting on and off board as separate actions.

case	action	resource	kind
car1	arriveA		create
	getonboardA	free place	seize
	getoffboardB	free place	release
	leaveB		dispose
car2	arriveB		create
	getonboardB	free place	seize
	getoffboardA	free place	release
	leaveA		dispose

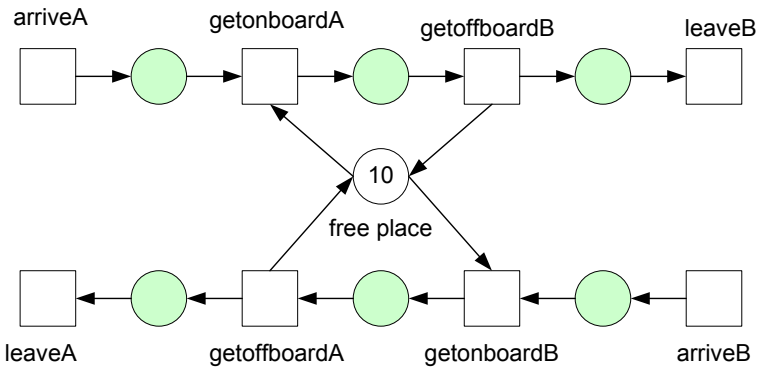

Figure 2: First ferry conceptual model.

Problem revision

As mentioned in the previous section, analyzing the basic modeling concepts may lead to a revision of the problem definition. The modeler must be aware of this possibility and consult the user if it should occur.

We notice the problem stated earlier: a free place cannot be seized anytime. A car at bank A can only seize a free place when the ferry is at bank A and it can only release it when the ferry has crossed and is at bank B.

Restricting the behavior

The firing of the transitions in Figure 2 must be further restricted. The way to achieve this in Petri net modeling is by adding extra places and arcs. The new places represent states of the ferry, respectively at bank A and at bank B. We add ferry transitions A2B and B2A to toggle the states and bidirectional arcs for the getonboard/getoffboard transitions. The resulting update net is given in Figure 3.

The modification still needed is to ensure that upon arrival, first all cars on it leave the ferry, then all cars needing transport (or the first 10 of them) enter it, before the ferry leaves the bank. Such features cannot be realized by classical Petri nets;

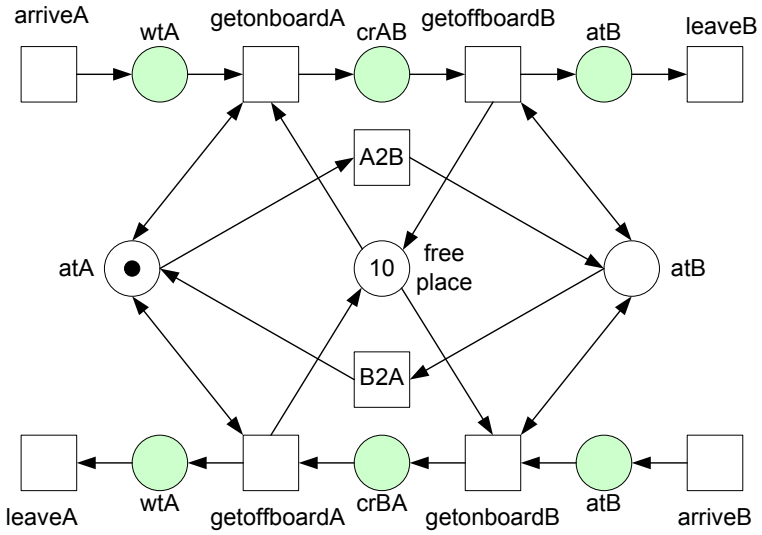

Figure 3: Second ferry conceptual model.

Priority

some extra feature is needed. Arena possesses a *priority* concept and this fits the bill perfectly. The getting off board transitions have the highest priority, followed by getting on board and finally setting sail. The ferry thus leaves when all possible firings of the getonboard transitions have occurred.

Verification

Resource invariant

Deadlock avoidance

Conceptual models can be *verified* by computing place invariants. Every resource should correspond to a place invariant. Our ferry model indeed satisfies freeplace + crAB + crBA = 10 and atA + A2B + atB + B2A = 1. Also, *deadlock-freedom* can be analyzed. If a conceptual model is not deadlock-free, the parameters (e.g. durations) may prevent the deadlock from occurring, but it is dangerous to depend on them. By adding places and arcs (i.e. by restricting the behavior) deadlocks can be avoided.

4.1.2 Photo example

We repeat the description of the model.
A civil servant creates documents for citizens, who are queueing for his help. On average, a new citizen needing help arrives every 5 minutes. A photo must be attached to the document, which is judged by the servant. The judgement takes 1 minute on average. In 40% of the cases, the photo is judged inadequate, upon which the citizen is deferred to a nearby photographer to make a new one. This advice takes on average 1.5 minutes, after which the next citizen is helped. It takes on average 20 minutes to obtain a new photo; when the citizen returns with it, he gets priority for his document and the photo is immediately judged correct. Making the document takes on average 3.5 minutes. The citizens complain about excessive waiting times. The city hall management wants to solve the problem. Two solutions are proposed:
1. Every citizen should have his photo taken at the preferred shop.
2. The judgement should be less harsh.
The alternatives should be compared by simulation.

Here we have one case class (citizens), with different paths, depending on whether

the photo is accepted or not. The actions are entering, judgement, document-making and (if the photo is rejected) instruction, making a new photo and returning. The initial table is as follows.

case	action	resource	kind
citizen	arrive	-	create
	decide	servant	seize
	make-doc	servant	release
	instruct	servant	release
	new-photo	-	
	return	servant	seize
	leave	-	dispose

State machine

When modeling the case workflow as a state machine net, it should be noted that the state between decide and instruct and the state between return and make-doc differ. So the acceptance of the photo and starting the make-doc action represents a state transition. We therefore add a dummy transition accept, going from a state ok?, where a choice is possible to a state ok! leading to make-doc, which is the same state after return. In state ok?, there is an option to continue by accept or instruct. The resulting net is given by Figure 4.

Adding transitions

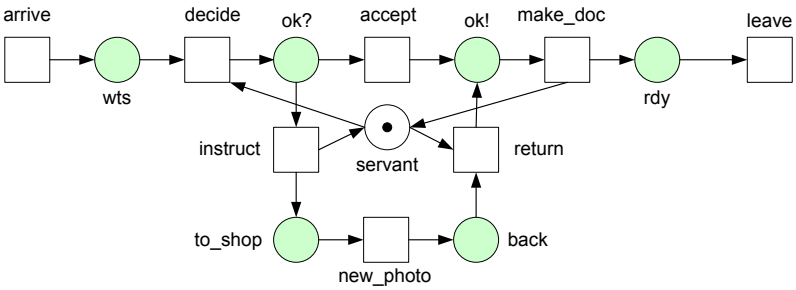

Figure 4: The photo conceptual model.

Priority

In Arena, a priority rule will be added: if both return and decide are enabled, the former gets priority for the servant resource. Verification shows resource invariance ($\mathsf{ok?} + \mathsf{ok!} + \mathsf{servant} = 1$) and freedom of deadlock.
It is possible to create an equivalent model without branching, by distinguishing between citizens with and without acceptable photo. The two classes follow a different path, as shown in Figure 5.
The models in Figs 4,5 are trace equivalent, giving equivalent simulation results.
The modeling of different life cycles for different classes of citizens has the advantage that possible dependencies are made explicit. For example, the make_doc duration may depend on the kind of photo offered. For example, the average duration of document making may be 3.1 minutes for new photos and 4.1 minutes for approved old photos. It is possible to adapt the branching model (Fig 4) if this happens to be the case. The solution is to add an *attribute* to each case that reflects whether the photo was accepted or not, influencing the processing time for the make_doc transition. The non-branching model (Fig 5) will be simpler, as there are two different make_doc transitions. No attributes have to be assigned in this case.

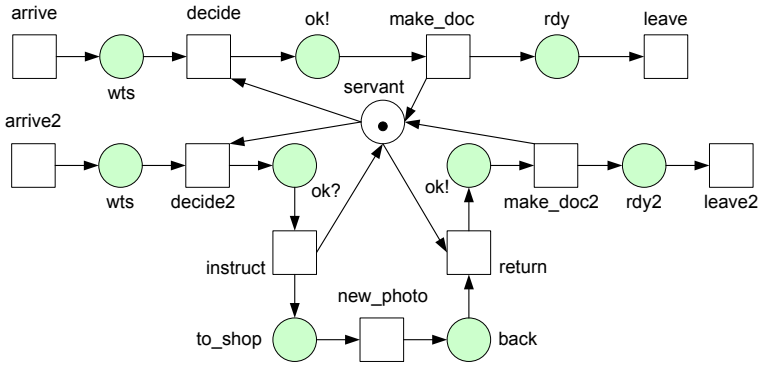

Figure 5: Alternative photo model.

Also, with the branching model there is a higher risk of not noticing the sketched phenomenon, which can have bad consequences for the advice backed up by simulation that will be eventually given. Without differentiation in the document making, the simulation might recommend the second alternative (less harsh judgement). However, in reality the resource time gained by less advice-giving (1.5 minute) in the second alternative is lost for two thirds by the extra time needed for document-making, whereas the first alternative has the added benefit of shorter document-making. This might tip the scale in favor of the first alternative. In a non-branching model, that risk is reduced, as it becomes more natural to obtain a correct parametrization.
Of course, for a large number of different paths (e.g. loops), a non-branching modeling approach is not possible anymore.

Exercise

Make conceptual models for the following situations. Make sure that resources define place invariants, keep an eye on possible deadlocks and invent ways to avoid them.

1. A hospital has three treatment rooms and five doctors. Patients needing treatment go to the waiting room and wait there until a doctor is available. The patient and the doctor then have a consultation. In 50% of the cases, a treatment is required. In this case, the patient and doctor need a treatment room to finish the consultation. In the first model, the doctor and patient wait together.
   Make a second model, where the patient waits alone. If the room is available, the patient is installed there (e.g. by a nurse) and the doctor joins him (with priority) when he is available again.

2. An operating system allows simultaneous execution of applications and has 250MB of free memory at its disposal. When starting, an application needs 20MB of memory. In 80% of the cases, a running application will claim another 10MB up to a maximum of 40MB. In the other 20% (or if 50MB has been claimed) termination will occur and the claimed memory is released.

3. A factory fabricates two types of products, X and Y and has (multiple) machines of types A, B and C respectively. Orders for product X have to pass through A and then C. Orders for Y have to pass through B and then C.

   Extension 1:
   A machine can break down; if this occurs it needs to be repaired before it can resume production. Repairing a machine requires a repairwoman, who repairs breakdowns in FIFO order.
   Hint: Model the breakdown of a machine resource as an "evil" case that seizes it with high priority.

   Extension 2:
   The machine types A and B are different configurations of the same machine. It is possible to reconfigure a machine from type A into type B and vice versa. This takes some time, but it may improve overall performance.

4.2 Implementing the conceptual model

The implementation of a conceptual model in Arena is in principle straightforward. It is important to distinguish between *case* and *resource* places. In our net, the case places are shaded in order to distinguish them from the resources. In the Arena model, the flow of control of a case is depicted by a model that only contain case places.

Create
Process
Dispose
Intensity

The Arena versions of the actions are shown in Figure 6. They represent the Create, Process and Dispose blocks. The Create blocks generate cases, according to parameters that need to be specified, the most important one being the *intensity*, the expected time between arrivals. We connect the "output" side of an action (a small

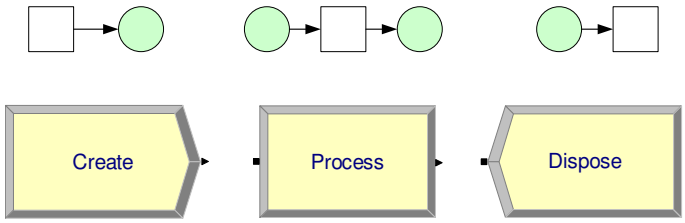

Figure 6: Simple Arena building blocks.

black triangle) with the "input" side of its successor (a square). The Arena flow of control is left-to-right, which replaces the arrowheads. The Process blocks possess a duration parameter and parameters w.r.t. their resource behavior. These blocks are the most frequently used. Some models (where all cases follow the same path) can be fully modeled by them.

4.2.1 Ferry example

The conversion of the ferry model in Figure 2 is rather straightforward.

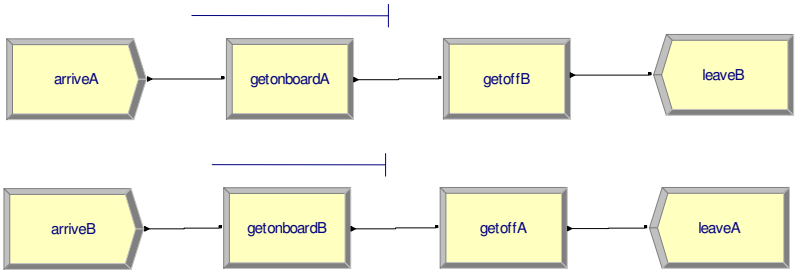

Figure 7: Ferry executable model (simplified).

Delay

Seize-delay

Delay-release

Seize-delay-release

Note that the default Process block is of the "delay" variety, which means that it neither seizes nor releases resources and only lets time pass. The getonboard blocks are modified to the "seize-delay" variety, adding a resource to be seized (free place). In doing so, a queue is added in the model, as cases may have to wait for the resource. Likewise, the getoffboard blocks are modified to the "delay-release" variety, releasing the free place resource as well.

The result is shown in Figure 7. Note that the seizing takes place before starting and the release after termination. There also exist "seize-delay-release" blocks, who seize and release the resources. Seized and released resources are in fact *bags* (various resource types with multiplicities) just like Petri nets. A seize-delay-release block must seize and release the same resource bag.

The model in Figure 3 cannot be converted to Arena directly. Resources cannot be seized and released independently; there should be actions, which requires a case. So we invent a new case type, which we call "ferry". We create only one ferry case and never dispose of it. We split the atA and atB places into resource and case places. Immediately after creation, the ferry seizes one of the resources (with high priority) and then the cycle starts. Also, the A2B and B2A transitions seize and release different resources, so they must be split as well, into setsail and berth process

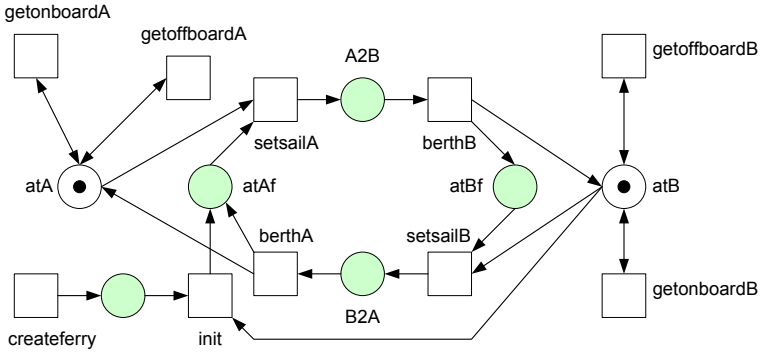

Figure 8: Adding the cyclic ferry case.

blocks. This gives the model indicated by Figure 8 (parts that have remained the same have been omitted)

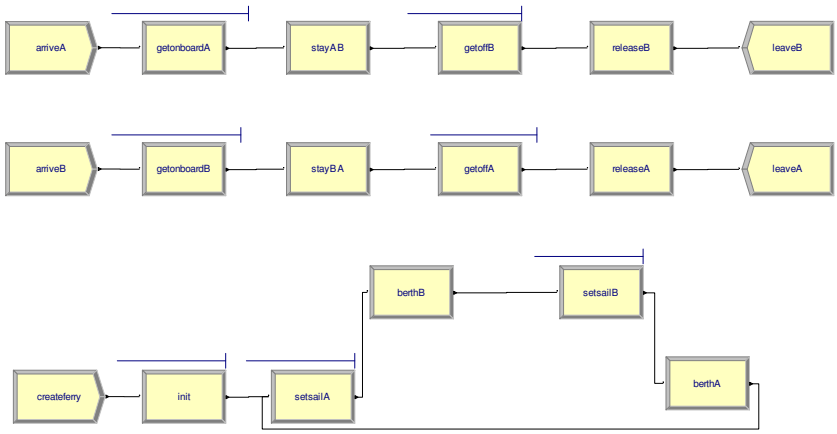

Figure 9: Arena ferry model.

In the net, the getonboard transitions seize a free place and an atA or atB resource and only release the latter. Similarly, the getoffboard transitions seize an atA or atB resource and release it together with a free place resource. Arena does not allow this, so these transitions must be split also.

Validation

The resulting Arena model is shown in Figure 9. We can add parameters and do some trial runs to look for mistakes. We now show the running model to users for validation. The events on the screen should be interpreted and the users should check whether they actually can occur in practice. Users should also comment about common phenomena in practice not occurring in the simulation (or vice versa). Sometimes, the parameters can be adapted to match the model to the modeled situation, otherwise the model should be adapted.

4.2.2 Photo example

Decide

With the blocks in Figure 6, resource conflicts (contention) can be modeled, but not alternative routings of cases, such as in Figure 4. This requires the Decide block, which has one input and two or more outputs. The simplest version randomly divides

the case flow, where a percentage of "true" outcomes must be specified. This version of the Decide block can be used for our photo example. We specify that the outcome "true" (straightly leading to make-doc) accounts for 60% of the cases. Together with the other blocks, this leads to the model in Figure 10.

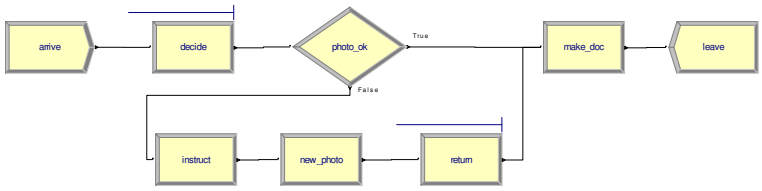

Figure 10: Photo executable model.

The non-branching model in Figure 5 can be simplified by replacing "seize-delay-release" transition sequences by single Process blocks, resulting in Figure 11.

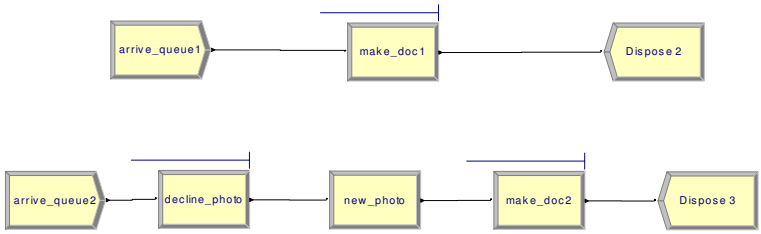

Figure 11: Alternative photo model.

Instead of generating one type of citizen with expected intensity of e.g. 5 minutes following paths A,B that split according to a 60/40 ratio, we generate A and B citizens with expected intensity of 8.333 and 12.5 minutes respectively. Thus, in 50 minutes 6 A-citizens and 4 B-citizens are expected, making a total of 10. The non-branching model is preferable; as a rule-of-thumb, random-choice Decide blocks should be used with care.

Other versions of the Decide block have a better use. For example, a decision can be made based on boolean expressions containing attribute values of the case being considered or global variables (or both). With Assign blocks, attributes and variables can be set or modified upon the passage of a case. This allows e.g. the programming of scheduling algorithms.

Assign

It is possible to show user-specified quantities in the simulation report by the Record block. This feature can e.g. be used to obtain the variance of throughput times, which is not part of the standard Arena report.

Record

Exercise

Model the conceptual models of the previous exercise into Arena.

1. Make Arena models for the hospital and production systems (extension 1). Add intensities and durations in such a way that the resource occupation rate is about 80%. Conduct some simulations to verify whether this actually is the case.

2. Make an Arena model for the deadlocking version of the operating system with parameters such that the deadlock actually occurs in the simulation. Also model and simulate the deadlock-free version with the same parameters.

3. Make an Arena model for the operating system that can be parameterized with any maximum memory size N instead of 50MB. Claims still occur in batches of 10MB. Make also a version that allows a process to release memory batches while running instead of at termination only.
   Hint: Use the amount of memory claimed as a case attribute and a decision rule for releasing.

4. Make a model for extension 2 of the production system with the following rule: *If no cases are waiting for resource A and the queue length for cases waiting for resource B exceeds N, a machine of type A is reconfigured and vice versa.*
   Hint: Use global variables for the queue lengths and a decision rule.

5 Parameterizing the models

We have seen validations during model construction. What is validated in this stage is the behavior of a few cases. When it comes to validating the characteristics of the system to be simulated, the parameters of the model play a vital role.
For an Arena model, this amounts to the intensity (time between arrivals) of the Create blocks, the duration (delay) of the Process blocks and the probability (percent true) of the random Decide blocks. Intensities and durations are sampled from random distributions. See appendices B,C,D for details.
During model construction, the distributions are arbitrarily selected, but for the simulation experiments, they should match the values that have been observed (or are predicted) to occur. In order to obtain faith in the simulation, known situations should be modeled and simulated. The simulation results obtained should match those observed in reality, which is an important criterion for validation.
The necessary parameters for a current system must be derived from sensors data (traffic), inferred from SAP logs (production/logistics) and obtained by direct observation. It is always advisable to observe the system for at least some time in order to understand what is going on. This helps in detecting errors in logs or sensor malfunction.
Of course, the most desirable option is to obtain sample data that actually occurred. Arena possesses an input analyzer to optimally "fit" a distribution based on these sample data. In many cases, this is impossible or unfeasible. A SAP log will usually

contain data in aggregated form only, such as the duration of a series of actions. The duration of the individual actions must be inferred some way or another from the log. A good way to start is to derive averages first. The expectations $\mathbb{E}(X_i)$ of the chosen distributions X_i should match the derived averages.

One of the main reasons for simulation is the assessment of queues. Queues are closely related to server capacity (resource utilization); it can be calculated from expected values of distributions only: arrival intensities, expected durations and decision probabilities. In the photo example (40% of the photos is rejected), a citizen that arrives requires on average $0.6(1+3.5)+0.4(1+1.5+3.5) = 5.1$ minutes of server capacity. An arrival intensity of a citizen every 5.0 minutes thus gives a capacity overflow. Occupation rates for resources can often be inferred from the data without simulation, like in the example above, see also appendix F. It is advisable to compute such characteristics in order to compare them to the same characteristics obtained by simulation.

If possible, obtaining variances for the samples is also important (although less important than averages). When the variance increases, simulation results such as queue lengths and waiting times have a tendency to grow. Based on some heuristics, distributions can be chosen that match the computed average and variance. Checks can be executed to assess their plausibility.

As a rule of thumb, Create blocks have an exponential distribution. By measuring a number N of arrivals during a period of M minutes, the exponential distribution with intensity M/N is often the right choice. There are some proviso's: N should not be too small and the period should not be too large. If, say, $N < 100$, random fluctuations will have a too large effect. If the period is too long, systematic fluctuations may occur. A first check consists of measuring the number of arrivals in subintervals, which should more or less match the ratio M/N (if N is not too small).

A second test for the plausibility of the exponential distribution is counting the

Consider e.g. traffic densities, which may be large between 7:00 and 9:00 and much smaller between 9:00 and 11:10. Measuring 5,000 arrivals 7:00 and 11:10 does not mean an intensity of 20 cars per minute. With 20 cars per minute arriving, simulation would indicate that a server that can handle 30 cars per minute is adequate. If 80% of the arrivals occur between 7:00 and 9:05, the intensity is 32 cars per minute and large queues will occur!

The parameter for Process blocks that determines resource usage is the average duration. This average can be inferred from observations, possibly complemented by logs. Care has to be taken in dealing with breakdowns: make sure to recognize and eliminate breakdowns in a log. In general a resource breakdown is modeled as a special "diabolic case" that seizes it with priority and releases it when repaired. Breakdowns are relatively rare phenomena and cannot be reliably assessed: a log containing enough breakdowns suffers from systematic fluctuations. Assumptions have to be made for the intensity and duration of a diabolic case. In the final report, these assumptions have to be mentioned as a proviso, e.g. *if breakdowns in machine X occur at most once per month and are repaired in less than one hour, the average throughput time of case Y is less than 7.5 hours with 95% reliability.*

Random Decide blocks should be avoided if possible. If the model contains them nonetheless, look out for dependencies. The model in Figure 12, features a processing step followed by a test. If the test is negative, the processing has to be redone. It is

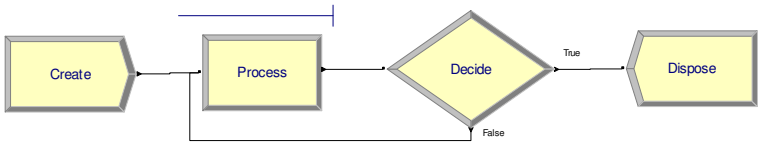

Figure 12: Iterative model.

possible that the various iterations (first, second, ...) possess different characteristics. For example, the average processing step may take on average 1.56 hours and 77% of the tests is positive. By closer inspection, the duration of the processing depends on the iteration number: the first processing step takes on average 1.7 hours and 70% of the tests is positive. In the remaining 30%, the second processing step takes on average 1.1 hours and 99% of the tests are positive. The third iteration has a short duration; all tests following it have been positive in the log studied. The average resource utilization is the same in both models, but the expected waiting time is higher in reality than a simulation from the above model would indicate.
Analyzing the measured data, consulting the users and keeping one's eyes open are all necessary to get a proper parametrization. A necessary validation (if possible) is to make a parameterized model of the current system, simulate it and check whether the simulation results (such as queue lengths, waiting times and resource utilization) match the actual results measured. Any mismatch should be investigated, explained and corrected.
In the ferry example, we installed video cameras to record the events. From the recorded data, we want to infer the arrival processes and the duration of the ferry crossing and getting on and off board. A problem that often occurs is that arrival intensities are not constant. If the ferry is heavily used for commuter traffic, higher intensities occur during workdays around 8am and 5pm. If the ferry is mainly used for leisure rides, intensities are higher during holidays with nice weather conditions. Together with the user, some representative periods must be selected. For these periods, the arrival times should be derived from the recorded images. The crossing time may be influenced by weather circumstances (wind, current) and the passing of other boats on the river. Here also, selecting representative periods may be necessary. Getting off and on board requires some preparations, that can be discounted in the crossing, followed by an amount of time proportional to the number of cars that get off and on board.
Obtaining the necessary data from the recordings requires some man-hours of tedious, yet important work. Fortunately, it is not necessary to look at the video recording in "real time". The video recording should display the time, so that we can use the "fast-forward" functionality, obtaining a trace of several hours, starting as shown.

```
time        event
06:59:57    A2B cars enter
07:00:07    4 cars entered
07:00:19    leave dock A
07:03:56    arrive at dock B
07:04:08    A2B cars leave
```

```
07:04:15    4 cars left
07:04:21    B2A cars enter
07:04:29    5 cars entered
07:04:36    leave dock B
...
```

From this information, parameters can be inferred. The Arena input analyzer can help in this respect. It is not always possible to parameterize by "standard" distributions. Examples are traffic light simulations (occurrence of "bursts") or machine breakdowns (two "breakdown classes" with different characteristics). Observing the process, noting comments and careful validation can help to avoid mistakes.

Exercise

1. Indicate how the parameters of the ferry model can be inferred from the a trace such as shown above.
2. A production plant possesses a breakdown-prone machine resource. The occurrence and duration of breakdowns in 2006 have been recorded, giving the following pattern.

from	0	3	6	9	12	15	18	21	24	27	30
to	3	6	9	12	15	18	21	24	27	30	∞
amount	3	11	8	5	3	2	4	3	2	2	4

How do you model the breakdowns. What kind of distribution you choose?

6 Processing the results

Simulation is used to support decisions, such as the number of resources of a certain type that has to be installed in order to guarantee a certain service level. For example, we want to guarantee that incoming telephone calls are answered within 20 seconds on average and that at least 90% of the calls is answered in less than 30 seconds. By simulation, it can be assessed for given combinations of infrastructure, procedures and available personnel whether this requirement can be met. It is very important to adequately report the extent of the statements made.

6.1 Observed quantities

Observations

During simulation, repeated *observations* are made of quantities such as waiting times, run times, processing times or stock levels. Also, occurrences of certain events can be counted. Individual observations have little interest. Instead, we are interested in statistical data about the observed quantities, such as means and variances.

Suppose we have k consecutive observations, called $x_1, x_2, ...x_k$. These observations are also called a *random sample*. The *mean* of a number of observations is also called the *sample mean*. We denote the sample mean of $x_1, x_2, ...x_k$ as $\overline{x}$:

Sample mean

$$\overline{x} = \frac{\sum_{i=1}^{k} x_i}{k}$$

Please note that the sample mean is an estimate of the true mean.

Sample variance

The *variance* of a number of observations is also called the *sample variance*. This

variance is a measure for the deviation from the mean. The smaller the variance, the closer the observations will be to the mean. We can find the sample variance s^2 by using the following formula:

$$s^2 = \frac{\sum_{i=1}^{k}(x_i - \overline{x})^2}{k-1}$$

We can rewrite this formula as:

$$s^2 = \frac{(\sum_{i=1}^{k} {x_i}^2) - \frac{1}{k}\left(\sum_{i=1}^{k} x_i\right)^2}{k-1}$$

During a simulation, Arena keeps track of the number of observations k, the sum of all observations $\sum_{i=1}^{k} x_i$ and the sum of the squares of all observations $\sum_{i=1}^{k} {x_i}^2$. In doing so, the sample mean and the sample variance can be obtained. The square root of the sample variance $s = \sqrt{s^2}$ is also called the *sample standard deviation*.

Sample standard deviation

Apart for the mean and the variance of a random sample, there is also the so-called *median* of k observations $x_1, x_2, \ldots, x_k$. The median is the value of the observation in the 'middle' after sorting the observations w.r.t. their value. An 'observation in the middle' only exists if the number of observations is odd. In case of an even number of observations, the average of the two observations in the middle is taken. In a simulation experiment, we have to save and sort all observations in order to calculate their median.

Median

Example 1

In a simulation experiment the following waiting times are measured: 2.3, 3.4, 2.2, 2.8, 5.6, 3.2, 6.8, 3.2, 5.3 and 2.1. Using the random sample we can determine the sample mean and the sample variance. The sample mean is 3.69 and the sample variance is 2.648. The median is 3.2.
Arena automatically reports mean and variance of waiting times for entities. can indicate whether e.g. 90% of the simulated incoming calls are answered within 30 seconds. This can also be assessed directly

6.2 Subruns and preliminary run

In a simulation experiment, we can easily determine the sample mean and the sample variance of a certain quantity. We can use the sample mean as an estimate for the expected true value of this quantity (e.g. waiting time), but we cannot determine how reliable this estimate is. A simulation experiment usually consists of a number of partial experiments (subruns) that allow us to assess the reliability of the simulation results.

6.2.1 Necessity

Example 2

Consider a post office with one counter. Customers enter the post office according to a Poisson process with intensity 6. The time between two consecutive customers is therefore distributed negative exponentially with an average of 10 minutes. The service time is also distributed negative exponentially. The average service time is 6 minutes. We want to determine the average waiting time of customers.
A simulation experiment that assesses the waiting times of 1000 consecutive customers, has yielded a mean waiting time of 12.6 minutes. (This result stems from

a truly conducted simulation experiment!) The expected waiting time can be calculated mathematically in this case (see appendix F) and yields 9 minutes. The difference between the simulated and calculated mean waiting time indicates that the experiment was stopped too soon. The experiment clearly does not allow to reliably assess the waiting time. Indeed, a second simulation run of 1000 customers (with a different start value for the random generator) yields a mean waiting time of 8.5 minutes. This poses an important question: how long should we simulate in order to obtain reliable conclusions.

Half width

We thus need a mechanism to determine the reliability of a certain result. Arena reports the *half width* of observed means. In the example above, a mean of 12.6 minutes is reported with a half width of 3.8 minutes. This signifies that with 95% reliability the reported mean lies between 8.8 and 16.4 minutes. This is called the *confidence interval*, which indeed (barely) contains the calculated (actual) mean (9 minutes). Clearly, a longer simulation should be conducted if we can only state that the average waiting time is "somewhere between 8.8 and 16.4 minutes".

Confidence interval

On some occasions, the half width cannot reliably determined by Arena. This is reported as "insufficient" or "correlated". In such cases, a longer simulation run is required.

Subruns

The determination by Arena of half widths is based on calculations involving *subruns*. The simulation run is divided into a series of smaller simulation runs, from which the results are compared.

Dividing the simulation run with 1000 customers into 10 consecutive runs with 100 customers each, we find for each of these subruns a separate sample mean, ranging between 7.2 and 18.6 minutes. The mean over these 10 subruns is 12.6 minutes; the various subrun means can be used to assess the half width of 3.8 minutes. We we will treat later how this is done.

6.2.2 Subruns and initial phenomena

Arena allows the replication of a simulation; in that case, the simulation is replayed each time with different start values of the random generator. In doing so, we need to distinguish between two situations:

(i) A *stable situation*, meaning the circumstances are constant. Quantities like e.g. the arrival intensities of customers are the same throughout the simulation. There are no structural peaks. Also, no startup effects occur: we are interested in the *stable state* of a process.

(ii) An *unstable situation*, where the circumstances change structurally during simulation. For instance, an "empty" state characterizes the beginning and end of the simulation. Often there is a clear start and end; this is called a *terminating process*.

Analyzing the average waiting time of a customer in the post office at a given time of the day can be done in stable situation. We assume that the arrival process and the service process have certain characteristics at this time of the day. However, the analysis of the average waiting time of customers during the whole day requires an

unstable situation. We might see a different arrival process at the end of the day than at 10 AM.
The reason that we differentiate between stable and unstable situations is the fact that this influences the format of the simulation experiment.

Stable situation

In a stable situation, the start state is often exceptional; e.g. all queues are initially empty. In order to allow the simulated system to evolve to a more normal state, preliminary "warm-up" runs can be specified. The results gathered in these runs are disregarded in the measurements. A warm-up period should be long enough to reach a stable state, which is checked by comparing simulation results for different warm-up lengths.

Unstable situation

In dealing with an unstable situation, the modeled initial state must reflect the initial state of the system. Subruns are often special time intervals. For example, if we want to analyze the waiting times in a post office for the entire day, each subrun represents one day. Each of these subruns starts in the state that there are no customers in the post office. At the end of the day no new customers are allowed and the subrun stops after all waiting customers have left.

Relevant questions when setting up a simulation experiment are:

- Is a preliminary run necessary?
- How long should the preliminary run be?
- How many subruns are necessary?
- How long should each subrun be?

Let us try and answer these questions. A preliminary run is necessary only if we are simulating a stable situation, where the initial state differs from the average state experienced during simulation.
The length of a preliminary run depends on how much the initial state differs from an average state and how fast the simulation reaches a stable situation. We can chart the development of some relevant quantities in time and estimate from this chart when a stable state has been reached.
The number of subruns needed depends on the desired reliability of the final results and the length of each subrun, as indicated below.
For unstable situations, the length of a subrun is determined by the period that is investigated, e.g. the morning rush for traffic simulation.
The total run length (subrun length times number of subruns) determines the half width (and thus the confidence interval) for measured quantities. By increasing the total run length, the confidence intervals become smaller and thereby the simulated estimates become more accurate. As a rule of thumb, halving the confidence interval requires quadrupling the total run length.

6.3 Analysis of subruns

In this subsection, we treat the method used by Arena to obtain half widths. Suppose we have executed n subruns and measured a certain result x_i for each subrun i. We

know there exists a "true" value (called μ) that each x_i approximates and we want to derive assertions about μ from the x_i. For example, x_i is the mean waiting time measured in subrun i and μ the "true" mean waiting time that we could find by conducting a hypothetical simulation experiment of infinite length. (Instead, we might consider the mean variance of the waiting time, the mean occupation rate of a server or the mean length of a queue.) We must be certain that the values x_i are mutually independent for all subruns. If Arena discovers dependencies, the half width is reported as "correlated". Given the results $x_1, x_2, \ldots, x_n$, we derive the sample mean $\overline{x}$:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

and the sample variance s^2:

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$

Note that the sample mean and the sample variance for the results of the various subruns should not be confused with the mean and the variance of a number of measures *within* one subrun! We can consider $\overline{x}$ as an estimate of μ. The value $\overline{x}$ can be seen as a sample from a random variable $\overline{X}$ called *estimator*. The value [1] $\frac{s}{\sqrt{n}}$ is an indication of the reliability of the estimate $\overline{x}$. If $\frac{s}{\sqrt{n}}$ is small, it is a good estimate.

6.3.1 The situation with over 30 subruns

If there is a large number of subruns, we can consider the estimator $\overline{X}$ (because of the central limit theorem) as normally distributed. We will therefore treat the situation with over 30 subruns as a special case.

When do we stop generating subruns?

The fact that $\frac{s}{\sqrt{n}}$ measures how well $\overline{x}$ approximates μ, allows us to determine the time that we can stop generating subruns.

(i) Choose a value d for the permitted standard deviation from the estimated value $\overline{x}$.

(ii) Generate at least 30 subruns and note per subrun the value x_i.

(iii) Generate additional subruns until $\frac{s}{\sqrt{n}} \leq d$, where s is the sample standard deviation and n the number of subruns executed.

(iv) The sample mean $\overline{x}$ is now an estimate of the quantity to be studied.

There are two other reasons why in this case at least 30 subruns have to be executed. In the first place, $\overline{X}$ is only approximately normally distributed with a large number of subruns. This is a compelling reason to make sure that there are at least 30 mutually independent subruns. Another reason for choosing an adequate number

[1] The variance of the estimator $\overline{X}$ is $\mathrm{Var}[\overline{x}] = \mathrm{Var}[\frac{1}{n}\sum_{i=1}^{n} x_i] = \frac{\sigma^2}{n}$. The standard deviation of $\overline{x}$ thus equals $\frac{\sigma}{\sqrt{n}}$. As s is a good estimate of σ, the amount $\frac{s}{\sqrt{n}}$ yields a good estimate for the standard deviation of $\overline{x}$.

of subruns is the fact that by increasing the number of subruns, s becomes a better estimate of the true standard deviation.

Confidence interval

Given a large number of independent subruns, we can also determine a *confidence interval* for the quantity to be studied. Because $\overline{x}$ is the average of a large number of independent measures, we can assume that $\overline{x}$ is approximately normally distributed. (Central Limit Theorem, Appendix E). From this fact, we deduce the probability that μ lies within a so-called confidence interval. Given the sample mean $\overline{x}$ and the sample standard deviation s, the true value μ conforms with confidence $(1-\alpha)$ to the following equation:

$$\overline{x} - \frac{s}{\sqrt{n}}\ z(\frac{\alpha}{2}) < \mu < \overline{x} + \frac{s}{\sqrt{n}}\ z(\frac{\alpha}{2})$$

where $z(\frac{\alpha}{2})$ is defined as follows. If Z is a standard normally distributed random variable, then $\mathbb{P}[Z > z(x)] = x$. For a number of values of x, $z(x)$ is shown in table 1. The value α represents the unreliability, that is the chance that μ does not conform to the equation. Typical values for α range from 0.001 to 0.100. The interval

$$\left[\overline{x} - \frac{s}{\sqrt{n}}\ z(\frac{\alpha}{2}), \overline{x} + \frac{s}{\sqrt{n}}\ z(\frac{\alpha}{2})\right]$$

is also called the $(1-\alpha)$-confidence interval for the estimated value μ.

x	z(x)
0.001	3.09
0.005	2.58
0.010	2.33
0.025	1.96
0.050	1.64
0.100	1.28

Table 1: $\mathbb{P}[Z > z(x)] = x$ where Z is standard normally distributed.

Example 3

We can illustrate the above with the following example. A company is worried about the workload of the help desk staff. This has become so high that absenteeism has increased substantially. To look into this situation a simulation study was done to determine how to decrease the workload. To assess the workload in the present situation, a simulation experiment consisting of 30 subruns was conducted. Each subrun represents one working day. The average occupation rate of help desk staff per subrun is shown in Table 2.
The sample mean is 0.9408 and the sample variance is 0.000617. So, all the data needed to set up a $(1-\alpha)$-confidence interval are known: $n = 30$, $\overline{x} = 0.9408$, $s^2 = 0.000617$ and therefore $s = 0.02485$. If we take α equal to 0.010 we will find the following confidence interval:

$$\left[0.9408 - \frac{0.02485}{\sqrt{30}}\ z(\frac{0.010}{2}), 0.9408 + \frac{0.02485}{\sqrt{30}}\ z(\frac{0.010}{2})\right]$$

subrun number	average load factor	subrun number	average load factor	subrun number	average load factor
1	0.914	11	0.894	21	0.898
2	0.964	12	0.962	22	0.912
3	0.934	13	0.973	23	0.943
4	0.978	14	0.984	24	0.953
5	0.912	15	0.923	25	0.923
6	0.956	16	0.932	26	0.914
7	0.958	17	0.967	27	0.923
8	0.934	18	0.924	28	0.936
9	0.978	19	0.945	29	0.945
10	0.976	20	0.936	30	0.934

Table 2: The average occupation rate per subrun.

As $z(0.005) = 2.58$ this is therefore the interval $[0.9291, 0.9525]$. The larger the unreliability α, the smaller the corresponding confidence interval. Arena will report a half width of 0.88, giving the 95% confidence interval $[0.9328, 0.9496]$. The 90% confidence interval is $[0.933, 0.9483]$. We can safely infer that the occupation rate for the help desk staff is quite high!

6.3.2 The situation with less than 30 subruns

In some cases we can do with less than 30 subruns. In this case, the results of the separate subruns (x_i) have to be approximately normally distributed. If the result of a subrun x_i is the average of a large number of observations, then (by the central limit theorem,) each x_i is approximately normally distributed. So if x_i is, say, the average waiting time, average service time or average throughput time of a large number of customers, x_i is approximately normally distributed. By using this property, we can deduce, given n subruns with a sample mean $\overline{x}$, sample deviation s and reliability $(1 - \alpha)$ the following confidence interval:

$$\left[\overline{x} - \frac{s}{\sqrt{n}}\ t_{n-1}(\frac{\alpha}{2}), \overline{x} + \frac{s}{\sqrt{n}}\ t_{n-1}(\frac{\alpha}{2})\right]$$

Student's t-distribution

where $t_v(x)$ is the critical value of a *Student's t-distribution*, also called t-distribution, with v degrees of freedom. Table 3 shows for a number of values of v and x the critical value $t_v(x)$.

Contrary to the method discussed earlier, we can determine the confidence interval in the way shown above if a limited number of subruns (say 10) is at our disposal. If we have a larger number of subruns at our disposal, it is better to apply the $(1 - \alpha)$ confidence interval mentioned earlier, even if we are convinced that the subrun results are normally distributed. For large n the confidence interval based on $t_{n-1}(\frac{\alpha}{2})$ is more accurate than the one based on $z(\frac{\alpha}{2})$; only the latter depends upon the central limit theorem concerning the number of subruns.

$t_v(x)$	$x =$			
	0.100	0.050	0.010	0.001
$v = 1$	3.08	6.31	31.82	318.31
2	1.89	2.92	6.96	22.33
3	1.64	2.35	4.54	10.21
4	1.53	2.13	3.75	7.17
5	1.48	2.02	3.37	5.89
6	1.44	1.94	3.14	5.21
7	1.41	1.89	3.00	4.79
8	1.40	1.86	2.90	4.50
9	1.38	1.83	2.82	4.30
10	1.37	1.81	2.76	4.14
15	1.34	1.75	2.60	3.73
20	1.33	1.72	2.53	3.55
25	1.32	1.71	2.49	3.45
50	1.30	1.68	2.40	3.26
100	1.29	1.66	2.35	3.17
∞	1.28	1.64	2.33	3.09

Table 3: The critical values for a Student's t-distribution with v degrees of freedom.

For any assertion concerning the reliability of $\overline{x}$, based on the results in table 1 we will find for $\alpha = 0.100$ the following confidence interval:

$$\left[0.9408 - \frac{0.02485}{\sqrt{30}}\ t_{29}(\frac{0.100}{2}), 0.9408 + \frac{0.02485}{\sqrt{30}}\ t_{29}(\frac{0.100}{2})\right]$$

As $t_{29}(0.050) = 1.699$ this yields the interval $[0.9331, 0.9485]$. This interval and the 90% confidence interval we deduced earlier are approximately the same. Keep in mind that we can only use the confidence interval based on the t distribution, if we are convinced that the average occupation rate per subrun is approximately normally distributed. Especially with a small number of subruns, this condition is extremely important.

6.4 Variance reduction

Using the techniques described above, we can analyze simulation results. Reliable assertions often require long simulation runs, which may turn out cost prohibitive. However, some advanced techniques allow reliable assertions from shorter simulation runs. As we have just seen, there is a linear connection between the size of the confidence interval and the standard deviation of the results measured per subrun. If the standard deviation is doubled, the size of the corresponding confidence interval is also doubled. In order to allow assertions with the same reliability, the number of subruns has to increase fourfold! So by decreasing the standard deviation between the subrun results the reliability will increase or the number of subruns needed will decrease. The standard deviation is the square root of the variance, so decreasing the

variance and the standard deviation goes hand in hand. The techniques that focus on decreasing the variance are called *variance reducing techniques* Some well-known techniques are:

Variance reducing techniques

- antithetic variates
- common random numbers
- control variates
- conditioning
- stratified sampling
- importance sampling

We will not explain all of these techniques at length and only summarize the first two.

Antithetic variates

In a simulation experiment random numbers are constantly being used and assigned to random variables. A good random generator will generate numbers that are independent of each other. It is, however, not necessary to generate a new set of random numbers for each subrun. If $r_1, r_2, \ldots, r_n$ are random numbers, so are $(1 - r_1), (1 - r_2), \ldots, (1 - r_n)$! These numbers are called antithetic. If we generate new random numbers for each odd subrun and we use antithetic random numbers for each even subrun, we only need half of the random numbers (if the total number of subruns is even). There is another bonus. The results for subrun $2k - 1$ and subrun $2k$ will probably be negatively correlated. If e.g. subrun $2k - 1$ is characterized by frequent arrivals (e.g. caused by sampling small random numbers), the arrivals in subrun $2k$ will be infrequent since antithetic, thus large numbers will be sampled. If x_{2k-1} and x_{2k} represent the results of two consecutive subruns, in all probability $\mathrm{Cov}[x_{2k-1}, x_{2k}]$ will be smaller than zero. This leads to a decrease in the variance of the mean of x_{2k-1} and x_{2k}, as:

$$\mathrm{Var}[\frac{X+Y}{2}] = \frac{1}{4}(\mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X, Y])$$

The total sample variance will then also decrease, narrowing down the confidence interval.

Common random numbers

If one wants to compare two alternatives, it is intuitively obvious that the circumstances should be as similar as possible. This means that the samples taken in the simulation runs of either alternative should correspond maximally. When simulating different arrangements of the post office, the alternatives may use the same random numbers for interim arrival times and service times. In this way the variance of the difference between both alternatives may be substantially reduced.

There are various advanced techniques for increasing the information obtained through simulation. These techniques have to be used with great care. More details can be found in [2, 10, 11].

6.5 Sensitivity analysis

Given a certain model, one can give well-founded estimates for the expected waiting times, occupation rates, fault frequencies etc., by using subruns and calculating confidence intervals. Since these results are based on a specific situation, it is unclear how *sensitive* they are. If we find an estimated average waiting time of 10 minutes for an arrival process with an average interarrival time of 5 minutes, what would the average waiting time be if the interarrival time is not 5 but 6 minutes? In general a model has a number of *parameters*; adjustable quantities, like average interarrival time, average service time and average response time. For a simulation experiment each of these quantities is given a certain value. This value is often estimated, as the exact value is unknown. Also the probability distribution chosen will only approximate the true distribution of values. It is therefore of the utmost importance to know how sensitive the results are to variations in the model parameters.

Sensitivity

Model parameters

Sensitivity analysis

A *sensitivity analysis* is carried out to assess dependencies between the model parameters and the results. To test the sensitivity, a number of experiments are conducted with slight variations in parameter settings. These experiments indicate the extent to which slight variations can influence the final result. Adjusting the setting of a certain parameter will often only mildly influence on the final result. Sometimes, however, a slight adjustment of a parameter will lead to completely different results. A resource with a high occupation rate will be more sensitive to fluctuations in its arrival process than a resource with a lower occupation rate. We also use the term *robustness*. A model is robust if slight deviations in parameter settings barely influence the final result.

Exercise

Assess the parameters used for the ferry simulation model given as example. Assess their reliability and their stability over time and decide a sensitivity analysis range for them. For example traffic densities have been obtained by collecting past data. The amount of traffic is influenced by many factors and may very well increase or decrease over time, so an extensive sensitivity analysis of these parameters is needed.

7 Pitfalls

As already indicated in the introduction, simulation is often used to support critical strategic decisions, where errors are very expensive. However, errors are manifold and often easily introduced when conducting a simulation study, so one has to be on the alert constantly. We will look at the phases in the life cycle of a simulation study and identify the dangers in each specific phase. Figure 13 shows the phases of a simulation study and the possible sources of errors.

Possible error sources

The problem definition can be inconsistent (contradictory) or incomplete (vague). A conceptual model is developed by a systems analyst and not by the user himself, so various mapping errors can occur during modeling. Implementing the conceptual model in a simulation language can also introduce errors. If validation is performed by the wrong persons or without the proper care, errors made earlier are not eliminated. Therefore, preferably the model should be validated by the user. During

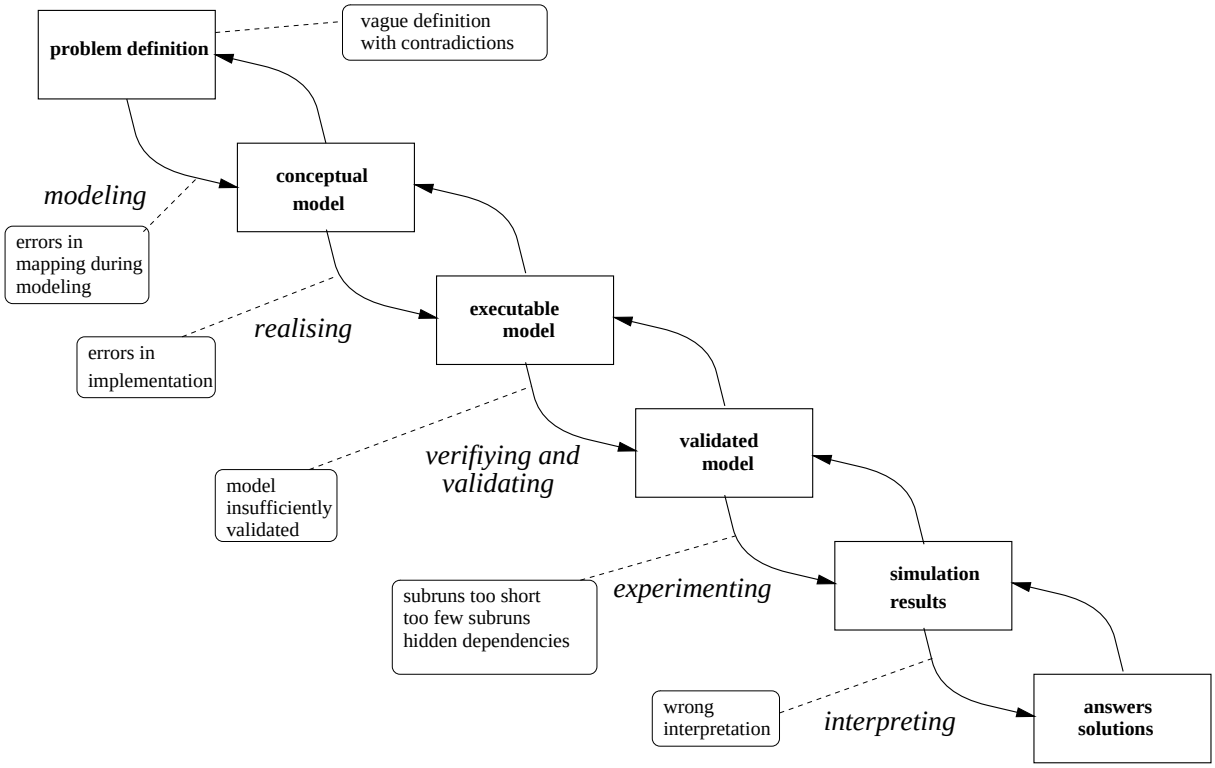

Figure 13: The dangers per phase in a simulation study.

experimentation, errors can arise from too short or too few subruns or from hidden dependencies between the subruns. Also, the initial run can be too short. Errors during experimentation lead to incorrect results. Even if all the previous traps have been avoided, things can still go wrong during interpretation, if faulty conclusions are drawn from the results gathered.

We list ten typical errors (pitfalls) frequently made. Anyone involved in a simulation study should be aware of them and avoid them and their likes.

Error 1: One-sided problem definition

A simulation study gets off on the wrong foot if the problem definition is drawn up exclusively by either the user or the systems analyst. The user may possess extensive knowledge of the problem area, but lacks the experience needed for defining his problem. The systems analyst knows the elements which should be present in a problem definition, but lacks the background of the specific problem. The systems analyst is also aware of the possibilities and impossibilities of simulation. The user on the other hand, generally knowing little about simulation, is barely informed on this issue. Therefore, for a simulation study to be successful, it is important that both parties closely cooperate in setting up the problem definition. The problem definition serves as a 'contract' between the user and the builder of the model. A *rule of thumb* for this situation is:

Rule of thumb

"Do not start a simulation study until it is clear to both user(s) and

analyst(s) which questions need to be answered!"

Error 2: Choice of a wrong level of detail

In making a simulation model, one chooses a certain level of detail. In a simulation model for a manufacturing department, a machine may be modeled as an object with serving time as its only parameter. Alternatively, it can be modeled in detail, taking into account aspects such as set-up times, faults, tool-loading, maintenance intervals etc. Many simulation studies fail because a wrong level of detail was chosen initially. Too much detail causes the model to become unnecessarily complex and introduces extra parameters that need to be assessed (with all the risks involved). A lack of adequate detail can lead to a simulation model that leaves the essential questions of the problem definition unanswered. The right level of detail is chosen if:

Characteristics of a good level of detail

(1) information is present that allows experiments with the model,

(2) the important questions from the problem definition are addressed by the model and

(3) the complexity of the model is still manageable for all parties concerned.

If it is impossible to choose a level of detail that meets this condition, the problem definition will have to be adjusted.

Error 3: Hidden assumptions

Many assumptions are needed in order to allow a simulation study to be carried out. These assumptions fill gaps in an incomplete problem definition or are made to keep the simulation model simple. The assumptions should be carefully documented and reported, as provisos for the conclusions. If possible, a sensitivity analysis should be carried out to assess how dependent the reported conclusions are of them. Assumptions that are made without proper documentation (i.e. hidden) have the tendency to evolve into unquestioned truths. Incorrect hidden assumptions may lead to the rejection of the simulation model. Also, hidden assumptions about future developments lead to incorrect conclusions. There is quite some difference between the conclusions *The capacity will be insufficient within three years* with and without the proviso *in case of 8% annual growth.*

Error 4: Validation by the wrong people

Sometimes, due to time pressure or indifference of the user, the simulation model is only validated by its maker(s). Discrepancies between the model and the ideas of the user may thus be discovered too late, if at all. Therefore, the user should be involved in the validation of the simulation model before any experiments are conducted.

Error 5: Forcing the model to fit

Frequently, in the validation phase, the results of the simulation model do not match the observed or recorded actual data. One is then tempted to make the model 'fit' by

changing certain parameter values. One fiddles around with the parameter settings until some match is found. This, however, is very dangerous, since this match with reality is most likely caused by sheer luck and not by a model that adequately reflects reality. Parameters should be adjusted only after having understood why the model deviates from reality. This prevents the conscious or unconscious obscuring of errors in the model.

Error 6: Underexposure of the sensitivity of the model

Certain model parameters (e.g. the intensity of the arrival process) are often set at one specific value. This chosen setting should be justified statistically. However, even if this is the case, small variations in the arrival process can make all assumptions about it invalid. Therefore, the sensitivity of the model to minor adjustments of the parameters should be seriously accounted for.

Error 7: No subruns

Some people say: "A sufficiently long simulation yields correct results!" They execute a simulation run for a night or weekend and then blindly trust e.g. the mean waiting time measured. This is a very risky practice, as it disallows any assertions about the reliability of the result found. The reported half width should be used to define reliability intervals. In a what-if analysis through simulation, the simulation should be carried out long enough to allow reliable comparison of alternatives.

Error 8: Careless presentation of the results

Interpreting the results of a simulation study may require complex statistical analysis. This is often a source of errors. Translating the results from statistics into language a user can understand, can be very tricky indeed. In Darrel Huff's book "How to lie with statistics" ([4]), there are numerous examples of sloppy and misleading presentations. As an example, suppose the final report of a simulation study contains the following conclusion "Waiting times will be reduced by 10 percent". This conclusion is very incomplete, as it contains no reference whatsoever to its reliability. It is good practice to give a confidence interval. The same conclusion suggests that waiting times will be reduced by 10 percent for each customer. This, however, may not be the case. The average waiting time may be reduced by 10 percent while it increases for certain customers and is reduced somewhat more for others.

Error 9: Dangers of animation

Simulation tools present nicely formatted reports and even possess animation facilities. This improves communication with the user. However, there is a large inherent danger in animation. As animation only shows the tangible aspects of the simulation model, the user may develop an unfounded faith in the model. The choice of parameters or decision making rules deeply influence the simulation results, yet are barely visible in an animation. A nice report does not replace a sound statistical analysis.

Error 10: Unnecessary use of simulation

Simulation is a flexible and varied analysis tool. Some people therefore are inclined to use it regardless of the circumstances. Often, however, a simple mathematical model (e.g. a queuing model) or a simple spreadsheet calculation is sufficient. In such cases simulation is 'overkill'. It should only be used if and when the situation requires it. Simulation is a means and not a goal!

8 Recommended further reading

A classic book on simulation is the book by Naylor et al. [9]. Other standard works about simulation are: Bratley, Fox and Schrage [2], Law and Kelton [7], Ross [11] and Shannon [12]. In particular the books of Ross ([11]) and Bratley, Fox and Schrage ([2]) are recommended for further study.

References

[1] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. *Journal of the Association of Computing Machinery*, 22(2):248–260, April 1975.

[2] P. Bratley, B.L. Fox, and L.E. Schrage. *A guide to simulation, 2nd ed.* Springer-Verlag, Berlin, 1987.

[3] D. Gross and C.M. Harris. *Fundamentals of queueing theory.* Wiley, London, 1985.

[4] D. Huff. *How to lie with statistics.* Penguin Books, New York, 1954.

[5] W.D. Kelton, R.P. Sadowski, and D. Sturrock. *Simulation with Arena, 3rd ed.* McGraw-Hill, New York, 2004.

[6] L. Kleinrock. *Queueing systems, Vol. 1:Theory.* Wiley-Interscience, London, 1975.

[7] A.M. Law and W.D. Kelton. *Simulation modeling and analysis, 3rd ed.* McGraw-Hill, New York, 2000.

[8] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* The MIT Press, Cambridge, 1986.

[9] T.H. Naylor, J.L. Balintfy, D.S. Burdick, and Kong Chu. *Computer simulation techniques.* Wiley, New York, 1966.

[10] M. Pidd. *Computer modelling for discrete simulation, 4th ed.* Wiley, Chichester, 2002.

[11] S.M. Ross. *A course in simulation, 3rd edition.* Academic Press, San Diego, 2002.

[12] R.E. Shannon. *Systems simulation: the art and science.* Prentice-Hall, Englewood Cliffs, 1975.

A Modeling, concurrency and scheduling with Petri nets

In this appendix we treat the concurrency aspect of simulation: actions may or may not occur simultaneously. This influences the order in which the actions are scheduled and their interaction with the simulated time. These concepts are vital to simulation models; they are best illustrated within Petri nets but are not limited to them alone.

Petri net modeling is all about the interplay of *actions* and *objects*. Actions may or may not occur depending upon the presence of certain objects. Of course, it is near impossible to take all kinds of objects into account. Whether or not a certain object must be modeled depends on the questions that need an answer. The same holds for actions.

We revisit our discotheque example. We are interested in the cloakroom queues. so the actions that we wish to model are entering, leaving one's coat and departing. There are possible queues for leaving and obtaining one's coat. A first approximation features customer objects as shown in Figure 14. A customer can be in five states: outside (wishing to enter), in the hall (queuing to leave his/her coat), inside, in the hall (queuing to get his/her coat) and outside (having left). These states are visible as circles and are called *places*. The actions are inscribed in squares called *transitions*. They move the customers from one state to the other. We add a transition that moves customers from *in* to *qgc*.

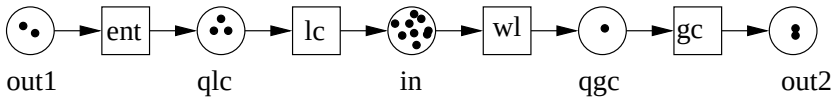

Figure 14: Discotheque simulation model.

In the figure, the customer objects are visible as *tokens*, the black dots within the places. From time to time, a customer moves from one state to another. This is understood as the *consumption* of a token and the *production* of a new token. In the depicted model, there are three customers waiting to leave their coat and one waiting to get it.

In Figure 14, a customer could move from place *qlc* ot place *in*. In order to determine the time at which such a state change occurs, every token possesses a *time stamp*.

Scheduler

By looking at the time stamps of the tokens, a *scheduler* of the simulation decides the next action that will occur and the *simulated occurrence time* of that action.

A logical rule for the scheduler is that an action cannot occur sooner than the time

Eagerness

stamps of the tokens consumed by it. Another rule is called the *eagerness* rule: if some action can occur, some action must occur. If an action occurs, the time stamps of the produced tokens will be greater than or equal to the simulated time of that action. Often, a *delay* depending on the action is added to the occurrence time.

We shall study the example in Figure 15. Initially there are three tokens with time stamps 5,6, and 6 in place a. Action x has a delay of 2 and y has a delay of 0. Action x can occur at time 5, so by the eagerness rule some action must occur at time 5. The only action that can occur is x, consuming the token with time stamp 5 and producing a token with time stamp 7 because of the delay. So we obtain the second token configuration. In this configuration, there are two possibilities for x

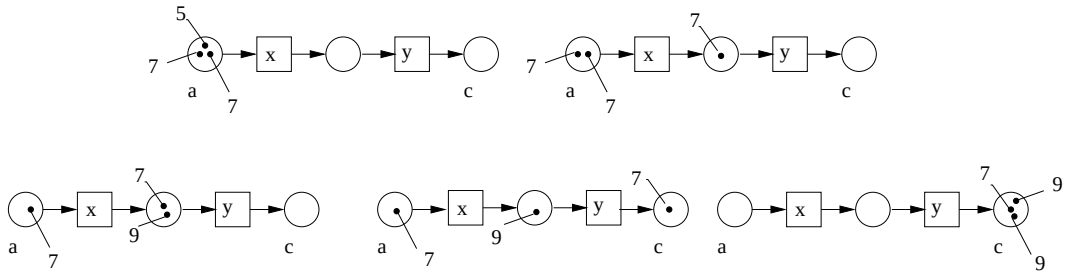

Figure 15: Scheduling of actions.

and one for y to occur at time 7, so one of them must occur. We can get the third configuration, where the other two can still occur at time 7, leading to the fourth. After some more actions, the fifth configuration is reached, in which no actions can occur anymore.

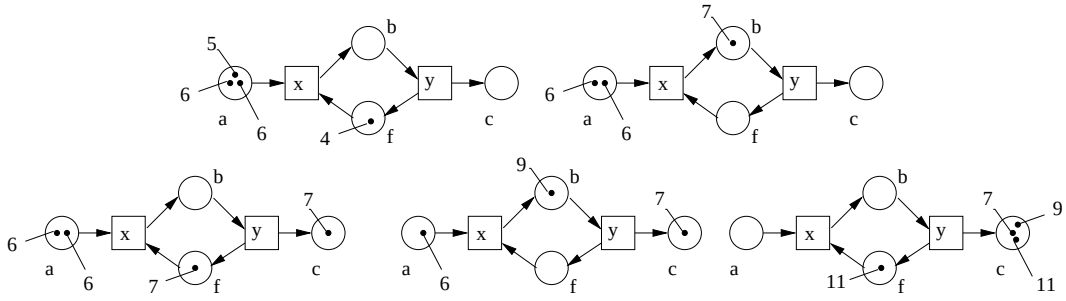

Figure 16: Scheduling of actions: 1-server queue.

It can be shown that all tokens in a will lead to tokens in c with their time stamps augmented by 2. This corresponds to a queue with service time 2 and an unbounded number of servers. If we want a bounded number of servers, we have to model them as separate objects. In Figure 16, we have added a place f (free server), containing a token with time stamp 4. Action x needs a token from a and another one from f to occur. So it can occur at time 5, leading to the second configuration. Here there is token in b with time stamp 7, which keeps the server occupied. By the occurrence of y at time 7, the server becomes free again, leading to the third configuration, freeing the server. The next stages become clear.

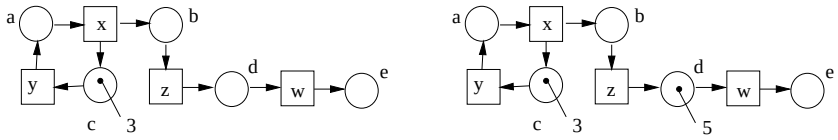

Figure 17: A livelock: action w will never occur.

Notice the alternation of the server between the free (f) and occupied (b) states. A 2-server queue is modeled by initializing a second token in f. Also notice that the

Livelock

scheduling rules sketched above make it possible to model *livelocks*: loops without delay. In Figure **??** such a livelock is depicted, where certain actions will never

occur. This is because there is always an action with a smaller execution time that must be scheduled to occur first.
The reason that the livelock occurs is that actions x and y have no delay. If one or both would have had some delay, tokens in d would eventually be consumed by an occurrence of w. This net is a token spawner or *generator*. The start place of simulation models (*out1* in our disco example) will be connected to some generator that spawns tokens with increasing time stamps.

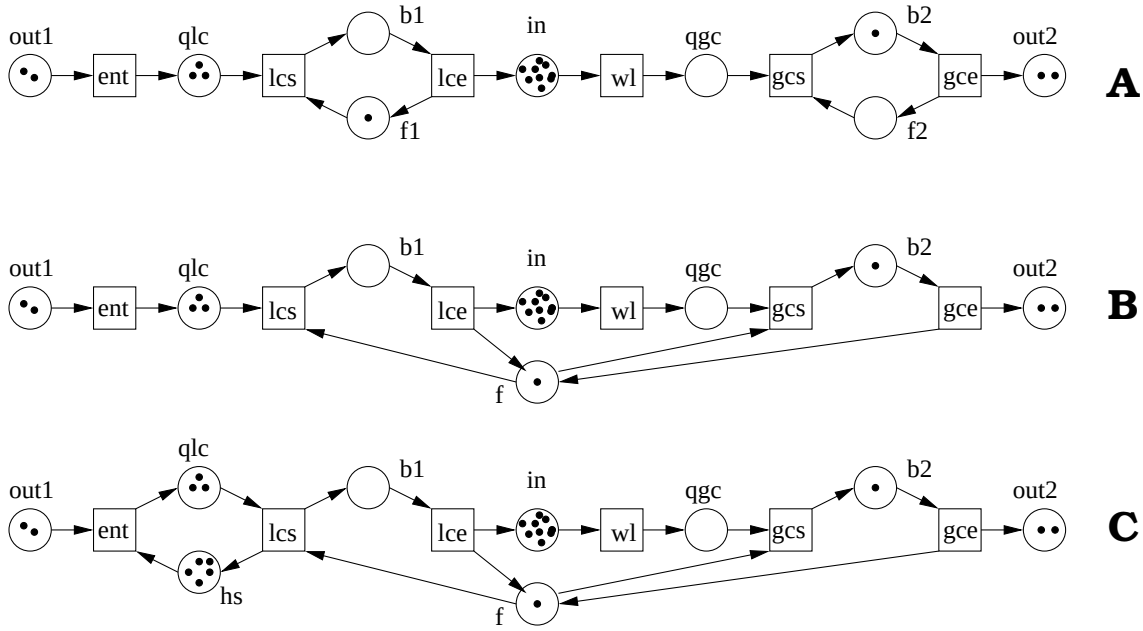

Figure 18: Discotheque with queues.

We continue with our discotheque example. In Figure 18 we see two one-server queues for the cloakroom (version A). There apparently are two staff members at the cloakroom, one of taking coats and the other returning them. Version B takes into account that staff members do not exclusively take or return coats; they do both. So if the "take" queue is empty, two servers can be present at the "return" queue. A third version models the fact that if the "take" queue is too long, the discotheque cannot be entered anymore. The space in the hall is indeed a resource, which is modeled as such.
Finally, Figure 19 models the fact that some customers do not wear a coat and thus bypass the cloakroom. The actions *lcs* (leave coat - start) and *nce* (no coat entry) are both possible; if one occurs, the other does not. Similarly, a customer can leave without coat. If this model is adopted, it must be decided which percentage of customers bypasses the cloakroom and the generation of customers must take care of it.
Simulation models have a tendency to become more and more complex as details are added. To allow structuring of these complex models, they can be divided into subparts that can be unfolded. Arena allows so called "submodels" to realize these conceptual models in the same structured way.

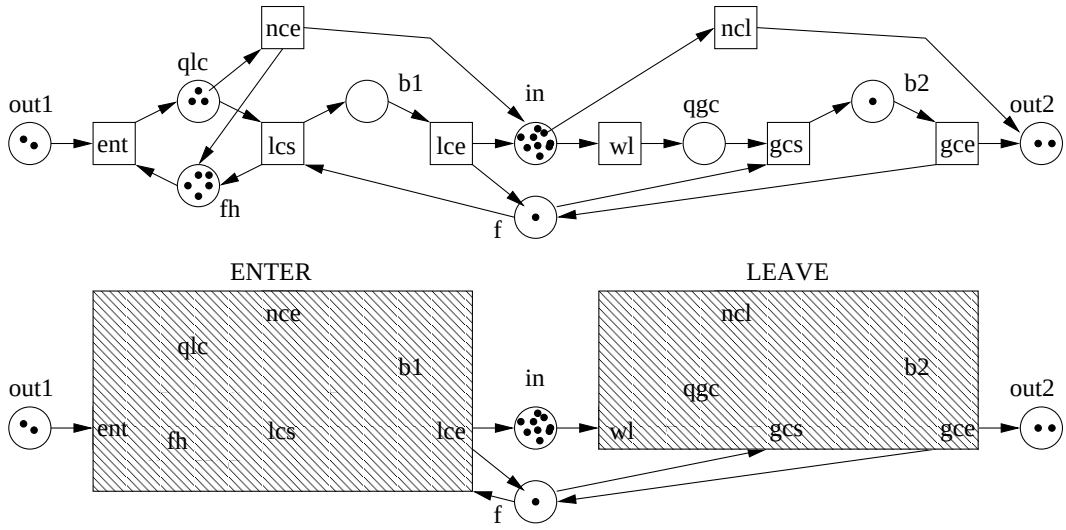

Figure 19: Choice and hierarchy.

B Basic probability theory and statistics

In this appendix, some relevant concepts from probability theory and statistics are treated.

B.1 Random and pseudo-random numbers

Environment
Monte Carlo method
Pseudo-random numbers
Random generator

A simulation experiment is little more than replaying a modeled situation. To replay this situation in a computer, we have to make assumptions for the modeled system together with its *environment*. The environment's behavior is determined by *randomization*. We do not know when and how many customers will enter a post office, but we can find out the mean and variation of customer arrivals. While simulating, the computer takes pseudo-random samples from a probability distribution. The computer is by nature a deterministic machine, so the numbers are not truly random. This has the advantage that the simulation experiment can be replicated.

A *random generator* is a piece of software for producing pseudo-random numbers. The computer does in fact use a deterministic algorithm to generate them, which is why they are called "pseudo" random. Most random generators generate pseudo-random numbers between 0 and 1, which are distributed *uniformly* over the interval between 0 and 1. After having generated random numbers $X_0 \ldots X_N$, the size of the subset of $\{X_0 \ldots X_N\}$ contained in a subinterval of length α will be about α/N. It depends on the random generator whether 0 and/or 1 themselves can be generated. Most random generators generate a series of pseudo-random numbers $\frac{X_i}{m}$ according to the formula:

$$X_n = (aX_{n-1} + b) \text{ modulo } m$$

For each i, X_i is a number from the set $\{0, 1, 2, \ldots, m-1\}$ and $\frac{X_i}{m}$ matches a sample from a uniform distribution between 0 and 1. There are choices for a, b and m so that the sequence appears to be truly random. There are several tests to check the

quality of a random generator (cf. [2, 10, 12, 7]): frequency test, correlation test, run test, gap test and poker test.
Good choices with m not too large are

$$X_n = 16807X_{n-1} \text{ modulo } (2^{31} - 1)$$

and

$$X_n = 3125X_{n-1} \text{ modulo } (2^{35} - 31).$$

Seed

The first number in the sequence (X_0) is called the *seed*. The seed completely determines the sequence of random numbers. In a good random generator, different seeds produce different sequences. Sometimes the computer selects the seed itself (e.g. based on the system clock). However, preferably the user should consciously select a seed himself, allowing the *reproduction* of the simulation experiment later. Reproducing a simulation experiment is important if an unexpected phenomenon occurs that needs further examination.

Most simulation languages and packages possess an adequate random generator. This generator can be seen as a black box: a device that produces (pseudo) random numbers upon request. However, beware: pseudo-random numbers are not truly random, since a deterministic algorithm is used to generate them. Do not use more than one generator and take care in selecting the seed.
The following well-known pitfalls illustrate the dangers in using random generators. The first mistake is using the so-called 'lower order bits' of a random sequence. If the numbers X_n are randomly generated number between 0 and 1, then so are the numbers $K.X_n$modulo K right? In fact, for large K, these numbers may behave cyclically with a quite short cycle. Another frequent mistake is the double use of a random number. If the same random number is used twice for generating a sample, a dependency occurs that does not exist in reality, which may lead to extremely deceptive results. Packages like Arena make it hard to directly access the random generator to avoid the above mistakes.

B.2 Probability distributions

Probability distribution

The random generator is used to obtain samples from various *probability distributions*. A probability distribution specifies which values are possible and how probable each of those values is.

Random variable

To simplify the discussion of random distributions and samples from probability distributions, we introduce the term *random variable*. A *discrete* random variable X represents an experiment that can be repeated and has a number $\mathbb{P}[X = a]$ between 0 and 1 (a probability) assigned to each outcome a. For example, we can model the throwing of a dice by means of a random variable X that can take on the values 1,2,3,4,5 and 6. The probability of obtaining any value a from this set is $\frac{1}{6}$. We can write this as follows:

$$\mathbb{P}[X = a] = \begin{cases} \frac{1}{6} & \text{if } a \in \{1, 2, 3, 4, 5, 6\} \\ 0 & \text{else} \end{cases}$$

A *continuous* random variable X has probabilities $\mathbb{P}[a \leq X \leq b]$ that the variable lies in the interval between a and b. If X is uniformly distributed between 0 and 1, then $\mathbb{P}[\frac{1}{3} \leq X \leq \frac{1}{2}] = \frac{1}{6}$, since $\min(\frac{1}{2}, 1) - \max(0, \frac{1}{3}) = \frac{1}{6}$ and $\mathbb{P}[\frac{1}{2} \leq X \leq \frac{3}{2}] = \frac{1}{2}$. The *probability density* $f_X(x)$ is defined by

$$f_X(x) = \lim_{\epsilon \downarrow 0} \mathbb{P}[x \leq X \leq x + \epsilon]/\epsilon.$$

We can make expressions with random variables, as with ordinary variables (also see below). If X, Y are discrete random variables representing different dice throws, the expression $X + Y$ satisfies e.g. $\mathbb{P}[X + Y = 5] = \Sigma_{a+b=5}\mathbb{P}[X = a \wedge Y = b]$. Random variables X, Y are called *independent* iff $\forall a, b :: \mathbb{P}[X = a \wedge Y = b] = \mathbb{P}[X = a].\mathbb{P}[Y = b]$. If the dice throws are independent, we have $\mathbb{P}[X + Y = 5] = \frac{1}{9}$.

Expectation

Given an expression $\mathcal{E}(X)$ containing a random variable X, we define its *expectation* $\mathbb{E}[\mathcal{E}] = \int_{-\infty}^{\infty} \mathcal{E}(x).f_X(x)dx$. In discrete cases this becomes $\Sigma_a \mathcal{E}(a).\mathbb{P}[X = a]$. The expectation $\mathbb{E}[X]$ of a random variable X is thus $\mathbb{E}[X] = \int_{-\infty}^{\infty} x.f_X(x)dx$ (or $\Sigma_a a.\mathbb{P}[X = a]$). Similarly, its *variance* $\mathrm{Var}[X]$ is defined by $\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$. The expectation (also called *mean*) $\mathbb{E}[X]$ of X is the average to be expected from a large number of samples from X. The variance $\mathrm{Var}[X]$ is a measure for the average deviation of X from the mean. If $\mathrm{Var}[X] = 0$, then $\mathbb{P}[X = \mathbb{E}(x)] = 1$ (every sample of X equals the mean). If $\mathrm{Var}[X]$ is large, the sample variation will be high. If a random variable X is clear from the context, its expectation is often denoted as μ, and its variance ($\mathrm{Var}[X]$) is denoted as σ^2. The expectation and variance satisfy the following equalities:

$\mathbb{E}[X]$

Variance

$\mathrm{Var}[X]$

$$\sigma^2 = \mathbb{E}[(X - \mu)^2] = \mathbb{E}[X^2] - \mu^2$$

Because of the above equation, σ is a better measure for the deviation from the mean than its square $\mathrm{Var}[X]$. We call $\sigma = \sqrt{\mathrm{Var}[X]}$ the *standard deviation* of X. The ratio of μ and σ measures the "wildness" of the distribution. In appendices C,D, some distributions are treated that appear in simulation models.

Standard deviation

C Discrete random distributions

The values that a discretely distributed random variable can take on have are separated; the distance between any two such values will exceed a certain minimum. Often, there are only finitely many such values. The following table gives an overview of discrete distributions to be treated hereafter.

distribution	**domain**	$\mathbb{P}[X=k]$	$\mathbb{E}[X]$	$\mathrm{Var}[X]$
Bernoulli $0 \leq p \leq 1$	$k \in \{0,1\}$	$\begin{cases} 1-p & k=0 \\ p & k=1 \end{cases}$	p	$p(1-p)$
homogeneous $a < b$	$k \in \{a, \ldots, b\}$	$\frac{1}{(b-a)+1}$	$\frac{a+b}{2}$	$\frac{(b-a)((b-a)+2)}{12}$
binomial $0 \leq p \leq 1$ $n \in \{1,2,\ldots\}$	$k \in \{0,1,\ldots,n\}$	$\binom{n}{k} p^k (1-p)^{n-k}$	np	$np(1-p)$
geometric $0 \leq p \leq 1$	$k \in \{1,2,\ldots\}$	$(1-p)^{k-1}p$	$\frac{1}{p}$	$\frac{1-p}{p^2}$
Poisson $\lambda > 0$	$k \in \{0,1,\ldots\}$	$\frac{\lambda^k}{k!}e^{-\lambda}$	λ	λ

C.1 Bernoulli distribution

Bernoulli distribution p

If X is a random variable distributed according to a *Bernoulli distribution* with parameter p, then it may take on the value 1 with probability p and 0 with probability $1-p$. So:

$$\mathbb{P}[X=0] = 1-p \quad \text{en} \quad \mathbb{P}[X=1] = p$$

The expectation $\mathbb{E}[X]$, i.e. the mean value, is p. The variance $\mathrm{Var}[X]$ equals $p(1-p)$. Often, the value 1 signifies success and 0 failure of an experiment.

C.2 Discrete homogeneous distribution

Homogeneous distribution a, b

A random variable X is distributed according to a *discrete homogeneous distribution* with a lower bound of a and an upper bound of b, if it can take on only integer values between and including a and b and each such value is equally probable. The lower bound a and the upper bound b are integers. In this case the probability of a certain value k equals:

$$\mathbb{P}[X=k] = \begin{cases} \frac{1}{(b-a)+1} & \text{if } k \in \{a, a+1, a+2, \ldots, b-1, b\} \\ 0 & \text{else} \end{cases}$$

The expectation ($\mathbb{E}[X]$) equals $\frac{a+b}{2}$. The variance ($\mathrm{Var}[X]$) equals $\frac{(b-a)(b-a+2)}{12}$. Rolling a dice corresponds to taking a sample from a discrete homogeneous distribution with lower bound 1 and upper bound 6.

C.3 Binomial distribution

Binomial distribution n, p

Suppose we do n experiments that can either succeed or fail. Per experiment the chance of success equals p. What is the probability of k successes? We can model this with a *binomial distribution* with parameters n and p. Suppose X is distributed binomially with parameters n and p. For $x \in \{0,1,\ldots,n\}$ we have:

$$\mathbb{P}[X=k] = \binom{n}{k} p^k (1-p)^{n-k}$$

The expectation equals np. The variance equals $np(1-p)$. Throwing 10 coins on a table and counting the tails corresponds to taking a sample from a binomial distribution with parameters $n = 10$ and $p = 0.5$. The case $n = 1$ of the binomial distribution is the Bernoulli distribution.

C.4 Geometrical distribution

Geometrical distribution

Suppose repeating an experiment with a chance of success p until we are successful for the first time. The number of experiments thus needed is a sample from a *geometrical distribution* with parameter p. Suppose X being distributed geometrically with parameter p, then for each positive integer number k:

$$\mathbb{P}[X = k] = (1-p)^{k-1}p$$

The expectation equals $\frac{1}{p}$. The variance equals $\frac{1-p}{p^2}$. This distribution is also called the Pascal distribution.

C.5 Poisson distribution

Poisson distribution

λ

The *Poisson distribution* is strongly related to the negative exponential distribution seen later in this book. If the time between two consecutive arrivals of a customer in a supermarket is distributed according to the negative exponential distribution with parameter λ, the number of customers entering the supermarket per time unit is distributed Poisson with parameter λ.
Suppose X is distributed Poisson with parameter λ, then for each integer number k the following holds:

$$\mathbb{P}[X = k] = e^{-\lambda}\frac{\lambda^k}{k!}$$

The expectation ($\mathbb{E}[X]$) equals λ. The variance ($\mathrm{Var}[X]$) also equals λ.
Think of a supermarket where customers enter according to a negative exponential distribution. The average time between two arrivals is 15 minutes (0.25 hour), that is to say $\lambda = \frac{1}{0.25} = 4$. The number of arrivals per hour is distributed Poisson with parameter $\lambda = \frac{1}{0.25} = 4$. The average number of arrivals per hour therefore equals 4. The variance also equals 4.

D Continuous random distributions

Probability density

Next, we discuss some continuous distributions. The notion $\mathbb{P}[X = k]$, giving the probability of a certain value k, satisfies $\mathbb{P}[X = k] = 0$ for any k. The important notion is the *probability density*. The larger the probability density $f_X(k)$ of a continuously distributed random variable X at the point k becomes, the greater the probability that a sample from X is close to k. We give an overview, then discuss the various distributions individually.

distribution	**domain**	$f_X(x)$	$\mathbb{E}[X]$	$\mathrm{Var}[X]$
uniform $a < b$	$a \leq x \leq b$	$\frac{1}{b-a}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
exponential $\lambda > 0$	$x \geq 0$	$\lambda e^{-\lambda x}$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
normal $\mu \in \mathbb{R}$ $\sigma > 0$	$x \in \mathbb{R}$	$\frac{1}{\sqrt{2\pi\sigma^2}}\ e^{\frac{-(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
gamma $r, \lambda > 0$	$x > 0$	$\frac{\lambda(\lambda x)^{r-1}\ e^{-\lambda x}}{\Gamma(r)}$	$\frac{r}{\lambda}$	$\frac{r}{\lambda^2}$
Erlang $\lambda > 0$ $r \in \{1, 2, \ldots\}$	$x > 0$	$\frac{\lambda(\lambda x)^{r-1}\ e^{-\lambda x}}{(r-1)!}$	$\frac{r}{\lambda}$	$\frac{r}{\lambda^2}$
χ^2 $v \in \{1, 2, \ldots\}$	$x > 0$	see gamma $r = \frac{v}{2}$ and $\lambda = \frac{1}{2}$	v	$2v$
beta $a < b$ $r, s > 0$	$a \leq x \leq b$	$\frac{1}{b-a}\ \frac{\Gamma(r+s)}{\Gamma(r)\Gamma(s)}$ $\left(\frac{x-a}{b-a}\right)^{r-1}\left(\frac{b-x}{b-a}\right)^{s-1}$	$a + (b-a)\frac{r}{r+s}$	$\frac{rs(b-a)^2}{(r+s)^2(r+s+1)}$

D.1 Uniform distribution

Uniform distribution
a, b

The *uniform distribution*, also called homogeneous distribution, is very simple. Between a lower bound a and an upper bound b all values are equally probable. A random variable X distributed uniformly with parameters a and b has probability density $f_X(x) = \frac{1}{b-a}$ for x between a and b ($a \leq x \leq b$) and $f_X(x) = 0$ for $x < a$ or $x > b$. The expectation ($\mathbb{E}[X]$) equals $\frac{a+b}{2}$ The variance ($\mathrm{Var}[X]$) equals $\frac{(b-a)^2}{12}$.

D.2 Triangular distribution

Triangular distribution
a, b, c

The *triangular distribution* is the default Arena distribution for process durations. It has three parameters, a minimum a, maximum c and a most likely value b, with $a \leq b \leq c$. As Figure 20 shows, the graph has triangular shape. A random variable X with distribution triangle(a, b, c) has probability density $f_X(x) = \frac{2(x-a)}{(c-a)(b-a)}$ for $a \leq x \leq b$, $f_X(x) = \frac{2(c-x)}{(c-a)(c-b)}$ for $b \leq x \leq c$ and $f_X(x) = 0$ for $x < a$ or $x > c$. The expectation ($\mathbb{E}[X]$) equals $\frac{(a+b+c)}{3}$ and the variance ($\mathrm{Var}[X]$) equals $\frac{(b-a)^2}{6}$. It is often advisable to choose a symmetric triangle, i.e. $b = \frac{a+c}{2}$. In this case, the expectation ($\mathbb{E}[X]$) equals b.

D.3 Negative exponential distribution

Negative exponential distribution
λ

The *negative exponential distribution* is often used to model arrival processes and is the default Arena distribution for the Create block. The negative exponential distribution has only one parameter λ. Let X be a negative exponentially distributed random variable with parameter λ. The corresponding probability density will be ($x \geq 0$):

$$f_X(x) = \lambda e^{-\lambda x}$$

Figure 21 shows this probability density for $\lambda = 1.0$, $\lambda = 0.5$ and $\lambda = 2.0$.

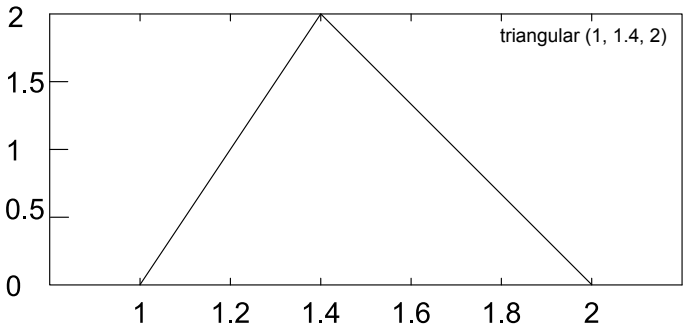

Figure 20: The triangular distribution with parameters $a = 1, b = 1.4$ and $c = 2$.

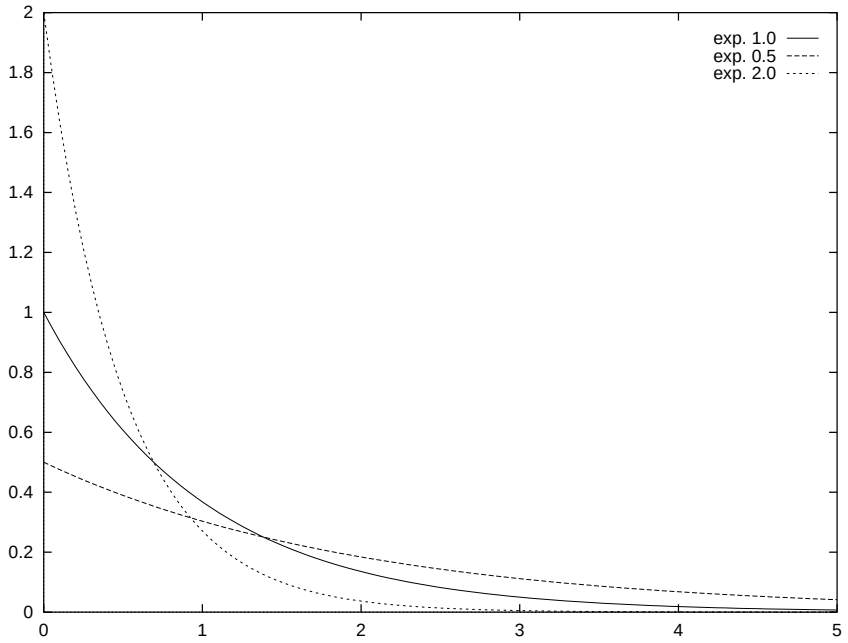

Figure 21: Negative exponential distributions with parameters $\lambda = 1.0$, $\lambda = 0.5$ and $\lambda = 2.0$.

The expectation equals $\frac{1}{\lambda}$. The variance equals $\frac{1}{\lambda^2}$. If a random variable X, negative exponentially distributed with parameter λ, defines an arrival process, λ is called the *intensity* of the arrival process. The random variable X is used to model the time between two consecutive arrivals. The higher the intensity of the arrival process (i.e. the larger λ), the larger the mean number of arrivals per time unit. If $\lambda = 10$, the expected inter-arrival time is 0.10 time units. The mean number of arrivals per time unit in this case equals 10. If $\lambda = 100$, then the expected inter-arrival time equals 0.01 time units and the mean number of arrivals per time unit equals 100.

Intensity

D.4 Normal distribution

Normal distribution

The *normal distribution* has many applications. This distribution is used for modeling e.g. processing times, response times, stock levels and transport times. Often, one does not know the exact distribution of a certain random variable, but its mean μ and variance σ are known. In such cases, the normal distribution with parameters μ and σ is often used as an approximation. This practice has its dangers; instead of $\mathsf{normal}(\mu, \sigma)$ one could also use $\mathsf{triangle}(\mu - \sqrt{6}\sigma, \mu, \mu + \sqrt{6}\sigma$. The probability density of a σ, μ-normally distributed random variable X is defined as follows:

μ, σ

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Figure 22 shows for a number of parameter values the corresponding probability densities. The probability densities are maximal around μ. As we digress from the mean, the probability density decreases. The smaller σ, the faster the probability density decreases, so for small σ, a value near μ is very likely. The expectation ($\mathbb{E}[X]$) equals μ. The variance ($\mathrm{Var}[X]$) equals σ^2.

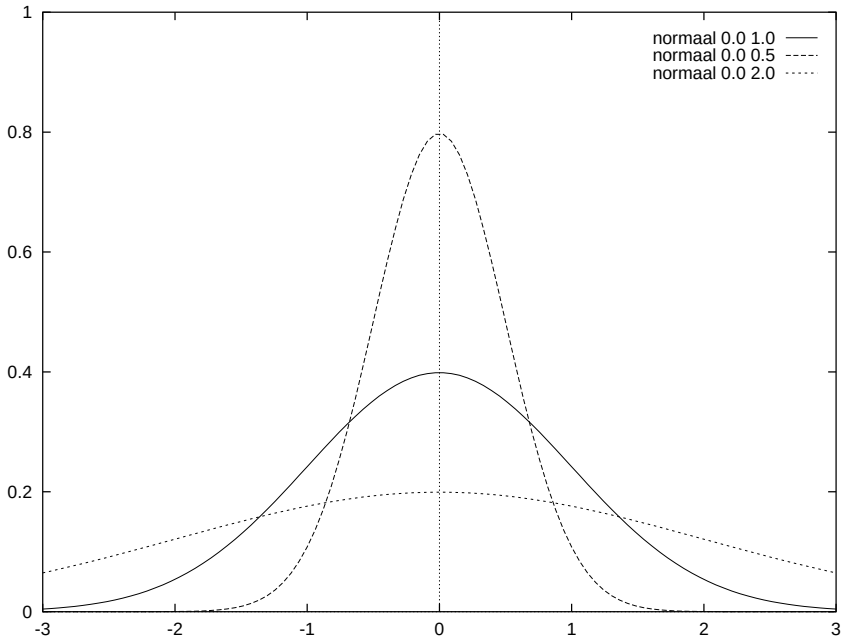

Figure 22: Normal distributions with parameters $\mu = 0.0$ and σ consecutively 1, 0.5 and 2.

Standard normal distribution

If $\mu = 0$ and $\sigma = 1$, we call this a *standard normal distribution*. If Y is a standard normally distributed random variable, then $X = \mu + \sigma Y$ is a normally distributed random variable with parameters μ and σ. Conversely, if X is a normally distributed random variable with parameters μ and σ, then $Y = \frac{X-\mu}{\sigma}$ is a standard normally distributed random variable.

If we use a normally distributed random variable for modeling time durations, like processing times, response times or transport times, we must be aware that this random variable can also take on *negative* values. In general negative durations are impossible; this may even cause a failure of the simulation software. For wildly distributed durations (like traveling from Eindhoven to Utrecht in a car), it is quite possible that a sample duration (3 hours due to a traffic jam) exceeds twice the mean duration of 1 hour. These durations are highly asymmetric: the minimum duration is, say, 50 minutes, 95% of the trips are in less than 1 hour and only in a few cases do long sample durations occur. In such cases, the next distribution is a better choice.

D.5 Gamma distribution

A characteristic of the normal distribution is its symmetry. We often need 'skewed' distributions instead, where the probability densities below and above the mean are distributed differently. In this case a *gamma distribution* might be chosen. A random variable X is gamma distributed with parameters $r > 0$ and $\lambda > 0$ if for all $x > 0$:

Gamma distribution r, λ

$$f_X(x) = \frac{\lambda(\lambda x)^{r-1}\ e^{-\lambda x}}{\Gamma(r)}$$

(The function Γ is the mathematical gamma function.) Figures 23, 24 and 25 show the probability densities for a number of parameter values.
These figures clearly show that the gamma distribution, depending on the parameter values has many manifestations. If r does not exceed 1, f_X is a monotonously decreasing function. In this case values close to 0 are most probable. If $r > 1$, the function will first grow to a maximum and then decrease monotonously. The parameter r therefore determines the shape of the distribution. The parameter λ determines the 'spread' of the distribution. The larger λ, the more probable is a value close to 0. The expectation ($\mathbb{E}[X]$) equals $\frac{r}{\lambda}$. The variance ($\mathrm{Var}[X]$) equals $\frac{r}{\lambda^2}$.

Modal value

The *modal value* of a probability distribution is the value for which the probability density is maximal, i.e. m is the modal value of an X if for each x: $f_X(m) \geq f_X(x)$. The modal value of a gamma distributed random variable X depends on r and λ. If $r \leq 1$, 0 is the most probable value, i.e. the modal value of X equals 0. If $r > 1$ the modal value equals $\frac{r-1}{\lambda}$. A special case of gamma distribution is the negative exponential distribution ($r = 1$).

D.6 Erlang distribution

Erlang distribution

The *Erlang distribution* is a special case of the gamma distribution. A gamma distribution with an integer parameter r, is also called an Erlang distribution. A

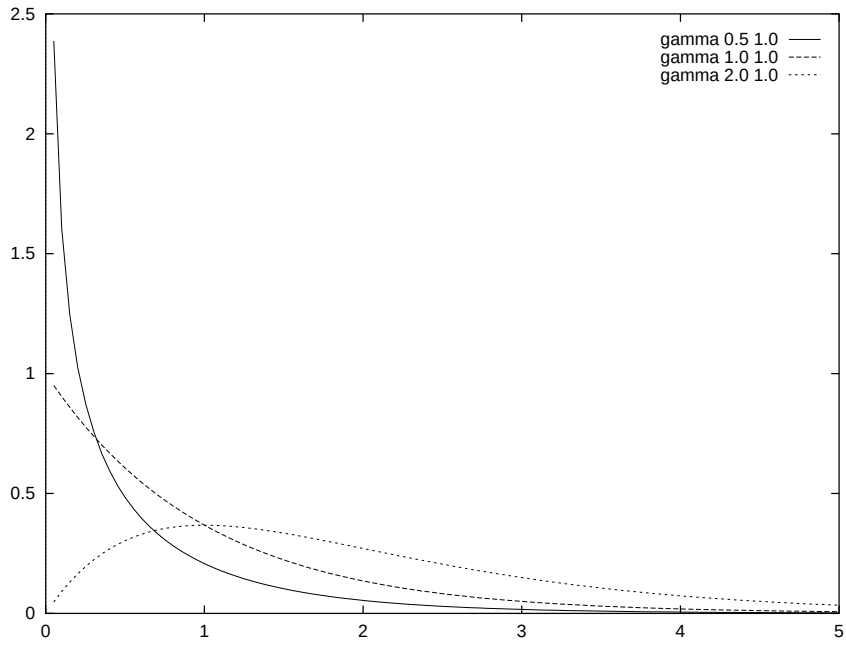

Figure 23: Gamma distributions with parameter $\lambda = 1$ and r consecutively 1, 0.5 and 2.

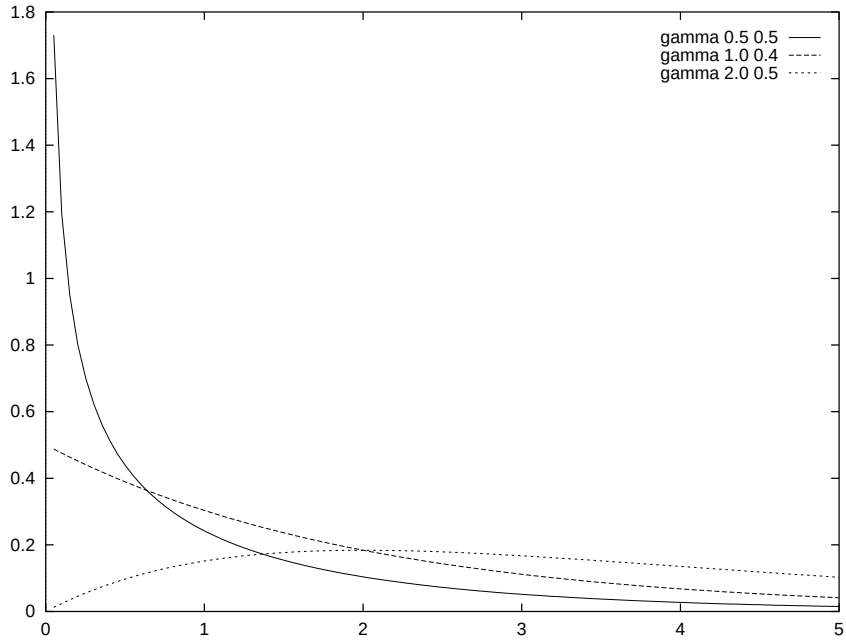

Figure 24: Gamma distributions with parameter $\lambda = 0.5$ and r consecutively 1, 0.5 and 2.

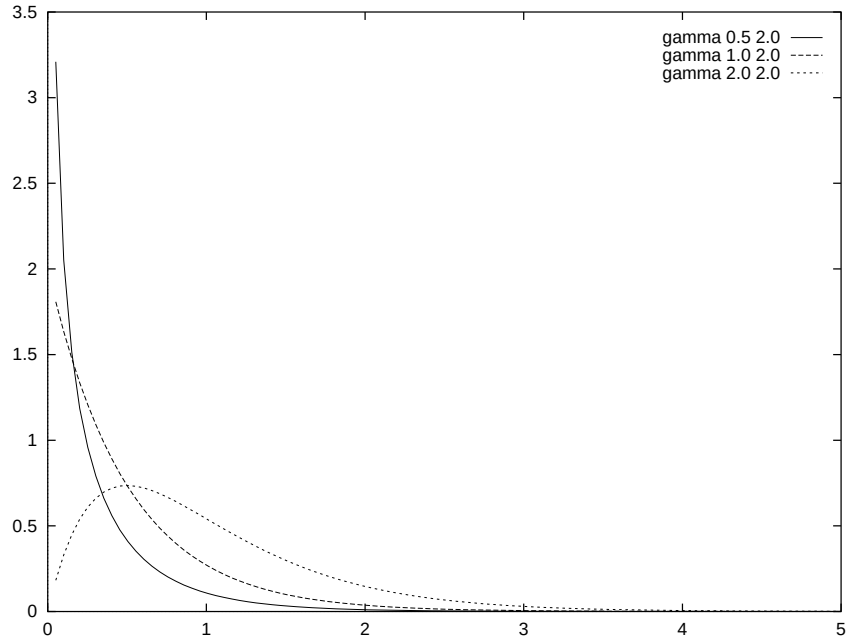

Figure 25: Gamma distributions with parameter $\lambda = 2$ and r consecutively 1, 0.5 and 2.

r, λ

random variable which is distributed Erlang with parameters r (integer) and λ can be considered the sum of r independent, negative exponentially distributed random variables with parameters λ. If X is distributed Erlang with parameters r and λ, then:

$$f_X(x) = \frac{\lambda(\lambda x)^{r-1}\ e^{-\lambda x}}{(r-1)!}$$

(Note that $n!$ is the factorial function: the product of all integers from 1 up to n. If r is an integer, then $\Gamma(r) = (r-1)!$.) Figure 26 shows for a number of values of r and λ the function f_X.
The expectation equals $\frac{r}{\lambda}$. The variance equals $\frac{r}{\lambda^2}$.

D.7 χ^2 distribution

χ^2 distribution

Degrees of freedom

The χ^2 *distribution* is another special case of the gamma distribution. A χ^2 distribution has a single parameter v. This v is a positive integer and represents the number of *degrees of freedom*. A random variable X distributed χ^2 with v degrees of freedom is the same as a random variable distributed gamma with parameters $r = \frac{v}{2}$ and $\lambda = \frac{1}{2}$. Figure 27 shows the probability density for a number of values v. The expectation equals v. The variance equals $2v$. There is also a connection between the normal distribution and the χ^2 distribution. If $X_1, X_2, \ldots, X_n$ are mutually independent standard normally distributed random variables, the random variable $X = X_1^2 + X_2^2 + \ldots + X_n^2$ is distributed exactly χ^2 with parameters $v = n$. The χ^2 distribution is used specifically for 'goodness-of-fit' tests.

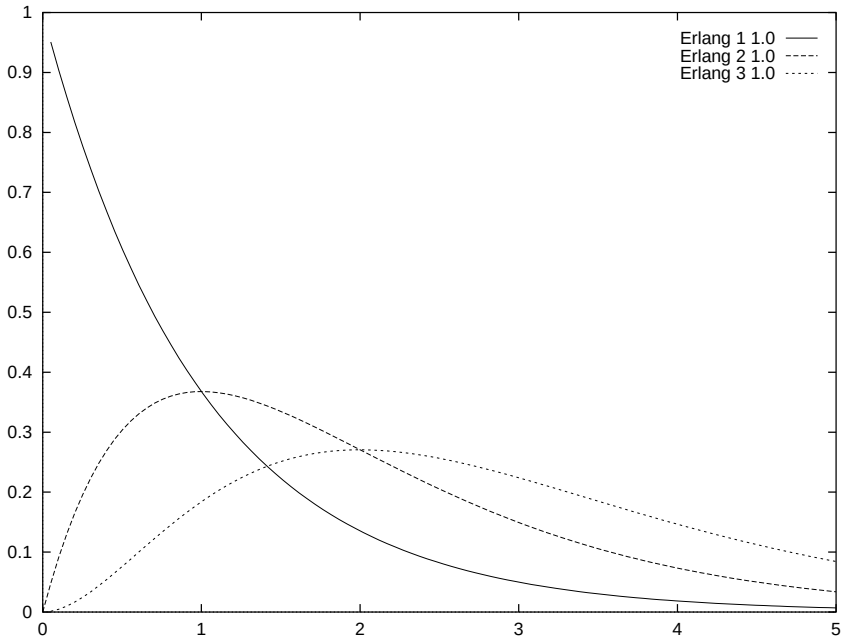

Figure 26: Erlang distributions with parameter $\lambda = 2$ and r consecutively 1, 2 and 3.

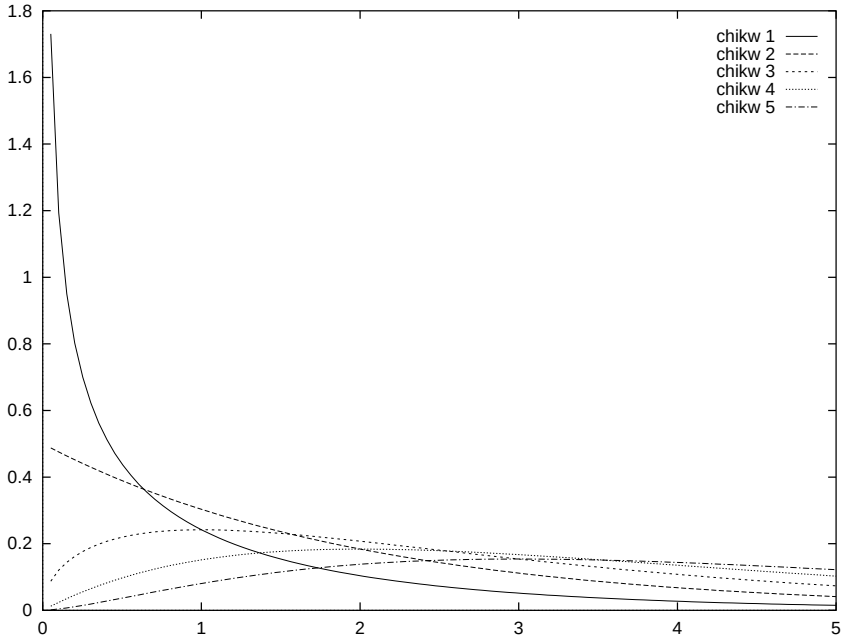

Figure 27: χ^2 distributions with $v = 1$, $v = 2$, $v = 3$, $v = 4$ and $v = 5$.

D.8 Beta distribution

Beta distribution

a, b, r, s

Like the uniform distribution, the *beta distribution* is distributed over a finite interval. We use it for random variables having a clear upper and lower bound. The beta distribution has four parameters a, b, r and s. The parameters a and b represent the upper and lower bounds of the distribution. The parameters r ($r > 0$) and s ($s > 0$) determine the shape of the distribution. For each x ($a \leq x \leq b$), the probability density is defined as follows:

$$f_X(x) = \frac{1}{b-a} \frac{\Gamma(r+s)}{\Gamma(r)\Gamma(s)} \left(\frac{x-a}{b-a}\right)^{r-1} \left(\frac{b-x}{b-a}\right)^{s-1}$$

Standard beta distribution

If $a = 0$ and $b = 1$ we call this a *standard beta distribution*. Figures 28, 29 and 30 show, assuming a standard beta distribution, for a number of values of r and s the corresponding probability densities. Clearly, the beta distribution is very varied.

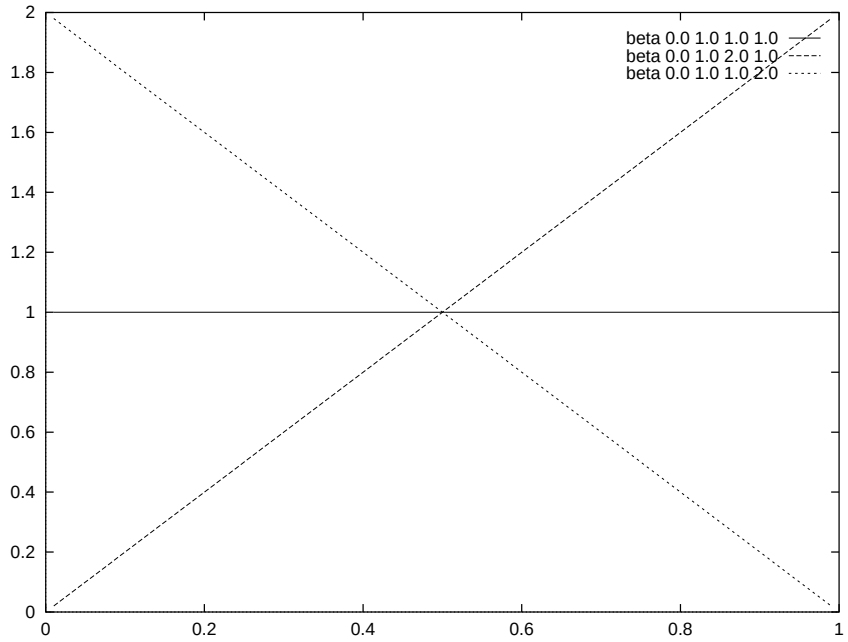

Figure 28: Beta distributions with parameters $a = 0$, $b = 1$ and r consecutively 1.0, 2.0 and 1.0, and s consecutively 1.0, 1.0 and 2.0.

If $r = s = 1$, X is distributed homogeneously with parameters a and b. The larger/smaller $\frac{r}{s}$ becomes, the more/less the chance of a value close to b. If $r < 1$, the probability density is large close to a. If $s < 1$, the probability density is large close to b. If $r > 1$ and $s > 1$, the modal value (the maximum probability density) lies somewhere between a and b. The expectation equals:

$$\mathbb{E}[X] = a + (b-a)\frac{r}{r+s}$$

The variance equals:

$$\mathrm{Var}[X] = \frac{rs(b-a)^2}{(r+s)^2(r+s+1)}$$

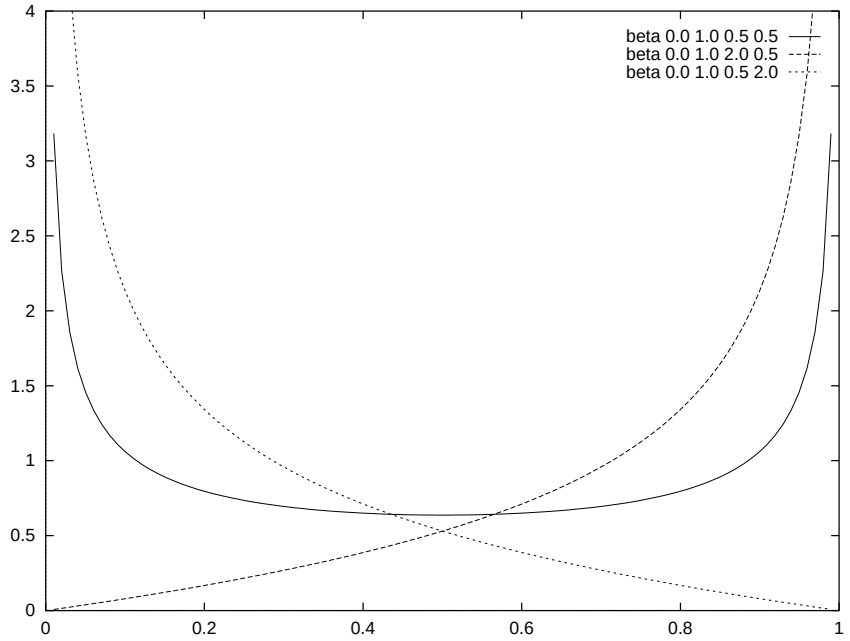

Figure 29: Beta distributions with parameters $a = 0$, $b = 1$ and r consecutively 0.5, 2.0 and 0.5, and s consecutively 0.5, 0.5 and 2.0.

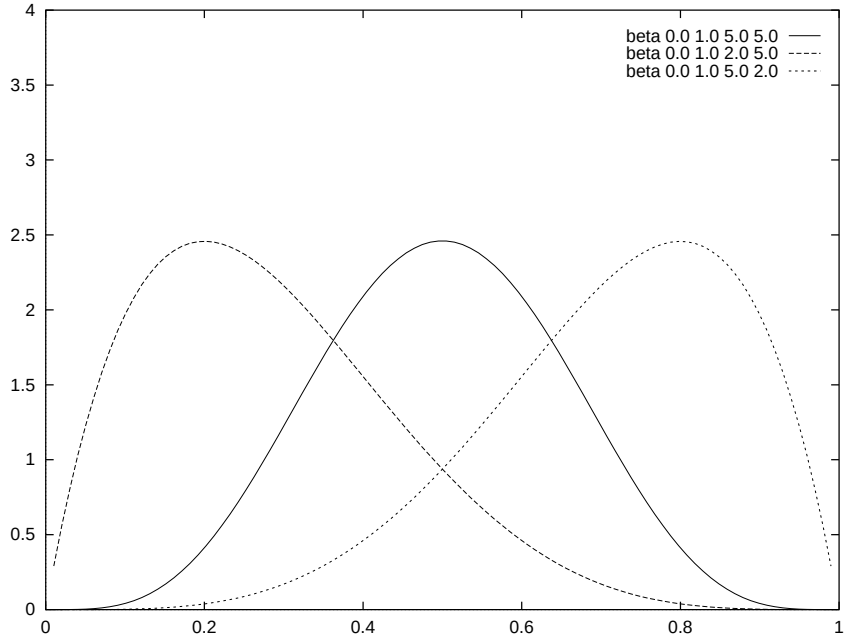

Figure 30: Beta distributions with parameters $a = 0$, $b = 1$ and r consecutively 5.0, 2.0 and 5.0, and s consecutively 5.0, 5.0 and 2.0.

A beta distribution is 'skewed' (not symmetrical) whenever $r \neq s$. Therefore, the most probable value (modal value) may differ from the expectation. If $r > 1$ and $s > 1$, the modal value equals $a + (b-a)\frac{r-1}{r+s-2}$. If $r < 1$ or $s < 1$, the maximal probability density is achieved at b (if $r < s$) or a (if $s < r$).
Suppose we want a standard beta distribution with an expectation μ and a variance σ^2. In that case we have to choose r and s as follows:

$$\begin{aligned} r &= \frac{\mu^2(1-\mu)}{\sigma^2} - \mu \\ s &= \frac{(1-\mu)^2\mu}{\sigma^2} - (1-\mu) \end{aligned}$$

This is only possible if the following condition is met:

$$\sigma^2 < \mu(1-\mu)$$

As each sample is between 0 and 1, there is an upper bound to the variance (at most 0.25).
A beta distributed random variable Y with lower bound a, upper bound b, expectation μ and variance σ^2, is obtained by constructing a random variable X, distributed standard beta with expectation $\frac{\mu-a}{b-a}$ and variance $\frac{\sigma^2}{(b-a)^2}$. Y is then defined by $Y = (b-a)X + a$.

PERT

A well-known application of the beta distribution is *PERT* (Program Evaluation and Review Technique). PERT is a technique used by project managers assess throughput times. For each activity PERT needs three estimates of its duration:

(i) an optimistic estimate (i.e. a lower bound),

(ii) a pessimistic estimate (i.e. an upper bound),

(iii) an estimate of the most probable duration (modal value).

If we model the duration of such an activity by means of a beta distribution, we use the first two estimates to determine the parameters a and b. If c is the most probable duration, the parameters r and s are set so that the expectation and the variance take on the following values:

$$\begin{aligned} \mu &= \frac{a + 4c + b}{6} \\ \sigma^2 &= \frac{(b-a)^2}{36} \end{aligned}$$

Therefore, if a lower and upper bound are given, the variance is fixed.

E Random variables

In this section we treat some elementary properties of random variables. X and Y are random variables with respectively expectation μ_X and variance σ_X^2, and expectation μ_Y and variance σ_Y^2

Concerning addition and multiplication of variates, the following universal properties apply:

$$\begin{aligned} \mathbb{E}[X+Y] &= \mu_X + \mu_Y \\ \mathbb{E}[aX+b] &= a\mu_X + b \end{aligned}$$

The variance of a random variable X ($\mathrm{Var}[X] = \sigma_X^2$) can be expressed in terms of the expectation of X and X^2.

$$\mathrm{Var}[X] = \mathbb{E}[(X-\mu_X)^2] = \mathbb{E}[X^2] - \mu_X^2$$

For the variance the following property is important:

$$\mathrm{Var}[aX+b] = a^2\sigma_X$$

The covariance of X and Y ($\mathrm{Cov}[X,Y]$) is also denoted as σ_{XY}^2.

$$\mathrm{Cov}[X,Y] = \mathbb{E}[(X-\mu_X)(Y-\mu_Y)] = \mathbb{E}[XY] - \mu_X\mu_Y$$

If X and Y are independent, then $C[X,Y] = 0$.
There is also a relation between variance and covariance:

$$\mathrm{Var}[X+Y] = \sigma_X^2 + \sigma_Y^2 + 2\mathrm{Cov}[X,Y]$$

So if X and Y are independent, $\mathrm{Var}[X+Y] = \sigma_X^2 + \sigma_Y^2$.

Markov's inequality

If a random variable X only takes on non-negative values, then for each $a > 0$:

$$\mathbb{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$$

Chebyshev's inequality

Given a random variable X with mean μ and variance σ^2, then for each $k > 0$:

$$\mathbb{P}[|X-\mu| \geq k\sigma] \leq \frac{1}{k^2}$$

Central limit theorem

For a set $X_1, X_2, \ldots, X_n$ of independent uniformly distributed random variables with expectation μ and variance σ^2, the random variable

$$\frac{(X_1 + X_2 + \ldots + X_n) - n\mu}{\sigma\sqrt{n}}$$

converges for $n \to \infty$ to a standard normal distribution.
Thus, the sum of a large number of independent random variables is approximately normally distributed. In interpreting simulation results we can assume (for $n > 10$) that the sum of n independent random variables X_i with expectation μ and variance

σ^2, is approximately normally distributed with expectation $n\mu$ and variance $n\sigma^2$. A similar statement holds for the mean of the X_i.

The extent of the normal distribution

The central limit theorem above shows the importance of the normal distribution for the interpretation of simulation results. Given a normal distribution with parameters μ (expectation) and σ (standard deviation), the probability of a sample lying between $\mu - \sigma$ and $\mu + \sigma$ is approximately 0.683. The following table shows for a number of intervals surrounding μ the probability of a draw from a normal distribution with parameters μ and σ in this interval.

interval	probability
$[\mu - \frac{\sigma}{2}, \mu + \frac{\sigma}{2}]$	0.383
$[\mu - \sigma, \mu + \sigma]$	0.683
$[\mu - 2\sigma, \mu + 2\sigma]$	0.954
$[\mu - 3\sigma, \mu + 3\sigma]$	0.997

So, the probability that a certain draw is between $\mu - 3\sigma$ and $\mu + 3\sigma$ is approximately 0.997.

F Queuing models

Queuing models

So-called analytical models can be analyzed directly without simulation. A well-known class of such models are the so-called *queuing models*. Here, we only treat single queue models. Important results have been derived for networks of queues (see e.g. Baskett et al. [1] and Marsan et al. [8]).

$A/B/c$-notation

A queue is often characterized by $A/B/c$ where A refers to the arrival process, B to the distribution of the serving times and c to the number of parallel identical servers. The letter M is used to indicate a negative exponential distribution. The letter E_r refers to an Erlang distribution. G denotes an arbitrary distribution.
One of the simplest queuing models is the $M/M/1$ queue, i.e. a queue with a Poisson arrival process (the interarrivals are negative exponentially distributed), negative exponentially distributed serving times and only 1 server (at most one customer is served at a time).

Little's formula

Before presenting some results, we will state *Little's formula*, which applies to each queue in a stable state without dependencies between the arrival process and the serving process. The formula is:

$$L = \lambda S$$

where L is the mean number of customers in the system, λ the mean number of customers arriving per time unit and S the mean system time (i.e. the time that customers stay within the system on average, so the sum of the waiting time and the serving time).

$M/M/1$ queue

The $M/M/1$ queue is specified by two parameters λ and μ. For the arrival process the

negative exponential distribution with parameter λ is used. The mean interarrival time is thus $\frac{1}{\lambda}$. For the serving process a negative exponential distribution with parameter μ is used. The mean serving time is thus $\frac{1}{\mu}$. The occupation rate ρ is:

$$\rho = \frac{\lambda}{\mu}$$

We can represent the state of the queue by the integer k that represents the total number of customers in the system. The probability that the queue is in state k at a given moment, is denoted as p_k:

$$p_k = (1-\rho)\rho^k$$

These probabilities are also called the *steady-state probabilities*. The mean number of customers in the system is L:

$$L = \frac{\rho}{1-\rho}$$

The mean systems time is S:

$$S = \frac{1}{(1-\rho)\mu}$$

The mean waiting time W is the difference between the systems time and the serving time:

$$W = \frac{\rho}{(1-\rho)\mu}$$

$M/E_r/1$ queue

The $M/E_r/1$ queue is specified by three parameters λ, r and μ. The serving time now is distributed Erlang with parameters λ and r. For the occupation rate ρ, the mean number of customers L, the mean system time S and the mean waiting time W, we find the following values:

$$\begin{aligned}
\rho &= \frac{r\lambda}{\mu} \\
L &= \frac{\lambda\rho(r+1)}{2\mu(1-\rho)} + \rho \\
S &= \frac{\rho(r+1)}{2\mu(1-\rho)} + \frac{r}{\mu} \\
W &= \frac{\rho(r+1)}{2\mu(1-\rho)} + \frac{r-1}{\mu}
\end{aligned}$$

$M/G/1$ queue

Less is known about the $M/G/1$ queue Once again, the arrival process is specified by the parameter λ. The mean serving time is $\frac{1}{\mu}$ and the variance of the serving time is σ^2. The variance of the serving time C is defined as follows: $C = \sigma\mu$. For

the occupation rate ρ, the mean number of customers L, the mean system time S and the mean waiting time W we find the following values:

$$\begin{aligned} \rho &= \frac{\lambda}{\mu} \\ L &= \rho + \frac{\rho^2}{2(1-\rho)}(1+C^2) \\ S &= \frac{1}{\mu} + \frac{\rho}{2\mu(1-\rho)}(1+C^2) \\ W &= \frac{\rho}{2\mu(1-\rho)}(1+C^2) \end{aligned}$$

The first formula is also known as Pollaczek-Khinchin's formula.

$M/G/\infty$ queue

The $M/G/\infty$ queue is specified by two parameters λ and μ. The variance of the distribution G is not relevant. Because there are always a sufficient number of free servers in the $M/G/\infty$ queue, the occupation rate cannot be defined (actually, it is 0). We now equate ρ with the amount of work arriving per time unit: $\rho = \frac{\lambda}{\mu}$. The steady-state probabilities p_k now are:

$$p_k = \frac{\rho^k}{k!}e^{-\rho}$$

For the mean number of customers L, the mean system time S and the mean waiting time W, we find the following values:

$$\begin{aligned} L &= \rho \\ S &= \frac{1}{\mu} \\ W &= 0 \end{aligned}$$

For more information on this subject we refer you to Kleinrock [6] and Gross and Harris [3].