Analysis of Manufacturing Systems

using Chi 3.0

I.J.B.F. Adan, A.T. Hofkamp, J.E. Rooda and J. Vervoort

Oct 2012

Technische Universiteit Eindhoven, Department of Mechanical Engineering
Systems Engineering Group, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

http://se.wtb.tue.nl/
a.t.hofkamp@tue.nl

Contents

Acknowledgements

This book is the result of years of research within the Systems Engineering Group of the Department of Mechanical Engineering of the Eindhoven University of Technology. Credits for the design of the language go to all members of the χ-club. In particular, we would like to mention the following contributions. J.M. van de Mortel-Fronczak stood at the base of the language [vdMFR95]. N.W.A. Arends was the first to describe the formalism [Are96]. W. Alberts and G.A. Naumoski wrote the first compiler (χ0.3) [AN98]. A.T. Hofkamp wrote the real-time compiling system [Hof01] and the latest compiler (Chi 3). V. Bos and J.J.T. Kleijn helped in providing a formal framework for χ [BK00]. D.A. van Beek gave suggestions for improving the language. Finally, close cooperation of the Systems Engineering Group with the Parallel Programming Group (formerly lead by M. Rem) and the Formal Methods Group (lead by J.C. Baeten) has improved the quality of the language.

In addition, we would like to thank all (former) students and (former) members of the Systems Engineering Group that have performed several projects involving the modelling and analysis of manufacturing systems using Chi in research and engineering environments.

Regarding this book, we thank L.F.P. Etman for his contribution to Chapter 3 and for preparing a number of exercises. We thank E.J.J. van Campen for providing us with photographs of the Crolles2 waferfab.

Chi 3

This edition is based on the language Chi 3. The original text has been modified, with minimal effort, to make the text suitable for Chi 3.

Some chapters have been deleted from this edition.

Chapter 1

Introduction

The title of this book is analysis of manufacturing systems. In this chapter we mark out the main focus of the book.

1.1 Manufacturing Systems (MSs)

Manufacturing stems from the Latin words manus (hand) and factus (make). Nowadays, by manufacturing we mean the process of converting raw material into a physical product (we do not consider services to be manufacturable). A system is a collection of elements forming a unified whole. Thus, a manufacturing system refers to any collection of elements that converts raw material into a product. We can identify manufacturing systems at four different levels.

- At factory level the manufacturing system is the factory (also referred to as plant, fabricator, or shortly fab). The elements of the system are areas and (groups of) machines.
- At area level the manufacturing system is an area of the factory with several machines or groups of machines (cells). The elements of the system are individual machines.
- At cell level the manufacturing system is a group of machines, that are typically scheduled as one entity.
- At machine level, the manufacturing system is the individual machine (also referred to as equipment or tools). The elements of the system are machine components.

These levels constitute a hierarchical breakdown of the factory, see Figure 1.1.

In this book we will focus on manufacturing systems at area and cell level. Though, many parts of this book are also applicable to analysis of entire factories or individual machines. For instance, the analysis of the output (for instance the number of products processed per hour) of a factory shows many similarities to analyzing the output of an area, or the output of machine. Scheduling tasks at machine level is in many ways analogous to scheduling orders at factory level or scheduling jobs at area level.

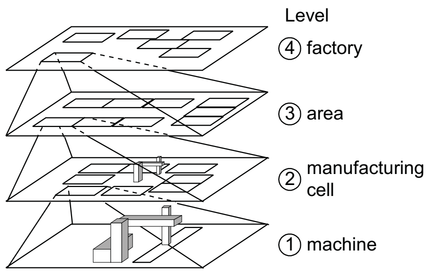


Figure 1.1: Manufacturing systems regarded at different levels

The types of manufacturing systems that can be analyzed using the approach described in this book are as diverse as there are products. Some examples are:

- a wafer fab (factory level), the lithography area (area level), a waferstepper (machine), see Figures 1.2 and 1.3,
- a CD-R fab (factory level), the CD-R mastering area (area level), a CD-R printing machine(machine level), see Figure 1.4,
- an automotive assembly factory (factory level), the bodywork assembly line (area level), the welding robots (cell level), a single welding robot (machine level), see Figure 1.5.

Figure 1.2: Lithographic area of the Crolles2 waferfab in Grenoble, France

Figure 1.3: Waferinspection machine and waferstepper in the lithographic area of the Crolles2 waferfab

Figure 1.4: CD optical mastering area, CD printing machine, CD packaging machine (source: http://www.sonopress.de)

Figure 1.5: Automotive assembly line (source: http://www.mazda.com)

In this book we consider different types of manufacturing systems. Depending on the specific industry we are in, we see for example shoes, a pod of wafers, or a printed-circuit board moving through the factory. Throughout this book we use the term *lot* to refer to the common transfer unit in the manufacturing system. If a machine processes multiple lots at the same time, we say the machine processes *batches*. A batch is a set of lots.

1.2 Analysis of Manufacturing Systems (AMS)

Some typical design questions when designing or optimizing a manufacturing system are:

- What machines do I choose? Do I choose a cheap machine that breaks down often, or a more expensive machine, that breaks down less often?
- How many machines do I use? Do I choose a single high-capacity machine, do I choose multiple low-capacity machines in parallel, or a combination?
- What configuration of workstations do I choose? Do I choose a flowline, a jobshop, or something in between?
- Do I want buffers and if so of what size?
- What control strategy do I use for my plant? Do I release new orders at a fixed rate and if so at what rate? Do I release orders according to a precalculated schedule or do I release orders based on the plant status?

This book does not go into much detail on the best answers, but presents a way to model and analyze a manufacturing system to help you answer these questions yourself. We analyze manufacturing systems with one objective: to evaluate and compare alternative manufacturing system designs and decide on the best alternative.

Question is: on what criteria do we evaluate and compare manufacturing systems? Many different performance criteria exist. Some commonly used performance indicators are:

- throughput δ: the number of lots per time-unit that leave the manufacturing system,
- flowtime φ: the time it takes a lot to travel through the system,
- coefficient of variation on the flowtime (c_φ): the amount of variability on the flowtime of a lot,
- utilisation u: the fraction of time a machine is processing lots, and
- work-in-process or wip-level w: the number of lots in the manufacturing system.

More on these and other performance indicators can be found in [GH85], [Tij94], [HS01] and in the remainder of this book.

In order to analyze a (design of a) manufacturing system, we need a way to model it and to determine its performance. We can make many different types of models. Often it is possible to make a rough estimation of the plant performance without requiring advanced modelling, see for example [HS01]. In the early stages of the design process we often can not do better than rough estimates, since we do not have enough detailed data. As we collect more data, it is possible, for simple plants, to do some more accurate calculations with simple queueing equations, see again [GH85, Tij94]. However, these queueing equations have a limited range of application and validity. If plants become more complex, we require advanced queuing theory (as can be found in for example [GH85, BS93, Tij94, AR01]) and stochastic process theory. However, these models require extensive mathematical knowledge, substantial effort, and even these models have a limited range of applicability. Here, we can no longer use analytical models[1] and we enter the field of (discrete-event) simulation. Figure 1.6 shows the stage in the design process, the amount of data needed, and the range of applicability related to the analysis methods mentioned.

Many commercial simulation packages are available that use computer power to calculate and predict the performance of the plant. These packages offer ready-to-use parametric building blocks of plant components such as machines and buffers. Examples are Flexsim [Fle02], Witness [Lan02], Automod [Bro03], Simscript II.5 [Cac02], and Extend 5.0 [Ima02]. However, when plants become more complex and components become more unconventional (due to for example batching policies, control strategies, specific routings, uncommonly used distributions), substantial low-level programming (in for example C or C++) is required and modelling and simulation become time consuming. Then, also these packages reach the boundaries of their applicability.

In this book, we propose a way to analyze manufacturing systems using the formalism Chi. The Chi-language can be used for modelling and simulating simple manufacturing systems, but is especially valuable for modelling and simulating complex manufacturing systems. It offers a high degree of freedom, without requiring extensive programming effort.

[1]By analytical models we mean mathematical models that can (sometimes only after a significant amount of effort) be explicitly solved and yield an exact solution.

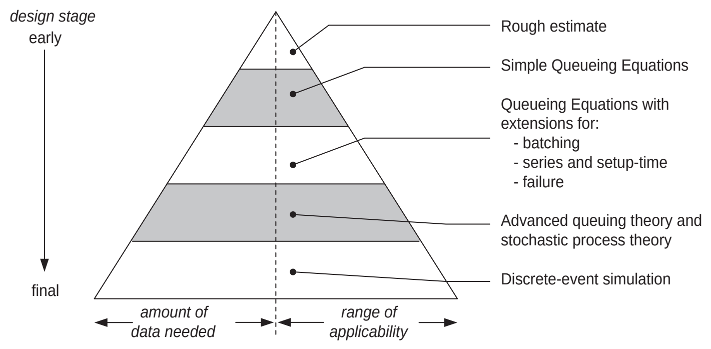

design stage
early
final
Rough estimate
Simple Queueing Equations
Queueing Equations with extensions for:
- batching
- series and setup-time
- failure
Advanced queuing theory and stochastic process theory
Discrete-event simulation
amount of data needed
range of applicability

Figure 1.6: Different analysis methods

Chi is a high-level formalism, tailored to describe concurrent behaviour (which typically is the case in plants). The Chi-language is based on computer science and mathematics [BW90, Fok00, Bru97]. It has been used in numerous cases to analyze the performance and behaviour of complex manufacturing systems. Examples are semi-conductor factories, chemical plants, automobile assembly lines, and factories for lithographic equipment as well as lithographic equipment itself, wafer furnaces, or a chemical clinical analyzer.

On reading this book

Readers familiar with the Chi-language can simply start at Chapter 2 and go from there. Readers unfamiliar with the Chi-language can familiarize themselves with it, by working through the "Chi 3 tutorial" [AHR12]. In the beginning of each chapter in this book, we indicate what elements of the Chi-language are required to understand the models that are presented.

Each chapter contains a number of exercises. Answers to some of these exercises are included in the back of this book.

Outline

The structure of this book is as follows. In Chapter 2 we discuss easy-to-use analytical ways to analyze a deterministic manufacturing system. In Chapter 3 we discuss analytical approaches to analyze stochastic manufacturing systems, for example with stochastic process times.

In Chapter 4 we start with modelling a single machine flowline. We present a basic model of a buffer and a machine. Then, in Chapter 5, we present a way to measure performance of a manufacturing system by simulation, amongst others, how to measure mean throughput, flowtime and utilisation. In Chapter 6, we continue with analyzing flowlines of multiple machines with and without buffering. Subsequently, in Chapters 7 and Chapter 8, we discuss the modelling and analysis of networks of machines, for example, parallel machines, jobshops, and re-entrant flowlines. Subsequently, in Chapter 9, we present ways to extend and improve machine models. We discuss aspects like more advanced distributions to represent the process time, adding machine failure and repair, and modelling batching and set-up times. In the final chapters we present a number of industrial cases, in which the Chi-language was used to analyze manufacturing systems.

Chapter 2

Analytical models of deterministic manufacturing systems

In Chapter 1 we saw that there are different ways to analyze a manufacturing system. In this chapter and the following chapter, we provide a brief view on analytical models that can be used to analyze manufacturing systems. We distinguish two classes of analytical models: deterministic and stochastic models.

A variable is deterministic if we know in advance what exact value it has and will have at every instant of time, that is, the value of the variable is predetermined. A constant deterministic variable is simply a constant, e.g. 1.54. A dynamic deterministic variable changes, but the changes are known in advance, for example a fixed sequence: {1,5,1,5,1,5,1,5}. A variable is stochastic if we do not know the exact value of the variable in advance. We only know its value obeys some probability distribution. An example of stochastic variable is a sample of a binomial distribution: {1,0,1,0,1,1,1,0,0,1}.

In this chapter we discuss the principles of manufacturing system analysis and present analytical approaches to analyze deterministic manufacturing systems. In Chapter 3 we present analytical approaches to analyze stochastic manufacturing systems.

In the subsequent chapters we occasionally compare the results of the analytical methods discussed in this and the subsequent chapter with simulation results to investigate the (range of) validity of the analytical approaches in an engineering environment.

2.1 Manufacturing system principles

Before we introduce analytical models of manufacturing systems, we first introduce a few basic quantities and the main principles for manufacturing system analysis.

Basic quantities

In this section we present the following basic quantities: raw process time t_0, throughput δ, flowtime φ, wip (work-in-process) w, and utilisation u. Other quantities exist, but are dealt with when encountered in the remainder of the book. Figure 2.1 illustrates throughput, flowtime, wip, and utilisation for manufacturing systems at factory and machine level.

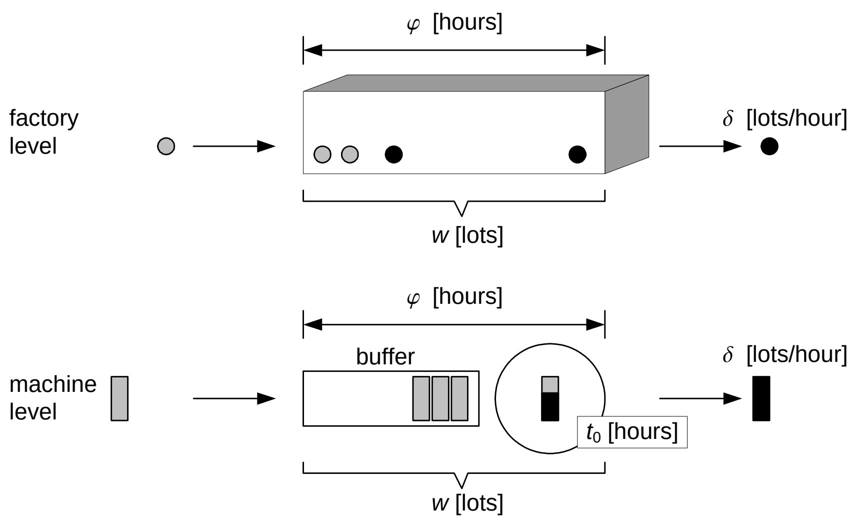


Figure 2.1: Basic quantities for manufacturing systems

- Raw process time t_0 denotes the net time a lot needs processing on a machine. This process time excludes additions such as setup time, breakdown, or other sources that may increase the time a lot is on the machine. The raw process time is typically measured in hours or minutes.
- Throughput δ denotes the number of lots per time-unit that leaves the manufacturing system. At machine level, this denotes the number of lots that leave a machine per time-unit. At factory level it denotes the number of lots that leave the factory per time-unit. The unit of throughput is typically lots/hour.
- Flowtime φ denotes the time a lot is in the manufacturing system. At factory level this is the time from release of the lot in the factory until the finished lot leaves the factory. At machine level this is the time from entering the machine (or the buffer in front of the machine) until leaving the machine. Flowtime is typically measured in days, hours, or minutes.
- Wip-level w denotes the total number of lots in the manufacturing system, i.e. in the factory or in the machine. Wip is measured in lots.
- Utilisation u denotes the fraction a machine is not idle. A machine is considered idle if it could start processing a new lot. Thus processing time as well as downtime, setup-time and preventive maintenance time all contribute to the utilisation. Utilisation has no dimension. Utilisation can never exceed 1.0.

In literature different formulas are presented to compute the utilisation in specific cases. We present only one formula:

$$u = \frac{T_{\text{nonidle}}}{T_{\text{total}}} \tag{2.1}$$

Herein T_{nonidle} denotes the time the machine is not idle during a total timeframe T_{total}. In the following examples we calculate the utilisation for a number of cases.

Example 2.1 Utilisation (1)
A machine has raw process time t_0 of 0.15 hours and processes 5 lots/hour. Calculate the utilisation.

Processing 5 lots takes $5 \times 0.15 = 0.75$ hours. Every hour the machine is idle for 0.25 hours, hence utilisation $u = 0.75$. ⊠

Example 2.2 Utilisation (2)
A machine has raw process time $t_0 = 0.15$ hours. Lots arrive at a fixed rate of 5.0 lots/hour. After every fourth lot, the machine needs recalibrating. Recalibration takes 0.10 hour. Calculate the utilisation.

To process four lots, the machine is occupied for $4 \times 0.15 + 0.10 = 0.70$ hours. These lots are processed over a total time of $4/5.0 = 0.80$ hours. Hence the mean utilisation is $0.7/0.8 = 0.875$. ⊠

Example 2.3 Utilisation for batch processing
A machine processes lots in fixed batches[1]. Processing a batch takes 3 hours. After every batch the machine needs to be cleaned and prepared for the next batch, which takes 1 hour. Lots arrive at a rate of 2 lots/hour. Calculate the utilisation for a batch size of 10 and 12 lots respectively.

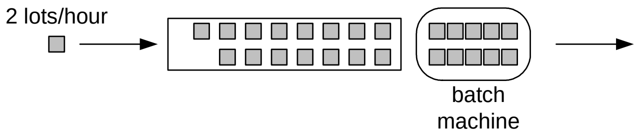


Figure 2.2: Batch machine processing in batches of 10 lots

In case of a batch size of 10 lots, the machine processes a batch every 5 hours. For every batch the machine is occupied for $3 + 1 = 4$ hours. The machine is idle for 1 hour every 5 hours, the utilization $u = 4/5 = 0.80$. In case of a batch size of 12 lots, the machine processes a batch every 6 hours. The utilisation is now $u = 4/6 = 0.67$. ⊠

First principle: Conservation of mass

The first principle for manufacturing systems is conservation of mass. Similar as for fluid flow systems, conservation of mass applies to manufacturing systems. Figure 2.3(a) shows a workstation (buffer and machine). Herein λ_1 denotes the arrival rate in lots/hour, δ_1

[1] A batch is a set of lots processed together.

denotes the throughput in lots/hour and w_1 denotes the number of lots in the workstation. In Figure 2.3(b), the total throughput δ_1 of workstation 1 splits into two flows, with throughput rates $\delta_{1,1}$ and $\delta_{1,2}$ respectively. In Figure 2.3(c), $\lambda_{2,1}$ and $\lambda_{2,2}$ together form the total arrival rate of workstation 2.

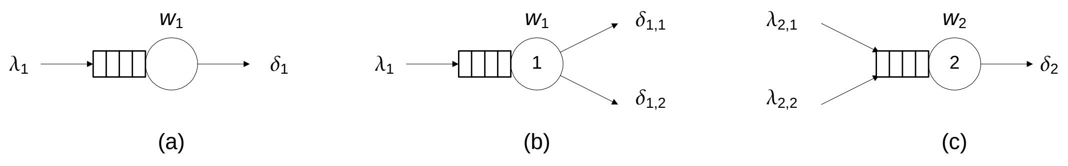


Figure 2.3: Three workstations

Conservation of mass for Figure 2.3(a) yields:

$$\frac{dw_1}{dt} = \lambda_1 - \delta_1 \tag{2.2}$$

This differential form of mass conservation states that the change in the number of lots dw/dt in workstation 1 is equal to the number of lots entering workstation 1 minus the the number of lots leaving workstation 1 per unit of time. On many occasions we consider systems in steady state. We say a manufacturing system is in *steady state* when the mean number of lots in the system is constant, that is, there are no lots piling up in the system, or $dw/dt = 0$. In this case, the mean number of lots entering the manufacturing system equals the mean number of lots leaving the system. Equation 2.2 becomes:

$$\lambda_1 - \delta_1 = 0 \tag{2.3}$$

Herein λ_1 and δ_1 represent the mean arrival rate and throughput in steady state. For the workstations in Figure 2.3(b) and (c), mass conservation in steady state yields:

$$\lambda_1 - \delta_{1,1} - \delta_{1,2} = 0 \qquad \lambda_{2,1} + \lambda_{2,2} - \delta_2 = 0 \tag{2.4}$$

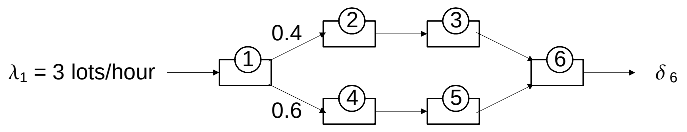
We denote the total throughput of workstation 1 as $\delta_1 = \delta_{1,1} + \delta_{1,2}$. Moreover, we have considered workstations here, but we can just as well consider factories or areas.

Example 2.4 Manufacturing areas
Figure 2.4 shows a factory consisting of 6 areas and the raw processing time of every area. Area 1 receives 3 lots/hour. After being processed in Area 1, 40% of all lots are processed in Area 2 and 3, and 60% of all lots are processed in Area 4 and 5. Finally all lots are processed in Area 6. Consider the factory in steady state and calculate the throughput of Area 2 and 4 and the utilisation for all 6 areas.

The throughput rates for Areas 2 and 4 follow directly from the given probability data:

$$\delta_2 = 0.4 \cdot \lambda_1 = 0.4 \cdot 3 = 1.2 \text{ lots/hour}$$
$$\delta_4 = 0.6 \cdot \lambda_1 = 0.6 \cdot 3 = 1.8 \text{ lots/hour}$$

Area 1 processes 3 lots/hour. Processing 3 lots per hour in Area 1 takes $3 \cdot 0.2 = 0.6$ hours. Every hour, the area is occupied for 0.6 hours, hence utilisation $u_1 = 0.6$.



Area	t_0 [hrs]
1	0.2
2	0.6
3	0.4
4	0.3
5	0.4
6	0.3

Figure 2.4: Material flow in six manufacturing areas

Area 2 processes 1.2 lots/hour. Processing 1.2 lots in Area 2 takes $1.2 \cdot 0.6 = 0.72$ hours, hence utilisation $u_2 = 0.72$.
In a similar way, we calculate the utilisation for the remaining areas. $u_3 = 0.48$, $u_4 = 0.54$, $u_5 = 0.72$, and $u_6 = 0.9$. As all utilisations are smaller than 1, lots do not pile up in this system. ⊠

Example 2.5 Rework
Consider the manufacturing system in Figure 2.5(a). The manufacturing system consists of two infinite buffers (B_1 and B_2) and two machines (M_1 and M_2). 50% of the lots processed by machine 1 need to be reprocessed by machine 1. 25% of the lots processed by machine 2, have to be reprocessed by machine 1 and 2 again. Lots arrive at machine 1 with a fixed arrival rate of 3 lots/hour. Calculate (a) the number of lots that machine 1 processes per hour and (b) the number of lots that leave the system via machine 2. Assume that the system is in steady state.

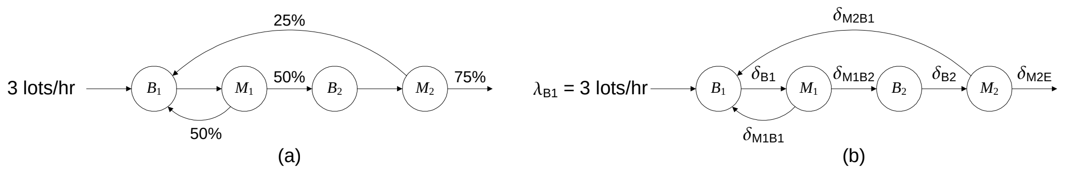


Figure 2.5: Manufacturing line with rework

The total throughput of machine 1 splits into two flows. We introduce throughput $\delta_{M_1B_1}$ and $\delta_{M_1B_2}$, denoting the number of lots/hour that are sent from machine M_1 back to the first buffer B_1 and from machine M_1 forward to the second buffer B_2 respectively. We introduce additional throughput rates, see Figure 2.5(b). Throughput δ_{M_2E} denotes the number of lots per hour that exit the system after being processed by machine M_2. From mass conservation and the probability data, we obtain the following equations:

$$\begin{array}{lll}
B_1 & \lambda_{B_1} + \delta_{M_1B_1} + \delta_{M_2B_1} - \delta_{B_1} = 0 & (1) \\
M_1 & \delta_{M_1B_2} = \delta_{M_1B_1} = 0.5\delta_{B_1} & (2) \\
B_2 & \delta_{B_2} = \delta_{M_1B_2} & (3) \\
M_2 & \delta_{M_2E} = 0.75\delta_{B_2} & (4a) \\
 & \delta_{M_2B_1} = 0.25\delta_{B_2} & (4b)
\end{array}$$

Filling in (2), (3), and (4) in (1) leads to:

$$\begin{array}{l}
\lambda_{B_1} + 0.5\delta_{B_1} + 0.25\delta_{B_2} - \delta_{B_1} = 0 \\
\lambda_{B_1} + 0.5\delta_{B_1} + 0.25\delta_{M_1B_2} - \delta_{B_1} = 0
\end{array}$$

$$\begin{aligned}&\lambda_{B_1} + 0.5\delta_{B_1} + 0.125\delta_{B_1} - \delta_{B_1} = 0\\ &\delta_{B_1} = \tfrac{8}{3}\lambda_{B_1} = 8 \text{ lots/hour}\end{aligned}$$

Thus 8 lots/hour enter machine M_1. In steady state, M_1 processes 8 lots/hour. Of these 8 lots/hour, 4 lots/hour proceed to buffer B_2 and 4 lots/hour are lead back to buffer B_1.

The number of finished lots that leave the system after being processed on machine M_2 is:

$$\begin{aligned}\delta_{M_2E} &= 0.75\delta_{B_2}\\ &= 0.75\delta_{M_1B_2}\\ &= \tfrac{3}{8}\delta_{B_1}\\ &= 3 \text{ lots/hour}\end{aligned}$$

The throughput of this line is 3 lots/hour. Note that the number of lots entering the line is equal to the number of lots leaving the line. Due to rework, machine M_1 has to process 8 lots/hour. ⊠

Second principle: Little's law

The second principle for manufacturing systems is Little's law. Little's law states that the mean wip-level (number of lots in a manufacturing system) w is equal to the product of the mean throughput δ and the mean flowtime φ [Lit61], provided the system is in steady state.

$$w = \delta \cdot \varphi \tag{2.5}$$

This can be intuitively interpreted by looking at Figure 2.6, which treats a manufacturing system as a pipeline where lots pass through. The total amount of fluid in the pipeline in [m^3] is equal to the flowrate in [m^3/s] multiplied by the time it takes an imaginary fluid element to flow through the pipe in [s].

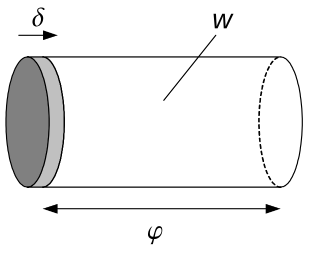


Figure 2.6: Manufacturing system as pipeline

Analogously, the mean number of lots in a manufacturing system (wip-level w in [lots]) is equal to the mean number of lots leaving the system per unit of time (throughput δ in [lots/timeunit]) multiplied by the mean time a lot remains in the system (flowtime φ in [timeunits]). Little's law only applies in steady state.

Example 2.6 Automotive factory
We study the automotive factory in Figure 2.7. By simple measurements we determine that

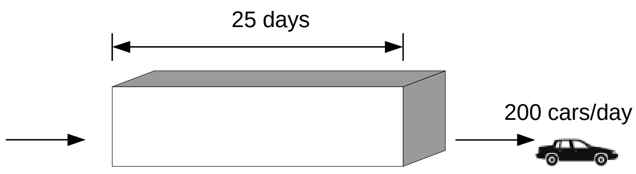
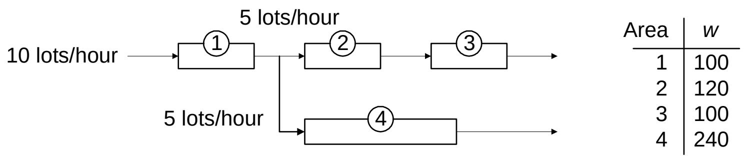


Figure 2.7: Automotive factory

200 cars per day leave the factory. On average a car (or its parts) stay for 25 days in the factory. Calculate the mean number of cars in the factory in steady state.

By Little's law we estimate that the factory contains $w = \delta \cdot \varphi = 200 \cdot 25 = 5000$ cars (or parts for 5000 cars). ☒

Example 2.7 Areas
A factory has four areas. 50% of the lots are processed in Area 1, 2, and 3. The remaining 50% is processed in Area 1 and 4. Through the year, we count the number of (semi-finished) lots in each area. The mean number of lots in each area are shown in Figure 2.8. Moreover, new lots arrive at Area 1 with a an arrival rate $\lambda = 10$ lots/hour. Calculate the mean flowtime per area.



Area	*w*
1	100
2	120
3	100
4	240

Figure 2.8: Factory with four areas

With Little's law we calculate the mean flowtime per area.
$\varphi_1 = w_1/\delta_1 = 100/10 = 10$ hours
$\varphi_2 = w_2/\delta_2 = 120/5 = 24$ hours
$\varphi_3 = w_3/\delta_3 = 100/5 = 20$ hours
$\varphi_4 = w_4/\delta_4 = 240/5 = 48$ hours
☒

2.2 Analysis of deterministic manufacturing systems

In this section we explore the possibility to estimate the performance of a factory (or area), without detailed data and/or extensive calculations. We analyze problems where:

1. the machines have deterministic process times,
2. lots arrive according to a deterministic arrival pattern,
3. the buffers have infinite capacity, and
4. the manufacturing system is in steady state.

Calculating maximum throughput

We discuss an example, in which we determine the maximum throughput for the manufacturing line from Example 2.5. The arrival rate λ is to be chosen. Machines M_1 and M_2 process lots with a raw process time of 0.25 and 0.40 hours respectively. Buffers B_1 and B_2 have infinite capacity. As in Example 2.5, 50% of all lots processed by machine M_1 need rework on machine M_1, and 25% of all lots processed by machine M_2 need rework on M_1 and M_2.

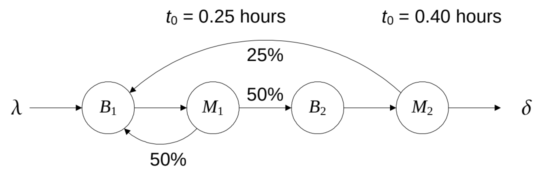


Figure 2.9: Manufacturing line with rework

We express the total throughput of machine 1 and 2 in terms of the arrival rate λ. Using mass conservation and the probability data we determine that machine 1 processes $\frac{8}{3}\lambda$ lots per hour (see Example 2.5). Machine 2 processes $\frac{4}{3}\lambda$ lots per hour.

Next, we express the utilisation per machine in terms of the arrival rate. Machine 1 processes $\frac{8}{3}\lambda$ lots per hour, which takes $\frac{8}{3}\lambda \cdot 0.25 = \frac{2}{3}\lambda$ hours. Utilisation $u_{M1} = \frac{2}{3}\lambda$. The utilisation of machine 2 is $u_{M2} = \frac{1.6}{3}\lambda$.

We conclude that machine 1 is the critical resource, or the *bottleneck*. The maximum throughput is obtained when machine 1 is fully utilized, that is, at an arrival rate of $\lambda = 3/2 = 1.5$ lots/hour. The line has a maximum throughput $\delta_{\max} = 1.5$ lots/hour. The utilisation of machine 2 is then $u_{M2} = \frac{1.6}{3} \cdot 1.5 = 0.8$. Further increasing the release rate λ does not increase the line throughput. Instead, the number of lots in buffer B_1 will keep increasing. Then the system is no longer in steady-state.

Basically we followed the following steps to determine the maximum throughput for a manufacturing system:

1. Assume deterministic process times, deterministic arrival rate, infinite buffers and a steady state system;
2. Express the throughput of each machine in terms of the release rate λ;
3. Express the utilisation of each machine in terms of the release rate λ;
4. Determine the bottleneck, the maximum allowable release rate, and the maximum throughput of the line.

Example 2.8 Maximum throughput of a manufacturing line
Consider the manufacturing line in Figure 2.10. The line consists of machine M_1 with a raw process time of 1.8 hours and 2 parallel machines M_2 with each a raw process time of 4 hours. 5% of the lots processed on machine M_1 need rework on machine M_1. Calculate the maximum throughput of this line.

Using mass conservation we derive the number of lots that machine M_1 has to process:

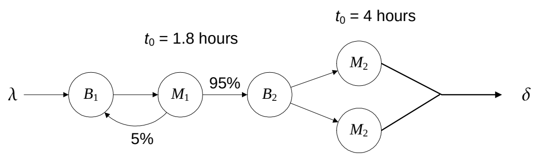


Figure 2.10: Manufacturing line with rework and parallel machines

$$\begin{aligned} \lambda + 0.05 \cdot \delta_{B_1 M_1} - \delta_{B_1 M_1} &= 0 \\ \delta_{B_1 M_1} = \lambda_{M_1} &= \frac{1}{0.95}\lambda \end{aligned}$$

Each machine M_2 has to process 0.5λ lots/hour. The utilisation for machines M_1 and M_2 is:

$$\begin{aligned} u_{M_1} &= 1/0.95 \cdot \lambda \cdot 1.8 = 1.89\lambda \\ u_{M_2} &= 0.5\lambda \cdot 4 = 2\lambda \end{aligned}$$

The two machines M_2 together form the bottleneck. The maximum throughput is attained, if both machines M_2 are fully utilized. This is the case for $\lambda = 0.5$ lots/hour. This manufacturing line has a maximum throughput $\delta_{\max} = 0.5$ lots/hour. ⊠

If we plot the throughput δ and the utilisation u of a deterministic manufacturing system against the arrival rate λ, we obtain diagrams very similar to the ones shown in Figure 2.11.

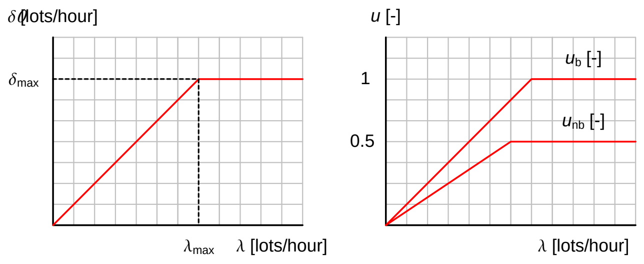


Figure 2.11: Throughput δ and utilisation u against release rate λ

If release rate λ is lower than maximum throughput $\delta_{\max}$, the throughput is equal to the release rate (the system is in steady state). If the release rate is increased beyond $\lambda_{\max}$ ($=\delta_{\max}$), the throughput remains equal to $\delta_{\max}$. The utilisation of the bottleneck u_{b} increases linearly with the release rate up to a maximum of 1. The utilisation of a non-bottleneck station (u_{nb}) also increases linearly with the release rate, but never reaches 1.

Calculating flowtime

Let us consider the manufacturing line in Figure 2.12. We have a two workstation[2] line. Machine M_1 has a raw process time of 2 hours, machine M_2 has a raw process time of 3 hours.

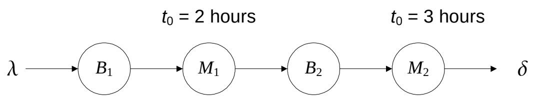


Figure 2.12: Manufacturing line

We assume that the processing times of the machines are deterministic, that the buffers have infinite capacity, and that the system is in steady state. The line in Figure 2.12 has a maximum throughput of 1/3 lots/hour (machine M_2 is the bottleneck).

The basic approach for determining the flowtime of a lot is virtually following a lot from the moment it enters the system until it leaves the system again. In Figure 2.13 we display the state of the system and the lots in it at different time instants. In this way we can follow different lots as they travel through the line. We release lots at the maximum rate of 1/3 lots/hour.

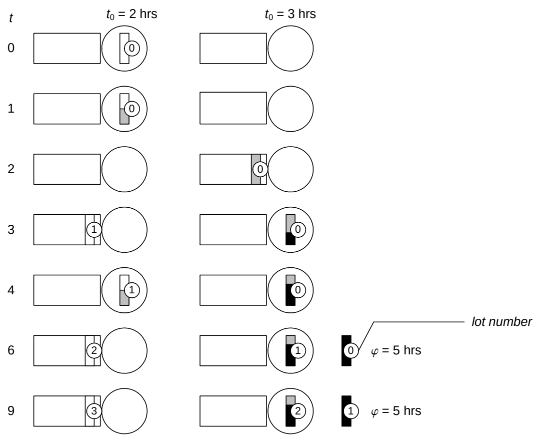


Figure 2.13: State of the manufacturing line at different time instants

From Figure 2.13 we determine the flowtime at 5 hours. Another way to follow lots as they travel trough the line, is with a lot-time-diagram. Herein, we plot the state of individual lots against time. Figure 2.14(a) and (b) show the lot-time-diagram for the line with a release rate of 1/3 and 1/2 lots/hour respectively.

For a release rate of 1/3 lots/hour we determine from Figure 2.14(a) that the flowtime is 5 hours. For a release rate of 1/2 lots/hour we see in Figure 2.14(b) that the flowtime keeps increasing. For any release rate under the maximum throughput the flowtime is 5 hours, for any release rate above the maximum throughput the flowtime grows to infinity.

[2]The term workstation is used here to denote a combination of a buffer and a machine or a buffer and a number of identical machines in parallel.

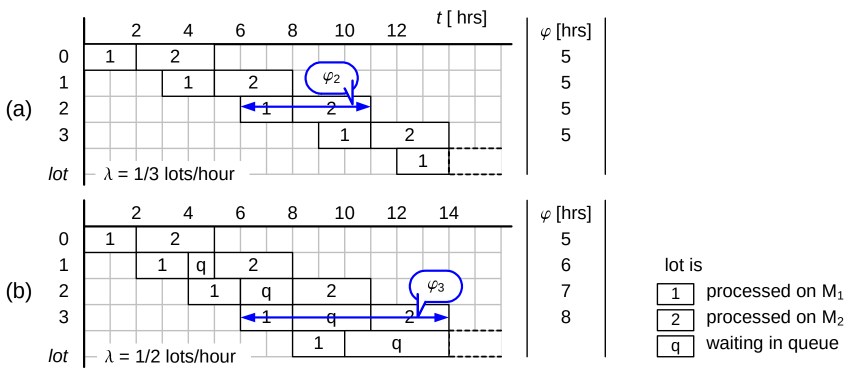


Figure 2.14: Lot-time-diagram for release rate of (a) 1/3 lots/hour and (b) 1/2 lots/hour

We can use the lot-time-diagram also to determine the mean wip-level. In Figure 2.15 we derive the wip-level over time from the lot-time-diagram. For instance, for a release rate of 1/3 lots/hour, at $t = 7$ there are two lots in the system (lot 1 and 2).

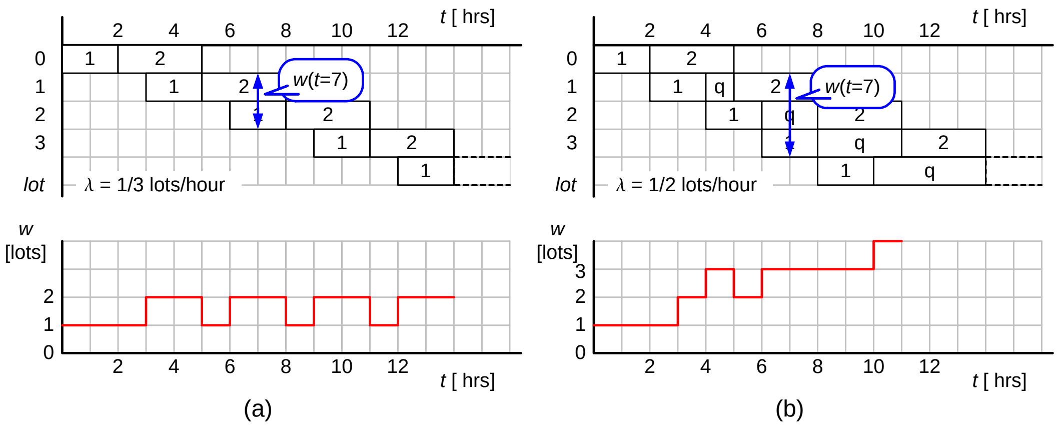


Figure 2.15: Lot-time-diagram and w-t-diagram for release rate of (a) 1/3 lots/hour and (b) 1/2 lots/hour

For a release rate of 1/3 lots/hour, the behaviour becomes periodic for $t > 3$ hours with a period of 3 hours, see Figure 2.15(a). We regard this periodic behaviour as steady state behaviour[3]. The mean wip-level is $\overline{w} = \frac{1}{3} \cdot 1 + \frac{2}{3} \cdot 2 = \frac{5}{3}$ lots. (Try! Is this result in correspondence with Little's law?) For Figure 2.15(b) the wip-level keeps increasing. For a release rate higher than the maximum throughput, the mean wip-level grows to infinity.

[3]We speak of steady state when the mean number of lots in the system is constant. If the number of lots in the system evolves periodically, the momentous number of lots in the system changes, but the mean number of lots in the system is constant.

We now extend the second workstation with a second machine M_2, see Figure 2.16. Workstation 2 is no longer the bottleneck: workstation 1 is now the bottleneck. The line has a maximum throughput of 1/2 lots/hour.

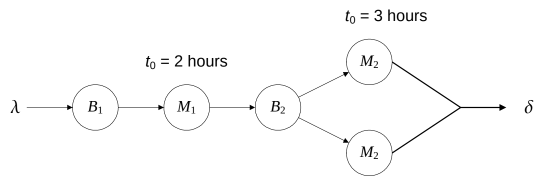


Figure 2.16: Manufacturing line with two parallel machines M_2

To determine the flowtime of lots, we again draw lot-time-diagrams. Figures 2.17(a) and (b) show the lot-time-diagram for a release rate of 1/3 and 1/2 lots/hour respectively.

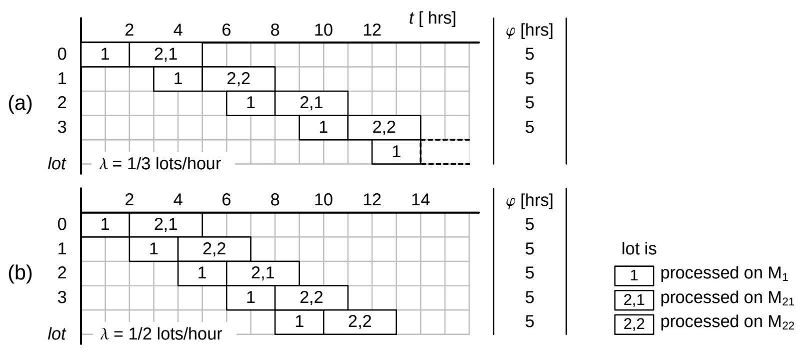


Figure 2.17: Lot-time-diagram for release rate of (a) 1/3 lots/hour and (b) 1/2 lots/hour

We see that for both release rates the flowtime is 5 hours. Doubling the capacity at workstation 2 increases the maximum throughput but does not influence the flowtime!

We can also use the lot-time-diagrams to determine the utilisation of the individual machines. From the lot-time-diagram we derive the state (idle or not idle) of the individual machines, see Figure 2.18.

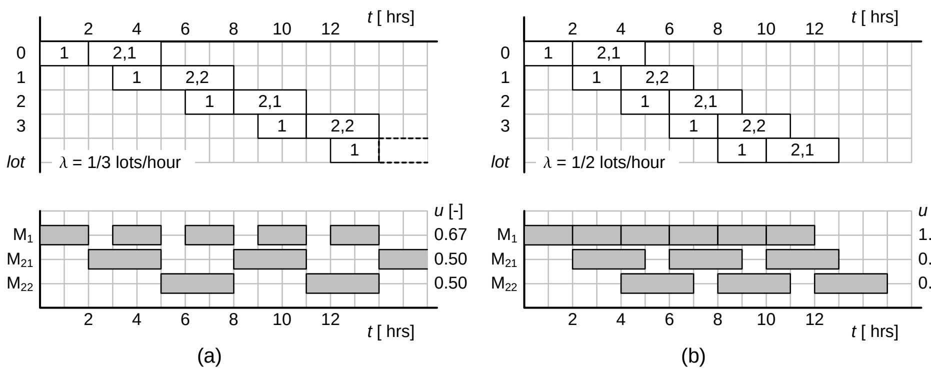


Figure 2.18: Lot-time-diagram and u-t-diagram for release rate of (a) 1/3 lots/hour and (b) 1/2 lots/hour

(Try! From Figure 2.17 derive a wip-time-diagram and determine the mean wip-level for both release rates. Verify the results by using Little's law.)

Example 2.9 Flowtime of a manufacturing line
Consider the manufacturing line in Figure 2.19. The manufacturing line consists of three workstations with 1, 4, and 2 machines respectively. The machines have raw process times of 0.15, 0.8 and 0.35 hours respectively.

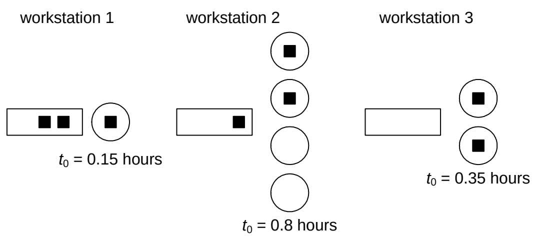


Figure 2.19: Three workstation flowline

(i) Calculate the maximum throughput. (ii) Draw a lot-time-diagram and determine the flowtime at $\lambda = \lambda_{\max}$. (iii) From the lot-time-diagram derive a wip-time-diagram and determine the mean wip-level in steady state. Verify that the results are in correspondence with Little's law.

(i) We calculate the utilisation for each machine in terms of release rate λ. At release rate λ lots/hour, machine 1 is processing 0.15λ hours every hour, $u_1 = 0.15\lambda$. Each machine in workstation 2 has to process $\frac{1}{4}\lambda$ lots/hour, thus each machine is processing for $\frac{1}{4}\lambda \cdot 0.8 = 0.2\lambda$ hours every hour, $u_2 = 0.2\lambda$. Similarly, $u_3 = \frac{1}{2}\lambda \cdot 0.35 = 0.175\lambda$. Workstation 2 is the bottleneck. The maximum throughput is attained when $u_2 = 1$. For $u_2 = 0.2\lambda = 1$, we get $\lambda_{\max} = \delta_{\max} = 1/0.2 = 5$ lots/hour.

(ii) Figure 2.20(a) shows the lot-time-diagram.

From the lot-time-diagram we determine $\varphi = 0.15 + 0.8 + 0.35 = 1.3$ hours.

(iii) Figure 2.20(b) shows the wip-time-diagram derived from the lot-time diagram. From

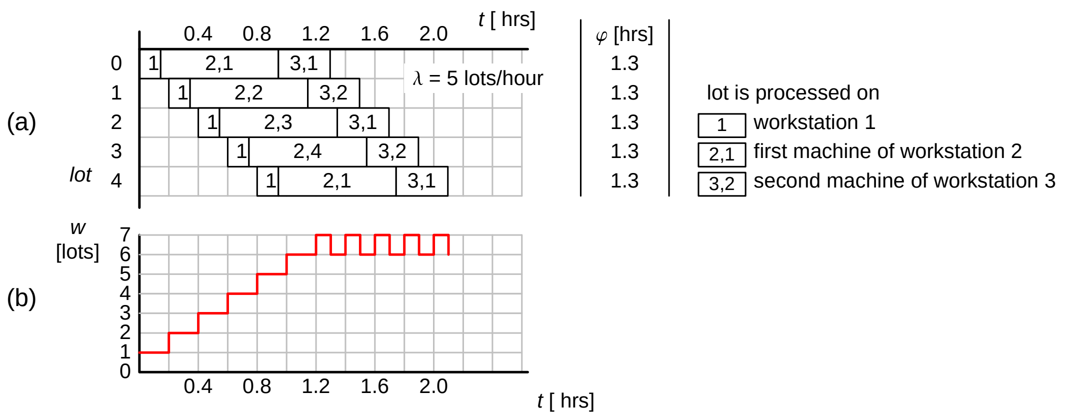


Figure 2.20: Lot-time-diagram and w-t-diagram for release rate of 5 lots/hour

the wip-time-diagram we see that the wip changes periodically for $t > 1.2$ hours. For $t > 1.2$ hours, the system is in steady state. The mean wip level in steady state is $\overline{w} = \frac{1}{2}\cdot 6 + \frac{1}{2}\cdot 7 = 6.5$ lots. Using Little's law we calculate the mean wip-level at $w = \delta \cdot \varphi = 5 \cdot 1.3 = 6.5$ lots. The results correspond. ⊠

For simple deterministic manufacturing lines, like the ones in Figure 2.12, Figure 2.16, and Example 2.9 the flowtime is simply determined. For release rates under the maximum throughput the flowtime is equal to the sum of the process times the lot encounters, and for release rates higher than the maximum throughput, the flowtime grows to infinity, see Figure 2.21.

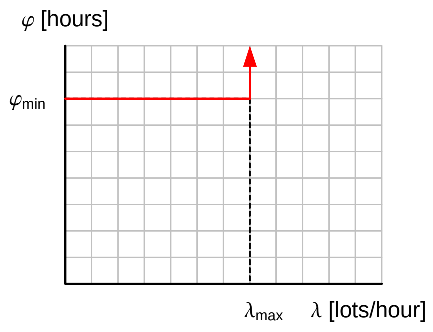


Figure 2.21: Flowtime against release rate λ

In these simple cases, following lots, or constructing a lot-time-diagram is not necessary to determine the flowtime. However, for manufacturing systems with a little more complexity, a lot-time-diagram may be a very useful approach to determine the flowtime. In the sequel, we regard two more complex examples in which we determine the flowtime of a lot by constructing a lot-time-diagram.

Example 2.10 Flowtime for a batch machine
Consider the batch machine in Figure 2.22 which processes batches with a fixed batchsize of $k = 6$ lots. Lots arrive with an arrival rate of λ lots/hour. Processing a batch takes 3

hours. (i) Determine maximum throughput $\delta_{\max}$ of the batch machine. (ii) Determine the (mean) flowtime of a lot for two release rates under $\delta_{\max}$. Is the flowtime independent of the release rate? (iii) Find an expression for the mean flowtime of a lot for a deterministic batch machine with fixed batch size k, batch processing time t_{batch} and release rate λ.

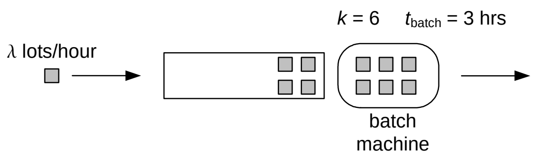


Figure 2.22: Batch machine

(i) The batch machine has to process a batch every $\frac{6}{\lambda}$ hours. Processing a batch takes 3 hours. The utilisation is thus $\frac{3}{6/\lambda} = \frac{1}{2}\lambda$. The utilisation becomes 1 for $\lambda = \lambda_{\max} = 2$ lots/hour.

(ii) To investigate the flowtime we again follow individual lots as they enter and leave the system. Figures 2.23(a) shows the lot-time-diagram for a release rate of $\lambda = 2$ lots/hour.

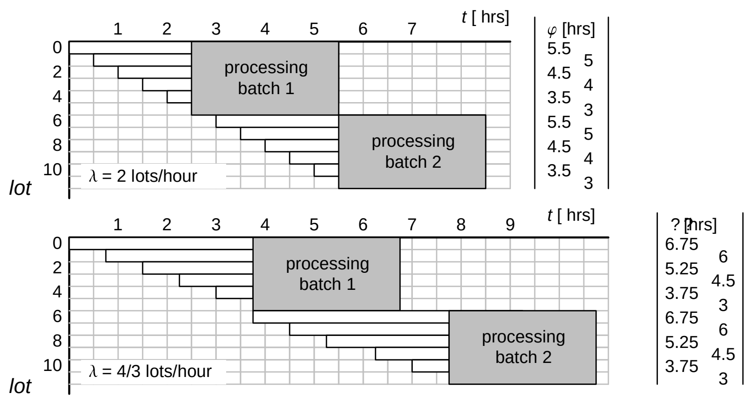


Figure 2.23: Lot-time-diagram for (a) $\lambda = 2$ lots/hour and (b) $\lambda = 4/3$ lots/hour

Note that the flowtime is not the same for all lots. The first lot in a batch has to wait for 5 subsequent lots before it is processed, while the last lot does not have to wait at all. The mean flowtime is (5.5+5+4.5+4+3.5+3)/6 = 4.25 hours. Figure 2.23(b) shows the lot-time-diagram for a release rate of $\lambda = 4/3$ lots/hour. The mean flowtime is now (6.75+6+5.25+4.5+3.75+3)/6 = 4.88 hours. Apparently, for a batch machine the flowtime is no longer independent of the release rate!

(iii) We now consider a batch machine with batch processing time t_{batch} and batch size k. Lots are released at rate λ lots/hour. The flowtime of the first and the last lot in a batch of k lots is respectively:

$$\varphi_{\text{first}} = \varphi_{\text{wait for batch}} + t_{\text{batch}} = \frac{k-1}{\lambda} + t_{\text{batch}}$$
$$\varphi_{\text{last}} = t_{\text{batch}}$$

As long as $\lambda < \lambda_{\max}$, mean flowtime $\overline{\varphi}$ is:

$$\overline{\varphi} = \frac{(k-1)/\lambda + t_{\text{batch}} + t_{\text{batch}}}{2} = \frac{k-1}{2\lambda} + t_{\text{batch}} \tag{2.6}$$

For release rates $\lambda > \lambda_{\max}$, the mean flowtime grows to infinity. $\boxtimes$

Example 2.11 Machine with alternating process times
Consider a two-workstation flowline. Each workstation consists of a single machine and an infinite buffer. Machine 1 has a constant process time of 3.0 hours. Machine 2 has an alternating process time of 1.0 and 5.0 hours, that is, the process time follows the deterministic sequence 1.0, 5.0, 1.0, 5.0, 1.0, ...

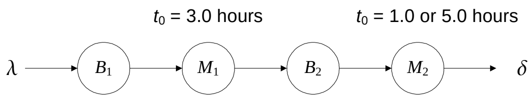


Figure 2.24: Two-workstation flowline with alternating process times for second machine

(i) What is the maximum throughput of this line? (ii) Determine the (mean) flowtime of a lot. (iii) Construct a wip-time-diagram and determine the mean wip-level in steady state. Verify your result with Little's law.

(i) If lots are released at a rate λ lots/hour, machine 1 and 2 have to process 2 lots per $\frac{2}{\lambda}$ hours. Processing two lots requires $2 \cdot 3.0 = 6.0$ hours on machine 1, and $1.0 + 5.0 = 6.0$ hours on machine 2. The utilisation for both machines is $u_1 = u_2 = \frac{6.0}{2/\lambda} = 3\lambda$. Both machines are bottlenecks. The maximum throughput is $\delta_{\max} = \lambda_{\max} = 1/3$ lots/hour.

(ii) We construct a lot-time-diagram for a release rate of 1/3 lots/hour, see Figure 2.25(a).

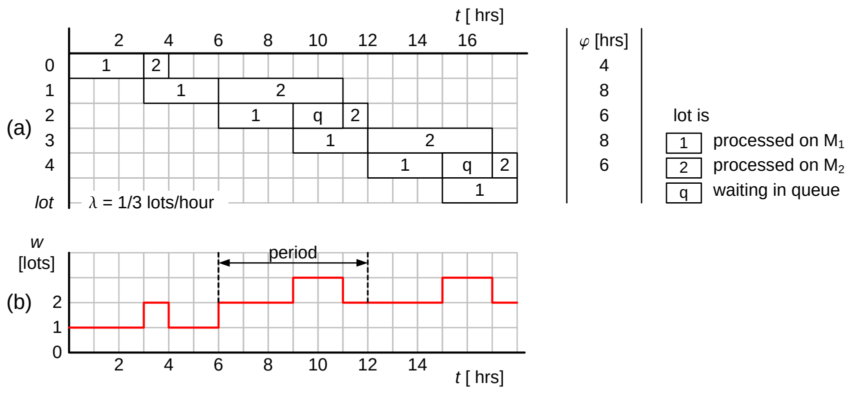


Figure 2.25: Lot-time-diagram and wip-time-diagram for $\lambda = 1/3$ lots/hour

From the lot-time-diagram we determine the mean flowtime at $(8{+}6)/2 = 7$ hours. Note that we determine the mean flowtime only after the system has reached steady state, that is, behaves periodically. Note that the flowtime again depends on the release rate.

(iii) Figure 2.25(b) shows the wip-time-diagram. For $t > 6$ hours the wip-level evolves periodically with a period of 6 hours. The mean wip level is $\overline{w} = \frac{2}{6} \cdot 3 + \frac{4}{6} \cdot 2 = \frac{14}{6} = \frac{7}{3}$ lots. With Little's law: $w = \delta \cdot \varphi = \frac{1}{3} \cdot 7 = \frac{7}{3}$ lots. ⊠

Please note, that determining the flowtime in the way we discussed here only works for deterministic systems!

For deterministic system systems the process times (or other properties) need not be constant, as long as the changes are deterministic! Recall that the second machine in Example 2.11 did not have a constant process times, but the changes were predetermined and therefore the process time was deterministic. If the process time was stochastic, for example the process time is either 1.0 or 5.0, both with 50% probability, we can only construct a lot-time-diagram that represents one of the many possible outcomes. For example it could happen that the process time of machine 2 follows the sequence {1.0,1.0,5.0,5.0,5.0,1.0,1.0,5.0}. For this sequence the flowtime is significantly higher. (Try! Determine the flowtime for this sequence.) But in case of stochastic process times, any other sequence is possible. We can no longer use the approach presented in this section. The same holds for stochastic arrival times, stochastic chance on rework, stochastic break-down, or any other source of stochasticity.

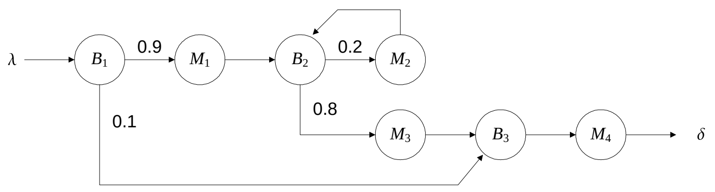
In Chapter 3 we consider mathematical models to calculate the flowtime for stochastic process and inter-arrival times.

Keyword overview

Modelling manufacturing systems	Analyzing manufacturing systems
	• mass conservation • utilisation u, bottleneck, maximum throughput • Little's law • throughput and flowtime calculation for deterministic manufacturing systems

2.3 Exercises

1. Consider the manufacturing system with rework and bypassing in Figure 2.26. The manufacturing system consists of three buffers and four machines.



machine	t_0 [hrs]
M_1	2.0
M_2	6.0
M_3	1.8
M_4	1.6

Figure 2.26: Manufacturing system with rework and bypassing

Lots are released at a rate of λ lots/hour. The numbers near the arrows indicate the fraction of the lots that follow that route. For instance, of the lots leaving buffer B_1 90% goes to machine M_1 and 10% goes to buffer B_3. The process time of each machine is listed in the table in Figure 2.26.

   (a) Express the throughput of machine M_1, M_3, and M_4 in terms of λ.

   (b) Express the throughput of machine M_2 in terms of λ.

   (c) Express the utilisation of each machine in terms of λ.

   (d) What machine is the bottleneck? Determine the maximum throughput of this system.

2. Consider a three-workstation flowline. The workstations each have an infinite buffer and contain 1, 3, and 2 machines respectively. The process time of the machines in workstation 1,2 and 3 is 0.9, 3.0, and 1.9 hours respectively.
   (a) Calculate the maximum throughput of the line.
   (b) Use a lot-time-diagram to determine the flowtime of a lot for release rates under the maximum throughput.
   (c) Determine the mean wip-level in the line for the maximum throughput.

3. We have a three-workstation flowline. Each workstation consists of an infinite buffer and a number of machines. The number of machines has still to be determined. The machines in workstation 1,2, and 3 have a process time of 0.10, 0.15, and 0.06 hours respectively. We want to establish a throughput of 60 lots/hour.
   (a) Determine the flowtime of a lot for release rates under the maximum throughput.
   (b) Determine the number of machines in workstation 1,2, and 3 required to attain the desired throughput of 60 lots/hour.
   (c) Which workstation (what workstations) is (are) the bottleneck?

4. Consider a two-workstation flowline with two machines. The two machines have a process time of 2.0 and 3.0 hours respectively. The machines may be placed in any order. The two alternatives are shown in Figure 2.27.

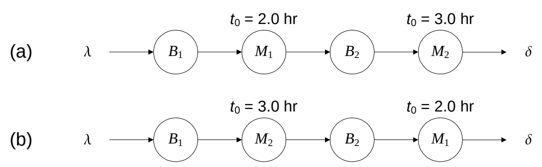


Figure 2.27: Two flowline configurations

   (a) What is $\delta_{\max}$ for the line in Figure 2.27(a)?
   (b) What happens when we set $\lambda > \delta_{\max}$?
   (c) What is $\delta_{\max}$ for the line in Figure 2.27(b)?
   (d) What happens when we now set $\lambda > \delta_{\max}$?

5. Consider the re-entrant flowline in Figure 2.28. Lots are released in the line by generator G at a rate of λ lots/hour. Each lot passes through the flowline twice and has the same fixed route, namely M_1, M_2, M_1, M_2. If a newly released lot and a re-entrant lot arrive at a machine at the same time, the re-entrant lot is processed first.

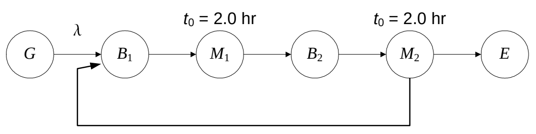


Figure 2.28: Re-entrant flowline

(a) Express the utilisation of machine 1 in terms of release rate λ and determine the maximum throughput.

(b) Construct a lot-time-diagram for $\lambda = \frac{1}{5}$ lots/hour and determine the (mean) flowtime of a lot.

(c) From the lot-time-diagram, derive a wip-time-diagram, and determine the mean wip-level in steady state.

(d) Verify your result using Little's law.

6. We have a two workstation flowline. Workstation 1 consists of an infinite buffer and a single-lot[4] machine. Workstation 2 consists of an infinite buffer and a batch machine. The single-lot machine has a process time of 2.0 hours. The batch machine processes in fixed batches of 5 lots. The process time of a batch is 8.0 hours.

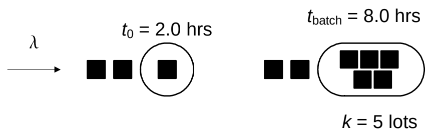


Figure 2.29: Single-lot and batch machine in series

(a) Express the utilisation of the single-lot machine and the batch machine in terms of release rate λ and determine the maximum throughput.

(b) Construct a lot-time-diagram for $\lambda = \frac{1}{2}$ lots/hour. Determine the (mean) flowtime of a lot.

(c) Calculate the mean flowtime of a lot without using the lot-time-diagram. You can use the equation found in Example 2.10(iii).

(d) Determine the mean wip-level.

(e) Can we decrease the mean wip-level by increasing the release rate?

We replace the batch machine by a cheaper one. This machine processes batches of size 4. Processing a batch takes 10 hours.

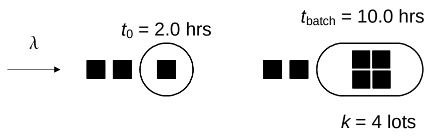


Figure 2.30: Single-lot and batch machine in series (2)

(a) Determine the maximum throughput of the line and determine the utilisation of the single-lot machine when the line attains the maximum throughput.

(b) Construct a lot-time-diagram for this line, for the maximum release rate. Determine the (mean) flowtime of a lot.

(c) Calculate the mean flowtime of a lot without using the lot-time-diagram. You can use the equation found in Example 2.10(iii).

[4]A single-lot machine processes one lot at a time.

7. We have a two-workstation flowline. Each workstation consists of an infinite buffer and a single machine. The flowline is used to process two types of products: type A and type B. Table 2.1 shows the process time on each machine per type.

	machine 1	machine 2
type A	3.0	2.5
type B	2.0	2.5

Table 2.1: Processing times of different product types in [hours]

New products are released in the flowline with a rate of λ prods/hour. The product types are released in a fixed sequence: {A,A,B,B,B,A,A,B,B,B,A,A,B,B,B,...}.

(a) Express the utilisation of machine 1 and 2 in terms of release rate λ. Determine the maximum throughput of this line.

(b) What is the maximum throughput specified per product type, that is, how many products of each type are processed per hour on average?

(c) Construct a lot-time-diagram for a release rate of $\lambda = 0.4$ prods/hour.

(d) Determine the mean flowtime per product in steady state.

(e) Calculate the average wip level in steady state.

(f) Calculate the average wip level in steady state, specified per product type.

2.4 Additional exercises

1. The manufacturing system in Figure 2.31 consists of 6 workstations. 20% of the lots processed by workstation W_4 need rework. The lot is first stripped in workstation W_5 and then reprocessed by workstation W_2 and W_3.

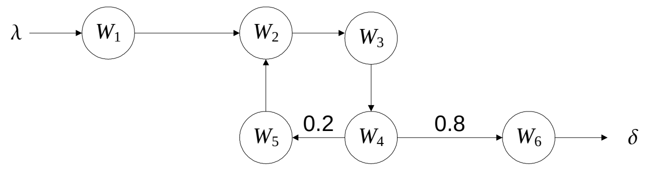


Figure 2.31: Manufacturing line

The numbers at the arrows indicate the fraction of the lots that follow that route.

(a) Calculate the total throughput of workstation 2,3, and 5 in terms of release rate λ.

(b) Calculate the throughput $\delta_{W_4W_5}$ and $\delta_{W_4W_6}$ for workstation 4.

(c) Verify that conservation of mass holds for system $W_2W_3W_4W_5$.

2. Figure 2.32 shows a flowline with bypassing loops going from workstation 1 to workstation 2, 3, and 4. The numbers at the arrows indicate the fraction of the lots that follow that route.

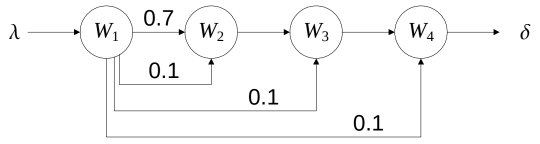


Figure 2.32: Manufacturing line with bypassing

   (a) Use mass conservation to determine the throughput of workstations 1,2,3, and 4.

   (b) Verify that conservation of mass holds for system $W_1W_2W_3W_4$.

3. Figure 2.33 shows a flowline of workstations with rework loops. 20% of the lots processed by workstation 2 need rework on workstation 1 and 2. 20% of the processed by workstation 3 need rework on workstation 2 and 3. Finally, 30% of the lots processed by workstation 4 need rework on 3 and 4.

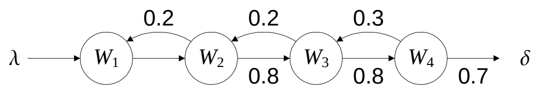


Figure 2.33: Manufacturing line with rework

   (a) Intuitively, what workstation will have to process the most lots?

   (b) Write down the mass conservation equations for workstation 1 through 4.

Mass conservation on system $W_1W_2W_3W_4$ yields that $\delta = \lambda$. For the total throughput of workstation 4 we have $\delta_{W_4} = \delta + \delta_{W_4W_3}$.

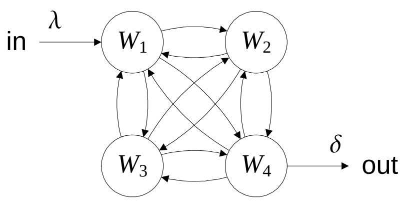
   (a) Calculate δ_{W_4} in terms of λ.

   (b) Calculate the throughput of each workstation in terms of λ.

   (c) Assuming that the raw process time t_0 for each workstation is 1.0 hours, what workstation is the bottleneck? (Does this correspond to your initial guess in a)?)

   (d) What is the maximum throughput of this line?

4. Figure 2.34 shows a so-called job shop. In a job shop lots can each have very different routings. The from-to-matrix is used to indicate the fraction of the lots that go from a specific workstation to another workstation. For example, 10% of lots that finish processing on workstation 1 go to workstation 2, 80% goes to workstation 3, and 10% goes to workstation 4. All lots enter the job shop via workstation 1 and leave the job shop via workstation 4.



		to				
		1	2	3	4	out
from	in	1.0	▨	▨	▨	▨
	1	▨	0.1	0.8	0.1	▨
	2	0.2	▨	0.7	0.1	▨
	3	0.1	0.2	▨	0.7	▨
	4	0.0	0.2	0.3	▨	0.5

Figure 2.34: (a)Job shop, (b) From-to-matrix

(a) Intuitively, what workstation processes the most lots in this job shop?

(b) Write down the mass conservation equations for each workstation.

(c) Show that we can write these equations as the following matrix equation.

$$\begin{bmatrix} -1 & 0.2 & 0.1 & 0 \\ 0.1 & -1 & 0.2 & 0.2 \\ 0.8 & 0.7 & -1 & 0.3 \\ 0.1 & 0.1 & 0.7 & -1 \end{bmatrix} \begin{bmatrix} \delta_{W_1} \\ \delta_{W_2} \\ \delta_{W_3} \\ \delta_{W_4} \end{bmatrix} = \begin{bmatrix} -\lambda \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(d) Solve the matrix equation (for example using Matlab).

(e) How can you easily verify that the throughput δ_{W_4} is indeed 2λ? Do the results correspond to your intuition built in part a)?

(f) The process time t_0 for workstation 1,2,3, and 4 is 1.0, 1.3, 0.7, and 0.8 hours respectively. Which workstation is the bottleneck?

Chapter 3

Analytical models of stochastic manufacturing systems

In the previous chapter we presented analytical approaches to determine the (maximum) throughput, the flowtime and the wip-level for deterministic manufacturing systems, that is, systems with deterministic process times, deterministic arrival rate, deterministic routings, and no other sources of stochasticity. At the end of the previous chapter, we argued that these analytical approaches are no longer useful when stochasticity is involved. In this chapter we present an analytical approach to analyze manufacturing systems, that feature stochastic process times, stochastic inter-arrival times, or other stochastic phenomena. Not being a book about queueing theory or stochastic process theory we do not go into details and derivations. Instead we refer to other books for derivations. We restrict ourselves to the basic queueing relations for a single workstation (Section 3.2), for a flowline (Section 3.3) and some extensions on these queueing relations for batching (Section 3.5).

3.1 Classification of manufacturing queueing models

The basic manufacturing queueing model is shown in Figure 3.1.

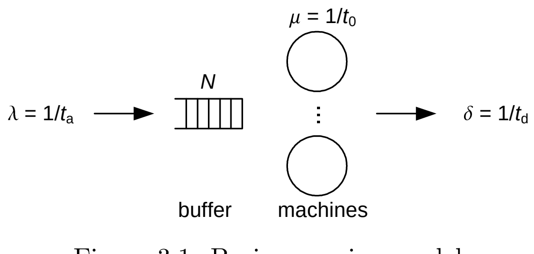


Figure 3.1: Basic queueing model

The basic model is a workstation that consists of a buffer and a number of identical parallel machines. The workstation is characterized by the following quantities.

- Inter-arrival time t_a (with arrival rate $\lambda = 1/t_\mathrm{a}$) denotes the time between two subsequent lots entering the system. It is characterized by a distribution, with mean t_a and coefficient of variation[1] c_a.

- Process time t_0 (with service rate $\mu = 1/t_0$) is the time it takes the machine to process a lot. Stochastic process time is characterized by a distribution with mean t_0 and coefficient of variation c_0.

- Buffer size N (or maximum queue length) is the maximum number of lots that can be stored in the buffer.

- The number m of parallel identical machines in the workstation.

- Inter-departure time t_d (with throughput $\delta = 1/t_\mathrm{d}$) denotes the time between two subsequent lots leaving the system.

Kendall [Ken53] introduced a shorthand notation for the characteristics of a workstation of the form $A/B/m/N$. Herein A and B represent the distribution of the inter-arrival time and the distribution of the process time respectively. We distinguish three classes of distributions: an M denotes an exponential (or Markovian, or memoryless) distribution, a D denotes a deterministic distribution, and a G denotes any (or a general) distribution. The m in the Kendall notation refers to the number of parallel machines in the workstation and the N denotes the maximum number of lots in the buffer.

For example, an $M/M/1/\infty$ workstation is a workstation with an infinite buffer where lots arrive with exponentially distributed inter-arrival times and one machine that has exponential processing times. In case of infinite buffers, the ∞ symbol is often omitted from the Kendall notation. So, a $G/D/1$ workstation is single-machine workstation with an infinite buffer, the machine has a deterministic process time and the inter-arrival time of the lots can be any distribution.

3.2 Queueing relations for single-machine workstation

Consider a workstation that consists of infinite buffer B_∞ and machine M, see Figure 3.2. Lots arrive at the buffer with a stochastic inter-arrival time. The inter-arrival time distribution has mean t_a and squared coefficient of variation c_a^2. The coefficient of variation relates the degree of variability in a distribution to the mean of the distribution. In queueing theory, the coefficient of variation is more often used than the standard deviation. The machine has stochastic process times, with mean process time t_0 and squared coefficient of variation c_0^2. Finished lots leave the machine with a stochastic inter-departure time, with mean t_d and squared coefficient of variation c_d^2.

Two preconditions are required: (1) the system is stable, that is, reaches steady-state, and (2) no lots are lost in the system. The first precondition implies that the mean inter-arrival time t_a should be greater than the mean process time t_0. If $t_\mathrm{a} < t_0$, the machine has insufficient capacity and an ever-increasing queue in the buffer occurs. The second

[1]The coefficient of variation c of a distribution is defined as the quotient of its standard deviation σ and its mean μ: $c = \sigma/\mu$.

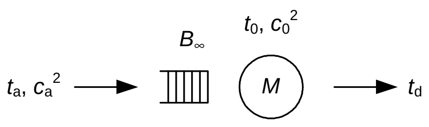


Figure 3.2: Single machine workstation

precondition implies that the mean inter-departure time t_d is equal to the mean inter-arrival time t_a. The approximation of Kingman [Kin61, Tij94] states that the mean waiting time φ_B in buffer B can be approximated by:

$$\varphi_\mathrm{B} = \frac{c_\mathrm{a}^2 + c_0^2}{2} \cdot \frac{u}{1-u} \cdot t_0 \tag{3.1}$$

Every t_a hours, the machine has to process a lot, which takes t_0 hours, thus the utilisation $u = t_0/t_\mathrm{a}$.

The Kingman relation in Equation 3.1 is composed of three factors: a variability factor $(c_\mathrm{a}^2 + c_0^2)/2$, a utilisation factor $u/(1-u)$ and a term with the process time t_0.

- The variability term indicates that if both inter-arrival and process times are deterministic (constant), the waiting time in the buffer is 0. The higher the variability in the inter-arrival time or in the process time, the longer the mean waiting time in the buffer becomes.
- The utilisation factor shows that the average waiting time of a lot in the buffer increases in a non-linear fashion for increasing utilisation. As an extremum the system becomes unstable if $u = 1$. Utilisations of 80% and higher give rise to long waiting times in systems with variability.
- The final factor is the mean process time. The mean waiting time is proportional with the mean process time.

Equation 3.1 is exact for an $M/G/1$ system (as already shown by Pollaczek-Khinchine [Pol30, Khi32]), that is, a single machine workstation with exponentially distributed inter-arrival times and any distribution for the process time. For other single machine workstations it is an approximation. As an example, let us consider an $M/M/1$ system.

Example 3.1 $M/M/1$ system
An $M/M/1$ system consists of an infinite buffer and a single machine. The inter-arrival times and process times are distributed according to an exponential distribution. For an exponential distribution, the mean is equal to the standard deviation. The mean inter-arrival time t_a is 4.0 hours, the mean process time t_0 is 3.0 hours. Determine the mean flowtime per lot.

For an exponential distribution holds: $c = \sigma/\mu = 1$, so $c_\mathrm{a}^2 = 1$ and $c_0^2 = 1$. Applying Equation 3.1 yields:

$$\varphi_\mathrm{B} = \frac{c_\mathrm{a}^2 + c_0^2}{2} \cdot \frac{u}{1-u} \cdot t_0 = \frac{1+1}{2} \cdot \frac{u}{1-u} \cdot t_0 = \frac{u}{1-u} \cdot t_0$$

The total flowtime becomes:

$$\varphi = \varphi_\mathrm{B} + t_0 = (\frac{u}{1-u} + 1) \cdot t_0 = \frac{t_0}{1-u} = \frac{3}{0.25} = 12 \text{ hours}$$

As the M/M/1 system is an element of the class of M/G/1 systems, the calculation of the flowtime is exact. ☒

An interesting parameter is the *flowtime factor* φ/t_0, which relates the total flowtime to the raw process time (minimal flowtime). For Example 3.1 the flowtime factor is 4, indicating that on average 75% of the flowtime is due to waiting in the buffer.

Example 3.2 Flowtime for a workstation with high and low process times
Consider a single machine workstation with an infinite buffer and a machine. The machine has a process time of either 1.0 or 5.0. The inter-arrival times are distributed according to an exponential distribution, with mean inter-arrival time $t_\mathrm{a} = 4.0$ hours. Determine the mean flowtime and the mean number of lots waiting in the buffer.

Since, the inter-arrival times are distributed according to an exponential distribution, $c_\mathrm{a}^2 = 1$. Next, we calculate the standard deviation s_0 on the process time.

$$s_0^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{(1.0-3.0)^2 + (5.0-3.0)^2}{2} = 4.0$$

The squared coefficient of variation becomes:

$$c_0^2 = \frac{s_0^2}{t_0^2} = \frac{4.0}{3.0^2} = \frac{4}{9}$$

Applying Equation 3.1 yields:

$$\varphi_\mathrm{B} = (\frac{c_\mathrm{a}^2 + c_0^2}{2} \cdot \frac{u}{1-u}) \cdot t_0 = (\frac{1+4/9}{2} \cdot \frac{0.75}{1-0.75}) \cdot 3.0 = 6.5 \text{ hours}$$

Using Little's law, we determine the mean number of lots in the buffer:

$$w_\mathrm{B} = \varphi_\mathrm{B} \cdot \delta = 6.5 \cdot 1/4 = 1.63 \text{ lots}$$

This manufacturing system is an $M/G/1$ system, so the calculated flowtime (and thus wip-level) are exact. ☒

Figure 3.3 shows the mean total flowtime φ for an $M/G/1$ system determined analytically as a function of the utilisation u, with $c_\mathrm{a} = 1$, $t_0 = 3.0$ hours, and $c_0 = \{0, 1, 3\}$.

No machine can ever be utilized for the full hundred percent, since almost all manufacturing systems contain some sort of variability, which causes the flowtime (and thus wip-level) to go to infinity for $u = 1$. In industry this simple observation is often overlooked. But also for utilizations smaller than one, we can clearly see that flowtime increases strongly at higher utilisation levels. The higher the machine is utilized the larger the average waiting time of lots in the queue is, which results in larger flowtimes. Especially in make-to-order industries one strives for short flowtimes to keep customers happy.

The only way one can improve the trade-off between utilization and flowtime is to decrease variability. Figure 3.3 shows that for a low variability level the waiting time stays relatively

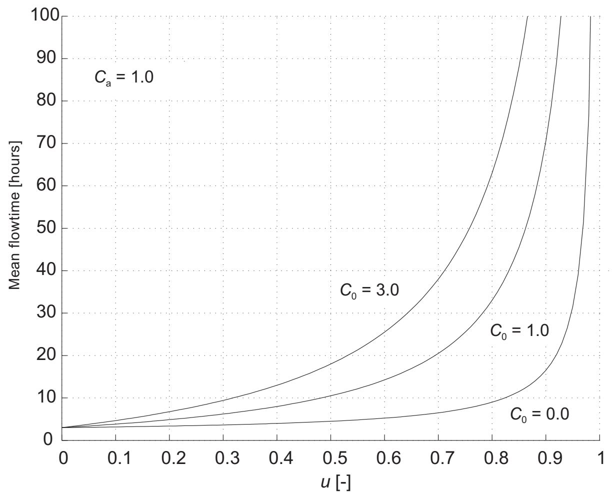


Figure 3.3: Mean flowtime for a $M/G/1$ system for different values of c_0

small even for high utilization levels. The nonlinear increase of waiting time occurs for very high utilizations. On the other hand, high variability levels cause the waiting time to strongly increase for low utilization levels.

In Chapter 5 we investigate the validity of this analytical relations by simulation.

3.3 Queueing relations for a flowline

In practice, we often encounter chains of buffers and machines. As an example, consider a three workstation flowline, see Figure 3.4.

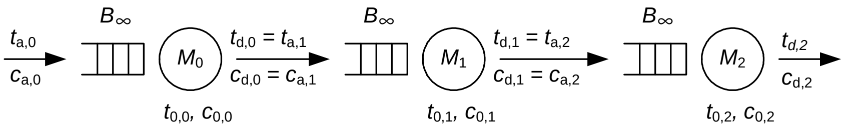


Figure 3.4: Three workstation flowline

We calculate the mean flowtime for workstation 0 by using Equation 3.1. To calculate the mean flowtime for workstation 1, we need the inter-arrival time distribution for workstation 1. The inter-arrival times of workstation 1 are equal to the inter-departure times of workstation 0 (mean $t_{\mathrm{d},0}$, coefficient of variation $c_{\mathrm{d},0}$). For a stable system, we have $t_{\mathrm{a},0} = t_{\mathrm{d},0} = t_{\mathrm{a},1} = t_{\mathrm{d},1} = t_{\mathrm{a},2} = t_{\mathrm{d},2}$ (conservation of mass). But how do we determine $c_{\mathrm{d},1}$? We can approximate the coefficient of variation c_d by Kuehn's linking equation [Kue79]:

$$c_\mathrm{d}^2 = (1 - u^2) \cdot c_\mathrm{a}^2 + u^2 \cdot c_0^2 \tag{3.2}$$

Equation 3.2 indicates that for low utilisations ($u \approx 0$) the squared coefficient of variation on the inter-departure times c_d^2 resembles the squared coefficient of variation on the inter-arrival times c_a^2 ($c_\mathrm{d}^2 \approx c_\mathrm{a}^2$). For low utilisations, lots seldomly have to wait in the buffer before being processed. The variability on the departure is inherited from the arrival pattern. For high utilisations ($u \approx 1$), c_d^2 resembles the squared coefficient of variation on the process times c_0^2 ($c_\mathrm{d}^2 \approx c_0^2$). For high utilisations, lots usually have to wait in the buffer before they are processed. The variability of the arrival is levelled out by the large queue in the buffer. The variability on the departure is dictated by the process time variability.

Example 3.3 Three workstations in line
Consider the three workstation flowline from Figure 3.4. For the inter-arrival time at workstation 0 we have $t_{\mathrm{a},0} = 4.0$ hours and $c_{\mathrm{a},0}^2 = 1$. The three workstations are identical with respect to the process times: $t_{0,i} = 3.0$ hours and $c_{0,i}^2 = 0.5$ hours for $i = 0, 1, 2$. Determine the mean total flowtime per lot.

Since $t_\mathrm{a} > t_{0,i}$ for $i = 0, 1, 2$, we have a stable system and $t_{\mathrm{a},i} = t_{\mathrm{d},i} = 4.0$ hours for $i = 0, 1, 2$. Subsequently, the utilisation for each workstation is $u_i = 3.0/4.0 = 0.75$ for $i = 0, 1, 2$.

We calculate the mean flowtime for workstation 0, using Equation 3.1.

$$\varphi_0 = \varphi_\mathrm{B} + t_{0,0} = \frac{c_{\mathrm{a},0}^2 + c_{0,0}^2}{2} \cdot \frac{u}{1-u} \cdot t_{0,0} + t_{0,0} = \frac{1+0.5}{2} \cdot \frac{0.75}{1-0.75} \cdot 3.0 + 3.0 = 9.75 \text{ hours}$$

Next, we determine the coefficient of variation on the inter-arrival time $c_{\mathrm{a},1}$ for workstation W_1 using Equation 3.2.

$$c_{\mathrm{a},1}^2 = c_{\mathrm{d},0}^2 = (1-u^2) \cdot c_{\mathrm{a},0}^2 + u^2 \cdot c_{0,0}^2 = (1-0.75^2) \cdot 1 + 0.75^2 \cdot 0.5 = 0.719$$

We calculate the mean flowtime for workstation 1.

$$\varphi_1 = \frac{0.719+0.5}{2} \frac{0.75}{1-0.75} 3.0 + 3.0 = 8.49 \text{ hours}$$

In a similar way, we determine that $c_{\mathrm{a},2}^2 = 0.596$ and $\varphi_2 = 7.93$ hours. We then calculate the mean total flowtime.

$$\varphi_\mathrm{tot} = \varphi_0 + \varphi_1 + \varphi_2 = 26.2 \text{ hours}$$

Note that the minimal flowtime without variability ($c_{\mathrm{a},0}^2 = c_{0,i}^2 = 0$) equals 9.0 hours. ☒

We investigate the validity of Kuehn's linking equation in Chapter 6.

3.4 Identical parallel machines

The relations from the previous sections allow us to determine the flowtime for a single-machine workstation and a flowline of a single-machine workstations. In this section we wish to determine the flowtime and the coefficient of variation on the inter-departure time for a workstation with identical parallel machines. Figure 3.5 shows a workstation that consists of an infinite buffer and m parallel identical machines.

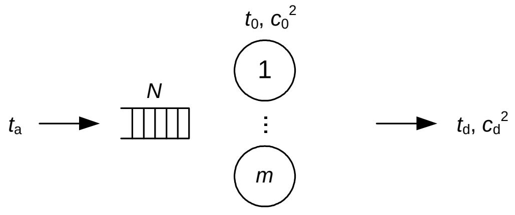


Figure 3.5: Workstation with m identical parallel machines

In this workstation, all lots wait in a single queue for the next available machine. For a workstation with m identical parallel machines, the waiting time in the buffer is approximated by the following equation [HS01].

$$\varphi_\mathrm{B} = \frac{c_\mathrm{a}^2 + c_0^2}{2} \cdot \frac{u^{\sqrt{2(m+1)}-1}}{m(1-u)} \cdot t_0 \tag{3.3}$$

A reasonable approximation for the coefficient of variation on the inter-departure times is the following linking equation [HS01].

$$c_\mathrm{d}^2 = 1 + (1-u^2)(c_\mathrm{a}^2 - 1) + \frac{u^2}{\sqrt{m}}(c_0^2 - 1) \tag{3.4}$$

Both equations are approximations, which are fairly accurate for high degrees of utilisation and less accurate for lower degrees of utilisation. Note that for $m = 1$ Equations 3.3 and 3.4 reduce to Equations 3.1 and 3.2. Be careful to calculate the utilisation in accordance with the number of machines!

Example 3.4 Parallel machines
Consider the flowline in Figure 3.6.

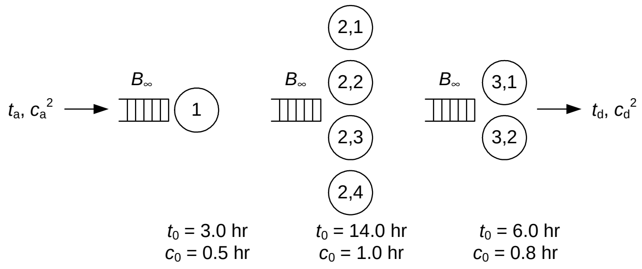


Figure 3.6: Flowline with multiple-machine workstations

The flowline consists of three workstations with 1,4, and 2 identical machine respectively. The raw process time and the coefficient of variation on the raw process time per machine is displayed in Figure 3.6. The lots arrive at workstation 1 with a mean inter-arrival time of 4.0 hours and a coefficient of variation of 1.0. Determine the mean flowtime per workstation.

We start with determining the utilisation for each workstation. Workstation 1 processes a lot every 4.0 hours, which takes 3.0 hours, $u_1 = 3.0/4.0 = 0.75$. Workstation 2 can process 4 lots simultaneously. Each machine in workstation 2 receives one lot every 16.0 hours, $u_2 = 14.0/16.0 = 0.875$. Similarly, we determine the utilisation for workstation 3, $u_3 = 6.0/8.0 = 0.75$.

Since the utilisation for all workstation is under 1.0, the system is stable and $t_{\mathrm{a},i} = t_{\mathrm{a}.1} = 4.0$ hours.

We now calculate the flowtime for workstation 1. Workstation 1 has a single machine, so we use Equation 3.1.

$$\begin{aligned}\varphi_1 &= \frac{c_{\mathrm{a},1}^2 + c_{0,1}^2}{2} \cdot \frac{u_1}{1-u_1} \cdot t_{0,1} + t_{0,1} \\ &= \frac{1+0.5^2}{2} \cdot \frac{0.75}{1-0.75} \cdot 3.0 + 3.0 = 8.625 \text{ hours}\end{aligned}$$

To calculate the flowtime for workstation 2, we need the coefficient of departure for workstation 1, using Equation 3.2.

$$c_{\mathrm{d},1}^2 = c_{\mathrm{a},2}^2 = (1-u_1^2) \cdot c_{\mathrm{a},1}^2 + u_1^2 \cdot c_{0,1}^2 = (1-0.75^2) \cdot 1 + 0.75^2 \cdot 0.5^2 = 0.578$$

We calculate the mean flowtime for workstation 2. Workstation 2 contains 4 parallel machines.

$$\begin{aligned}\varphi_2 &= \frac{c_{\mathrm{a},2}^2 + c_{0,2}^2}{2} \cdot \frac{u_2^{\sqrt{2(m_2+1)}-1}}{m_2(1-u_2)} \cdot t_{0,2} + t_{0,2} \\ &= \frac{0.578+1.0}{2} \cdot \frac{0.875^{\sqrt{10}-1}}{4\cdot(1-0.875)} \cdot 14.0 + 14.0 = 30.55 \text{ hours}\end{aligned}$$

We calculate the coefficient of variation on the inter-departure times.

$$\begin{aligned}c_{\mathrm{d},2}^2 &= c_{\mathrm{a},3}^2 = 1 + (1-u_2^2)\cdot(c_{\mathrm{a},2}^2 - 1) + \frac{u_2^2}{\sqrt{m_2}}(c_{0,2}^2 - 1) \\ &= 1 + (1-0.875^2)\cdot(0.578-1) + 0 = 0.901\end{aligned}$$

In a similar way, we determine that $\varphi_3 = 12.09$ hours and $c_{\mathrm{d},3}^2 = 0.8136$. The mean total flowtime is

$$\varphi_{\mathrm{tot}} = \varphi_1 + \varphi_2 + \varphi_3 = 51.27 \text{ hours}$$

Note that the minimal flowtime without variability ($c_{\mathrm{a}}^2 = c_{0,i}^2 = 0$) equals 24.0 hours. ⊠

3.5 Batch machines

In this chapter we have regarded single-lot machines only. In the previous chapter we discussed batch machines with deterministic process and inter-arrival times. In this section we take a look at batch machines with stochastic process and inter-arrival times.

Consider a batch machine that processes lots in batches. The batches have a fixed size k. The machine starts processing the batch only after the batch is complete. The batch machine is depicted in Figure 3.7.

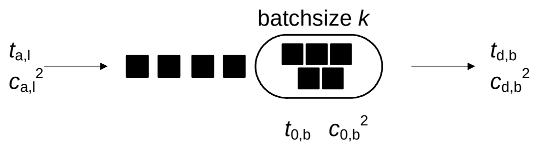


Figure 3.7: Machine processes batches of fixed batch size k

Lots arrive with a mean inter-arrival time $t_{\mathrm{a,l}}$ and squared coefficient of variation $c_{\mathrm{a,l}}^2$. Processing a batch takes on average $t_{0,\mathrm{b}}$, with a squared coefficient of variation $c_{0,\mathrm{b}}^2$. Question now is: what is the mean flowtime?

We take a closer look at the lots in the buffer. We, imaginarily, divide the buffer into two parts. In the first part of the buffer, lots wait until they have formed a complete batch. In the second part the batches of lots wait for the machine to become idle. The mean flowtime for the batch machine is the sum of the wait-to-batch-time φ_{Bk}, the queueing time of the batches φ_{Bq}, and the processing time.

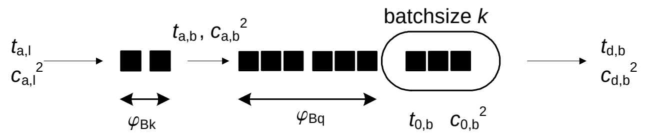


Figure 3.8: Lots first form a batch and then wait to be processed

The wait-to-batch-time φ_{Bk} is not equal for all lots in a batch. The first lot in the batch has to wait for $(k-1)$ lots to arrive, while the last lot in the batch does not have to wait at all before the batch is complete. The mean wait-to-batch-time is:

$$\varphi_{\mathrm{Bk}} = \frac{\varphi_{\mathrm{Bk}}(\text{first}) + \varphi_{\mathrm{Bk}}(\text{last})}{2} = \frac{(k-1)\cdot t_{\mathrm{a,l}} + 0}{2} = \frac{(k-1)}{2}\cdot t_{\mathrm{a,l}}$$

Note that this result is identical to the result obtained in Example 2.10. After the batch is complete, it is possible that the machine is not idle and the batch has to wait. In fact, we can regard the batch machine and the queue of batches in the same way as a basic single-lot workstation. From Section 3.2 we know that the waiting time in the queue is:

$$\varphi_{\mathrm{Bq}} = \frac{c_{\mathrm{a,b}}^2 + c_{0,\mathrm{b}}^2}{2}\cdot\frac{u}{1-u}\cdot t_{0,\mathrm{b}}$$

Herein $c_{\mathrm{a,b}}$ is the coefficient of variation on the inter-arrival time of batch! However, we only know the variation on the inter-arrival time of a lot. Question is: how do we derive $c_{\mathrm{a,b}}$ from $c_{\mathrm{a,l}}$? From basic statistics we know that for k independent stochastics the variance of the sum of these stochastics is equal to the sum of the variances, see for example [MR02].

$$\mathrm{Var}(X_1 + X_2 + \ldots + X_{\mathrm{k}}) = \mathrm{Var}(X_1) + \mathrm{Var}(X_2) + \ldots + \mathrm{Var}(X_{\mathrm{k}})$$

The inter-arrival time of a batch is the sum of the inter-arrival times of k lots.

$$t_{\mathrm{a,b}} = t_{\mathrm{a,1}} + t_{\mathrm{a,1}} + \ldots + t_{\mathrm{a,k}}$$

If the inter-arrival times for the lots are independent samples of the same distribution we get for the variance:

$$\begin{aligned}\mathrm{Var}(t_{\mathrm{a,b}}) &= \mathrm{Var}(t_{\mathrm{a,l}} + t_{\mathrm{a,2}} + \ldots + t_{\mathrm{a,k}}) \\ &= \mathrm{Var}(t_{\mathrm{a,1}}) + \mathrm{Var}(t_{\mathrm{a,2}}) + \ldots + \mathrm{Var}(t_{\mathrm{a,k}}) \\ &= \mathrm{Var}(t_{\mathrm{a,l}}) + \mathrm{Var}(t_{\mathrm{a,l}}) + \ldots + \mathrm{Var}(t_{\mathrm{a,l}}) \\ &= k \cdot \mathrm{Var}(t_{\mathrm{a,l}})\end{aligned}$$

For the squared coefficient of variation we get:

$$c^2_{\mathrm{a,b}} = \frac{\sigma^2_{\mathrm{a,b}}}{t^2_{\mathrm{a,b}}} = \frac{k \cdot \sigma^2_{\mathrm{a,l}}}{(k \cdot t_{\mathrm{a,l}})^2} = \frac{1}{k}\frac{\sigma^2_{\mathrm{a,l}}}{t^2_{\mathrm{a,l}}} = \frac{c^2_{\mathrm{a,l}}}{k}$$

We can now calculate the queueing time of a batch in the buffer. Remember that the utilisation of a batch machine can be determined as follows. Each $t_{\mathrm{a,b}} = k \cdot t_{\mathrm{a,l}}$ the batch machine processes a batch, which takes $t_{0,\mathrm{b}}$. Hence, utilisation is $u = t_{0,\mathrm{b}}/(k \cdot t_{\mathrm{a,l}})$.

Example 3.5 Single-lot and batch machine in series
We have a flowline of a single-lot machine and a batch machine. Lots arrive at the single lot machine with a mean inter-arrival time of 4.0 hours and a squared coefficient of variation c^2_{a} of 1. Processing a lot on the single-lot machine takes on average 3.0 hours, with $c^2_0 = 0.5$. The batch machine processes lots in fixed batches of 3 lots. Processing a batch takes 8.0 hours on average with a squared coefficient of variation $c^2_{0,\mathrm{b}} = 0.1$, see Figure 3.9. Calculate the mean flowtime of a lot.

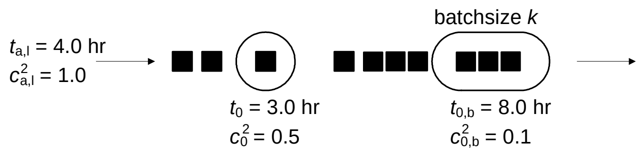

batchsize k
$t_{\mathrm{a,l}}$ = 4.0 hr
$c^2_{\mathrm{a,l}}$ = 1.0
t_0 = 3.0 hr
c^2_0 = 0.5
$t_{0,\mathrm{b}}$ = 8.0 hr
$c^2_{0,\mathrm{b}}$ = 0.1

Figure 3.9: Single-lot machine and batch machine in series

First we calculate the utilisation for the single-lot machine and the batch machine. These are $u_{\mathrm{M}} = 3/4$ and $u_{\mathrm{batch}} = 8/(3 \cdot 4) = 2/3$. Since both utilisations are under 1 we have a stable system and we can calculate a mean flowtime. For the queueing time for the single-lot machine we have:

$$\varphi_{\mathrm{B}_1} = \frac{c^2_{\mathrm{a}} + c^2_0}{2} \cdot \frac{u}{1-u} \cdot t_0 = \frac{1+0.5}{2} \cdot \frac{0.75}{1-0.75} \cdot 3 = 6.75 \text{ hours}$$

Next we determine the coefficient of variation of the lots leaving the single-lot machine and entering the batch machine.

$$c_\mathrm{d}^2 = u^2 c_0^2 + (1-u^2)c_\mathrm{a}^2 = 0.75^2 \cdot 0.5 + (1-0.75^2)\cdot 1 = 0.7188$$

We now calculate the wait-to-batch-time and the queueing time of batches in front of the batch machine.

$$\varphi_\mathrm{Bk} = \frac{k-1}{2}\cdot t_\mathrm{a,l} = \frac{3-1}{2}\cdot 4.0 = 4.0 \text{ hours}$$

For the queueing time we need to calculate the coefficient of variation on the inter-arrival time of a batch.

$$c_\mathrm{a,b}^2 = \frac{c_\mathrm{a,l}^2}{k} = \frac{0.7188}{3} = 0.2396$$

$$\varphi_\mathrm{Bq} = \frac{c_\mathrm{a,b}^2 + c_\mathrm{0,b}^2}{2}\cdot\frac{u}{1-u}\cdot t_\mathrm{0,b} = \frac{0.2396+0.1}{2}\cdot\frac{2/3}{1/3}\cdot t_\mathrm{0,b} = 2.717 \text{ hours}$$

The total mean flowtime of a lot in the line is

$$\varphi_\mathrm{tot} = \varphi_{\mathrm{B}_1} + t_0 + \varphi_\mathrm{Bk} + \varphi_\mathrm{Bq} + t_\mathrm{0,b} = 6.75 + 3 + 4 + 2.717 + 8 = 24.5 \text{ hours}$$

⊠

The lots leaving a batch machine, leave the machine in batches. With Equation 3.2 we can approximate the coefficient of variation on the inter-departure time. The inter-departure time of batches that is! If we were to have a single-lot machine behind the batch machine of Example 3.5, we can use the coefficient of variation on the inter-departure times of batches to determine the coefficient of variation on the inter-departure times of lots leaving the batch machine. But keep in mind that these inter-departure times are no longer independent. If we use Equation 3.1 for the single-lot machine behind the batch machine we can only approximate the queueing time!

3.6 Discussion

In this chapter we have presented analytical approaches to determine the flowtime for a flowline of workstations with stochastic inter-arrival times and stochastic processing times. We have even discussed ways to calculate the flowtime for batch machines. In some cases, the presented queueing relations allow an exact calculation of the flowtime. In many cases, the queuing relations allow for a reasonable approximation of the flowtime.

However, in practice, many manufacturing systems exist that still can not be analyzed using the queueing relations in this chapter. To illustrate this, consider the following examples:

- The queueing relations are applicable to simple flowlines of workstations. They can not be used for flowlines that feature rework or bypassing.

- The queueing relations assume infinite buffers that dispatch lots in a FIFO (First-In-First-Out) manner. They can not be used for manufacturing systems that have dispatching policies or that have finite buffer sizes.
- The queueing relations assume machines that process one lot (or batch) type at a time, where the process time for all lots (batches) is according to the same distribution. They can not be used for machines that process different lots in a different manner or that require set-up time.
- We have presented some simple extensions that allow us to calculate the flowtime for batch machines with fixed batch sizes. The relations presented there are unapplicable if we use a batch forming strategy that allows only a maximum wait-in-batch-time or variable batch size.

In short, there are a large number of examples, encountered in practice, that can not be analyzed with the relations from this chapter. Using probability theory and advanced queueing theory, it is possible to derive relations for these specific cases. However, this requires significant mathematical knowledge and effort and even then we have only an approximation. Moreover, if the systems become larger, the mathematical relations quickly become large and complex. The analytical approaches from previous and this chapter are very valuable in giving us a first impression of the system's performance. It provides us with a sense of the system's bottle-necks, the role of variability sources, and the order of magnitude of the flowtime, throughput, and wip-level.

However, for realistically sized problems, with realistic properties such as dispatching rules, finite buffers, and batching strategies, accurately determining the system's performance is best done with simulation. In the remainder of this book we present an approach to analyze different kinds of manufacturing systems using the discrete-event simulation language Chi. We start with simple systems and extend the models step by step. At the end of this book, we can virtually analyze any manufacturing system using Chi. At some points in the subsequent chapters, we discuss the practical useability of the relations presented in this chapter, by comparing simulation results with analytical results.

Keyword overview

Modelling manufacturing systems	Analysis of manufacturing systems
	• Kendall's notation
	• flowtime approximation for a $G/G/1$ workstation
	• Pollaczek-Khinchine's approximation
	• squared standard deviation
	• squared coefficient of variation
	• Kuehn's linking equation
	• flowtime approximation for a $G/G/m$ workstation
	• flowtime approximation for batch machines

3.7 Exercises

1. We have a manufacturing line with three workstations, see Figure 3.10. From measurements done, we have the following data available for each workstation:

   - Workstation 1 consists of an infinite buffer and 5 identical machines with $t_0 = 15.0$ min/lot and $c_0^2 = 1.0$.
   - Workstation 2 consists of an infinite buffer and 12 identical machines with $t_0 = 30.0$ min/lot and $c_0^2 = 1.0$.
   - Workstation 3 consists of an infinite buffer and one machine with $t_0 = 3.0$ min/lot and $c_0^2 = 1.0$.

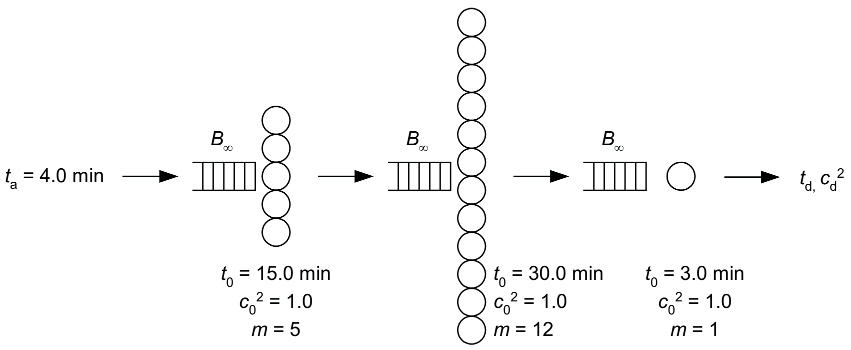


Figure 3.10: Manufacturing line with multiple-machine workstations

   We assume that lots arrive at workstation 1 with an inter-arrival time that is distributed according to an exponential distribution with mean 4.0 minutes.

   (a) Calculate the utilisation for workstation 1,2, and 3.
   (b) Calculate the mean waiting time in the buffer for workstation 1.
   (c) Calculate the mean inter-arrival time $t_{a,2}$ and squared coefficient of variation $c_{a,2}^2$ for lots arriving at workstation 2.
   (d) Calculate the mean waiting time in the buffer for workstation 2.
   (e) Calculate the mean waiting time in the buffer for workstation 3.
   (f) Calculate the mean number of products in buffer 1,2, and 3.
   (g) Explain the low wip-level in buffer 2.
   (h) Calculate the mean flowtime of a lot for the entire manufacturing line.
   (i) What is the mean number of products in the line (the wip-level)?

2. Consider a manufacturing line consisting of three workstations. Each workstation consists of an infinite buffer and a single machine. In addition we have the following data:

   - Lots arrive according to a distribution with mean t_a and squared coefficient of variation $c_a^2 = 2.0$.

- Workstations 1,2, and 3 have a mean process time of $t_{0,1} = 1.0$ hr, $t_{0,2} = 4.0$ hr, and $t_{0,3} = 1.0$ hr respectively and a squared coefficient of variation $c_{0,1}^2 = 4.0$, $c_{0,2}^2 = 1.0$, and $c_{0,3}^2 = 1.0$.

Figure 3.11 shows a graphical representation of the system.

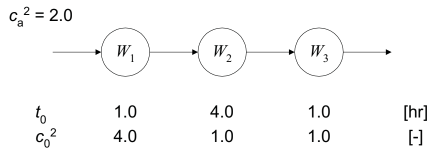


Figure 3.11: Three-workstation manufacturing line

(a) Calculate the maximum throughput $\delta_{\max}$.

(b) What is the mean flowtime of a product if we require a throughput $\delta = \delta_{\max}$?

We wish to obtain a throughput $\delta = 0.9\delta_{\max}$.

(a) Calculate the utilisation of machine 1,2, and 3.

(b) Calculate the mean flowtime of the line.

(c) Calculate the mean number of products in buffer 1,2, and 3.

(d) If you were to improve the flowtime and/or throughput of this system, with which machine would you start?

The management decides to invest in workstation 2. We have two alternatives. In alternative one we invest in speed. We speed up the machine in workstation 2 so that $t_{0,2} = 1.0$ hr, however this induces an increase in variability: the squared coefficient of variation $c_{0,2}^2$ becomes 4.0. In alternative two we simply add 3 similar machines to the line: workstation 2 consists of 4 identical machines with $t_{0,2} = 4.0$ hr and $c_{0,2}^2 = 1.0$. Both alternatives are graphically represented in Figure 3.12.

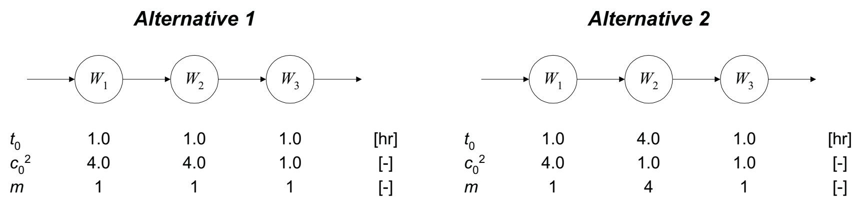


Figure 3.12: Two alternatives

(a) What is the maximum throughput for both alternatives?

(b) Calculate the mean flowtime for alternative 1 for $\delta = 0.9\delta_{\max}$.

(c) Calculate the mean flowtime for alternative 2 for $\delta = 0.9\delta_{\max}$.

(d) Calculate the wip-level for both alternatives.

(e) Which alternative do you prefer?

(f) At first sight, the alternatives yield only a small decrease (or even an increase) in flowtime, despite the investments done. Are these investments futile?

3. We have a batch machine that processes batches of fixed size k. Lots arrive with mean inter-arrival time $t_{\mathrm{a,l}} = 4.0$ hours and squared coefficient of variation $c^2_{\mathrm{a,l}} = 1.0$. The process time for a batch is proportional with the batch size: $t_{0,\mathrm{b}} = k \cdot t_0$, the squared coefficient of variation is inversely proportional with the batch size: $c^2_{0,\mathrm{b}} = c^2_0/k$. For this machine we have $t_0 = 3.0$ hours and $c^2_0 = 0.50$.

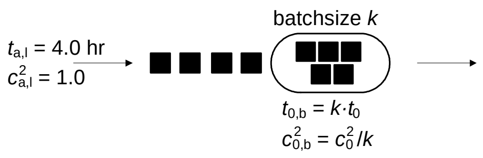

batchsize k
$t_{\mathrm{a,l}}$ = 4.0 hr
$c^2_{\mathrm{a,l}}$ = 1.0
$t_{0,\mathrm{b}} = k \cdot t_0$
$c^2_{0,\mathrm{b}} = c^2_0/k$

Figure 3.13: Batch machine process batches of fixed size k

(a) Express the mean total flowtime for this batch machine in batch size k.

(b) For what value of k is the flowtime minimal? (Do not forget to assure a stable system.)

4. We have three identical batch machines in parallel. The batch machines have a single buffer in which batches are formed. The machines process batches of a fixed size $k = 4$ lots. Lots arrive at the buffer with $t_{\mathrm{a,l}} = 0.50$ hours and $c^2_{\mathrm{a,l}} = 1.0$. The batch process time is characterized by $t_{0,\mathrm{b}} = 5.0$ hours and $c^2_{0,\mathrm{b}} = 0.5$.

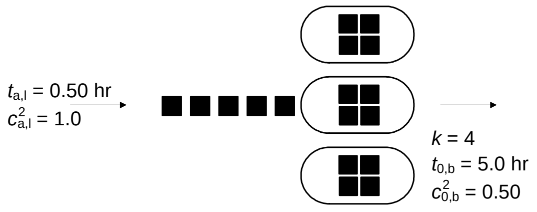

$t_{\mathrm{a,l}}$ = 0.50 hr
$c^2_{\mathrm{a,l}}$ = 1.0
k = 4
$t_{0,\mathrm{b}}$ = 5.0 hr
$c^2_{0,\mathrm{b}}$ = 0.50

Figure 3.14: Three batch machines in parallel

(a) Calculate the utilisation of the batch machines.

(b) Calculate the wait-to-batch-time in the buffer.

(c) Calculate the queueing time for a batch in the buffer.

(d) Calculate the mean total flowtime.

Chapter 4

Single workstation

In this chapter we take the first step into simulation-based analysis of a manufacturing system. By means of a single workstation, we present the first simple Chi-specification of a manufacturing system, and discuss the most elementary components of a manufacturing system.

4.1 Single machine model

We consider a single machine. This machine processes lots. To analyze the behavior of this machine, we define three processes, together forming model *GME*, see Figure 4.1.

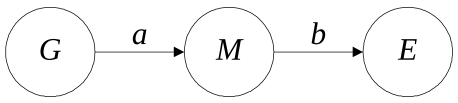


Figure 4.1: Model *GME*

Generator process `G` sends unfinished lots to machine `M`. Machine `M` processes these lots and sends finished lots to exit process `E`. We interpret these three processes as a representation of an entire factory, where process `G` represents the part of the factory before machine `M` and process `E` represents the part of the factory after machine `M`. In the sequel, we frequently encounter the concept of introducing a generator process and an exit process.

For this first analysis we assume that generator `G` generates a lot every 4.0 hours. For machine `M` this implies an *inter-arrival time* (time between two lot arrivals) t_a of 4.0 hours. Moreover, we assume that processing a lot on machine `M` takes also 3.0 hours, i.e. the *process time* t_0 equals 3.0 hours. Exit process `E` is always able to receive lots. We specify the processes `G`, `M`, `E`, and model `GME` as follows.

```
type lot = int;

proc G(chan! lot a):
   lot x;
   while true:                                              Chi-4.1
      a!x;
      delay 4.0;
      x = x + 1
   end
end

proc M(chan? lot a; chan! lot b):
   lot x;
   while true:
      a?x;                                                  Chi-4.2
      delay 3.0;
      b!x
   end
end

proc E(chan? lot a):
   lot x;
   for i in range(7):
      a?x;                                                  Chi-4.3
      write("%10.1f\tE\treceived lot %5d\n", time, x)
   end
end

model GME():
   chan lot a, b;                                           Chi-4.4
   run G(a), M(a,b), E(b)
end
```

Every lot is represented by a natural number, namely its id number. Generator `G` sends a lot (`a!x`) every 4.0 time units (1.0 time unit ≡ 1.0 hour). Every lot has a unique id number. The first lot has id number 0. After machine `M` has received a lot (`a?x`), the lot is processed for 3.0 hours, and then sent to exit `E` (`b!x`). Exit `E` can permanently receive lots. After each lot, exit `E` writes the current time (`time`) and the lot id number (`x`) to the screen.

In the model specification we connect generator `G` to machine `M` and machine `M` to exit `E`. Recall that an experiment environment is to be defined in order to simulate the model. Compiling and simulating this model (ams31.chi) yields the following output.

```
 3      E   received lot    0
 7      E   received lot    1
11      E   received lot    2
15      E   received lot    3
19      E   received lot    4
23      E   received lot    5
27      E   received lot    6
```

From the simulation output we tell that exit `E` receives finished lot 0 at `time = 3.0`. Subsequently, lots `1, 2, 3, ...` are received at equidistant intervals of 4.0 time units (here: hours). This corresponds to the expected simulation outcome. The expected behaviour is as follows. At `time = 0.0`, generator `G` sends the first lot 0 to machine `M`. At `time = 3.0`, machine `M` has finished the lot and sends it to exit `E`. So, lot 0 arrives at `time = 3.0` at exit `E`. As soon as `M` has sent the processed lot to `E`, it can receive a new lot from `G`. Only, at `time = 4.0` `G` is ready to send the next lot (lot 1) to `M`.

Below, we describe the behaviour of model3 `GME` over time by an execution scenario.

`time`	`G(a)`	`M(a,b)`	`E(b)`	`x`
0.0	`a!x`	`a?x`		0
0.0	`delay 4.0`	`delay 3.0`		
3.0		`b!x`	`b?x`	0
3.0			`write(...)`	
4.0	`x = x + 1`			
4.0	`a!x`	`a?x`		1
4.0	`delay 4.0`	`delay 3.0`		
7.0		`b!x`	`b?x`	1
7.0			`write(...)`	
8.0	`x = x + 1`			
8.0	`a!x`	`a?x`		2
...				

Table 4.1: Execution scenario for *GME*

In the execution scenario of Table 4.1 we see the statements that each of the parallel processes executes at different time instants. At `time = 0.0`, `G` sends the value of `x` over channel `a` to `M`. After this synchronous communication, variable `x` in process `M` has been assigned the value 0. Note that at `time = 4.0` several statements are performed. Here, we see that communication, output, and assignment statements are timeless. Time proceeds only when a time delay (delay statement) is encountered.

4.2 Single machine with varying process time

In practice, processing times are seldom deterministic. In most cases, we have variability on the process time, i.e. the process time is distributed according to some distribution. For this example we assume that the processing time is either 1.0 or 5.0 hours. Both possibilities have 50% chance. The specification of machine `M` becomes:

```
proc M(chan? lot a; chan! lot b):
   dist bool d = bernoulli(0.5);
   lot x;
   while true:
      a?x;
      if sample d:
         delay 1.0                                   Chi-4.5
      else:
         delay 5.0
      end;
      b!x
   end
end
```

On average it still takes machine `M` $(1.0 + 5.0)/2 = 3.0$ hours to process a lot. Simulation of model `GME` with this new machine `M`, yields the following results:

```
 5     E     received lot   0
 6     E     received lot   1
10     E     received lot   2
18     E     received lot   3
23     E     received lot   4
28     E     received lot   5
```

Though, on average process `M` still processes one lot every 3.0 hours, and generator `G` tries to generate a lot every 4.0 hours, on average process `E` receives a lot only every 4.5 hours instead of one lot every 4.0 hours as for the former example. We determine the average of one lot per 4.5 hour, by running the simulation for a longer time interval, for example for 10000 hours and divide the total simulated time by the total number of products.

Closer investigation of the behavior of generator `G` and machine `M` is required to explain this drop in performance. Hereto, we use the execution scenario in Table 4.2 that belongs to the simulation results shown.

Read the following explanation, while studying the execution scenario. At `time = 0.0` machine `M` receives lot 0 from generator `G`. In this simulation, the first sample from bernoulli distribution d yields `false`, resulting in a process time of 5.0 hours. At `time = 4.0` generator `G` is willing to send lot 1, but machine `M` is still processing lot 0. At `time = 5.0` machine `M` finishes lot 0 and sends it to exit `E`. Exit `E` receives the first lot at `time = 5.0`. Immediately, machine `M` receives lot 1 from `G`. The next sample yields `true`, resulting in a process time of 1.0 hour. At `time = 6.0` `M` sends lot 1 to exit `E`. `E` receives the second lot at `time = 6.0`. Machine `M` is willing to receive the next lot, however, generator `G` provides the next lot at `time = 5.0 + 4.0 = 9.0`. Only at `time = 9.0`, machine `M` can start processing on lot 2. That lot is finished at `time = 10.0`.

In the first situation, where generator `G` cannot send the next lot, we speak of *blocking*, i.e. machine `M` blocks the manufacturing system. In the second situation, where machine `M` has to wait for generator `G` to send the next lot, we speak of *starving*, i.e. machine `M` is starving.

The effect of variability on the process time is that the average output is lower than the average maximum output of 0.25 lots per hour. The effects of variability are even worse,

τ	$G(a)$	$M(a,b)$	$E(b)$	x
0.0	`a!x`	`a?x`		0
0.0	`delay 4.0`	`delay 5.0`		
4.0	`x= x + 1`			
5.0		`b!x`	`b?x`	0
5.0			`write(time, ...)`	
5.0	`a!x`	`a?x`		1
5.0	`delay 4.0`	`delay 1.0`		
6.0		`b!x`	`b?x`	1
6.0			`write(time, ...)`	
9.0	`x= x + 1`			
9.0	`a!x`	`a?x`		2
9.0	`delay 4.0`	`delay 1.0`		
10.0		`b!x`	`b?x`	2
10.0			`write(time, ...)`	
10.0	`x= x + 1`			
...				

Table 4.2: Execution scenario 2 for *GME*

if we apply a same bernoulli distribution with 1.0 and 5.0 hours on the inter-arrival times (Try!). Moreover, note that individual simulations do not yield identical results!

To cope with variability on process times as well as with variability on inter-arrival times we use buffers.

4.3 Single machine with buffer

We continue to investigate the model of the previous section, where machine `M` has process times of either 1.0 or 5.0 hours. We saw that the average output in case of stochastic process times was less than in case of deterministic process times, whilst the average process time for both cases was identical. We identified the cause: blocking and starving. To reduce the effect of blocking and starving we introduce a buffer with infinite capacity before machine `M`. We obtain model `GBME`, see Figure 4.2.

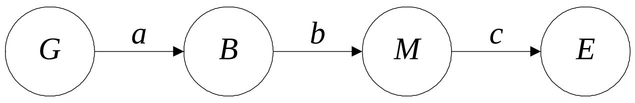


Figure 4.2: Model *GBME*

The specification of generator `G`, machine `M`, and exit `E` remains unchanged and is found in Specifications Chi-4.1, Chi-4.5, and Chi-4.3 respectively. We now present the specification of buffer `B`.

```
proc B(chan? lot a; chan! lot b):
   list lot xs; lot x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end
```

Chi-4.6

Buffer B is always able to receive lots via channel a. If a lot is received, it is added at the end of list xs. As long as there are lots in the buffer (not empty(xs)), buffer B is willing to send the first lot of the list (b!xs[0]). After sending the lot, the buffer removes the first lot from the list (xs = xs[1:]).

The model specification becomes:

```
model GME():
   chan lot a, b, c;
   run G(a), B(a, b), M(b, c), E(c)
end
```

Chi-4.7

Compilation and simulation yields the following output.

```
 1     E     received lot   0
 9     E     received lot   1
10     E     received lot   2
17     E     received lot   3
22     E     received lot   4
27     E     received lot   5
32     E     received lot   6
```

After running the simulation for a longer time interval, we determine that on average exit E now receives a lot every 4.0 hours. The buffer has eliminated the effect of blocking and starving on the long term average output. We investigate the behaviour of model *GBME* by means of the execution scenario in Tables 4.3 and 4.4 that belongs to the simulation results shown.

τ	$G(a)$	$B(a,b)$	$M(b,c)$	$E(c)$
0.0	`a!x`	`a?x`		
0.0	`delay 4.0`	`xs = xs + [x]`		
0.0		`b!xs[0])`	`b?x`	
0.0		`xs=xs[1:]`	`delay 1.0`	
1.0			`c!x`	`c?x`
1.0				`write(time, ...)`
4.0	`x= x + 1`			
4.0	`a!x`	`a?x`		
4.0	`delay 4.0`	`xs = xs + [x]`		
4.0		`b!xs[0]`	`b?x`	
4.0		`xs = xs[1:]`	`not sample d: delay 5.0`	
8.0	`x= x + 1`			
8.0	`a!x`	`a?x`		
8.0	`delay 4.0`	`xs = xs + [x]`		
9.0			`c!x`	`c?x`
9.0				`write(time, ...)`
9.0		`b!xs[0]`	`b?x`	
9.0		`xs = xs[1:]`	`sample d: delay 1.0`	
10.0			`c!x`	`c?x`
10.0				`write(time, ...)`
...				

Table 4.3: Execution scenario for *GBME*

τ	$G(a)$	$B(a,b)$	$M(b,c)$	$E(c)$
...				
12.0	`x= x + 1`			
12.0	`a!x`	`a?x`		
12.0	`delay 4.0`	`xs = xs + [x]`		
12.0		`b!xs[0]`	`b?x`	
12.0		`xs = xs[1:]`	`not sample d:  delay 5.0`	
...				
16.0	`x= x + 1`			
16.0	`a!x`	`a?x`		
16.0	`delay 4.0`	`xs = xs + [x]`		
17.0			`c!x`	`c?x`
17.0				`write(time, ...)`
17.0		`b!xs[0]`	`b?x`	
17.0		`xs = xs[1:])`	`not sample d:  delay 5.0`	
20.0	`a!x`	`a?x`		
20.0	`delay 4.0`	`xs = xs + [x]`		
...				

Table 4.4: Execution scenario for *GBME* (continued)

Read the following explanation, while studying the execution scenario. At `time = 0.0` generator `G` sends a lot to the buffer. Machine `M` is available, and at the same time instant this lot is sent from the buffer to the machine. Sampling from distribution d yields `true`; processing the first lot takes 1.0 hour. At `time = 1.0`, the exit receives the first finished lot. At `time = 1.0`, buffer `B` is empty: machine `M` has to wait, until the buffer receives a lot. Machine `M` is starving. At `time = 4.0` the buffer receives lot 1. The buffer immediately sends this lot to machine `M`. Processing this lot takes 5 hours. At `time = 8.0` generator `G` wants to sent lot 2. Though machine `M` is still in process, the buffer allows the lot to be sent. Buffer `B` prevents the generator from being blocked by machine `M`. After machine `M` finished processing lot 1, at `time = 9.0`, it can immediately continue with lot 2 from the buffer.

Here, buffer `B` is always able to receive new lots. It has an infinite number of buffer places. We speak of an *infinite buffer*. In practice, buffers have a limited number of buffer places. A finite buffer with N buffer places is specified as follows:

```
proc B(chan? lot a; chan! lot b; int N):
   list lot xs; lot x;
   while true:
      select
         size(xs) < N, a?x:
            xs = xs + [x]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end
```

Chi-4.8

As long as there are less than `N` lots in the buffer (`size(xs) < N`), the buffer is able to receive new lots via channel `a`.

4.4 Displaying simulated behaviour

There are multiple ways to display simulation results. Apart from textual output, we presented an execution scenario. The execution scenario provides a complete and accurate way to describe the behaviour of the simulation model. However, for quick insight on the behaviour of the simulation model, we prefer to visualize the simulated behaviour. Here we present two approaches to graphically display the simulation behaviour: the lot-time and the resource-time diagram. In a lot-time-diagram, which we encountered in Chapter 2, we plot the state of individual lots against time. In a resource-time-diagram we plot the state of the individual resources against time.

Figure 4.3 shows a lot-time-diagram distilled from the execution scenario in Tables 4.3 and 4.4.

In a resource-time diagram we record per resource what job or lot is processed. These diagrams are often referred to as Gantt charts [Gan19]. Figure 4.4 shows a resource-time-diagram corresponding to the execution scenario in Tables 4.3 and 4.4.

Besides the lot-time-diagram in which we follow lots and the resource-time-diagram in which we follow resources, we can plot any quantity or state against time. In Figure 4.5 the number of lots in the buffer and the state of the machine (idle or busy) are plotted against time.

All these visualizations provide easy insight on the simulated behaviour. They are valuable in *verifying* your model, that is, checking whether your simulation model behaves as intended. For deterministic systems we used a lot-time-diagram to determine the mean flow-time and wip-time-diagram to determine the mean wip-level, see Section 2.2. For a simulated stochastic system, such a diagram visualizes only (a short period of) one of the many possible simulation outcomes. Where it does provide insight on the behaviour of the

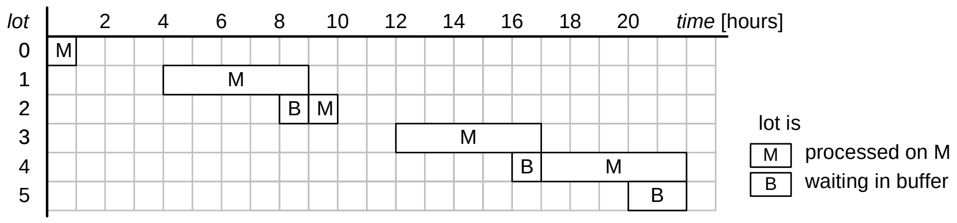


Figure 4.3: Lot-time-diagram of the simulation in Section 4.3

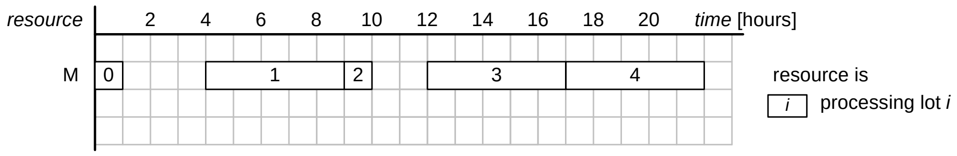


Figure 4.4: Resource-time-diagram of the simulation in Section 4.3

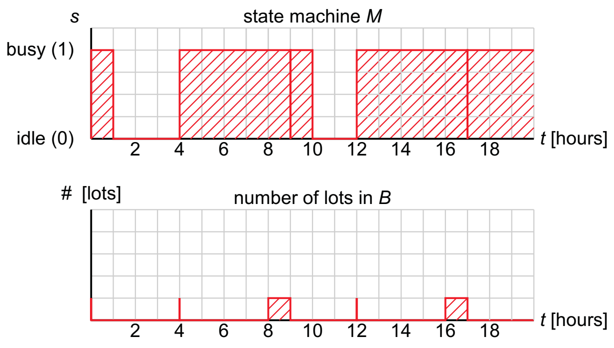


Figure 4.5: (a) State of machine M (idle or busy), (b) Number of lots in buffer B

simulated manufacturing system, it is of little use in determining the mean flowtime, wip-level or any quantity. In the next chapter, we discuss how to determine the most important performance indices for manufacturing systems.

Keyword overview

Modelling manufacturing systems	Analysing manufacturing systems
• generator	• starving and blocking
• exit	• process time variability
• machine with deterministic process time	
• machine with stochastic process time	
• finite buffer	
• infinite buffer	
• execution scenario	

4.5 Exercises

1. Machine N processes at two different speeds. In fifty percent of the cases the process time is 0.3 hour, and the other fifty percent the process time is 0.1 hour. We have the following Chi-specification:

```
type lot = int;

proc G(chan! lot a):
   lot n;
   while true:
     a!n; n = n + 1
   end
end

proc N(chan? lot a; chan lot b):
   dist int u = uniform(0,2);
   lot x;
   while true:
      a?x;
      if ...:
         ...
      else:
         ...                                             Chi-4.9
      end;
      b!x
    end
end

proc E(chan? lot a):
   int i; lot x;
   while true:
      a?x; i = i + 1;
      write("%10.1f\tE\treceived lot %5d\n", time, x)
   end
end

model GME():
   chan lot a, b;
   run G(a), M(a,b), E(b)
end
```

   (a) Make a graphical representation of this system.
   (b) Predict the mean number of lots that leave the system per hour.
   (c) Finish the specification and simulate the model.
   (d) Determine the mean number of lots that leave the system per hour by simulation.
   (e) Compare the outcome determined by simulation with your predicted outcome.

2. Here is a specification of model S that consists of generator G, 4-place buffer B, and exit process E. The generator ten times generates a lot every 10 seconds, and the exit

process removes lots from the buffer each 5 seconds. If a lot is added or removed, the new buffer contents are printed.

```
type lot = int;

proc G(chan! lot a):
   for i in range(10):
      a!i;
      delay 10.0
   end
end

proc B(chan? lot a; chan! lot b; int N):
   list int xs; int x;
   while true:
      select
         size(xs) < N, a?x:
            xs = xs + [x]
      alt
         size(xs) > 0, b!xs[0]:
            xs = xs[1:]
      end;
      write("%10.1f %s\n", time, xs)
   end
end

proc E(chan? lot a):
   int x;
   while true:
      a?x; delay 5.0
   end
end

model S():
   chan lot a,b;
   run G(a),  B(a, b, 4), E(b)
end
```

Chi-4.10

(a) Compile the specification, simulate and explain the results.

(b) What happens if the buffer size is set to two instead of four?

(c) Change the specification such that the generator needs 5 seconds to generate a lot, and the exit needs 10 seconds to consume a lot. Study again a 4-place and a 2-place buffer.

3. Consider a workstation consisting of infinite buffer B and machine M. The machine can process three types of products. The product type is denoted by a natural (0,1,2). The processing time depends on the product type. The processing time is 1.0, 3.0, and 6.0 minutes for product type 0,1, and 2 respectively, see Figure 4.6.

   The type of the products arriving at B is uniformly distributed. Products arrive at buffer B with a constant inter-arrival time of 4 minutes. Finished products immediately can leave the workstation.

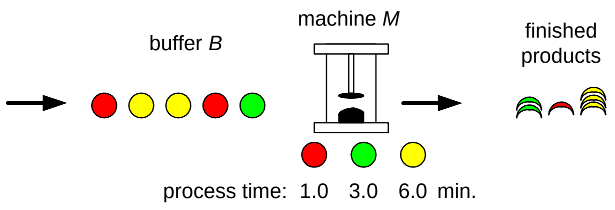


Figure 4.6: Work station processes with three product types

(a) Complete the specification below that represents the workstation and its environment.

```
type lot = int;

proc G(chan! int a):
   dist int d = uniform(...);
   while true:
      a!sample d; delay ...
   end
end

proc M(chan? lot a; chan! lot b):
   lot x;
   ...
end

proc B(chan? lot a; chan! lot b):
   ...
end

proc E(chan? lot a):
   lot x;
   while true:
      a?x
   end
end

model GBME():
   chan lot a, b, c;
   run G(a), B(a, b), M(b, c), E(c)
end
```

Chi-4.11

(b) Adapt the specification of exit `E`, so that it prints the number of lots that have been processed by the manufacturing system to the screen. How many products are produced by the system per hour?

(c) Determine the number of products that are produced per hour, specified per product type.

(d) We wish to change the first-in-first-out behavior of the buffer. Each product type has a different priority. Priority increases with the product type number, i.e. product type 2 has the highest and product type 0 has the lowest priority. So products of type 2 are processed first, then products of type 1 and then products of type 0. Adapt the specification and determine by simulation how many products are processed per hour, specified per product type. (Hint: use (a tuple of) three lists in buffer `B` or use list sorting.)

Chapter 5

Measuring performance

In the examples of the previous chapter we have investigated the behavior of a simple manufacturing line. Amongst others, we determined the number of lots that the manufacturing line processes per hour. In Chapter 2 we presented a number of quantities that characterize the performance of a manufacturing system. In this chapter, we present an approach to determine the four most frequently used performance indicators by simulation. These performance indicators are: throughput δ, flowtime φ, utilization u, and work-in-process or wip-level w.

Typically, the throughput, utilization, and wip-level change over time. The flowtime varies per lot. Let us reconsider the simulation from Section 4.3. From Figure 4.3 we determine that the flowtime for lot 0,1,2, and 3 is 1.0, 5.0, 2.0, and 5.0 hours respectively. We see that wip-level w is 1 at for instance $\tau = 1.0$ and 0 at $\tau = 3$. For $0.0 \leq \tau \leq 20.0$ the throughput is $4/20 = 0.2$ lots/hour. From Figure 4.4 we determine that the utilization for $0.0 \leq \tau \leq 20.0$ is $15/20 = 0.8$.

In general, we are not interested in the momentous value of performance indicators, but in the *mean* throughput $\overline{\delta}$, mean flowtime $\overline{\varphi}$, mean utilization $\overline{u}$ and mean wip-level $\overline{w}$. Herein, throughput, utilization, and wip-level are averaged over time, and flowtime is averaged over the lots! In the sequel, we present an approach to calculate the mean values of the throughput, flowtime, utilization and wip-level. We propose a way to adapt your Chi-specification to determine these performance indicators during simulation and apply it on the manufacturing system from Section 4.3.

5.1 Mean throughput

The mean throughput $\overline{\delta}$ at time instant τ is calculated by dividing the number of lots n that have exited the system up to time instant τ by the current time τ. In formula:

$$\overline{\delta} = \frac{n}{\tau} \tag{5.1}$$

To determine the mean throughput by simulation, we adapt the specification of exit `E`.

```
proc E(chan? lot a):
   lot x;
   for i in range(1, 1000):
      a?x;                                                 Chi-5.1
      write("%10.2f\tE\tmean throuhgput =\t%10.3f\n", time, i / time)
   end
end
```

We use i to count the number of lots that has left the system. The mean throughput is calculated by dividing the number of lots i by the current (simulation) time τ. Note that in the previous specifications, a lot is represented by an increasing id number, starting with 0. So, in this case, we could also calculate the mean throughput by $(x + 1)/\tau$.

Simulation of model *GBME* from Section 4.3 with this adapted exit `E` yields:

```
    5   E   mean throughput =   0.2
   10   E   mean throughput =   0.2
   11   E   mean throughput =   0.272727
   17   E   mean throughput =   0.235294
 ...
   85   E   mean throughput =   0.258824
   93   E   mean throughput =   0.247312
   94   E   mean throughput =   0.255319
 ...
 9985   E   mean throughput =   0.249975
 9990   E   mean throughput =   0.24995
 9995   E   mean throughput =   0.249925
10000   E   mean throughput =   0.2499
```

After simulating for a large time interval the mean throughput converges to 0.25 lots/hour (1 time unit in this specification is equivalent to 1 hour). This throughput corresponds to the results we found at the end of Section 4.3, where exit `E` received one lot every 4.0 hours.

5.2 Mean flowtime

To calculate the mean flowtime per lot, we record the time at which a lot enters the system. Hereto, we represent a lot by a record tuple of type (nat,real), where the first element denotes the id number and the second element the time at which the lot entered the system. The type declaration and generator `G` become:

```
type lot = real;

proc G(chan! lot a):
   while true:                                             Chi-5.2
      a!time; delay 4.0
   end
end
```

When exit `E` receives a lot it calculates flowtime φ of that lot by $\tau - x.1$. One way of calculating mean flowtime $\overline{\varphi}$ of n lots is adding up the flowtime of all individual lots and dividing it by the number of lots.

$$\overline{\varphi}_n = \frac{1}{n}\sum_{i=1}^{n}\varphi_i \tag{5.2}$$

Herein $\overline{\varphi}_n$ is the mean flowtime of n lots and φ_i the flowtime of the i-th lot. We prefer to use the following iterative (and numerically more accurate) approach to calculate the mean flowtime:

$$\overline{\varphi}_i = \begin{cases} 0 & \text{for } i = 0 \\ \dfrac{i-1}{i}\cdot\overline{\varphi}_{i-1} + \dfrac{\varphi_i}{i} & \text{for } i = 1, 2, 3, \ldots \end{cases} \tag{5.3}$$

Herein $\overline{\varphi}_i$ is the mean flowtime of i lots. In exit `E` we implement this iterative calculation as follows:

```
proc E(chan? lot a):
   real mphi; lot x;
   for i in range(1, 100000):
      a?x;
      mphi = (i - 1) / i * mphi + (time - x) / i;
      write("%10.2f\tE\tmean flow time =\t%10.3f\n", time, mphi)
    end
end
```

Chi-5.3

Note that in this Chi-specification $\overline{\varphi}_i$ is represented by $\overline{\varphi}$. Moreover, $\overline{\varphi}$ is initialized for $i = 0$ to 0.0. Simulation yields:

```
    1   E       mean flowtime = 1
    9   E       mean flowtime = 3
   14   E       mean flowtime = 4
   15   E       mean flowtime = 3.75
   21   E       mean flowtime = 4
   26   E       mean flowtime = 4.33333
 ...
   89   E       mean flowtime = 2.54545
   90   E       mean flowtime = 2.52174
   97   E       mean flowtime = 2.625
   98   E       mean flowtime = 2.6
...
49989   E       mean flowtime = 4.15468
49990   E       mean flowtime = 4.1545
49997   E       mean flowtime = 4.15457
49998   E       mean flowtime = 4.1544
```

After simulating for a sufficient long time interval the mean flowtime converges to 4.15 hours. On average, processing a lot takes 3.0 hours, thus on average a lot waits for 1.15 hour in the buffer.

5.3 Mean utilization

The mean utilization of a machine is defined as the fraction of time that a machine is busy. In Figure 5.1 the state s (idle or busy) of machine M is plotted against time.

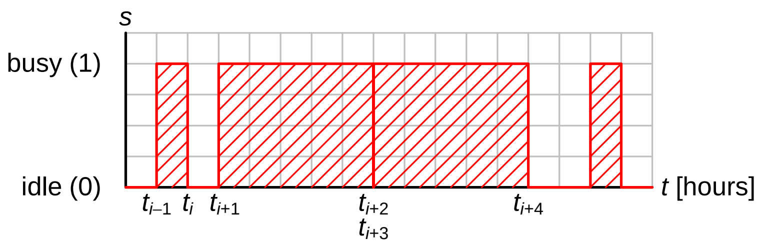


Figure 5.1: State (idle or busy) of machine M

From this data we can calculate mean utilization $\overline{u}_n$ after n events as follows:

$$\overline{u}_n = \frac{1}{t_n} \sum_{i=1}^{n} (t_i - t_{i-1}) \cdot s_{i-1} \tag{5.4}$$

Herein $\overline{u}_n$ is the mean utilization after the n-th event, t_i is the time instant of the i-th event and s_i is the state after the i-th event. $s_i = 0$ if the machine is idle after the i-th event, and $s_i = 1$ if the machine is busy after the i-th event. We prefer the following iterative (and numerically accurate) way of calculating the mean utilization:

$$\overline{u}_i = \begin{cases} 0 & \text{for } i = 0 \\ \dfrac{t_{i-1}}{t_i} \cdot \overline{u}_{i-1} + \dfrac{t_i - t_{i-1}}{t_i} \cdot s_{i-1} & \text{for } i = 1, 2, 3, \ldots \end{cases} \tag{5.5}$$

Herein, $\overline{u}_i$ is the mean utilization after i events. We have to recalculate mean utilization $\overline{u}_i$ after each event that changes the state s_i. To this end, we adapt the specification of machine M as follows:

```
proc M(chan? lot a; chan! lot b):
   real tim1, u;
   dist bool d = bernoulli(0.5);
   lot x;
   while true:
      a?x;
      tim1, u = tAver(tim1, time, u, 0);
      write("%10.2f\tM\tmean utilisation =\t%10.2f\n", time, u);
      if sample d:
         delay 1.0
      else:
         delay 5.0
      end;
      b!x;
      tim1, u = tAver(tim1, time, u, 1);
      write("%10.2f\tM\tmean utilisation =\t%10.2f\n", time, u)
   end
end
```

Chi-5.4

Variable $\overline{u}$ represents mean utilization $\overline{u}_i$. Every time the machine receives or sends a lot, state s_i changes and mean utilization $\overline{u}_i$ has to be updated. The time instant t_i of that update is the current simulation time τ. The time instant of the previous update is t_{i-1}. Recalculation of mean utilization $\overline{u}$ and updating the time instant t_{i-1} of the most recent update is performed by function tAver.

```
func tuple(real t; real u) tAver(real tim1, ti, uim1; int sim1):
   if ti == 0.0:
      return (ti, uim1)
   else:
      return (ti, tim1 / ti * uim1 + ( ti - tim1 ) / ti * sim1)
   end
end
```

Chi-5.5

As long as the time instant of the update is zero, the new average and last time instant remain unchanged. Otherwise, the new average is calculated conform Equation 5.5. If machine M receives a lot, the machine goes from idle to busy; the previous state $s_{i-1} = 0$. After the machine has sent the lot, the machine state switches from busy to idle; the previous state $s_{i-1} = 1$.

Compiling and simulating yields:

```
     5   M      mean utilization =     1
     6   M      mean utilization =     1
    13   M      mean utilization =     0.846154
    14   M      mean utilization =     0.857143
  ...
    85   M      mean utilization =     0.588235
    89   M      mean utilization =     0.573034
    93   M      mean utilization =     0.55914
  ...
```

```
49993   M       mean utilization =      0.753746
49994   M       mean utilization =      0.75375
49997   M       mean utilization =      0.753725
```

Running the simulation for a sufficient long time interval causes the mean utilization to converge to 0.75. This corresponds to the expected behavior. The manufacturing system receives a lot every 4.0 hours. On average it takes the machine 3.0 hours to process a lot. Thus, on average the machine is busy for 75% of the time.

5.4 Mean wip-level

Figure 5.2 shows how the wip-level of the manufacturing line can change over time.

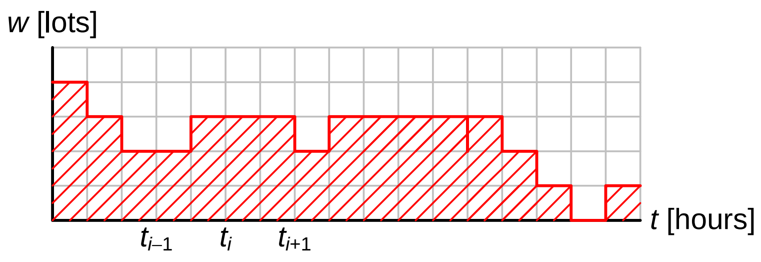


Figure 5.2: Work-in-process level over time

The time averaged wip-level $\overline{w}_n$ after n events is given by:

$$\overline{w}_n = \frac{1}{t_n}\sum_{i=1}^{n}(t_i - t_{i-1})\cdot w_{i-1} \tag{5.6}$$

Herein, $\overline{w}_n$ is the mean wip-level after n events, t_i the time instant of the i-th event, and w_i the wip-level after the i-th event. Iteratively:

$$\overline{w}_i = \begin{cases} 0 & \text{for } i = 0 \\ \dfrac{t_{i-1}}{t_i}\cdot\overline{w}_{i-1} + \dfrac{t_i - t_{i-1}}{t_i}\cdot w_{i-1} & \text{for } i = 1, 2, 3, ... \end{cases} \tag{5.7}$$

Here, $\overline{w}_i$ is the mean wip-level after the i-th event. We have to recalculate mean wip-level $\overline{w}_i$ after every event that affects the current wip-level w_i. To determine the wip-level during simulation we need to adapt both generator `G` and exit `E`. We choose to let exit `E` calculate the wip-level. `E` is aware of lots leaving the manufacturing system, however `E` is unaware of lots entering the manufacturing system. Therefore, we let generator `G` send a signal to `E` to notify `E` a lot has entered the system. We introduce channel z from `G` to `E`. The graphical representation of model `GBME` is shown in Figure 5.3.

The specification of generator `G` and exit `E` become:

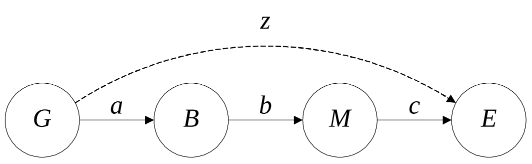


Figure 5.3: Model *GBME* with wip-update channel

```
proc G(chan! lot a; chan! void z):
   while true:
      a!time; z!; delay 4.0                                  Chi-5.6
   end
end

proc E(chan? lot a; chan? void z):
   real tim1, wmean; int w;
   lot x;
   for i in range(100000):
      select
         a?x:
            tim1, wmean = tAver(tim1, time, wmean, w);
            write("%10.2f\tM\tmean wip =\t%10.2f\n", time, wmean);
            w = w - 1                                        Chi-5.7
      alt
         z?:
            tim1, wmean = tAver(tim1, time, wmean, w);
            write("%10.2f\tM\tmean wip =\t%10.2f\n", time, wmean);
            w = w + 1
      end
   end
end
```

Every time a lot enters the system, generator G sends a signal via channel z to exit E. In exit E, variable w represents current wip-level w_i, and $\overline{w}$ represents mean wip-level $\overline{w}_i$. Exit E is always able to receive lots via channel a and signals from G via channel z. After either communication, mean wip-level $\overline{w}$ is recalculated and current wip-level w is updated. Note that the recalculation of mean wip-level $\overline{w}$ occurs before updating current wip-level w. (Try!: Look at Figure 5.2 and Equation 5.7 and argue why this is true.) For this recalculation we use the same function tAver as we did for calculating the mean utilization, see Specification Chi-5.5.

Simulation yields:

```
   0   E      mean wip =      0
   4   E      mean wip =      1
  10   E      mean wip =      1.3
  11   E      mean wip =      1.27273
...
  94   E      mean wip =      1.08511
```

```
   96   E       mean wip =      1.0625
  100   E       mean wip =      1.06
 ...
49996   E       mean wip =      1.04572
49999   E       mean wip =      1.04578
50000   E       mean wip =      1.04578
```

After running the simulation for a sufficient long time we see that the mean wip-level converges to 1.05.

From mean throughput $\overline{\delta}$ determined in Section 5.1 and mean flowtime $\overline{\varphi}$ determined in Section 5.2, we can calculate mean wip $\overline{w}$ by applying Little's law: $\overline{w} = \overline{\delta} \cdot \overline{\varphi} = 0.25 \times 4.15 = 1.04$ lots. This corresponds to mean wip-level $\overline{w}$ determined by simulation.

5.5 Variance

Apart from the mean of a quantity we are sometimes interested in its variance (squared standard deviation) or coefficient of variation (which we can determine from the variance and the mean). For instance, the variance on the flowtime allows us to predict delivery reliability.

We iteratively determine the squared standard deviation with the following relation [Lef03]:

$$\begin{cases} s_i^2 = \dfrac{i-2}{i-1} \cdot s_{i-1}^2 + \dfrac{(x_i - \overline{x}_{i-1})^2}{i} & \text{for } i > 1 \\ \\ s_i^2 = 0 & \text{for } i = 0, 1 \end{cases} \tag{5.8}$$

Herein s_i^2 is the estimated variance of i drawings x_i, and $\overline{x}_{i-1}$ is the estimated mean value of $(i-1)$ drawings x_i. The squared coefficient of variation c_i^2 is obtained by dividing squared standard deviation s_i^2 by squared mean $\overline{x}_i^2$.

In exit process `E` below, we use this iterative formula to determine the variance on the flowtime.

```
proc E(chan? lot a):
   real phi, phi2, mphi, s2phi;
   lot x;
   for i in range(1, 100000):
      a?x; phi = time -x; phi2 = phi * phi;
      if i == 1:
         s2phi = 0.0
      else:
         s2phi = (i - 2) / (i - 1) * s2phi + phi2 / i;
         mphi  = (i - 1) / i * mphi + phi / i
      end;
      write("%10.2f\tE\tmean flow time =\t%10.3f\t", time, mphi);
      write("var  flow time =\t%10.3f\n", mphi);
   end
end
```

Chi-5.8

Note that we first recalculate the squared standard deviation, before we recalculate the mean! From Equation 5.8, we see that in the calculation of s_i^2 for i lots, we use the mean $\overline{x}_{i-1}$ of $(i-1)$ lots.

5.6 Measuring other quantities

It is possible to measure other quantities than the ones discussed in this chapter, such as the mean number of lots in a buffer, the mean number of operations on a lot, the mean inter-arrival time at a specific machine. Hereto, you can use an approach analogous to the ones discussed. With respect to determining the mean of a quantity, carefully consider the following:

- Is the quantity to be averaged over time (for example in case of the mean number of lots in a buffer) or over the lots (for example in case of the mean number of operations per lot)?
  - If the quantity is to be averaged over time, use an approach analogous to the one discussed for the mean utilization or mean wip-level.
  - If the quantity is to be averaged over the lots, use an approach analogous to the one discussed for the mean flowtime.
- Recalculate the mean for every event that affects the momentous value of the quantity. This recalculation should be performed just before updating the momentous value of that quantity.

5.7 Simulation vs Approximation (1): single machine workstation

In this section we compare analytical results with simulation results. In Chapter 3 we presented the Pollaczek-Khintchine approximation for the waiting time in a buffer. We argued that it was exact for a M/G/1 queueing system. In this section, we consider two manufacturing systems to investigate this.

Example 5.1 $D/G/1$ system
First, we investigate the manufacturing system of Section 4.3, consisting of a single workstation with an infinite buffer and a machine with a process times of 1.0 or 5.0 hours, both with 50% probability. The inter-arrival time is fixed, at 4.0 hours. We are thus dealing with a $D/G/1$ system.

In Section 5.2 we determined that the flowtime for this system was 4.15 hours. Now let us see what the analytical approach from Chapter 3 yields. We calculate the coefficient of variation on the process times.

$$s_0^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{(1.0-3.0)^2 + (5.0-3.0)^2}{2} = 4.0$$

$$c_0^2 = \frac{s_0^2}{t_0^2} = \frac{4.0}{3.0^2} = \frac{4}{9}$$

The inter-arrival times are fixed and thus the coefficient of variation on the inter-arrival times is 0. The mean utilization is 0.75. We calculate the flowtime:

$$\varphi = (\frac{c_a^2 + c_0^2}{2} \cdot \frac{u}{1-u}) \cdot t_0 + t_0 = (\frac{0 + 4/9}{2} \cdot \frac{0.75}{1-0.75}) \cdot 3.0 + 3.0 = 5.0 \text{ hours}$$

This analytically determined flowtime deviates by about 20% from the simulation results. For this $D/G/1$ system, the Pollaczek-Khintchine formula is nothing more than an approximation! ⊠

Example 5.2 $M/G/1$ **system**
Once more we look at the workstation from Section 4.3, but we let the inter-arrival times be distributed exponentially with mean 4.0 hours. The process times remain 1.0 or 5.0 hours, both with 50% chance. We now have an $M/G/1$ system and the Pollaczek-Khintchine approximation should be exact.

In Example 3.2 we calculated the flowtime for this workstation:

$$\varphi = (\frac{c_a^2 + c_0^2}{2} \cdot \frac{u}{1-u}) \cdot t_0 + t_0 = (\frac{1 + 4/9}{2} \cdot \frac{0.75}{1-0.75}) \cdot 3.0 + 3.0 = 9.5 \text{ hours}$$

Now let us see what simulation yields. We use the model from Sections 4.2 and 4.3. We adapt generator `G` so that it releases lots with exponentially distributed inter-arrival times.

```
proc G(chan! lot a):
   dist real d = exponential(4.0);
   while true:
      a!time; delay sample d
   end
end
```

Chi-5.9

We use the exit process `E` from Section 5.2 to determine the mean flowtime. Simulation yields:

```
[user@host chi]./ams37
41109.3 E       mean flowtime = 9.08576
280679  E       mean flowtime = 9.32406
441615  E       mean flowtime = 9.29845
521513  E       mean flowtime = 9.37252
...
8.03191e+06     E       mean flowtime = 9.48781
8.07225e+06     E       mean flowtime = 9.489
8.1125e+06      E       mean flowtime = 9.489
```

We see that after simulating for a sufficient long time, after over 2 million lots[1], the mean flowtime starts to converge to 9.5 hours. For this $M/G/1$ system the Pollaczek-Khintchine approximation is exact! ⊠

[1](Try!: How can you estimate the number of lots from the simulation output shown above?)

5.8 Discussion

In the sequel, we frequently mention the performance indicators throughput δ, flowtime φ, utilization u and wip-level w. Unless explicitly stated otherwise, we are always referring to the mean throughput, mean flowtime, mean utilization and mean wip-level. For ease of reading, we omit the bar above the performance indicator symbols. We use φ_i to refer to the flowtime of the i-th lot, and we use φ to refer to the mean flowtime. Similarly, we use w_i or $w(t)$ to refer to the wip-level at a certain point in time, and we use w to refer to the mean wip-level.

Keyword overview

Modelling manufacturing systems	Analyzing manufacturing systems
	• by simulation, determining • mean flowtime φ, • mean throughput δ, • mean work-in-process w, • mean utilization u, and • variance • approximation vs simulation

5.9 Exercises

1. By hand, prove that Equation 5.3 is equal to Equation 5.2 for $i = 1, 2, 3, 4$.
2. For a workstation system in a factory, lot movement is recorded in a logfile. The workstation consists of a buffer and a machine. In Figure 5.4 a part of the logfile is shown.

   The data contains a time stamp (time instant in hours), a lot id, and a description of the event that occurs. From this data estimate the mean throughput, the mean flowtime, the mean wip-level, and the mean utilization.
3. Verify the results found for the $M/G/1$ system in Section 5.7 by simulation.
4. By simulation determine the mean throughput, mean flowtime, mean utilization and mean wip-level of the manufacturing system from Exercise 4.1.
5. Reconsider the workstation from Exercise 4.3 consisting of infinite buffer B and machine M. The machine can process three types of products. The product type is denoted by a natural (0,1,2). The processing time depends on the product type. The processing time is 1.0, 3.0, and 6.0 minutes for product type 0,1, and 2 respectively, see Figure 5.5.

   You can use the specification from Exercise 4.3(a).

   (a) Adapt the specification of exit `E` and determine the mean throughput $\overline{\delta}$ of the workstation.

   (b) Adapt the specification of exit `E` and determine the mean flowtime per product.

   (c) Adapt the specification of exit `E` and determine the mean flowtime per product, specified per product type.

875	LOT0174	ENTERED SYSTEM
875	LOT0174	ENTERED BUFFER
876.049	LOT0173	LEAVES MACHINE
876.049	LOT0173	LEAVES SYSTEM
876.049	LOT0174	LEAVES BUFFER
876.049	LOT0174	ENTERED MACHINE
880	LOT0175	ENTERED SYSTEM
880	LOT0175	ENTERED BUFFER
881.396	LOT0174	LEAVES MACHINE
881.396	LOT0174	LEAVES SYSTEM
881.396	LOT0175	LEAVES BUFFER
881.396	LOT0175	ENTERED MACHINE
885	LOT0176	ENTERED SYSTEM
885	LOT0176	ENTERED BUFFER
887.024	LOT0175	LEAVES MACHINE
887.024	LOT0175	LEAVES SYSTEM
887.024	LOT0176	LEAVES BUFFER
887.024	LOT0176	ENTERED MACHINE
890	LOT0177	ENTERED SYSTEM
890	LOT0177	ENTERED BUFFER
892.638	LOT0176	LEAVES MACHINE
892.638	LOT0176	LEAVES SYSTEM
892.638	LOT0177	LEAVES BUFFER
892.638	LOT0177	ENTERED MACHINE
895	LOT0178	ENTERED SYSTEM
895	LOT0178	ENTERED BUFFER
897.496	LOT0177	LEAVES MACHINE
897.496	LOT0177	LEAVES SYSTEM
897.496	LOT0178	LEAVES BUFFER
897.496	LOT0178	ENTERED MACHINE
900	LOT0179	ENTERED SYSTEM
900	LOT0179	ENTERED BUFFER
901.62	LOT0178	LEAVES MACHINE
901.62	LOT0178	LEAVES SYSTEM
901.62	LOT0179	LEAVES BUFFER
901.62	LOT0179	ENTERED MACHINE
905	LOT0180	ENTERED SYSTEM
905	LOT0180	ENTERED BUFFER
906.938	LOT0179	LEAVES MACHINE
906.938	LOT0179	LEAVES SYSTEM
906.938	LOT0180	LEAVES BUFFER
906.938	LOT0180	ENTERED MACHINE
910	LOT0181	ENTERED SYSTEM
910	LOT0181	ENTERED BUFFER
911.051	LOT0180	LEAVES MACHINE
911.051	LOT0180	LEAVES SYSTEM
911.051	LOT0181	LEAVES BUFFER
911.051	LOT0181	ENTERED MACHINE
915	LOT0182	ENTERED SYSTEM
915	LOT0182	ENTERED BUFFER
915.345	LOT0181	LEAVES MACHINE
915.345	LOT0181	LEAVES SYSTEM
915.345	LOT0182	LEAVES BUFFER
915.345	LOT0182	ENTERED MACHINE
919.669	LOT0182	LEAVES MACHINE
919.669	LOT0182	LEAVES SYSTEM
920	LOT0183	ENTERED SYSTEM
920	LOT0183	ENTERED BUFFER
920	LOT0183	LEAVES BUFFER
920	LOT0183	ENTERED MACHINE
924.686	LOT0183	LEAVES MACHINE
924.686	LOT0183	LEAVES SYSTEM
925	LOT0184	ENTERED SYSTEM
925	LOT0184	ENTERED BUFFER
925	LOT0184	LEAVES BUFFER
925	LOT0184	ENTERED MACHINE
930	LOT0185	ENTERED SYSTEM
930	LOT0185	ENTERED BUFFER
930.777	LOT0184	LEAVES MACHINE
930.777	LOT0184	LEAVES SYSTEM

Figure 5.4: Logdata for single machine workstation

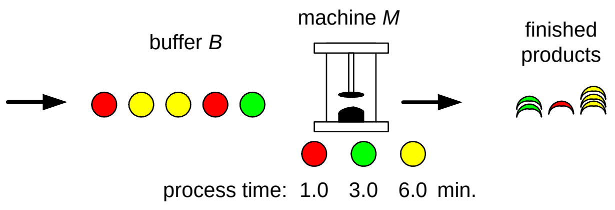


Figure 5.5: Work station process three product types

(d) Adapt the specification of buffer B and determine the mean number of products in the buffer.

6. We reconsider the workstation of the previous exercise. The previous exercise showed that the mean number of products in the infinite buffer B is 0.59. We replace the infinite buffer by a buffer with finite capacity $N = 1$.

    (a) Adapt the specification of the previous exercise and determine the mean throughput for this system.

    (b) How can the throughput be less than $1/4.0 = 0.25$ products/minute, whilst the generator is set to generate a lot every 4.0 minutes?

    (c) Increase buffer capacity N to 2 products. Again, determine the mean throughput.

Chapter 6

Machines in series

Chapter 4 discusses a single machine with a buffer. In this chapter we take a look at manufacturing lines containing multiple machines in series.

6.1 Deterministic flowline

The first flowline we consider consists of three workstations. Each workstation consists of an infinite buffer and one machine, see Figure 6.1 and 6.2.

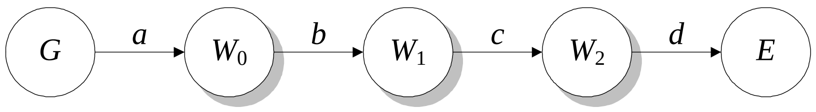


Figure 6.1: Three workstation flowline

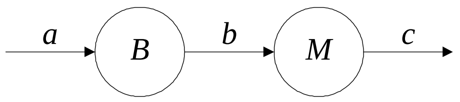


Figure 6.2: Workstation

At first, we choose deterministic inter-arrival and processing times. We take inter-arrival time $t_\mathrm{a} = 4.0$ hours. All machines have the same process time $t_0 = 3.0$ hours.

For this simple deterministic system we are not confined to simulation. The analytical approaches from Chapter 2 are perfectly suitable to analyze this flowline. Figure 6.3 shows a lot-time and resource-time diagram for a release rate of $\lambda = 1/t_a = 0.25$ lots/hour.

We see that the mean flowtime is 9.0 hours, the mean throughput is 0.25 lots/hour and the mean utilisation is 0.75. In general, for `M` deterministic identical workstations in series,

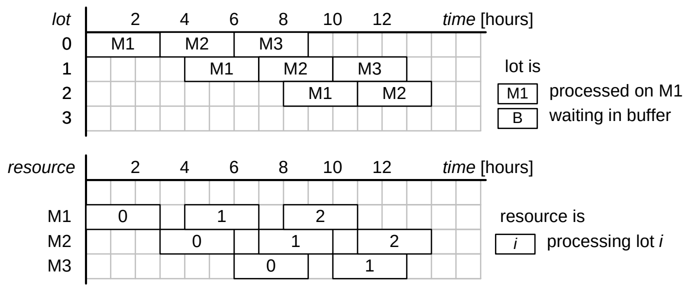


Figure 6.3: Analytically constructed (a) lot-time diagram and (b) resource-time diagram

consisting each of an infinite buffer and a single machine, we can deduce:

$$\begin{cases} \delta = \lambda \qquad u = \lambda t_0 \qquad \varphi = \sum_{i=1}^{m} t_{0,i} = m t_0 & \text{if } \lambda \le \lambda_{\max} \\ \delta = \lambda_{\max} \qquad u = 1 \qquad \varphi \to \infty & \text{if } \lambda > \lambda_{\max} \end{cases} \tag{6.1}$$

Or similarly:

$$\begin{cases} \delta = \dfrac{1}{t_\mathrm{a}} \qquad u = \dfrac{t_0}{t_\mathrm{a}} \qquad \varphi = \sum_{i=1}^{m} t_{0,i} & \text{if } t_\mathrm{a} \ge t_0 \\ \delta = \dfrac{1}{t_0} \qquad u = 1 \qquad \varphi \to \infty & \text{if } t_\mathrm{a} < t_0 \end{cases} \tag{6.2}$$

Now let us reproduce these results by simulation. We specify generator `G`, machine `M`, infinite buffer `B`, and exit `E` as follows.

```
type lot = real;

proc G(chan! lot a):
   while true:
      a!time; delay 4.0
   end
end

proc B(chan? lot a; chan! lot b):
   list lot xs; lot x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end

proc M(chan? lot a; chan! lot b):
   lot x;
   while true:
      a?x;
      delay 3.0;
      b!x
   end
end

proc E(chan? lot a):
   real mphi; lot x;
   for i in range(1, 100000):
      a?x;
      mphi = (i - 1) / i * mphi + (time - x) / i;
      write("%10.2f\tE\treceived lot %5d\tmean flow time =\t%10.3f\n", time, i, mphi)
    end
end
```

Chi-6.1

We specify process `W` and model `GWWWE` as follows.

```
proc W(chan? lot  a; chan! lot c):
   chan lot b;
   run B(a, b), M(b, c)
end

model GWWWE():                                                    Chi-6.2
   chan lot a, b, c, d;
   run G(a),
       W(a, b), W(b, c), W(c, d),
       E(d)
end
```

Simulation yields the following output:

```
 9      E       Received lot    0
13      E       Received lot    1
17      E       Received lot    2
21      E       Received lot    3
25      E       Received lot    4
29      E       Received lot    5
```

By simulation, we establish that the mean throughput δ equals 0.25 lots/hour, the mean flowtime φ equals 9 hours, and the mean utilisation u is 0.75. This corresponds to the analytical results.

Figure 6.4 shows how the wip-level changes over time. After 8 hours, the wip-level shows a repetitive pattern. The first 8 hours show a start-up effect, since the line started empty. From Figure 6.4 we determine that, after the start-up behaviour ends, mean wip-level $w = \frac{1}{4} \cdot 3 + \frac{3}{4} \cdot 2 = 2.25$ lots. This satisfies Little's law: $w = \delta \cdot \varphi = 0.25 \cdot 9 = 2.25$.

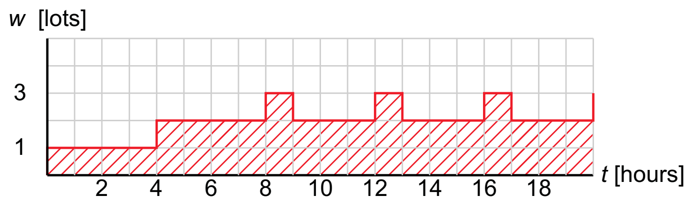


Figure 6.4: Work-in-process level over time

By simulation we investigate how mean flowtime φ, and mean wip-level w respond to different inter-arrival times. The results are shown in Table 6.1.

Note that all results are in accordance with Little's Law (Try!). For $t_a \geq 3.0$ hours, the utilisation equals t_0/t_a and the throughput equals $1/t_a$. For $t_a < 3.0$ hours, the line capacity is insufficient to process all lots: the utilisation equals 1 and the throughput equals $1/t_0$. The number of lots in the system and thus the mean flowtime go to infinity. This corresponds to the analytical results.

t_a [hrs]	u [$-$]	δ [lots/hour]	φ [hour]	w [lots]
12.0	0.25	$1/12 = 0.08$	9	0.75
9.0	0.33	$1/9 = 0.11$	9	1.0
6.0	0.5	$1/6 = 0.17$	9	1.5
5.0	0.6	$1/5 = 0.2$	9	1.8
4.0	0.75	$1/4 = 0.25$	9	2.25
3.5	0.86	$1/3.5 = 0.29$	9	2.57
3.0	1.0	$1/3 = 0.33$	9	3.0
2.0	1.0	$1/3 = 0.33$	∞	∞

Table 6.1: u, δ, φ, and w for $t_0 = 3.0$ hours and different t_a, determined by simulation

6.2 Unbalanced flowline

We now explore how the behaviour of the three workstation manufacturing line from Section 6.1 is affected if not all machines have the same (but all still deterministic) process time. As an experiment, we increase the process time for the second machine from 3.0 hour to 3.5 hour.

With Chapter 2 we easily deduce that the flowtime is 9.5 hours for release rates under the maximum release rate. The maximum release rate is 1/3.5 lots/hour, or similarly, the minimum inter-arrival time is 3.5 hours. For inter-arrival times less than 3.5 hours, the flowtime grows to infinity.

We now use simulation to reproduce these results. To simulate this unbalanced line, we specify machine `M` by the following parametric process.

```
proc M(chan? lot a; chan! lot b; real t0):
   lot x;
   while true:
      a?x;
      delay t0;                                    Chi-6.3
      b!x
   end
end
```

Process `W` and model `GWWWE` become:

```
proc W(chan? lot  a; chan! lot c; real t0):
   chan lot b;
   run B(a, b), M(b, c, t0)
end

model GWWWE():                                     Chi-6.4
   chan lot a, b, c, d;
   run G(a),
       W(a, b, 3.0), W(b, c, 3.5), W(c, d, 3.0),
       E(d)
end
```

Simulation shows that for inter-arrival times greater than 3.5, the flowtime is indeed 9.5 hours.

6.3 Machines in series without buffers

In Sections 4.2 and 4.3 we discussed the influence of buffers in a workstation for a machine with varying process times. In this section we explore the influence of buffers in a flowline. We start with the flowline from Section 6.1 and replace the three deterministic machines, by three machines that each have a process time that is either 1.0 or 5.0 hours. We keep the inter-arrival time fixed at 4.0 hours. We remove the infinite buffers. We specify machine `M` as follows.

```
proc M(chan? lot a; chan! lot b; real t0, s0):
   dist bool d = bernoulli(0.5);
   lot x;
   while true:
      a?x;
      if sample d:
         delay t0 - s0
      else:
         delay t0 + s0
      end;
      b!x
   end
end
```

Chi-6.5

Here `t0` is the mean process time on the machine and `s0` is the standard deviation on the process time. Applying statistics to determine the standard deviation of this distribution, shows that `s0` is indeed the standard deviation. (Try!)

Model `GMMME` becomes:

```
model GMMME():
   chan lot a, b, c, d;
   run G(a),
       M(a, b, 3.0, 2.0),
       M(b, c, 3.0, 2.0),
       M(c, d, 3.0, 2.0),
       E(d)
end
```

Chi-6.6

Simulation shows that the mean throughput is no longer 0.25 lots/hour, but drops down to 0.21 lots/hour. The mean flowtime is no longer 9.0 hours, but increases to 10.96 hour. Just as for the single machine, see Section 4.2, the three machine flowline suffers from starving and blocking. Figure 6.5 shows a resource-time-diagram for a single simulation.

At `time = 0.0`, lot 0 is released in the system. It is processed by machine 1, 2, and 3 in 1.0 hour. Lot 1 is released at `time = 4.0`. It is processed on machine 1 for 5.0 hours. At `time = 8.0` the generator tries to release the next lot, but it is blocked. Lot 2 is released at `time = 9.0`. Lot 2 takes 1.0 hour processing on machine 1, but when it is finished machine 2 is still processing lot 1. Machine 1 is blocked. And even though, at `time = 13.0`, machine 1 is not processing a lot, it is still blocked and can not start processing the next lot. In the mean time, machine 3 has not processed any lot from `time = 3.0` till `time = 14.0`: machine 3 is starving.

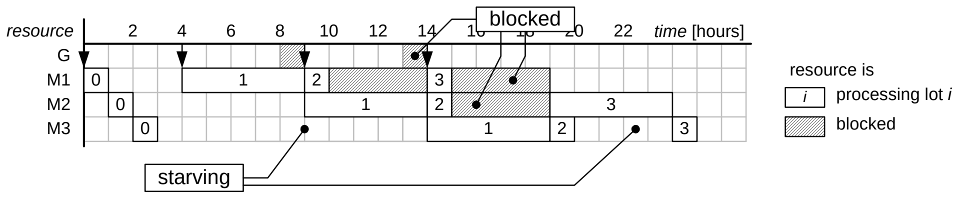


Figure 6.5: Resource-time-diagram for machines in series without buffers

Introducing finite buffers between the machines reduces the effect of starving and blocking. For infinite buffers the mean throughput is 0.25 lots/hour and the mean flowtime is 14.3 lots/hour. In the exercises, the reader is invited to investigate how the mean throughput and mean flowtime depend on the buffer size.

6.4 Simulation vs Approximation (2): flowline

Example 6.1 Validity of Kuehn's linking equation
To closer investigate the validity of Kuehn's linking equation, we consider a single workstation, consisting of an infinite buffer and one machine. The process time of the machine is distributed according to a gamma distribution with $t_0 = 3.0$ and $c_0^2 = 0.5$. We consider two different inter-arrival time distributions. First, we let the inter-arrival time be distributed according to an exponential distribution (an $M/G/1$ system) with $t_\mathrm{a} = \{4.0, 6.0, 12.0\}$ and $c_\mathrm{a}^2 = 1$. Then, we let the inter-arrival time be distributed according to a gamma distribution (a $G/G/1$ system) with $t_\mathrm{a} = \{4.0, 6.0, 12.0\}$ and $c_\mathrm{a}^2 = 2.0$.

We let exit E determine the mean inter-arrival time at the exit (thus the mean inter-departure time for the machine) and the squared standard deviation. In Section 5.5 we presented an approach to iteratively determine the variance.

```
proc E (chan? lot a):
   real tp, ta, tai, s2a;
   lot x;
   tp = time;
   for i in range(1, 100000):
      a?x; tai = time - tp; tp = time;
      if i <= 1:
         s2a = 0.0
      else:
         s2a = (i - 2) / (i - 1) * s2a + (tai - ta) * (tai - ta) / i
      end;
      ta = (i - 1) / i * ta + tai / i;
      write("%10.2f\tE\tmean flow time%10.4f\tvar flow time%10.4f\n",
            time, ta, s2a / ta / ta)
   end
end
```

Chi-6.7

Every time a lot arrives at the exit process, the inter-arrival time $t_{\mathrm{a},i}$ is determined, by subtracting the time instant t_p that the previous lot arrived at the exit from the current time. Subsequently, squared standard deviation s_a^2 on the inter-arrival time and mean inter-arrival time t_a are calculated. The squared standard deviation can only be calculated for $i > 1$ and is calculated before calculating the mean, see Equation 5.8.

In Table 6.2 the analytically determined c_d^2 is compared with the c_d^2 determined by simulation.

	exponentially distr. inter-arrival time		gamma distr. inter-arrival time	
t_a	$c_\mathrm{d}^2(analytical)$	$c_\mathrm{d}^2(simulation)$	$c_\mathrm{d}^2(analytical)$	$c_\mathrm{d}^2(simulation)$
4.0	0.719	0.719	1.156	1.050
6.0	0.875	0.875	1.625	1.497
12.0	0.968	0.968	1.906	1.828

Table 6.2: Comparing c_d^2 from analytical results with simulation results

We see that Kuehn's linking equation is accurate for the $M/G/1$ system and less accurate for the $G/G/1$ system. ⊠

Example 6.2 Flowline from Example 3.3 revisited
In Chapter 3 we discussed how to analytically determine the flowtime for a flowline, using Pollaczek-Khintchine's queueing relation and Kuehn's linking equation. In Example 3.3 we calculated the mean flowtime for a three-workstation flowline, with exponentially distributed inter-arrival times with mean $t_\mathrm{a} = 4.0$ hours and a process time distribution with $t_0 = 3.0$ hours and $c_0^2 = 0.5$. The mean flowtime was calculated at 26.2 hours. Let us try and reproduce these results by simulation.

Lots arrive at the flowline with an inter-arrival time that is exponentially distributed, with mean inter-arrival time t_a. A lot is represented by an identification number and a time stamp.

```
type lot = real;

proc G(chan! lot a; real ta):
   dist real d = exponential(ta);
   while true:
      a!time; delay sample d
   end
end
```

Chi-6.8

We specify the machines with mean process time t_0 and a squared coefficient of variation c_0^2. We choose the gamma distribution to represent the process times[1]. In the Chi Reference Manual [HR12] we find that the gamma distribution $\Gamma(a, b)$ has parameters a and `B`, with mean $\mu = ab$ and variance $\sigma^2 = ab^2$. If we have mean process time t_0 and squared coefficient of variation c_0^2 we obtain the following parameters (Try!):

$$\Gamma(a, b) \qquad \text{with } a = \frac{1}{c_0^2} \qquad \text{and } b = c_0^2 \cdot t_0$$

We specify machine `M` as follows:

[1]This is an arbitrary choice. Chapter 9 goes into more detail on choosing an appropriate distribution for process times.

```
proc M(chan? lot a; chan! lot b; real t0, c20):
   dist real d = gamma(1 / c20, c20 * t0);
   lot x;
   while true:                                              Chi-6.9
      a?x; delay sample d; b!x
   end
end
```

We let the buffer process record the mean flowtime in the previous workstation and the coefficient of variation on the inter-arrival time of lots arriving in that buffer. In Section 5.5 we presented an iterative way to determine the variance. We specify buffer `B` as follows.

```
proc B(chan? lot a; chan! lot b):
   int i; real ta, tp, tai, s2a, phi;
   list lot xs; lot x;
   while true:
      select
         a?x:
            i = i + 1; tai = time - tp; tp = time;
            if i <= 1:
               s2a = 0.0
            else:
               s2a = (i - 2) / (i - 1) * s2a
                     + (tai - ta) * (ta - ta) / i           Chi-6.10
            end;
            ta  = (i - 1) / i * ta  + tai / i;
            phi = (i - 1) / i * phi + (time - x) / i;
            xs = xs + [time]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end
```

Every time a lot arrives at the buffer, the variance on the inter-arrival time and the mean inter-arrival time are recalculated. Subsequently, the mean flowtime φ for the previous workstation is determined. The flowtime φ of the lot in the previous workstation is equal to the current time minus the time instant of entering the previous buffer ($x.1$). After these calculations, the time stamp of the lot ($x.1$) is updated to the current time, so that the next buffer can determine the flowtime of its preceding workstation.

To determine the mean flowtime in the last workstation, we use the following exit process.

```
proc E (chan? lot a):
   lot x; real mphi;
   for i in range(1, 10000):
      a?x;                                                  Chi-6.11
      mphi = (i - 1) / mphi + (time - x) / i
   end
end
```

Finally we have the following process and model:

```
proc W(chan? lot a; chan! lot c):
   chan lot b;
   run B(a, b), M(b , c, 3.0, 5.0)
end

model GWWWE():                                                    Chi-6.12
   chan lot a, b, c, d;
   run G(a, 4.0),
       W(a, b), W(b, c), W(c, d),
       E(d)
end
```

After adding output statements to the buffer and the exit specification, and running the simulation for a sufficient long time, we obtain the following simulation output:

```
39771.8        B0  phi   0          ca2   1.10149
39787.6        B1  phi   9.24992    ca2   0.74992
39792.3        B2  phi   8.87956    ca2   0.68941
39829          E   phi   8.25412
...
8.45643e+07    B0  phi   0          ca2   1.00149
8.45644e+07    B1  phi   9.74992    ca2   0.71992
8.45644e+07    B2  phi   8.37956    ca2   0.65941
8.45644e+07    E   phi   8.05412
...
```

Modelling and simulation of Example 3.3 yields a mean total flowtime $\varphi = 26.2$ hours. At first sight, the results seem to exactly correspond with the analytical results, but let us closer compare the analytical and simulation results. Table 6.3 shows the flowtime φ and the coefficient of variation on the inter-arrival time c_a for each workstation, determined analytically and by simulation.

	Analytical approximation		*Simulation*	
W_0	$\varphi_0 = 9.75$ hrs	$c^2_{a,0} = 1$	$\varphi_0 = 9.75$ hrs	$c^2_{a,0} = 1.00$
W_1	$\varphi_1 = 8.49$ hrs	$c^2_{a,1} = 0.719$	$\varphi_1 = 8.38$ hrs	$c^2_{a,1} = 0.719$
W_2	$\varphi_2 = 7.92$ hrs	$c^2_{a,2} = 0.596$	$\varphi_2 = 8.05$ hrs	$c^2_{a,2} = 0.659$
total	$\varphi = 26.2$ hrs		$\varphi = 26.2$ hrs	

Table 6.3: Analytical approximation results versus simulation results

In Table 6.3, we see that for workstation W_0 the analytical approximation of the mean flowtime is equal to the mean flow time obtained by simulation. This is correct, since we are dealing with an $M/G/1$ system. Moreover, we see that the analytical approximation of $c^2_{\mathrm{d},0} = c^2_{\mathrm{a},1}$ is also equal to the simulation results. We saw in the previous example that the for an $M/G/1$ system, the linking equation is fairly accurate, however, it is only exact for an $M/M/1$ system[2]. The subsequent analytical approximations deviate from the simulation results. For workstation W_1 and subsequent workstations the inter-arrival times are no longer exponentially distributed. As a consequence, the analytical approximations no longer yield the exact flowtime and the coefficient of variation on the inter-arrival times deviates stronger. In this particular case, the deviations cancel each other out, and the total flowtime determined analytically corresponds (up to three significant numbers) to the one determined by simulation. In general, this is not the case! ⊠

Keyword overview

Modelling manufacturing systems	Analysis of manufacturing systems
• flowline • parametric machine model • gamma-distributed process time	• starving and blocking • throughput and flowtime for a flowline • influence of buffers on throughput and flowtime • analytical vs simulation (2)

6.5 Exercises

1. Consider a flowline with three workstations. Each workstation consists of a finite buffer with N places and a machine with mean process time $t_0 = 3$ hours and standard deviation $\sigma_0 = 2$ hours on the process time. Lots arrive at the system with an inter-arrival time of $t_a = 4$ hours. We measure the mean throughput δ and the mean flowtime φ.

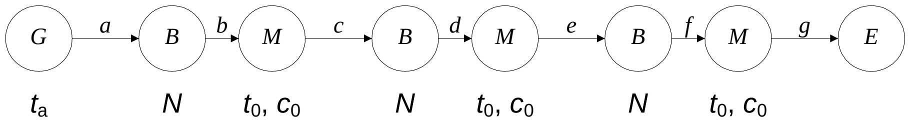


Figure 6.6: Three workstation flowline

Are the following statements true or false? (Try to determine your answer by reason. If desired, use simulation to verify your answer.)

(a) If buffer capacity N increases, the influence of blocking and starving decreases.
(b) If buffer capacity N increases, mean throughput δ increases.
(c) If buffer capacity N is zero, mean flowtime $\varphi = 9$ hours.
(d) If buffer capacity N goes to infinity, mean throughput δ goes to 0.25 lots/hour.
(e) If buffer capacity N goes to infinity, the throughput and flowtime are equal to the throughput and flowtime in case of deterministic process times of 3.0 hours.

2. Reconsider the flowline from Exercise 1. Are the following statements true or false? (Try to determine your answer by reason. If desired, use simulation to verify your answer.)

[2]For an $M/M/1$ system the linking equation always yields the trivial result that $c_a^2 = c_d^2 = 1$.

(a) For $N = \infty$, increasing σ_0 has no effect on mean throughput δ.

(b) For $N = 0$, increasing σ_0 has no influence on mean throughput δ.

(c) For $N = \infty$, increasing σ_0 has no effect on mean flowtime φ.

(d) For $N = 0$, increasing σ_0 has no influence on mean flowtime φ.

3. Reconsider the flowline from Exercise 1. We now also measure the mean wip-level w. Are the following statements true or false? (Try to determine your answer by reason. If desired, use simulation to verify your answer.)

(a) If $N = \infty$, the minimum flowtime that occurs for a single lot is 9 hours.

(b) If $N = 0$, the maximum flowtime that occurs for a single lot is 9 hours.

(c) If $\sigma_0 = 0$, mean flowtime $\varphi = 9$ for any buffer capacity N.

(d) For $t_a > t_0$ and $N = \infty$, mean flowtime φ and mean wip-level w go to infinity.

(e) For $t_0 > t_a$ and $N = 0$, mean wip-level $w = 3$.

(f) For $t_a > t_0$ and $N = \infty$, mean throughput $\delta = 1/t_a$.

4. We have a manufacturing system that consists of two machines M. Both machines are identical except for the process times: the first one has a mean process time of 0.2 hour with a variation of plus or minus 0.1 hour, whereas the second one has a mean process time of 0.4 hour with a variation of plus or minus 0.2 hour. The manufacturing system is modelled by the following specification:

```
type lot = real;

proc G(chan! lot a):
   while true:
      a!time
   end
end

proc M(chan? lot a; chan! lot b; real m, s):
   dist int d = uniform(0, 2);
   lot x;
   list(2) real pt = [m - s, m + s];
   while true:
      a?x; delay pt[sample d]; b!x
   end
end;                                                  Chi-6.13

proc E(chan? lot a):
   lot x;
   for i in range (1, 10000):
      a?x;
      write("%10.2f\tE\tMean output\t%10.5f\n", time, i / time)
   end
end

model S():
   chan lot a, b, c;
   run G(a),
   M(a, b, 0.2, 0.1), M(b, c, 0.4, 0.2),
   E(c)
end
```

Notice that machine M is specified as a process with parameters `M` and s, denoting the mean and standard deviation on the process times. In this way the mean process times and standard variations can easily be changed.

(a) Draw a graphical representation of the model and simulate the specification.

(b) Determine the mean number of lots processed per hour and explain the results.

(c) Insert a finite buffer with N places between both machines M. Determine the output in [lots/hour] for different buffer sizes (0,1,2,...). Plot the output in [lots/hour] against the buffer size N. Explain the results.

(d) What buffer size would you advice in this case?

5. Figure 6.7 shows a part of a printed circuit board (PCB) assembly line. This part contains two component placement robots (CPRs) that place electrical components on printed circuit boards (PCBs). A PCB is positioned under the placement robot with a conveyor belt.

   Placing one component on a PCB takes 0.8 to 1.1 seconds. The placement time is uniformly distributed. The number of components that have to be placed by a robot varies from 10 to 14 components (in Chi: uniform(10,15)) for each robot and is also

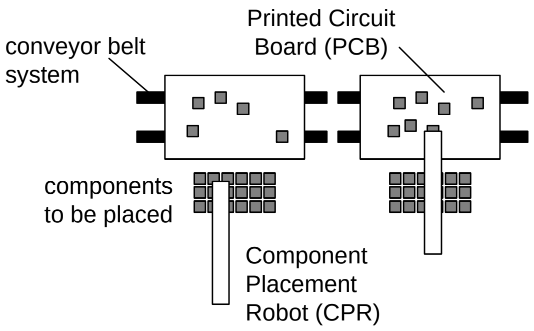


Figure 6.7: PCB assembly line

uniformly distributed. We neglect the conveyor transport time. In the first configuration there are no buffer places before and after the placement robots, see Figure 6.7. We use a Chi-specification to analyze the performance of the PCB-assembly line. A part of the specification is shown below.

```
type pcb = real;

proc G(chan! pcb a):
   while true:
      a!time; delay ...
   end
end

proc CPR(chan? pcb a; chan! pcb b):
   dist real pt = uniform(0.8,  1.1);
   dist int  nc = uniform( 10, 15);
   pcb x;
   int i, n;
   while true:
      a?x; n = ...; i = 0;
      while ...:
         delay ...;
         ...                                           Chi-6.14
      end;
      b!x
   end
end

proc E(chan? ... a):
   pcb x;
   for i in range(1, 100000):
      a?x;
      write("%10.2f\tE\treceived PCB\t5d\tflow time%10.2f\n", time, i, time - x)
   end
end

model PcbAssembly():
   chan ... a, b, c;
   run G(a), CPR(a, b), CPR(b, c), E(c)
end
```

(a) What is the time unit of this specification?

(b) Estimate the mean process time per PCB by hand.

(c) We assume that a new PCB arrives at the PCB assembly line every 15 seconds. Finish the specification of the PCB assembly line. Determine the mean throughput in PCBs/seconds and the mean flowtime in seconds.

(d) Investigate whether increasing the arrival rate results in a higher throughput. What maximum throughput can be attained? Is the mean flowtime affected by increasing the arrival rate?

6. As an alternative for the PCB assembly line of Exercise 5 we have a configuration that has a buffer place for one PCB between the placement robots, see Figure 6.8.

   (a) By simulation, determine the mean throughput and mean flowtime for the PCB assembly line if the arrival rate is 4 PCBs/minute.

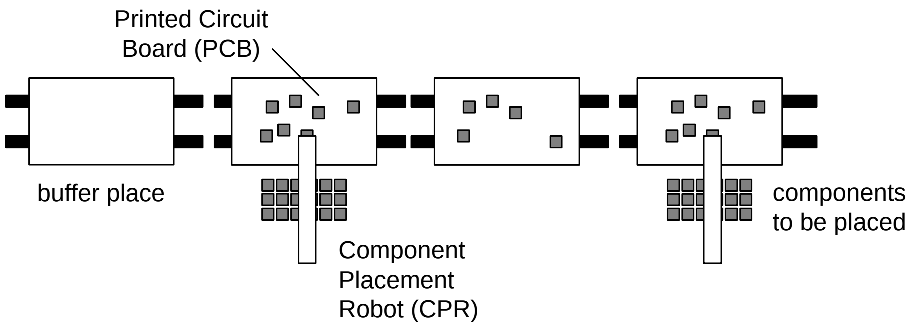


Figure 6.8: PCB assembly line with buffers

(b) What is the maximum attainable throughput if the arrival rate can be freely chosen? In case of maximum throughput, what is the mean flowtime?

6.6 Case

"There is something completely wrong," the plant manager pointed out, "You said you would buy three machines of equal capacity! But the line seems to be completely unbalanced." "But I did buy three machines of equal capacity", the design engineer replied, "I really did." "Why then is the buffer in front of the first machine generally almost three times as full as the buffer of the third machine?", the manager continued." "Well, sir, to be honest, I don't know. The lots arrive in the first buffer at a constant rate of 4 lots/hour. The capacity of each of the machines is much larger: 5 lots/hour. So that seems okay to me." "Okay? , a utilization of 80% you call that okay? Those machines are expensive, I want the full 100%. Why don't you increase the release rate to 5 lots/hour?" "Well, I'm not sure whether that is a good thing to do. I feel there is some trade-off between throughput and flowtime, and currently the flowtime is already near your maximum allowed flowtime of 3 hour." " What trade-off? I want a maximum throughput with a minimum flowtime. And if all lots are waiting in the buffer of the first machine, I do understand that that does no good for the flowtime. I demand an explanation." "Sir, I really don't understand it, but I'll try to find out why the buffers are not equally filled and whether or not a trade-off between throughput and flowtime exists. I hope we can further increase the release rate without surpassing the flowtime of 3 hour." "Report me at the end of this week. I hope you understand we cannot afford additional machines." The design engineer left the office of the manager and walked to the production line. 'I really do not understand', he thought, 'such a simple production line. A line with just three machines and a buffer in front of each machine. The vendor of the machines guaranteed that each of the three machines has a mean effective process time of 0.2 hour per lot. Wait, he also shortly mentioned something about variation coefficients. Ah, yes, I remember. He told me that the first machine has a coefficient of variation of about 1.5, and the second and third machine a variation coefficient of 0.5. But what difference does that make? The first machine is simply much less reliable due to its sensible and highly complex production process. On average the available capacity is still equal for all machines. So why bother and spend money to reduce the variability? And it is not for nothing I have put buffers in front of the machines.'

1. We would like to make a Chi-model of the three machine production line to evaluate the hypothesis of the design engineer that a trade-off is present between throughput and flowtime. Part of the specification of the production line is given below:

```
type lot = real;

proc G(chan! lot a; real m):
   while true:
      delay m; a!time
   end
end

proc B(chan? lot a; chan! lot b):
   ...
end

proc M(chan? lot a; chan! lot b; real m, s):
   ...
end

proc E(chan? lot a):
   ...
end

model S():
   chan lot a, b, c, d, e, f, g;
   run G(a, 0.25),
       B(a, b), M(b, c, 0.2, 0.3),
       B(c, d), M(d, e, 0.2, 0.1),
       B(e, f), M(f, g, 0.2, 0.1),
       E(g)
end
```

Chi-6.15

   (a) Make a graphical representation of the model.
   (b) Complete the specification of processes `B` and `M`. Note that we deal with an infinite buffer. Use a gamma-distribution for the process time of machine `M`, with `M` the mean and s the standard deviation.
   (c) Complete the specification of process `E` such that the mean throughput is computed, as well as the mean flowtime from generator to exit.
   (d) Run the program and compute the throughput and flowtime. Try to obtain about two significant digits. Run the program for several other release levels as well, and investigate whether there is a trade-off between throughput and flowtime. For which release level (throughput) is the maximum allowed flowtime reached?

2. To evaluate the observation of the plant manager with respect to the buffer contents, we must slightly extend our Chi-model. Besides the flowtime from start to exit, we must compute the average waiting times in the buffers as well. The buffer with the longest waiting time has the largest number of the lots in its queue (note that we can compare the waiting times of the buffers since the throughput of the machines are identical, see Little's Law).

To compute the mean waiting times, the buffers record the lot arrival time and compute the waiting time in the buffer when the lot leaves the buffer.

(a) Change the specification of buffer `B` such that, when a lot has been received, it records the time of arrival of that lot, and when a lot is sent, it computes the waiting time of that lot and the mean waiting time.

(b) Run the program for the original release rate of 4 lots/hour. Let it run some time until the large fluctuations have died out.

(c) What do your observe with respect to the mean waiting times in the three buffers? Do you agree with the observation of the plant manager?

(d) Change the coefficient of variation of the first machine into 0.5, and instead replace the coefficient of variation of the third machine by a value of 1.5. Rerun the program. What happens to the overall flowtime and the waiting times in the buffers?

(e) How is the average number of lots waiting in the queue of a machine affected?

3. The Chi simulation model is a powerful tool to analyze the three machine production line. However, for this simple production line, we can also use analytical approximations to estimate mean throughput and flowtime. These relations enable to better understand the actual physical behavior.

   Consider again the three machine production line where the first machine has a process time coefficient of variation of 1.5, and the other two machines a coefficient of 0.5.

   (a) What is the utilization of each of the machines, given the constant arrival rate of 4 lots/hour?

   (b) Estimate the average waiting time of lots in the buffer of the first machine, and compute the corresponding WIP-level (i.e. average number of lots waiting in the queue of the first machine).

   (c) Estimate the departure coefficient of variation of the first machine.

   (d) Compute the mean waiting times and WIP-levels for the second and third machine as well.

   (e) Compare the estimated waiting times with the waiting times obtained via simulation. Is the approximation reasonable?

   (f) Can you explain why the lots are not equally distributed over the three buffers? What will happen when the process time coefficient of variation of 1.5 applies for the third machine instead of the first machine?

Chapter 7

Machines in parallel

In the previous chapter we considered modelling machines in series. In this chapter we model machines in parallel.

7.1 Deterministic machines in parallel

Consider three identical machines in parallel with a single (infinite) buffer, see Figure 7.1. Each machine has a deterministic process time of 9.0 hours. The generator releases a new lot in the line every 4.0 hours.

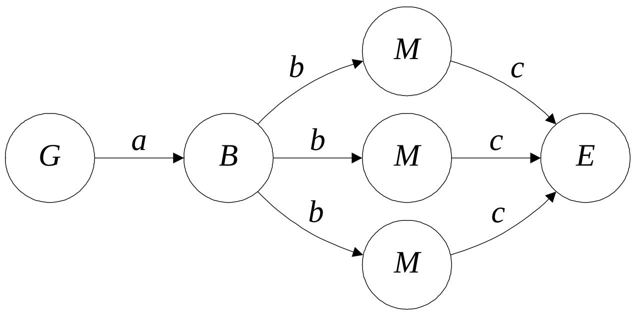


Figure 7.1: Three parallel machines

With the analytical techniques from Chapter 2 we expect a mean throughput of $\delta = 1/4$ lot/hour, an utilisation of 0.75, and a mean flowtime $\varphi = 9.0$ hours. We wish to verify our expectations by simulation.

The specifications of generator G and machine M are more or less identical to the ones presented in Chapter 4.

```
type lot = tuple(real t; int k);

proc G(chan! lot a):
   while true:
      a!(time, -1); delay 4.0
   end
end

proc M(chan? lot a; chan! lot b; int k):
   lot x;
   while true:
      a?x; x.k = k; delay 9.0; b!x
   end
end
```

Chi-7.1

Buffer B receives lots from the generator and can send lots to one of the three parallel machines. We specify buffer B as follows.

```
proc B(chan? lot a; chan! lot b):
   list lot xs; lot x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end
```

Chi-7.2

As long as the buffer contains lots, the buffer tries to send a lot to one of the machines over channel B. The buffer sends the lot to the machine that can receive the lot first. If more than one machine can receive the lot at the same time, the buffer chooses a machine to send the lot to in a non-deterministic fashion.

Exit E can always receive lots from the machines.

```
proc E (chan? lot a):
   lot x;
   for j in range(10):
      a?x;
      write("%10.2f E received lot %10.1f from machine %2d\n", time, x.t, x.k);
   end
end
```

Chi-7.3

Finally, we have model Parallel.

```
model Parallel():
   chan lot a, b; chan lot c;
   run G(a), B(a, b),
       unwind j in range(3):
          M(b, c, j)
       end,
       E(c)
end
```

Chi-7.4

Simulation yields:

```
 9.00 E received lot        0.0 from machine  0
13.00 E received lot        4.0 from machine  1
17.00 E received lot        8.0 from machine  2
21.00 E received lot       12.0 from machine  0
25.00 E received lot       16.0 from machine  1
29.00 E received lot       20.0 from machine  2
33.00 E received lot       24.0 from machine  0
37.00 E received lot       28.0 from machine  1
41.00 E received lot       32.0 from machine  2
45.00 E received lot       36.0 from machine  0
```

We extend exit `E` to measure the mean throughput, mean flowtime, and mean wip level. In correspondence with our expectations, we establish that $\delta = 0.25$ lots/hour, $\varphi = 9.0$ hours, and $w = 2.25$ lots.

A flowline with a single machine with a mean process time of 3.0 hours has the same capacity of 1/3 lots/hour. In case of deterministic arrival time and process time, the flowtime for the single-machine line is 3.0 hours. However, in case of high variability on the arrival time and process times, the parallel-machine line yields a lower variability on the flowtime and even may yield a lower flowtime than the single machine flowline with the same capacity due to shorter waiting times. We investigated this in Chapter 3 Exercise 2.

7.2 Analytical vs Simulation (3): Parallel machines

In Chapter 3 we presented analytical approximations for the flowtime in a workstation consisting of multiple identical machines in parallel. In this section we wish to compare this analytical approximation with simulation results. As a test case we consider a single workstation with three identical parallel machines. Lots arrive with a gamma-distributed inter-arrival times with mean t_a and coefficient of variation c_a. The process times of each machine are distributed according to a gamma distribution with mean t_0 and coefficient of variation c_0. We set $c_\mathrm{a}^2 = 1.0$, $t_0 = 9.0$, and $c_0^2 = 0.25$. We let t_a vary within the set $\{3.5, 4.0, 5.0, 6.0, 9.0, 12.0\}$. In Table 7.1 we compare the mean flowtime determined by simulation with the mean flowtime determined analytically.

We see that, for this test case, the approximation from Chapter 3 is a fair approximation for the flowtime; all deviations are 1% in order of magnitude. For higher degrees of variability, for example $c_\mathrm{a}^2 = c_0^2 = 2.0$, the approximation is less accurate.

		$c_a^2 = 1.0, c_0^2 = 2.0$	
t_a	u	φ(analytical)	φ(simulation)
3.5	0.86	18.90	18.93
4.0	0.75	13.43	13.37
5.0	0.60	10.84	10.75
6.0	0.50	10.06	9.94
9.0	0.33	9.38	9.28
12.0	0.25	9.20	9.12

Table 7.1: Comparing φ from analytical results with simulation results

7.3 Parallel machines for different lot types

Again, consider three machines in parallel with separate buffers. The generator now releases lots of three different types in the line. Lot type 0 can be processed on machine M_0, lot type 1 can be processed on machine M_1, and lot type 2 can be processed on machine M_2. Lot type 0,1,2 are released in a ratio of 4:3:3, with a release rate of 1 lot every 4.0 hours. Each machine takes 9.0 hours processing. Each buffer stores one lot type only. We wish to determine the mean throughput and flowtime per lot type and the utilisation per machine. Figure 7.2 shows a graphical representation of the flowline.

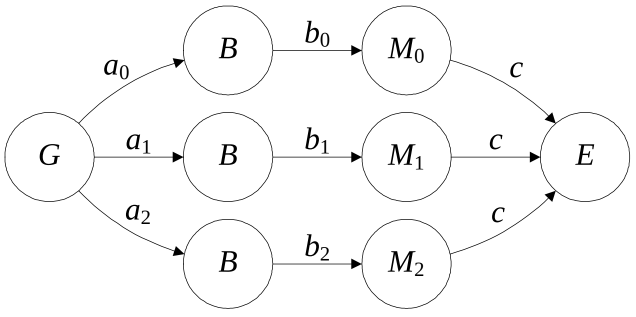


Figure 7.2: Three parallel machines processing different lot types

The type lot is now of type `tuple(real t; int k)` denoting the lot type {0,1,2}, and the time of release. We specify generator G as follows.

```
type lot = tuple(real t; int k);

proc G(list(3) chan! lot a):
   dist int d = uniform(0, 10);
   list(10) int ka = [0, 0, 0, 0, 1, 1, 1, 2, 2, 2];
   int k;                                                Chi-7.5
   while true:
      k = ka[sample d];
      a[k]!(time, k);
      delay 4.0
   end
end
```

Every 4.0 time units, generator G generates a lot. Vector ja and uniform distribution d are used to represent the creation of different lot types. In this way, lot types 0,1, and 2 are generated in the ratio of 4:3:3. Lot type i is sent over channel a_i. Buffers B_i are standard infinite buffers. A specification can be found in Chapter 4.

The specification of machines M_i and exit `E` are identical to the ones presented in Section 7.1. We specify model *Multtype* as follows.

```
model Parallel():
   list(3) chan lot a, b;
   chan lot c;
   run G(a),
       unwind j in range(3):                             Chi-7.6
          B(a[j], b[j]), M(b[j], c)
       end,
   E(c)
end
```

By simulation we determine the mean throughput per lot type and the mean flowtime per lot type. The results are shown in Table 7.2.

lot type	δ	φ	w
0	0.1	31.1	3.11
1	0.075	14.3	1.08
2	0.075	14.3	1.08
all	0.25	21.02	5.25

Table 7.2: Performance for three-type flowline

Note that, though we still have deterministic process times, the mean flowtime of a lot is no longer 9.0 hours. This is because the lot type is stochastic. Chances are $9/100 + 9/100 + 16/100 = 0.34$ that two lots of the same type are generated consecutive. Generating two lots of the same type in a row, causes an increase in flowtime for that lot. The mean utilisation for machine M_0, M_1, and M_2 is 0.9, 0.68, and 0.68.

Alternative model

In the specification of the multi-type flowline we used separate buffers for each lot type. A different modelling approach is to use a single buffer process that stores the lot types in separate lists. The graphical representation of the model is depicted in Figure 7.3.

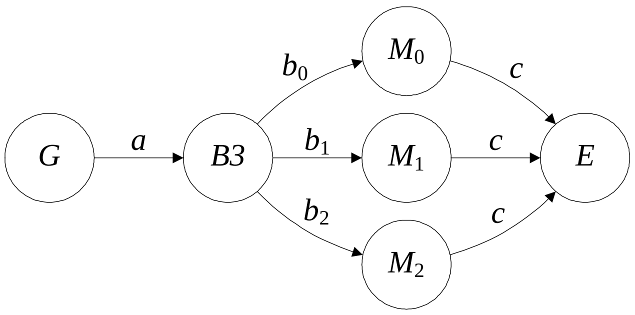


Figure 7.3: Three parallel machines processing different lot types, alternative model

The specification of generator G becomes.

```
proc G(chan! lot a):
   dist int d = uniform(0, 10);
   list(10) int ka = [0, 0, 0, 0, 1, 1, 1, 2, 2, 2];
   int k;
   while true:
      k = ka[sample d];
      a!(time, k);
      delay 4.0
   end
end
```
Chi-7.7

We specify multi-type buffer $B3$ as follows.

```
proc B3(chan? lot a; list(3) chan! lot b):
   list(3) list lot xss; lot x;
   while true:
      select
         a?x:
            xss[x.k] = xss[x.k] + [x]
      alt
         unwind j in range(3):
            not empty(xss[j]), b[j]!xss[j][0]:
               xss[j] = xss[j][1:]
         end
      end
   end
end
```
Chi-7.8

Buffer `B` uses vector xs of lists. Each list in the vector stores a different lot type. The buffer can always receive a lot over channel a. The lot is added to the list of the corresponding type. If there are lots of lot type 0 ($\text{len}(xs.0) > 0$), buffer `B` tries to send the first element of the list over channel b_0 to machine M_0. If multiple lot types are available, `B` sends a lot to the machine that can receive a lot first. If multiple lot types are available, and multiple machines are willing to receive a lot at the same time, buffer B chooses in a non-deterministic fashion.

The specifications of machine M and exit E are identical to the ones presented earlier. The model specification becomes:

```
model Parallel():
   chan lot a;
   list(3) chan lot b;
   chan lot c;
   run G(a), B3(a, b),
       unwind j in range(3):
          M(b[j], c)
       end,
       E(c)
end
```

Chi-7.9

By simulation we obtain the same results as for Specification Chi-7.5.

Keyword overview

Modelling manufacturing systems	Analysis of manufacturing systems
• identical parallel machines • conveyor model for parallel machines • parallel machines for different lot types	• analytical vs simulation (3)

7.4 Exercises

1. Reconsider the flowline from Chapter 3 Exercise 1. Determine the mean flowtime for this flowline by simulation. Compare simulation results with the analytical results.
2. Consider a multi-product flowline with three workstations. Each workstation has three machines that can perform different operations. Figure 7.4 shows a graphical representation of the multi-product flowline.

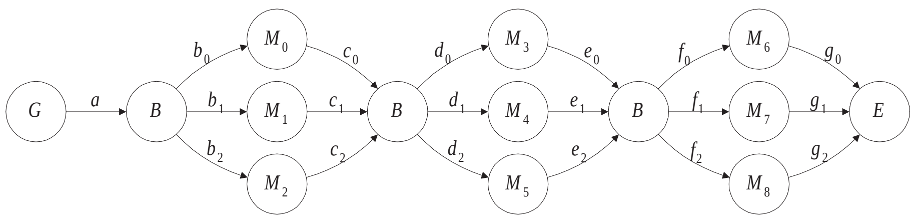


Figure 7.4: Multi-product flowline

The flowline is used to process six different lot types. Each lot type requires processing on different machines. However, each lot type only requires processing on one machine per workstation. Table 7.3 shows the required process steps per lot type. Moreover, the table shows what percentage of the lots is of that specific type.

lot type	percentage of assortment	routing
0	10%	M_0, M_3, M_6
1	10%	M_0, M_3, M_7
2	20%	M_0, M_4, M_6
3	20%	M_0, M_4, M_7
4	20%	M_1, M_5, M_8
5	20%	M_2, M_5, M_8

Table 7.3: Product assortment and required processes

We release new lots at a rate of 1.0 lot/hour.

(a) By hand, estimate the mean number of lots that arrive at each machine per hour.

(b) By hand, determine the required process time per machine, to attain a mean utilisation of 0.9.

(c) By hand, determine the minimal flowtime per lot type.

(d) Build a model of the multi-product flowline. Lots are of type $(\mathrm{nat}, \mathrm{nat}, \mathrm{real})$, representing the identification number, the lot type, and the time of release in the line. Assume that the machines have a fixed process time. You can use fragments of the specifications discussed in Section 7.3.

(e) By simulation, determine the mean flowtime per lot type. Compare these flowtimes with the minimal flowtime per lot type. Explain the increase in flowtime, while all machines have deterministic process times.

Chapter 8

Other configurations

In the previous chapter we considered the elementary configurations of buffers and machines: machines in series and machines in parallel. In practice, many other configurations of buffers and machines exist, see for example [RT98, CAJ01, HS01]. Figure 8.1 shows a number of possible configurations of machines. Table 8.1 provides a description for each configuration.

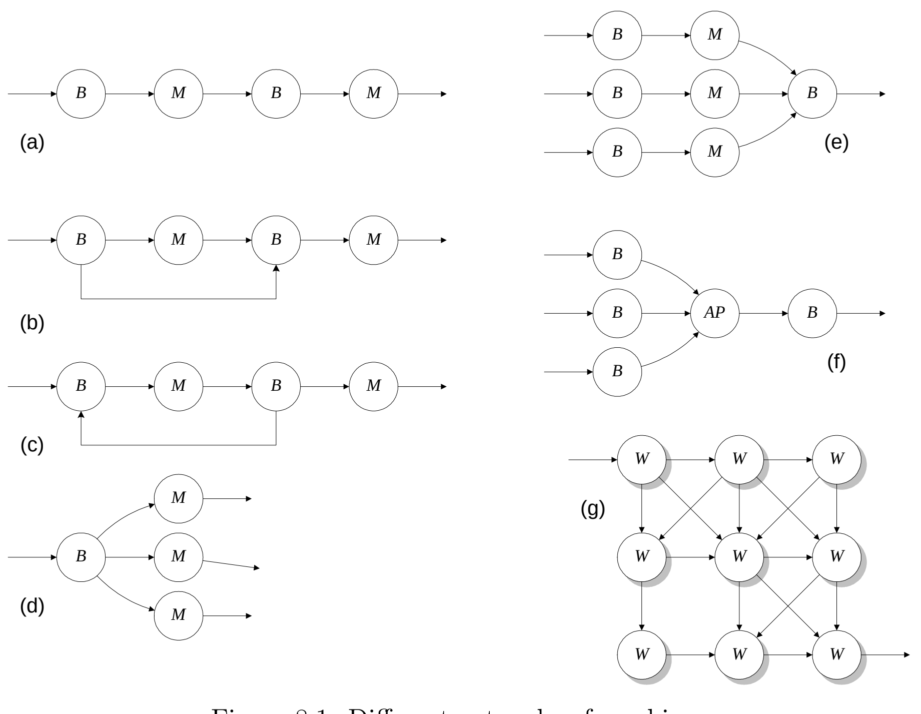


Figure 8.1: Different networks of machines

Depending on the desired properties of the production process (e.g. production volume, flexibility in production volume, flexibility in product type) a different configuration is desired. In theory, an unlimited number of possible configurations exist. In the previous

Configuration	Description
(a) Flowline	Machines and buffers are placed in series.
(b) Flowline with bypassing	Some lots skip an operation. One may think of an automotive assembly line, where some autos skip the installation of an airbag.
(c) Flowline with backtracking	(Some) lots revisit a previous operation. One may think of lots that need rework, of lots that repeatingly require an operation (e.g. painting multiple layers with intermediate drying), or of a re-entrant flowline for the production of IC's.
(d) Parallel machines	From a single buffer lots can be processed on each of the parallel machines. Parallel machines increase the capacity.
(e) Converging flowlines	Parallel flowlines converge into a single buffer.
(f) Single machine, multiple buffers	Machine processes lots from multiple buffers. One may also think of an assembly process, where the machine assembles a number of parts.
(g) Jobshop	We have several workstations, consisting of one or more machines and buffers. Each lot can have a different routing.

Table 8.1: Different configurations of machines and buffers

chapters we discussed configurations (a), (d), and (e). In the sequel, we discuss the remaining four configurations. We present a way to specify these configurations and perform a basic analysis. Here, we are not interested in conducting a thorough comparison study. Instead, we provide a number of model components together with an approach to analyze different configurations. For more on different configurations and lay-outs, we refer to [CAJ01].

8.1 Bypassing

Consider a flowline consisting of three machines and three infinite buffers. The flowline processes two lot types. The generator releases a new lot every 4.0 hours. The chance that the new lot is of type 0 is 50%. Lots of type 0 require processing on all three machines subsequently, lots of type 1 only require processing on machine M_0 and M_2. Processing a lot takes 3.0 hours on machine M_0 and M_2 and 6.0 hours on `M1`. The minimal flowtime for lots of type 0 is 12.0 hours, for lots of type 1 it is 6.0 hours. Thus, on average the minimal flowtime is 9.0 hours. We wish to determine the mean flowtime by simulation. Figure 8.2 shows a graphical representation of the model.

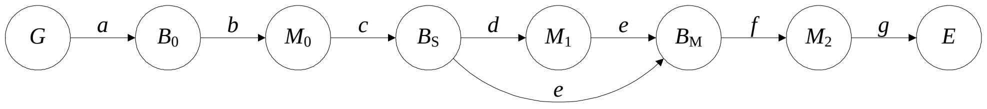

$$\begin{tikzcd} G \arrow[r, "a"] & B_0 \arrow[r, "b"] & M_0 \arrow[r, "c"] & B_S \arrow[r, "d"] \arrow[rr, bend right, "e"'] & M_1 \arrow[r, "e"] & B_M \arrow[r, "f"] & M_2 \arrow[r, "g"] & E \end{tikzcd}$$

Figure 8.2: Three-workstation flowline with bypassing

A lot is of type `tuple(int num; real t; int k)`, denoting the identification number, the time of release, and the lot type `0` or `1`. We specify generator `G` as follows.

```
type lot = tuple(int num; real t; int k);

proc G(chan! lot a):
   int i;
   dist int d = uniform(0, 2);
   while true:                                         Chi-8.1
      a!(i, time, sample d);
      i = i + 1;
      delay 4.0
   end
end
```

Buffer `B0` in front of machine `M0` is a standard buffer, in which the lots are all stored in a single list in order of arrival. Buffer `BS` in front of machine `M1` is a splitting buffer. Buffer `BS` receives lots and sends lots of type 0 via channel `b` to machine `M1` and lots of type 1 via channel `c` to the buffer in front of machine `M2`: lots of type 1 bypass machine `M1` and are directly sent to buffer `BM`. We can specify buffer `BS` in different ways. Here, we choose to store all lots in a single list.

```
proc Bs(chan? lot a; chan! lot b, c):
   list lot xs; lot x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         checktype(xs, 0), b!xs[0]:
            xs = xs[1:]
      alt
         checktype(xs, 1), c!xs[0]:
            xs = xs[1:]                                Chi-8.2
      end
   end
end

func bool checktype(list lot xs; int n):
   if empty(xs):
      return false
   else:
      return xs[0].k == n
   end
end
```

Buffer `BS` is always willing to receive lots. The function `checktype(xs, 0)` checks whether the first lot in list `xs` is of type 0. If the first lot in `xs` is of type 0, the function call `checktype(xs, 0)` returns true. If the first element in list `xs` is of type 0, `BS` tries to send the head of the list via channel *b*. If the first element in list `xs` is of type 1, `BS` tries to send the head of the list via channel `c`. Note that we use the function `checktype(xs, 0)` as guard in the selective waiting in `BS` instead of the guard `size(xs) > 0 and xs[0].k = n`. In this way, we avoid taking the head of an empty list [1], when `size(xs) = 0`.

Buffer `BM` is a merging buffer and has one incoming channel, both for lots that have been processed on machine `M1` and a for lots that bypass machine `M1`. We specify buffer `BM` as a standard buffer:

```
proc B(chan? lot a; chan! lot b):
   list lot xs; lot x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end
```

Chi-8.3

Buffer `BM` is always willing to receive lots via channel `a`, which in the model definition is connected to `Bs` and `M1`.

We specify machine `M` with a parametric process time.

```
proc M(chan? lot a; chan! lot b; real t):
   lot x;
   while true:
      a?x; delay t; b!x
   end
end
```

Chi-8.4

Finally, we specify model `Bypass`.

```
model Bypass():
   chan lot a, b, c, d, e, f, g;
   run
      G(a),
      B(a, b),     M(b, c, 3.0),
      Bs(c, d, e), M(d, e, 6.0),
      B(e, f),     M(f, g, 3.0),
      E(g)
end
```

Chi-8.5

After adding output statements to `E`, simulation yields.

[1] In the implementation of Chi3.0, we can use this boolean expression, as the `and` operator is a conditional and. This means that for a boolean expression `bool-expr-1 and bool-expr-2` the second boolean expression is only evaluated if the first boolean expression evaluates true.

```
10.00 E received lot     1
13.00 E received lot     0
20.00 E received lot     2
23.00 E received lot     4
26.00 E received lot     3
29.00 E received lot     5
32.00 E received lot     6
35.00 E received lot     7
44.00 E received lot     8
50.00 E received lot     9
```

We see that lot 1 has passed lot 0. Lot 1 is of type 1 and does not require processing at machine `M1`. The flowtime of lot 1 is therefore 6.0 hours. The flowtime of lot 0 is 13.0 hours, of which 1.0 hour is due to waiting induced by lot 1. We establish a mean throughput of 0.25 lots/hour. The mean flowtime equals 12.5 hours. Part of this flowtime is caused by the fact that buffer `BS` only regards the first lot in the buffer. Suppose buffer `BS` contains three lots. The first lot is of type 0, the second and third lot are of type 1. If machine `M1` is busy, all lots in the buffer have to wait, even when machine M_2 is idle. To let buffer B_s forward lots of type 1, regardless the type of the first lot in line, we use a separate list for each lot type.

```
proc Bs(chan? lot a; list(2) chan! lot b):
   list(2) list lot xs; lot x;
   while true:
      select
         a?x:
            xs[x.k] = xs[x.k] + [x]
      alt
         unwind j in range(2):
            size(xs[j]) > 0, b[j]!xs[j][0]:
               xs[j] = xs[j][1:]
         end
      end
   end
end
```

Chi-8.6

Simulation now yields a mean flowtime of 11.6 hours. However, in this way the mean flowtime of lots of type 0 increases.

8.2 Backtracking

As an example of a flowline with backtracking, we consider a three-workstation flowline with possible rework on the middle machine. Machine `M1` performs an operation, which has 90% chance of being successful. In buffer `BT` behind machine `M1` lots are tested. Lots that require rework are sent back to the buffer in front of machine `M1`. The generator releases new lots in the line at a rate of 0.25 lot/hour. On every machine, processing a lot takes 3.0 hours. We wish to determine the mean utilisation of the middle machine by simulation. Figure 8.3 shows a graphical representation of the flowline.

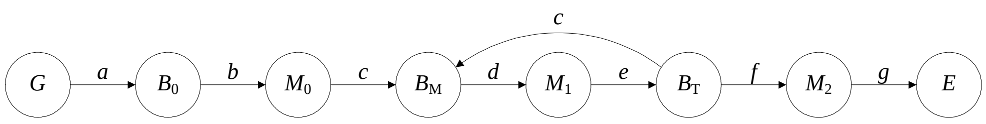


Figure 8.3: Three-workstation flowline with rework

Lots are of type (nat, real), representing the identification number and the time of release. Machine `M1` sends a tuple of a lot and a bool, in which the bool represents the result of the operation on machine `M1`. The last element is true, if the operation on machine `M1` was successful, otherwise it is false. We specify generator `G` and machine `M1` as follows.

```
type lot = tuple(int num; real t),
     bot = tuple(lot lt; bool ok);

proc G(chan! lot a):
   int i;
   while true:
      a!(i, time);
      i = i + 1;
      delay 4.0
   end
end
```

Chi-8.7

```
proc M1(chan? lot a; chan! bot b):
   dist bool d = bernoulli(0.3);
   lot x;
   while true:
      a?x;
      delay 3.5;
      b!(x, sample d)
   end
end
```

Testing buffer `BT` tests the lots that have been processed on machine `M1`. If the result is satisfactory, the lot is added to the list of waiting lots for machine M_2, otherwise it is sent back to merging buffer B_M. We specify buffer `BT` as follows.

```
proc BT(chan? bot a; chan! lot b, c):
   list lot xs, ys; lot x; bool ok;
   while true:
      select
         a?(x, ok):
         if ok:
            xs = xs + [x]
         else:
            ys = ys + [x]
         end
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      alt
         not empty(ys), c!ys[0]:
            ys = ys[1:]
      end
   end
end
```

Chi-8.8

We use a standard buffer specification (for instance Specification Chi-8.3) to represent the merging buffer. We connect it with the correct channels in accordance with Figure 8.3 by specifying model *Rework* as follows.

```
model Rework():
   chan lot a, b, c, d;
   chan bot e;
   chan lot f, g;
   run
      G(a),
      B(a, b),     M (b, c),
      B(c, d),     M1(d, e),
      BT(e, f, c), M (f, g),
      E(g)
end
```

Chi-8.9

By simulation, we determine a mean throughput of 0.25 lots/hour and a mean utilisation of 0.833 for machine `M1`. By hand, we can verify this result. Machine M_0 sends 0.25 lots/hour to machine `M1`. Of these 0.25 lots/hour 10% need rework. Of this 10%, 10% needs rework again, and so on. In total, $0.25 + 0.1 \cdot 0.25 + 0.01 \cdot 0.25 + ... = 1.111 \cdot 0.25 = 0.2778$ lots/hour arrive at the buffer for machine `M1`. This leads to a mean utilisation of $0.2778/0.3333 = 0.833$.

We can also deduce this using mass conservation from Chapter 2. Mass conservation on workstation $B_{\mathrm{M}}M_1$ yields:

$$\delta_{M_0} + 0.1\delta_{M_1} = \delta_{M_1}$$

With $\delta_{M_0} = \lambda$ we get:

$$\lambda = 0.9\delta_{M_1} \rightarrow \delta_{M_1} = \lambda/0.9$$

The utilisation of machine `M1` then is:

$$u_{M_1} = \delta_{M_1} \cdot t_{0,M_1} = \lambda/0.9 \cdot t_{0,M_1} = 0.25/0.9 \cdot 3.0 = 0.833$$

8.3 Assembly line

Consider an assembly process AP that assembles three different parts p_0, p_1, and p_2. Each part is stored in a separate buffer. We use separate generator processes to represent the arrival pattern of the parts (from other areas of the factory). We assume that parts arrive with a fixed inter-arrival time of 4.0 hours. The assembly process takes one part from each buffer. After the process has all the required parts, the assembly is started. The assembly takes 3.0 hours. We wish to investigate the performance by simulation. Figure 8.4 shows a graphical representation of the assembly line.

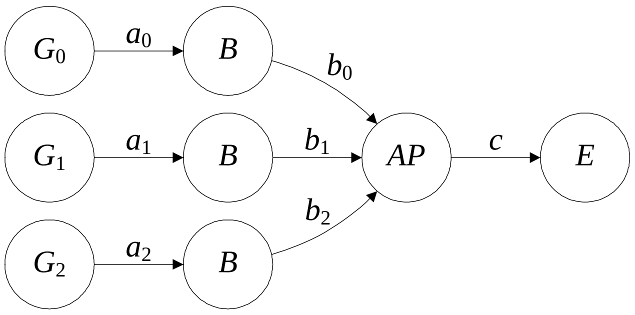


Figure 8.4: Line for assembling three parts

We specify process `G` as follows.

```
type lot = real,
     asy = list(3) real;

proc G(chan! lot a):
   while true:
      a!time;
      delay 4.0
   end
end
```

Chi-8.10

Buffers B are standard buffers storing parts. We specify assembly process AP as follows.

```
proc AP(list(3) chan? real a; chan! asy b):
   asy ps; lot x;
   set int cond;
   while true:
      cond = {0, 1, 2};
      select
         unwind j in range(3):
            j in cond, a[j]?x:
               ps[j] = x; cond = cond - {j}
         end
      end;
      b!ps
   end
end
```

Chi-8.11

Process *AP* uses variable *rcv* to monitor what parts have been received. If part p_i is not yet received ($rcv.i = \mathsf{false}$), it tries to take a part from the buffer over channel a_i. After all parts have been received, the parts are assembled. The assembly is sent over channel b. We specify model *Assembly* as follows.

```
model Assembly():
   list(3) chan real a, b;
   chan asy c;
   run
      unwind j in range(3):
         G(a[j]), B(a[j], b[j])
      end,
      AP(b, c), E(c)
end
```

Chi-8.12

After changing exit E to log the assemblies that leave the system, simulation yields:

```
3.000000     received assembly <(0,0.000000),(0,0.000000),(0,0.000000)>
7.000000     received assembly <(1,4.000000),(1,4.000000),(1,4.000000)>
11.000000    received assembly <(2,8.000000),(2,8.000000),(2,8.000000)>
15.000000    received assembly <(3,12.000000),(3,12.000000),(3,12.000000)>
19.000000    received assembly <(4,16.000000),(4,16.000000),(4,16.000000)>
23.000000    received assembly <(5,20.000000),(5,20.000000),(5,20.000000)>
27.000000    received assembly <(6,24.000000),(6,24.000000),(6,24.000000)>
31.000000    received assembly <(7,28.000000),(7,28.000000),(7,28.000000)>
```

The i-th element of the assembly represents part p_i. We attain a mean throughput of 0.25 assembly/hour. On average, a part is for 3.0 hours in the assembly line. Note that, because of the deterministic arrival times of the parts, the assembly process receives all parts at the same time.

We change the arrival times of the parts to become stochastic. We let the lots arrive according to a negative exponential distribution with a mean of 4.0 hours. A single simulation yields:

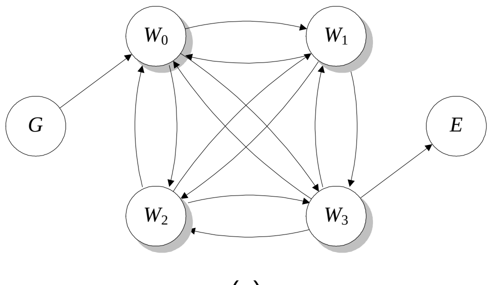
```
3.0000     received assembly <(0,0.00000),(0,0.00000),(0,0.0000)>
```

```
6.0000     received assembly <(1,1.78935),(1,2.1233),(1,0.919814)>
9.3672     received assembly <(2,2.84998),(2,4.6275),(2,6.3672)>
22.2245    received assembly <(3,3.54575),(3,19.2245),(3,10.9058)>
25.2245    received assembly <(4,5.14347),(4,19.2386),(4,20.3083)>
29.4133    received assembly <(5,5.19210),(5,19.5010),(5,26.4133)>
34.1696    received assembly <(6,8.51579),(6,23.7321),(6,31.1696)>
```

We can see that at $\tau = 22.2245$ for assembly number 3 part p_1 (that arrives at $\tau = 19.2245$) causes a substantial delay for the assembly process. By simulation, we establish that the mean throughput is still 0.25 assembly/hour. However, the mean time a part is in the system increases significantly. (Try! Can you tell from the simulation output whether the assembly time is still deterministic?)

8.4 Jobshop

The last configuration we discuss, is the jobshop. In a jobshop, we have multiple workstations. Each lot can have a different routing. In principle, a lot could go from every workstation to any other workstation. As an example, we consider a four workstation jobshop. Each job requires 6 operations. The routing varies per job. The first operation is always performed on workstation 0, the last operation is always performed on workstation 3. The remaining four operations are evenly distributed over all workstations, for instance, the chance that the second operation is to be performed on workstation 1 is 25%, just as the chance that it is to be performed on any other workstation.



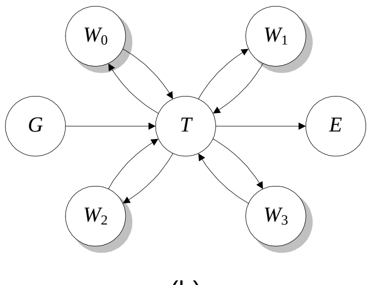


Figure 8.5: 4-workstation jobshop

Each workstation is connected to every other workstation. Generator `G` is connected to workstation 0 and workstation 3 is connected to exit E, see Figure 8.5(a). Each workstation consists of an infinite buffer and a machine with parametric process time t.

We let the generator release new jobs at a rate of 0.25 lots/hour. We wish to determine process time t for each workstation, so that the utilisation is 0.80 for each workstation.

It is possible to make a Chi-model of the jobshop in the way depicted in Figure 8.5(a). However, this model requires a large number of channels and a rather extensive model specification. Therefore, we propose an alternative model. In this model, we introduce a central process that connects all workstations. We call this process the transportation process. The graphical representation of this model is shown in Figure 8.5(b).

A job is of type [nat, [nat], real], representing the identification number, the job routing (e.g. $[0, 2, 1, 2, 2, 3]$), and the time of release in the system. We specify generator `G` as follows:

```
type job = tuple(int num; list(6) int rout; real t);

proc G(chan! job a):
   int i;
   dist int d = uniform(0, 4);
   list(5) int js;
   while true:
      js = [0];
      for j in range(4):                              Chi-8.13
         js = js + [sample d]
      end;
      js = js + [3];
      a!(i, js, time);
      i = i + 1;
      delay 4.0
   end
end
```

The generator generates a routing js, where the first and last element are 0 and 3 and the middle four elements are each drawn from a separate discrete uniform distribution `d`i[2].

Next, we specify transportation process T. We assume that the transportation takes no time.

```
proc T(chan?  job a; list(4) chan! job b; chan! job be):
   job x;
   while true:
      a?x;
      if empty(x.rout):
         be!x                                         Chi-8.14
      else:
         b[x.rout[0]]!x
      end
   end
end
```

The transportation process receives a job from the generator or from any workstation via channel a. Then, T checks how many operations are still to be performed. If no operations are left ($\text{len}(x.1) = 0$), the job is sent to the exit process. If there are operations left, the job is sent to the corresponding workstation.

Each workstation consists of a machine with deterministic processing time t and a standard infinite buffer. We specify machine M, buffer$B$, and process W as follows.

[2]Numerical random generators are never really random. They are pseudo-random generators. This delicate discussion is beyond the scope of this book. As a rule of thumb: use a separate distribution for every quantity.

```
proc B(chan? job a; chan! job b):
   list job xs; job x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end

proc M(chan? job a; chan! job b; real t):                    Chi-8.15
   job x;
   while true:
      a?x;
      delay t;
      x.rout = x.rout[1:];
      b!x
   end
end

proc W(chan? job a, c; real t):
   chan job b;
   run B(a, b), M(b, c, t)
end
```

Exit *E* is always able to receive finished jobs (not shown here). Finally we specify model *Jobshop*.

```
model Jobshop(real t0, t1, t2, t3):
   chan job a, be;
   list(4) chan job b;
   list(4) real t = [t0, t1, t2, t3];
   run
      G(a),                                                  Chi-8.16
      T(a, b, be),
      unwind j in range(4):
         W(b[j], a, t[j])
      end,
      E(be)
end
```

We add output statements to the generator, the machines, and the exit. If we let an operation on each workstation take 3.0 hours, simulation of a single job yields:

```
0.00000000000000000    G releases job     (0,[0,2,1,0,0,3],0)
3.00000000000000000    M0 processes job   (0,[0,2,1,0,0,3],0)
6.00000000000000000    M2 processes job   (0,[2,1,0,0,3],0)
```

```
9.00000000000000000    M1 processes job    (0,[1,0,0,3],0)
12.00000000000000000   M0 processes job    (0,[0,0,3],0)
15.00000000000000000   M0 processes job    (0,[0,3],0)
18.00000000000000000   M3 processes job    (0,[3],0)
18.00000000000000000   E received job      (0,[],0)
```

Now, we wish to obtain an utilisation of 0.80 for each machine. First, we make an analytical estimation. On average, workstation W_0 receives $1 + 4 \cdot P(W_0) = 1 + 4 \cdot 0.25 = 2$ lots per job, that is 0.5 lots/hour. W_3 also receives 0.5 lots/hour. Workstations W_1 and W_2 receive $4 \cdot 0.25 = 1$ lot per job, that is 0.25 lots/hour. In order to obtain an utilisation of 0.80, we estimate that we require a process time of 1.6 hour on machines M_0 and M_3 and a process time of 3.2 hours on machines `M1` and M_2.

Simulating the jobshop with these process times yields a throughput of 0.25 lots/hour, a mean flowtime of 26.6 hours and an utilisation of 0.80 on each machine. Note that the mean flowtime is significantly higher than the mean process time per job ($t_0 + t_3 + 4 \cdot \frac{t_0+t_1+t_2+t_3}{4} = 1.6+1.6+4 \cdot 2.4 = 12.8$ hours). Though the process times are deterministic, the stochastically generated jobs cause waiting times in the buffers!

Keyword overview

Modelling manufacturing systems	Analysis of manufacturing systems
• flowline with backtracking (rework)	
• flowline with bypassing	
• assembly line	
• jobshop	
• bundles	

8.5 Exercises

1. Consider an assembly process that assembles four different parts. Each part has a different inter-arrival time. We assume that we are dealing with fixed inter-arrival times. Table 8.2 shows the number of parts required per assembly and the inter-arrival time of that part.

   Each part is stored in a separate buffer. After the assembly processes has collected all required parts, the assembly commences. Assembling the parts takes 0.8 hours. The assembly is sent to exit E. Figure 8.6 shows a graphical representation of the assembly line.

   We represent an assembly by an vector of four lists. The i-th list in the assembly stores the parts of type i. We have the following fragments of specification.

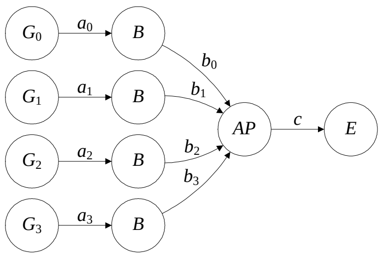


Figure 8.6: Assembly line

part	# in assembly	mean inter- arrival time [hour]
0	1	1.0
1	2	0.5
2	4	0.25
3	5	0.2

Table 8.2: Parts needed in assembly

Chi-8.17

```
type prt = real,
     asy = list(4) real;

proc G(chan! lot a):
   while true:
      a!time;
      delay 4.0
   end
end

func tuple(real t; real m) tAver(real ti_1, ti, mb; int b):
   if ti == ti_1:
      return (ti, mb)
   else:
      return (ti, mb * ti_1 / ti + b * (ti - ti_1) / ti)
   end
end

proc B(chan? prt a; chan! prt b; int n):
   list prt xs; prt x;
   real mb; int ti_1;
   int i;
   ti_1 = time;
   while true:
      select
         a?x:
            ti_1, mb = tAver(ti_1, time, mb,size(xs));
            xs= xs ++ [x]
      alt
         not empty(xs), b!xs[0]:
            ti_1, mb = tAver(ti_1, time, mb,size(xs));
            xs = xs[1:];
            i = i + 1;
            if i mod 10000 == 0:
               write("%10.2f\tB\t%5d\tmb =\t%10.2f\n", time, n, mb)
            end
      end
   end
end
```

```
proc AP(list(4) chan? prt a; chan! asy b; real t0):
   list(4) int pqr, rqr = [1, 2, 4, 5];
   list(4) prt ptc;
   prt p;
   while true:
      while pqr != rqr:
         select
            unwind j in range(4):
               pqr[j] < rqr[j], a[j]?p:
                  pqr[j] = pqr[j] +  1;
                  ptc[j] = ptc[j] + [p]
            end
         end
      end;
      delay t0;
      b!ptc;
      pqr = [0, 0, 0, 0];
      ptc = <pts>[ ]
   end
end
```

Chi-8.18

```
proc E(chan? asy a):
   asy x;
   while true:
      a?x
   end
end

model Assembly():
   list(4) chan prt a, b;
   chan asy c;
   list(4) real t = [1.0, 0.5, 0.25, 0.2];
   run
      unwind j in range(4):
         G(a[j], t[j]),
         B(a[j], b[j])
      end,
      AP(...),
      E(c)
end
```

(a) By hand, estimate the utilisation of the assembly process.

(b) Explain, whether the arrival rate of the parts is balanced, that is, does the assembly process need to wait for the same specific part over and over again?

(c) Finish the fragments of specification into a complete model.

(d) By simulation, determine the average buffer level in each part buffer.

(e) Figure 8.7 shows part of the buffer levels over time.
Finish the diagrams for each part buffer. Verify the simulation results.

(f) Change the arrival rate of the parts to become stochastic. Even after simulating for a substantial long time, we see that one or more buffer levels keep increasing. Can you explain this?

2. We have a manufacturing line that processes two lot types. The manufacturing line consists of 3 workstations. The first workstation has three identical machines M_0. The second and third workstation each have one machine. Each workstation has an infinite buffer. Lots of type 0 require processing on all three workstation. Lots of type

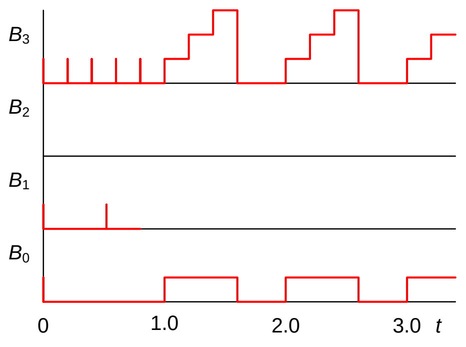


Figure 8.7: Number of parts in buffer over time

1 require processing on M_0 and M_2 only. Processing a lot on machine M_0, `M1`, and M_2 takes 9.0, 5.0, and 3.0 hours respectively. Figure 8.8 shows a graphical representation of this manufacturing line.

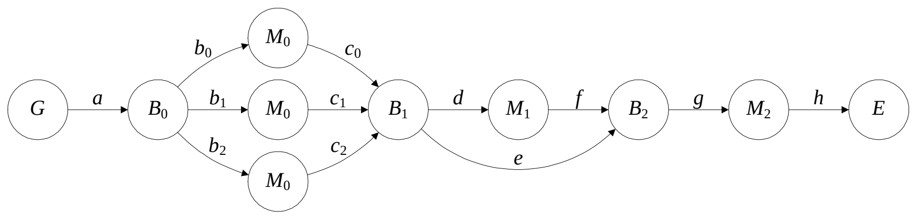


Figure 8.8: Manufacturing line

Lots are of type (nat, nat, real), representing the identification number, the lot type, and the time of release in the line.

(a) By hand, estimate the utilisation for each machine if we release lots in the line at a rate of 0.25 lots/hour.

(b) Build a model of this manufacturing line. Use fragments of the specifications discussed in Sections 8.1, 8.2, and 7.1.

(c) By simulation, determine the mean throughput and flowtime of this line for a release rate of 0.25 lots/hour.

(d) By hand, calculate the mean wip level of this line.

(e) By simulation, determine the mean utilisation of each machine. Compare your results with your estimations from part 2a.

3. In re-entrant flowlines, a lot runs through the flowline multiple times. In this way, we can perform a sequence of process steps more than once on a single lot. This is often encountered in the semi-conductor industry, where wafers are processed in layers. Here, we study the behaviour of a simple re-entrant flowline. The flowline has two machines and three buffers. The last buffer checks whether the lot needs to re-enter the flowline or whether it is finished. Figure 8.9 shows a graphical representation of the re-entrant flowline.

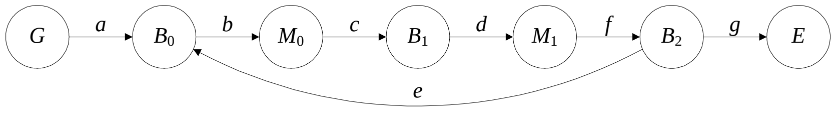


Figure 8.9: Re-entrant flowline

Processing takes 0.2 hours on each machine. There are three lot types. The lot types need 1,2, and 3 loops through the flowline respectively. The generator releases lots at a fixed arrival rate. On average, each lot type is generated in the same quantity.

(a) By hand, estimate the release rate for which the utilisation of both machines is 0.8.

(b) Let lots be of type (nat, real, nat), where the elements represent the identification number, the time of release, and the remaining number of loops through the line. Make a model of this manufacturing line. Below, the specification of buffer B_2 is shown.

```
proc B2(chan? lot a; chan! lot b, c):
   list lot xs, ys; lot x;
   while true:
      select
         a?x:
            x.rl = x.rl - 1;
            if x.rl > 0:
               xs = xs + [x]
            else:
               ys = ys + [x]
            end
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      alt
         not empty(ys), b!ys[0]:
            ys = ys[1:]
      end
   end
end
```

Chi-8.19

(c) By simulation, determine the mean utilisation per machine for different release rates. For which release rate is the utilisation 0.8? Does this correspond to your estimation?

Chapter 9

Extending machine models

In the previous chapters we have encountered several models of machines. On some occasions, we assumed fixed (constant) process times. On others, we treated process times as being stochastic. We then proposed a distribution to describe the process time. In this chapter we address in more detail how to accurately model machines. We will discuss the following aspects:

- using a distribution to represent process time,
- modelling the failure and repair of machines, and
- using effective process time to represent all sources of variability.

9.1 Representing process time by a distribution

In this section we present a brief stepwise approach to represent process times by distributions. This approach can also, at least to a large extent, be applied to describe the arrival pattern of raw materials or demand pattern of finished goods. One of the main tasks is to find and properly fit a distribution to the measured process time data. There is a large quantity of literature available on fitting a distribution to a set of data. We present the main techniques involved in fitting distributions, without addressing the statistical background in detail. More on the statistical background can be found in [LK00, MR02]. We discuss using distributions to represent process times in six steps.

1. Collect data.
2. Determine whether the observations are independent and identically distributed.
3. Plot a histogram and calculate basic statistical quantities.
4. Choose candidate distributions.
5. Determine the distribution parameters by fitting the chosen distribution to the sample data and determine how well the fitted distribution fits the data.

6. Implementing the distribution in your Chi-model.

Many packages support statistical analysis of data, amongst others Matlab [Mat03], SAS [SAS03], SPSS [SPS03], and Statgraphics [Man03]. In this chapter, we use Statgraphics. Statgraphics is more accessible than the more extensive packages SAS and SPSS, while having about the same functionality with respect to fitting distributions.

Step 1: Collecting data

The first step is collecting process time data. In modern high-tech fabs, times at which a machine starts and finishes processing is often recorded in log files. In many fabs we still need to collect data manually, by means of clocking. There is much that can go wrong in collecting data. One may think of censored data (the operator reports only fast (or only slow) jobs or one may think of observations that are not independent (see next step). More on incorrect data can be found in [BFS87].

Another delicate issue regards the number of observations that is needed to reasonably fit a distribution. No hard guidelines for determining this number are available. In general, more observations allow more accurate fitting of the distribution and more profound judgements regarding the validity of the fitted distribution.

Throughout this section we consider the sample data in Table 9.1 adopted from [Lee99].

105.84	28.92	98.64	55.56	128.04	45.60
67.80	105.12	48.48	51.84	173.40	51.96
54.12	68.64	93.12	68.88	84.12	68.64
41.52	127.92	42.12	17.88	33.00	

Table 9.1: Measured process times in seconds

Step 2: Checking data independency

If we wish to fit a distribution to process time data, the individual observations need to be independent and identically distributed (*idd*), that is, the observations are independent samples of a single distribution. Situations in which the idd assumption is not valid include:

- A new operator has been hired and the process time strongly depends on the operator experience, that is the process time is subject to a steep learning curve. Process time measurements are negatively correlated.
- The machine processes in shifts, and the machine has to warm up in the beginning of each shift. Process time measurements in the beginning of each shift are negatively correlated.
- We measure the flowtime of lots at a single workstation. If a lot has a large flowtime, the next lot is likely to have a large flowtime too. The flowtimes are positively correlated.

Many processes are non-stationary, that is, distributions change over time. For assessing time-dependency we can plot the observations against time or against sequence number. When we observe a pattern, we can use simple regression to analyze the degree of time-dependency. Figure 9.1 shows the time-series plot of the data from Table 9.1. The data shows no clear pattern.

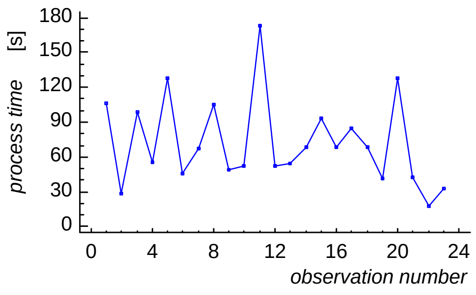


Figure 9.1: Time-series plot

From Figure 9.1, we assume that the observations are time-independent.

We present two graphical approaches to assess (auto)correlation[1] between subsequent observations. The first one is making a scatter diagram for observations $X_1, X_2, ..., X_n$ in which we plot the pairs (X_i, X_{i+1}). If the observations are independent, the points are scattered randomly. The nature of the random scattering depends on the underlying distribution. If correlation exists, the points lie on a line, in case of positive correlation on an inclining line and in case of negative correlation on a declining line. The second graphical approach is the correlation plot. The correlation plot is a graph of the estimated autocorrelations $\hat{\rho}_j$ for $j = 1, 2, ..., l$. The estimated autocorrelation $\hat{\rho}_j$ is an estimation of the true correlation ρ_j between two observations that are j observations apart in time. If the observations $X_1, X_2, ...X_n$ are independent, correlations ρ_j are zero. When the estimations $\hat{\rho}_j$ deviate significantly from zero, this is strong evidence that the observations are not independent. For calculating the autocorrelation $\hat{\rho}_j$ we refer to [MR02]. Here, we used Statgraphics to draw a scatter diagram and calculate the autocorrelation, see Figure 9.2.

Figure 9.2(a) shows the scatter diagram and Figure 9.2(b) shows the correlation plot for the observations from Table 9.1. Figure 9.2(b) shows the estimated autocorrelation for different lags ({0,1,2,...,7}). Also shown are 95.0% probability limits around 0.0. If the probability limits at a particular lag do not contain the estimated coefficient, there is a statistically significant correlation at that lag at the 95.0% confidence level. In this case, there is no evidence that autocorrelation exists.

Apart from these graphical methods, several statistical tests exist. For these, one is referred to [LK00, MR02]. Judging from the diagrams in Figure 9.2, we assume that the observations are independent and identically distributed.

[1]Statistically, we prefer speaking of autocorrelation, since the observations stem from the same source distribution.

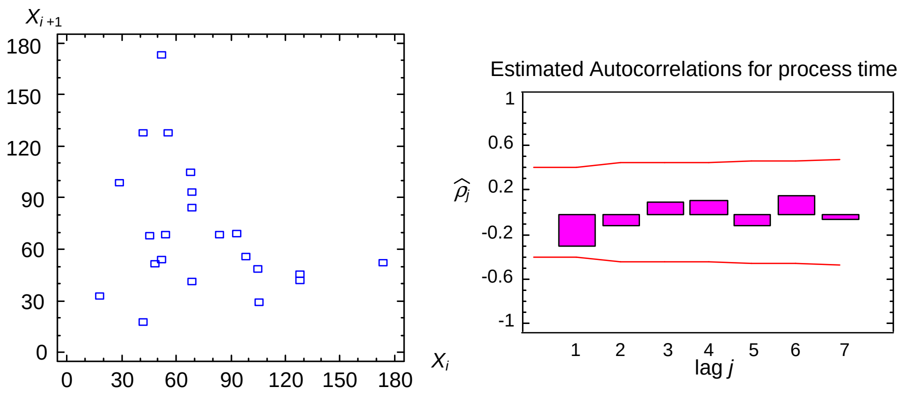


Figure 9.2: (a) Scatter diagram, (b) Estimated autocorrelation

Step 3: Basic statistical analysis

The next step is to perform a basic statistical analysis. We start by plotting a histogram of the data. Hereto, we divide the range of data into a number of intervals, and count the number of observations within each interval. The number of intervals can strongly influence the appearance of the histogram. A rule of thumb is to take a number of intervals equal to the square root of the number of observations. Figure 9.3 shows the histogram for the data of Table 9.1, drawn by Statgraphics.

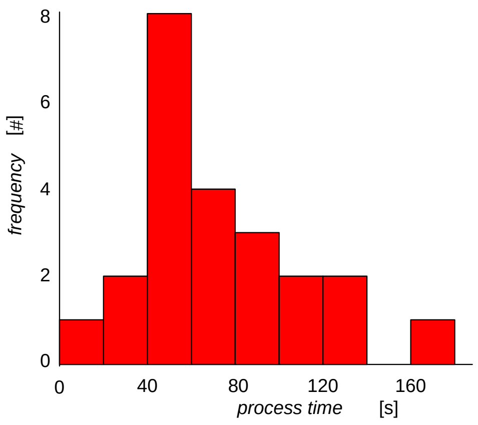


Figure 9.3: Histogram of process time measurements

Please note that in histograms, the upper limit of each interval is excluded from that interval. In this case, the second interval includes observations in the range $20 \leq t < 40$.

Next, we calculate some basic statistic quantities, like the range R, mean $\overline{x}$, standard deviation s, and standardized skewness[2] ν. They are calculated as follows:

$$\begin{aligned} R &= \max(x) - \min(x) & \overline{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ s^2 &= \frac{1}{n-1}\textstyle\sum_{i=1}^{n}(x_i-\overline{x})^2 & \nu &= \frac{1}{(n-1)s^3}\sum_{i=1}^{n}(x_i-\overline{x})^3 \end{aligned}$$

The mean $\overline{x}$ and standard deviation s are commonly used statistics. Skewness ν is an indication for the distribution symmetry. A skewness close to 0 indicates a symmetrical distribution. Positive values of skewness indicate that the upper tail of the curve is longer than the lower tail; negative values indicate a longer lower tail. Standardized skewness values outside the range of -2 to $+2$ indicate significant deviation from normality.

For the data of Table 9.1 we determine the following:

$$R = 155.52, \quad \overline{x} = 72.22, \quad s = 37.5, \quad \nu = 1.97$$

Thus, we are dealing with a distribution that has a mean of 72.22 seconds, a standard deviation of 37.5 seconds, and a small positive skewness, indicating a small deviation from symmetry, with the upper tail being a little longer.

Step 4: Selecting candidate distributions

Many different types of theoretical distributions exist. Here, we discuss the distributions listed in Table 9.2. These distributions are some of the most often encountered distributions in modelling manufacturing systems.

discrete distributions	continuous distributions
uniform	uniform
bernoulli	triangular
	normal
	exponential/gamma
	lognormal

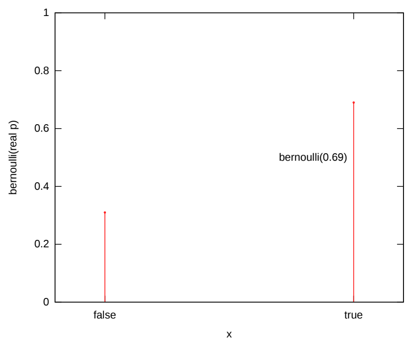
Table 9.2: Commonly used distributions

In the sequel, we present the typical shape and the parameters of each distribution, along with its range of common application. This overview is based on [LK00]. Additionally, we discuss how the distribution can be used in Chi. Next to its name and parameters, we indicate the type of parameters and the type of the distribution samples. For example `dist bool bernoulli(real p)` indicates that the Bernoulli distribution has one parameter of type real and that samples from the distribution are of type bool.

Other distributions include the binomial, the Erlang, the Weibull, the beta, the Pearson, the Johnson, and the log-logistic distribution. More on these distributions can be found in

[2]In literature, more than one formula is encountered to calculate skewness. Statgraphics calculates the so-called standardized skewness, which is calculated as presented here.

Bernoulli



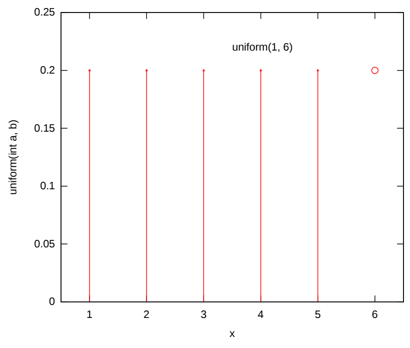
Discrete distribution that has two possible outcomes: `false` and `true`.

Function
`dist bool bernoulli(real p)`

Parameters
`p`: Chance on sampling `true`

mean	p
variance	$p(1-p)$

Discrete uniform



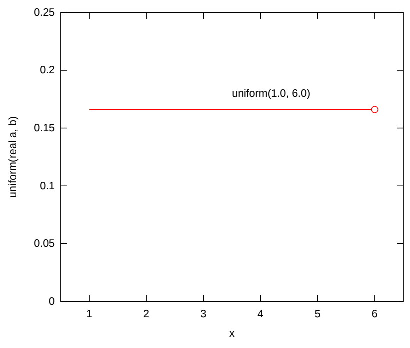
Discrete distribution that has several equally likely outcomes, the numbers $\{a, a+1, a+2, \ldots, b-2, b-1\}$. Note that b is **not included**.

Function
`dist int uniform(int a, b)`

Parameters
`a`: Lower bound
`b`: Upper bound (exclusive!)

mean	$(a+b-1)/2$
variance	$((b-a)^2-1)/12$

Continuous uniform



Continuous distribution with equal chance of sampling each value in the range $[a, b)$. Note that b is **not included**.

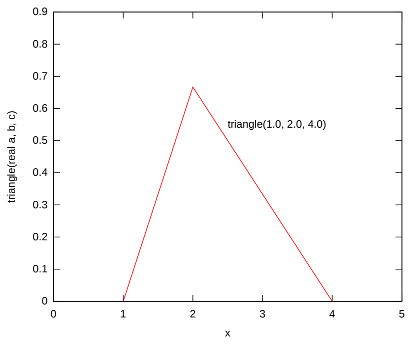
Function
`dist real uniform(real a, b)`

Parameters
`a`: Lower bound
`b`: Upper bound (exclusive!)

mean	$(a+b)/2$
variance	$(b-a)^2/12$

[LK00]. Next to using these theoretical distributions we can also use empirical or trace-driven models. A trace-driven model simply reproduces the collected data (in the same order). In case of an empirical model, we generate random values from the collected data. We can draw randomly from the set of data or we can use some sort of interpolation. More on empirical distributions can be found in [Ban98].

Triangular



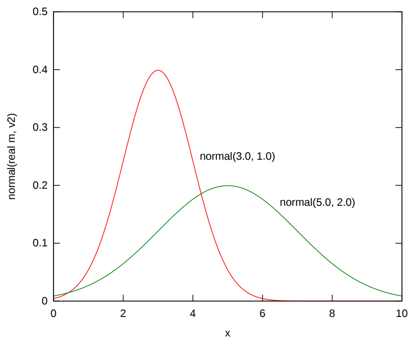
Continuous distribution producing samples in the range $[a, c]$ $(a < b < c)$.

Function
`dist real triangle(real a, b, c)`

Parameters
`a`: Lower bound
`b`: Mode
`c`: Upper bound

mean	$(a + b + c)/3$
variance	$(a^2 + c^2 + b^2 - a * b - a * c - b * c)/18$

Normal



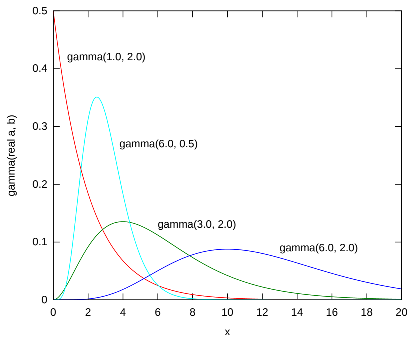
Bell-shaped curve around a with 'width' $2\sqrt{b}$.

Function
`dist real normal(real a, b)`

Parameters
`a`: Mean value
`b`: Variance

mean	a
variance	b

Gamma



Distribution which has either a decreasing probability function, or a peak.

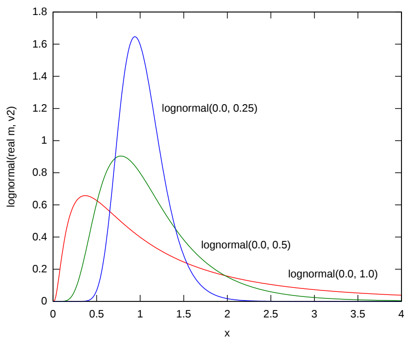
Function
`dist real gamma(real a, b)`

Parameters
`a`: Shape parameter
`b`: Scale parameter

mean	ab
variance	ab^2

Instead of fitting the data to all possible distributions, we first select possible candidate distributions that reasonably fit the collected data. Often, we can rule out distributions based on prior knowledge. For example, normal distributions are unlikely to be a good candidate for process times, since a sample from a normal distribution can be negative. Next, we compare the histogram from step 3 to the typical shapes of the distribution and

Lognormal



Takes on shapes similar to `gamma(p, q)` with `p` > 1. It can have large peaks close to $x = 0$.

Function
`dist real lognormal(real a, b)`

Parameters
`a`: Shape parameter
`b`: Scale parameter
mean $e^{a+b/2}$
variance $e^{2*a+b}(e^b - 1)$

select possible candidates. Finally, we use Statgraphics (or another statistical program) to draw probability plots. In a probability plot we use specific probability paper for each distribution. We order the observations in increasing order. We then plot the ordered list of observations against there cumulative frequency. Figure 9.4 shows the probability plot for the data from Table 9.1 for a uniform, normal, exponential, and lognormal distribution.

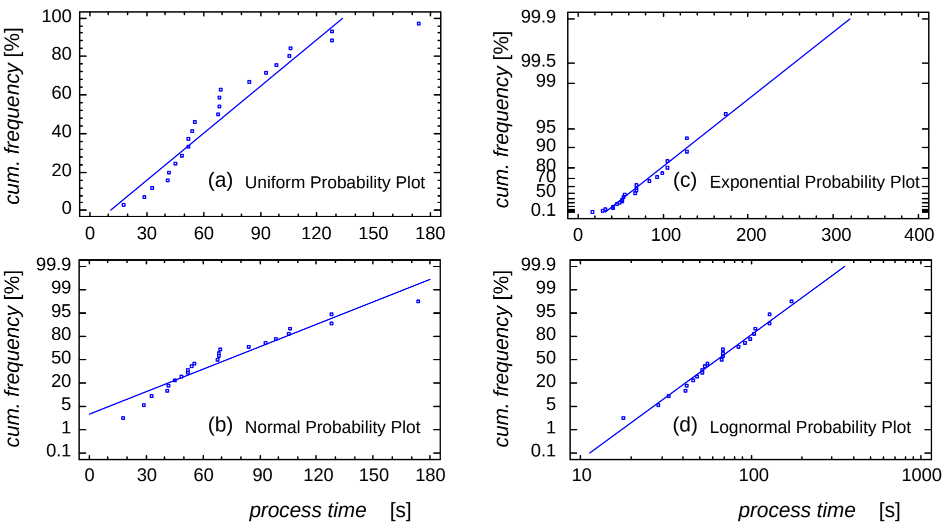


Figure 9.4: Probability plot for (a) uniform, (b) normal, (c) exponential, and (d) lognormal distribution

Note that the x- and y-axis can be different for each distribution type. If the plotted points fall approximately along a straight line, the corresponding distribution adequately describes the data. In this example, we see that the exponential and lognormal distribution are good candidates.

Step 5: Estimation of parameters

The next step is to estimate the parameters of the hypothesized distribution. Most statistical packages have built-in functionality to estimate these parameters. For more statistical background on how these parameters are estimated, we refer to [LK00].

We use Statgraphics to estimate the parameters for an exponential, a lognormal, and a gamma distribution. The fitted distributions are displayed in Figure 9.5, along with a histogram of the original observations.

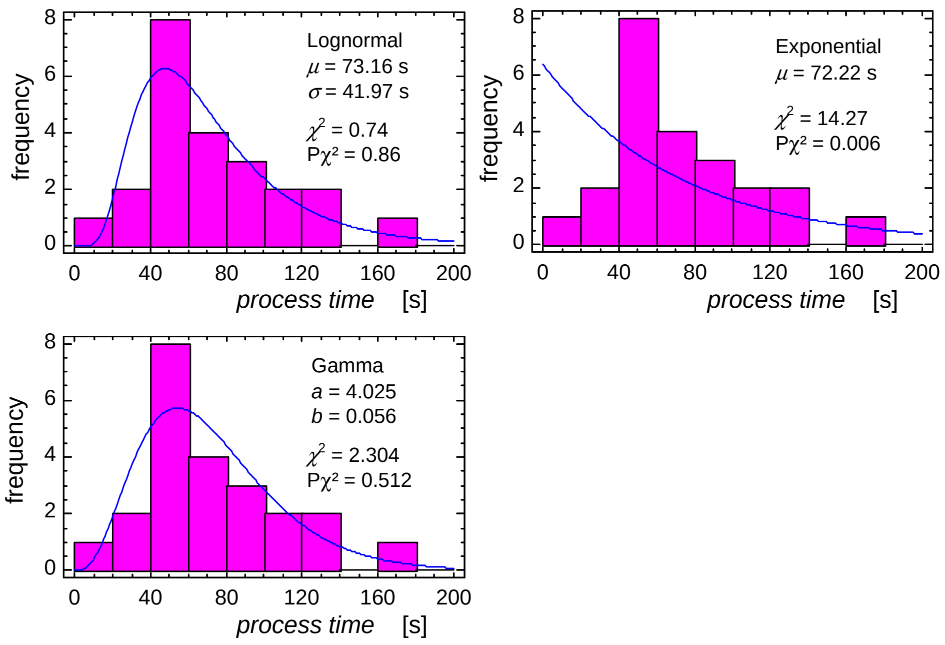


Figure 9.5: Fitted distributions

To assess the goodness-of-fit, two basic approaches exist. In the first approach we compare the frequency plot of the fitted distribution with the frequency plot from the observations. This can be done graphically by plotting the frequency plot of the fitted distribution together with the histogram of the original observations, as has been done in Figure 9.5.

Quantitatively comparing the histogram with the fitted distribution can be done using a χ^2-test [Pea00]. In this test we divide the range of the fitted distribution into k adjacent intervals. Then, we determine the expected frequency E_j of samples in the j-th interval if we were sampling from the fitted distribution. The χ-square test compares this expected frequency E_j with the actually measured frequency O_j. The test statistic is given by:

$$\chi^2 = \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j} \tag{9.1}$$

For small values of χ^2, the fitted distribution adequately represents the data. Statgraphics calculates the P-value associated with the calculated value of χ^2, indicating the chance that the set of observations are a set of samples from the hypothesized distribution. If P is small enough, we reject the hypothesis that the observations are from the fitted distributions with $(1 - P)$ confidence. Generally, if P is greater than 0.10, there is insufficient evidence for rejecting the hypothesized distribution. In Figure 9.5, the values of χ^2 and the associated P-value are calculated for each distribution. Based on the χ^2-test, we reject the exponential distribution with 99.4% confidence. We do not reject the gamma and lognormal distribution. The χ^2 values indicate that the lognormal distribution is a better fit.

Please note, that the χ^2-test results are strongly influenced by how we choose the intervals. Different intervals can yield totally different results. [LK00] advises intervals that each include at least 5 observations.

The second approach to assess goodness-of-fit is comparing the cumulative probability of the fitted distribution with the cumulative frequency of the original observations. This can be done graphically or by using a test statistic. Figure 9.6 shows the cumulative probability for the fitted distributions, along with the cumulative frequency of the observations.

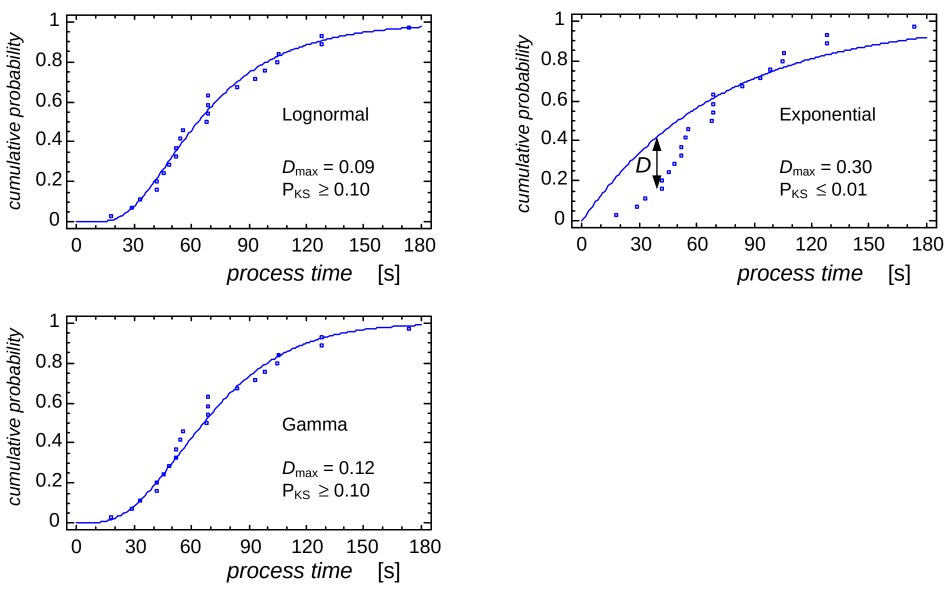


Figure 9.6: Cumulative probability plot with cumulative frequency of observations

A commonly used test is the Kolmogorov-Smirnov test [Kol33, Smi33]. This test calculates the maximum vertical distance D_{max} of the observations to the cumulative frequency diagram, as indicated for the exponential distribution in Figure 9.6. Using Statgraphics we have determined the maximum vertical distance D_{max} for each distribution. The maximum distance that is allowed, depends on the number of observations and the specific distribution type. Statgraphics calculates a P_{KS}-value that indicates the confidence with which the fitted distribution is adequate. If P_{KS} is sufficiently small, we reject the hypothesized distribution with $(1-P_{\text{KS}})$ confidence. Generally, if P_{KS} is greater than 0.10, there is insufficient evidence for rejecting the hypothesized distribution. The values for D_{max} and P_{KS} are displayed in Figure 9.6. Based on the Kolmogorov-Smirnov test, we reject the exponential distribution with at least 99% confidence. We can not reject the lognormal and gamma distribution. As for the χ^2-test, we again see that the lognormal distribution is the best fit.

For the Kolmogorov-Smirnov test we do not need to define intervals and we can use the test for any sample size n. However, unlike the χ^2-test, which can be used for testing against both continuous and discrete distributions, the Kolmogorov-Smirnov test is only appropriate for testing data against a continuous distribution.

Other goodness-of-fit tests exist, such as the Anderson-Darling, Kuiper, and Watson test. More details on goodness-of-fit tests can be found in [Ste74].

Step 6: Implementation

The final step is to implement the fitted distribution to represent the process time in the Chi-specification. In many cases, the parameters calculated by statistical packages correspond to the parameters with which the distribution is instantiated in Chi. In some cases, the calculated parameters have to be converted. In case of the lognormal distribution fitted to the data of Table 9.1, Statgraphics calculates mean $\mu = 73.16$ and standard deviation $\sigma = 41.97$, see Figure 9.5. However, in Chi we require the shape and scale parameters a en b, see the distribution overview from Step 4. Below, we present the conversion from μ and σ to shape and scale parameters a and b for a gamma and a lognormal distribution.

$$\begin{array}{lll} \text{gamma(a,b)} & a = \dfrac{\mu^2}{\sigma^2} & b = \dfrac{\sigma^2}{\mu} \\ \text{lognormal(a,b)} & a = \ln \mu - 0.5 \times \ln{(1 + \dfrac{\sigma^2}{\mu^2})} & b = \ln{(1 + \dfrac{\sigma^2}{\mu^2})} \end{array} \tag{9.2}$$

For the fitted lognormal distribution, the resulting machine model is shown below.

```
proc M(chan? lot inp; chan! lot outp):
   real t0  = 73.16,
        t02 = t0 * t0,
        s0  = 41.97,
        s02 = s0 * s0,
        a = ln(t0) - ln(1 + s02 / t02) / 2,
        b = ln(1 + s02 / t02);
   dist real d = lognormal(a, b);
   lot x;
   while true:
      inp?x; delay sample d; outp!x
   end
end
```

Chi-9.1

9.2 Machine failure and repair

In the previous section, we discussed ways to use a distribution to represent the process time of a lot. In practice, machines may break down and require repair, causing a lot to longer occupy the machine. In this section we discuss ways to model machine failure and repair. We distinguish two types of failure [Ger94]:

- Operation Independent Failure (OIF) and

- Operation Dependent Failure (ODF).

In case of OIF, the machine breaks down after being up for a certain amount of time, independent of the amount of time that the machine has been processing lots. By the amount of time a machine is up, we mean the time that a machine is turned on and has not broken down. When a machine is up and not processing, the machine is blocked (the subsequent buffer is full), the machine is starving (the preceding buffer is empty), or the machine is idle due to for example the control strategy. In case of ODF, the machine breaks down after processing for a certain amount of time, or after processing a certain number of lots. In Figure 9.7, the different types of failure are schematically depicted.

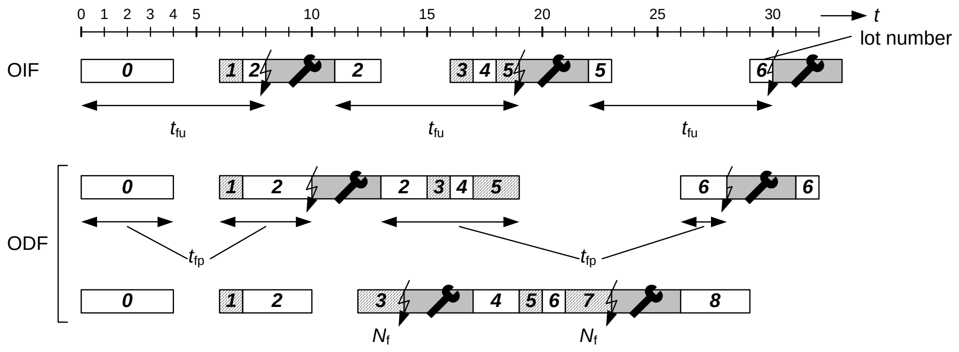


Figure 9.7: Operation Independent Failure (OIF) and Operation Dependent Failure (ODF)

In the upper case, the machine breaks down after being up for a fixed amount of time t_{fu} (OIF). In the middle case, the machine breaks down after processing for a fixed amount of time t_{fp} (ODF). In the bottom case, the machine breaks down after processing $N_{\mathrm{f}} = 4$ lots (ODF).

Adopted from [Lew95], we define the following quantities for characterizing failure and repair behaviour:

- Actual time between failure (ATBF): the time between two subsequent failures, denoted by t_{fa}.
- Processing time between failure (PTBF): the amount of processing time between two subsequent failures, denoted by t_{fp}.
- Up time between failure (UTBF): the amount of up time between two subsequent failures, denoted by t_{fu}.
- Cycles between failure (CBF): number of cycles (number of lots) processed between two subsequent failures, denoted by N_{f}.
- Time to repair (TTR): amount of time required for repair, denoted by t_{r}.

These quantities are schematically depicted in the state diagram of Figure 9.8.

All quantities can be stochastic. In that case, we have mean process time between failure $\bar{t}_{\mathrm{fp}}$, mean uptime between failure $\bar{t}_{\mathrm{fu}}$, mean number of cycles between failure $\overline{N}_{\mathrm{f}}$ and mean time to repair $\bar{t}_{\mathrm{r}}$.

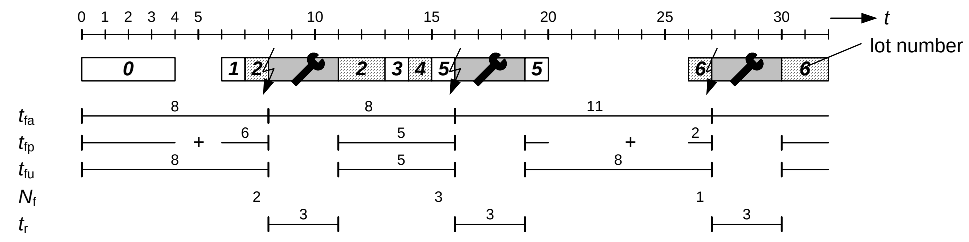


Figure 9.8: Quantities characterizing failure

In the sequel we present ways to model machine failure. We consider a single workstation, consisting of a infinite buffer and a machine. The process time t_0 of the machine is distributed according to a gamma distribution with mean 3.0 and coefficient of variation of 1.0. We consider four cases of different failure behaviour.

1. The machine fails after processing a certain number of lots.
2. The machine fails after processing for a certain amount of time. After the machine is repaired, the machine continues processing where it left off at the moment the machine broke down.
3. The machine fails after processing for a certain amount of time. After repair, the machine starts processing the lot from the beginning.
4. The machine fails after being up for a certain amount of time. After the machine is repaired, the machine continues processing where it left off at the moment the machine broke down.

In all cases, we let lots arrive at the buffer with a fixed arrival rate of 0.25 lots/hour. The exit can always receive lots.

1. Failure after processing a number of lots

On average, the machine fails every 100/3 = 33.3th lot. The number of lots between two failures is exponentially distributed and is a whole number. The repair time is also exponentially distributed with an average of 10.0 hours. We use a standard specification of generator G, buffer B, and exit E. The machine is specified as follows.

```
proc M(chan? lot a; chan! lot b):
   dist real dt0 = gamma(1.0 / (1.0 * 1.0), 3.0 / (1.0 * 1.0)),
             dNf = exponential(100.0 / 3.0),
             dtr = exponential( 10.0);
   int Nf;
   lot x;
   Nf = round(sample dNf);
   while true:
      if Nf == 0:
         delay sample dtr;                                    Chi-9.2
         Nf = round(sample dNf)
      else:
         a?x;
         delay sample dt0;
         b!x;
         Nf = Nf - 1
      end
   end
end
```

The machine specification contains three distributions. Distribution dt_0 represents the process time, and distributions dN_f and dt_r represent the number of lots between failure and the repair time respectively. We use variable N_f to record the number of lots that can be processed before failure. We draw N_f from an exponential distribution. Since the exponential distribution yields real samples, and we want N_f to be a whole number, we round the sample from the distribution to its nearest integer[3] If the number of lots before failure N_f is zero, the machine is repaired, which takes $\Delta\sigma dt_\mathrm{r}$ hours, and a new value for N_f is drawn. If the number of lots before failure N_f is greater than 0, the machine is able to receive a lot, process it, and send it to the next process. The number of lots before failure is decreased by 1.

By simulation we establish a mean actual time between failure of 135 hours and a downtime percentage of 7.4%. Adding the machine failure, causes the mean flowtime of a lot to increase from 6.5 hours to 12.5 hours. This increase is only for a small part ascribed to the repair time (on average every 33th lot stays for 3.0 + 10.0 = 13.0 hours on the machine). The largest part of the flow time increase is due to the increase in variability on the occupation time of a lot on the machine, which increases the average waiting time in the buffer.

2. Failure after processing for an amount of time

On average, the machine breaks down after processing for 100.0 hours. On average, repair takes 10.0 hours. The processing time between failure and the time to repair are both exponentially distributed. We use standard specifications for the generator G, buffer B, and exit E. The machine with failure is specified as follows.

[3]Since all samples taken from a exponential distribution are greater than zero, the rounded sample is a natural. However, the built-in round function yields an integer. We use the built-in i2n function to convert the integer into a natural.

```
proc M(chan? lot a; chan! lot b):
   dist real dt0  = gamma(1.0 / (1.0 * 1.0), 3.0 / (1.0 * 1.0)),
             dtf = exponential(100.0),
             dtr  = exponential( 10.0);
   timer t0, tf;
   real remaining_f, remaining_t0;
   lot x;

   remaining_f = sample dtf;
   while true:
      a?x;
      tf = timer(remaining_f);
      t0 = timer(sample dt0);
      while true:                                        Chi-9.3
         select
            ready(t0): break
         alt
            ready(tf):
               remaining_t0 = real(t0);
               delay sample dtr;
               t0 = timer(remaining_t0);
               tf = timer(sample dtf)
         end
      end;
      remaining_f = real(tf);
      b!x
   end
end
```

The distributions dt_0, dt_{fp}, dt_{r} represent the process time, the process time between failure, and the repair time respectively. The machine can only fail while processing. After receiving a lot, we take a sample from the process time distribution for that lot. The variable t_0 represents the remaining process time for the lot in process. The variable t_{fp} represents the remaining process time till failure. As long as t_0 is greater than 0.0, the machine finishes processing (Δt_0) or fails (Δt_{fp}), depending on what happens first. If the machine finishes processing, the remaining processing time till failure is decreased by t_0 and the remaining process time is nulled. If the machine fails, the remaining process time is decreased by t_{fp}, the machine is repaired, taking $\Delta\sigma dt_{\mathrm{r}}$, and a new sample from distribution dt_{fp} is drawn to determine the next moment of failure. Subsequently, the processing in continued. After the lot is finished processing ($t_0 = 0.0$) the lot is send via channel b.

By simulation, we determine that the machine is down for 7.5% of the time. The mean actual time between failure is 132 hours. Note that, though the mean process time between failure is 100.0 hours and the mean repair time is 10.0 hours, the percentage down is not 10% and the mean actual time between failure is not 110 hours. Note that this caused by the machine's being idle some of the time[4]!

Without machine failure, the mean flowtime is 6.5 hours. With machine failure, the mean flowtime is 14.2 hours. Again the increase in variability is the main contributor to the

[4]In case of exponential failure and repair times, we can relatively easy determine the flowtime and percentage downtime by probability calculations. For such calculations, one is referred to [Lew95] or [Ros00]

increase in flowtime.

3. Failure after processing for an amount of time, restart processing

We reconsider the machine from the previous example. However, in this case the machine has to restart processing from the beginning after repair, the specification becomes:

```
proc M(chan? lot a; chan! lot b):
   dist real dt0 = gamma(1.0 / (1.0 * 1.0), 3.0 / (1.0 * 1.0)),
             dtf = exponential(100.0),
             dtr = exponential( 10.0);
   timer t0, tf;
   real remaining_f, pt0;
   lot x;

   remaining_f = sample dtf;
   while true:
      a?x;
      pt0 = sample dt0;
      tf  = timer(remaining_f);
      while true:                                                Chi-9.4
         t0 = timer(pt0);
         select
            ready(t0): break
         alt
            ready(tf):
               delay sample dtr;
               tf = timer(sample dtf)
         end
      end;
      remaining_f = real(tf);
      b!x
   end
end
```

By simulation, we determine a slight increase in downtime: 7.8%, a slight decrease in the mean actual time between failure: 129 hours and a substantial increase in mean flowtime: 20.2 hours. Because, the machine has to restart processing after repair, the percentage idle time slightly decreases, causing a slight increase in the number of failures per time unit, that is, a slight increase in percentage downtime and a slight decrease in mean actual time between failure. This all together causes a small increase in utilization, leading to a substantial increase in mean flowtime.

4. Failure after being up for an amount of time

In this last case, we consider a machine that fails after being up for an amount of time. On average, the machine fails after being up for 100.0 hours. Repair, on average, takes 10.0 hours. The mean uptime between failure and the repair time are exponentially distributed. The machine can break down while waiting for a lot, while processing a lot, and while trying to send the lot to the next process.

A possible specification of machine M is:

Chi-9.5

```
proc M(chan? lot a; chan! lot b):
   dist real dt0 = gamma(1.0 / (1.0 * 1.0), 3.0 / (1.0 * 1.0)),
             dtf = exponential(100.0),
             dtr = exponential( 10.0);
   timer tf, t0;
   real remaining_t0;
   lot x;

   tf = timer(sample dtf);
   while true:
      # Receive a product
      while true:
         select
            ready(tf):
               delay sample dtr;
               tf = timer(sample dtf)
         alt
            a?x:
               break
         end
      end;
      # Process it
      t0 = timer(sample dt0);
      while true:
         select
            ready(tf):
               remaining_t0 = real(t0);
               delay sample dtr;
               tf = timer(sample dtf);
               t0 = timer(remaining_t0)
         alt
            ready(t0):
               break
         end
      end
      # Send a product
      while true:
         select
            ready(tf):
               delay sample dtr;
               tf = timer(sample dtf)
         alt
            b!x:
               break
         end
      end;

   end
end
```

We use the variables τ_f and τ_0 to represent the time instants at which the machine fails or finishes processing a lot respectively. As long as the machine is idle, the machine is waiting for a lot via channel a. While waiting for a lot, the machine may fail. Upon failure, the machine is repaired, and time instant τ_f of the next failure is determined. After a lot has been received, time instant τ_0 is determined at which the machine is finished processing the lot. As long as the lot is not finished processing ($finshd = \mathsf{false}$), the machine may break down. Upon failure, the machine is repaired and the time instant at which the lot is finished is increased by the repair time ($\tau_0 := \tau_0 + t_\mathrm{r}$). After the lot is finished, the machine tries to send the lot to the next process. While trying to send the lot, the machine may again fail.

Chapter 10

Case: (Part of) a waferfab

In previous chapters we have presented the main building blocks for modelling and analysis of manufacturing systems. With these building blocks we can practically model any manufacturing system. In this and the subsequent chapter we illustrate the use of Chi by means of two industrial cases. Both cases involve the manufacturing of chips. Most electronic devices contain one or more integrated circuits (chips). Figure 10.1(a) shows a PC ROM chip. Chips are produced on wafers. Figure 10.1(b) shows such a wafer.

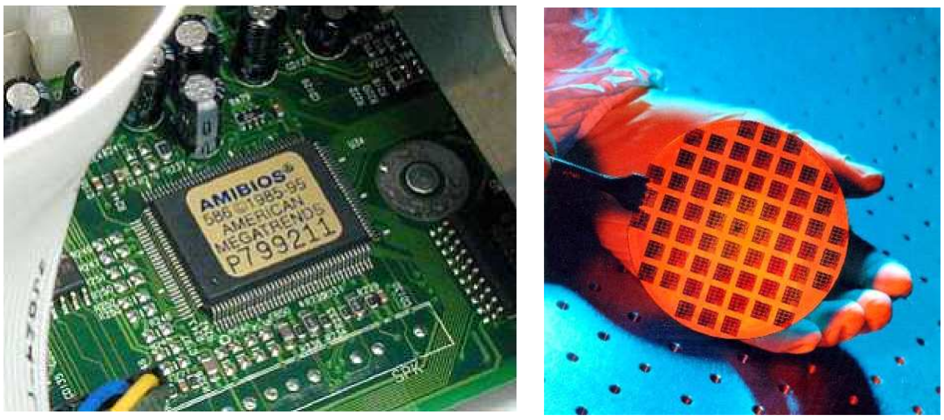


Figure 10.1: (a) ROM chip in PC, (b) Multiple chips (dies) on a wafer

The first case discusses a strongly simplified, but highly illustrative, analysis of a small part of a wafer fab. The second case discusses a more detailed and extensive model of a larger part of a wafer fab.

The cases are distilled from actual research conducted by the Systems Engineering group at Philips and ASML. For reasons of confidentiality, numbers are fictitious.

10.1 The manufacturing of Integrated Circuits (ICs)

Manufacturing a chip is a process involving hundreds of steps and requiring from a few days to up to three months of processing time to complete. Figure 10.2 shows the main steps in making chips.

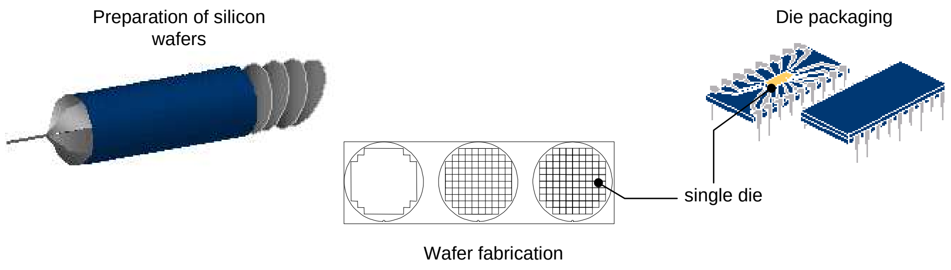


Figure 10.2: Main steps in chip making

Raw silicon is first extracted from common sand. A series of chemical steps refine it until the purity level reaches 99.9999999 percent [Int04]. Silicon ingots are then ”grown” from the purified molten silicon and sliced into round wafers ranging from 150mm to 300mm. On this wafer we create multiple chips (dies). Each die is made up of many layers, each of which contains a detailed pattern. The wafer manufacturing process consists of forming this sequence of layers very precisely. The wafer fabrication is discussed in more detail in the next section. When all layers of the wafer are ready, the microscopic circuitry of each microprocessor is tested. Then the wafer is cut with a diamond saw, separating the dies. Each die is then inserted into a protective package which allows it to connect to other devices. In the chip industry, the wafer processing is called the ’front-end’ and the die packaging is called the ’back-end’.

10.2 Wafer processing

The wafer processing takes a large number (typically over 400) of various operations. In this section we give a brief impression of the main steps. For more detailed information we refer to [Wol90]. Figure 10.3 shows the first set of operations performed on the wafer.

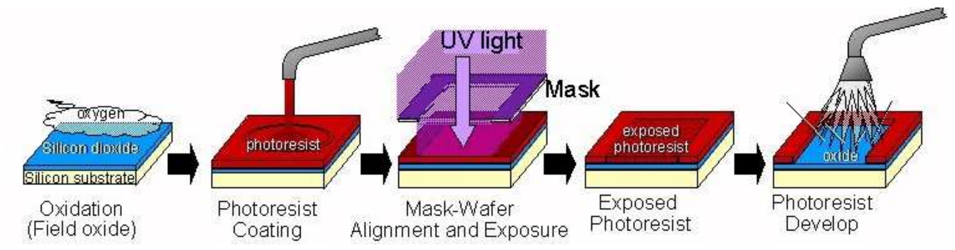


Figure 10.3: First operations in wafer processing (source: American Naval Acadamy)

First, a layer of silicon dioxide is grown by exposing it to gas and extreme heat. The wafer is then coated with a substance called photoresist. Photoresist becomes soluble when exposed to ultraviolet light. In the photolithography process, ultraviolet light is passed through a patterned mask, onto the silicon wafer. The mask protects parts of the wafer from the light. The light turns the exposed areas into a gooey layer of photoresist. Each layer on the microprocessor uses a mask with a different pattern. The gooey photoresist is completely dissolved by a solvent. This reveals a pattern of photoresist made by the mask on the silicon dioxide.

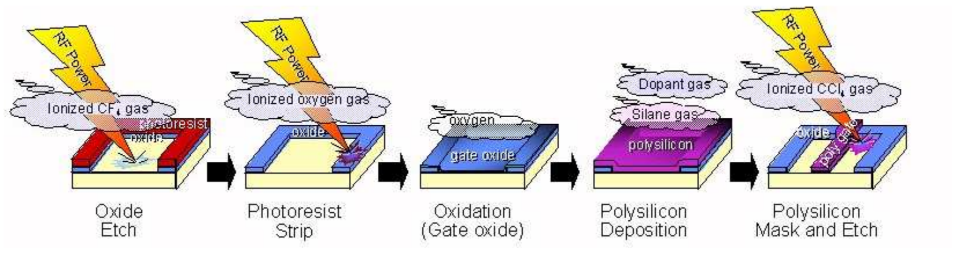


Figure 10.4: Second series of operations in wafer processing (source: American Naval Acadamy)

The revealed silicon dioxide is etched away with chemicals. Subsequently, the remaining photoresist is removed. Then a new layer of silicon dioxide is grown and a layer of polysilicon is applied. With another photolithography process a part of the polysilicon is maintained. In a similar way several layers with specific properties are added to the wafer. Sometimes parts of the wafer are doped with ions, sometimes parts of the wafer are exposed to specific gases, or sometimes metals are deposited on (parts) of the water.

Figure 10.5 shows a simplified flowchart of the processes in the IC manufacturing process.

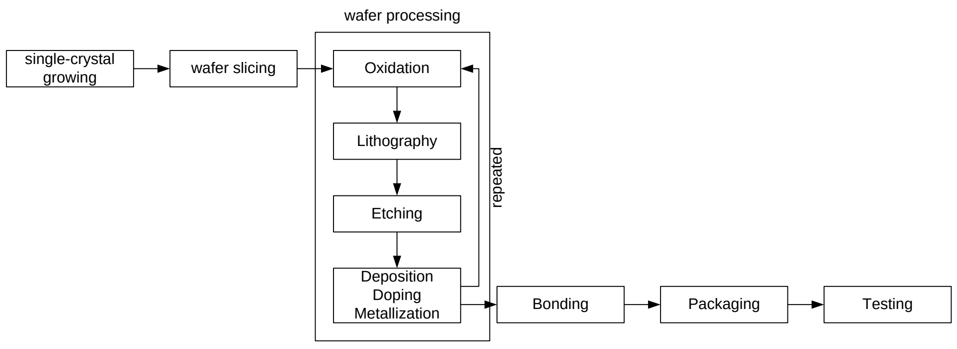


Figure 10.5: Flowchart of processes in IC manufacturing

The processing of wafers shows a typical re-entrant flow: a single wafer revisits a single area. Figure 10.6 shows an example wafer fab layout. (The diffusion area contains the oxide furnaces.) Each area has a stocker. Operators take lots that are to be processed from the stocker and put lots that are ready processing back into the stocker. An AGV system moves lots from one stocker to another.

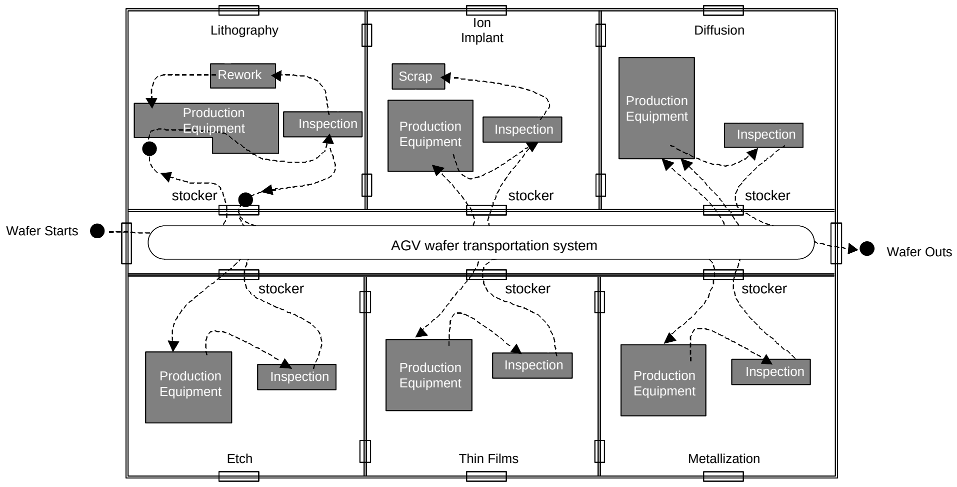


Figure 10.6: Example layout of a waferfab

In this case study, we focus on the oxide furnaces in the diffusion area and the lithographic equipment in the lithography area.

10.3 The oxide furnace and lithographic equipment

Figure 10.7(a) shows the furnace bay of a wafer fab (the Philips SMST facility in Böblingen) and a single furnace. We see that about 100 wafers are processed simultaneously in the furnace. Figure 10.7(c) shows the lithography area of a wafer fab (the Crolles 2 facility in Grenoble).

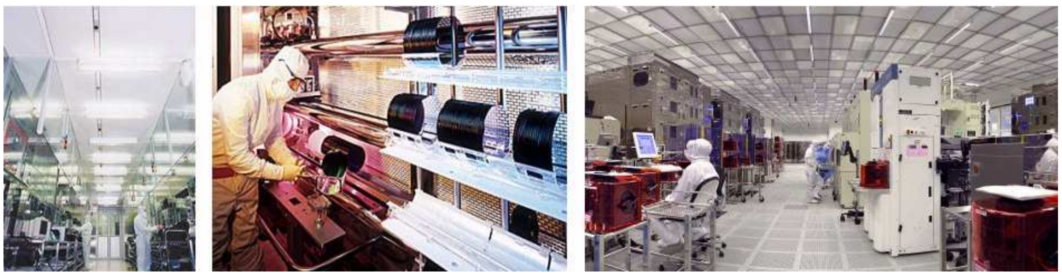


Figure 10.7: (a) Furnace bay (source: Micron Technologies), (b) Single (horizontal) furnace (source: Micron Technologies), and (c) Lithographic area (source: Crolles2 waferfab)

Wafers move through the factory in wafer pods. In Figure 10.7(c) a wafer pod is encircled. From now on, we refer to a wafer pod as a lot, since it is the main transport unit in the factory. A single lot can contain up to 25 wafers. (The number of wafers can be smaller if a customer order requires less than 25 wafers.) A lithographic machine processes lot for lot. The oxide furnace processes multiple lots (typically more than 25 wafers) simultaneously.

In other words, the lithographic machine is a single-lot machine and the furnace is a batch machine.

We wish to investigate the following:

> What is the effect of different batch sizes on the throughput, flowtime, and flowtime variability of the lots.

For this case we consider the lithography and furnace area only. Moreover, we assume that each area contains a single machine. This study does not yield a flowtime estimation for a real wafer fab, but it does provide us with valuable insight on the influence of the batch size on the performance of the most critical resource of the wafer fab: the lithography area. Figure 10.8 shows a schematic representation of the 'minifab'.

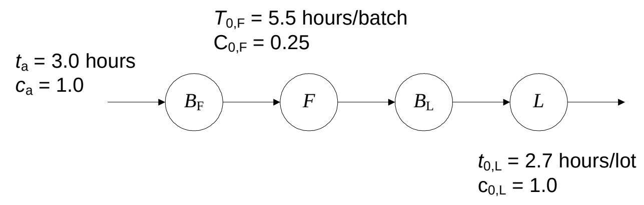


Figure 10.8: Schematic representation of the minifab

We assume an infinite buffer B_F for furnace F and an infinite buffer B_L for lithographic machine L. Moreover, we assume the following characteristics (see also Figure 10.8): lots arrive with a mean inter-arrival time $t_a = 3.0$ hours with coefficient of variation $c_a = 1.0$ hours. The furnace processes batches of fixed size with process time $T_{0,F} = 5.5$ hours/batch and coefficient of variation $C_{0,F} = 0.25$ hours. The lithographic machine processes lots and has process time $t_{0,L} = 2.7$ hours/lot and $c_{0,F} = 1.0$ hours. Note that we use capital letters for quantities that are related to batches.

In the sequel, we first investigate the flowtime of lots in this minifab analytically, and then by means of simulation.

10.4 Analytical approach

For analytically determining the flowtime of the minifab we consider the queueing system in Figure 10.9.

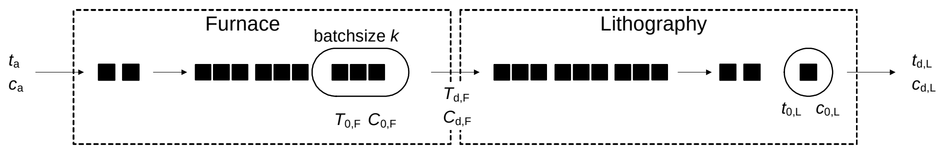


Figure 10.9: Queueing system representation of the minifab

At the furnace, lots arrive with a mean inter-arrival time of t_a hours. The lots wait to form a batch of size k and batches queue up before the batch machine. The furnace processes batches. The mean process time is $t_{0,\mathrm{F}}$ hours/batch. Batches leave the furnace with mean inter-departure time $t_{\mathrm{d,F}}$ hour and arrive at the lithographic machine. Here a queue of batches arises. The first batch in line is broken down lot for lot, as the lithographic machine process one lot at a time. The mean process time is $t_{0,\mathrm{L}}$ hours/lot. After processing, lots leave the machine with mean inter-departure time $t_{\mathrm{d,L}}$.

Please note that t_a, $t_{0,\mathrm{L}}$, and $t_{\mathrm{d,L}}$ are related to single lots and that $T_{0,\mathrm{F}}$ and $T_{\mathrm{d,F}}$ are related to batches!

For the flowtime calculation of the flowtime in the furnace area we use the equations from Section 3.5. We get:

$$\varphi_\mathrm{F} = \frac{k-1}{2} \cdot t_\mathrm{a} + \frac{c_\mathrm{a}^2/k + C_{0,\mathrm{F}}^2}{2} \cdot \frac{u_\mathrm{F}}{1-u_\mathrm{F}} \cdot T_{0,\mathrm{F}} + T_{0,\mathrm{F}} \quad \text{with } u_\mathrm{F} = \frac{T_{0,\mathrm{F}}}{k \cdot t_\mathrm{a}} \tag{10.1}$$

Here the first term represents the wait-to-batch-time, the second term represents the queueing time of a batch, and the third term represents the process time of the batch.

The coefficient of variation on the inter-departure time $T_{\mathrm{d,F}}$ is determined with Kuehn's linking equation.

$$C_{\mathrm{d,F}}^2 = u_\mathrm{F}^2 \cdot C_{0,\mathrm{F}}^2 + (1-u_\mathrm{F}^2) \cdot C_\mathrm{a}^2 = u_\mathrm{F}^2 \cdot C_{0,\mathrm{F}}^2 + (1-u_\mathrm{F}^2) \cdot c_\mathrm{a}^2/k \tag{10.2}$$

Note that this coefficient on the inter-departure time relates to batches. Hence, we require the coefficient of variation on the inter-arrival time related to batches (C_a^2). This coefficient of variation can be determined from the inter-arrival time of lots: $C_\mathrm{a}^2 = c_\mathrm{a}^2/k$, see Section 3.5.

For the litho machine, the flowtime also consists of three terms:

$$\begin{aligned}
\varphi_\mathrm{L} &= \frac{C_{\mathrm{d,F}}^2 + C_{0,\mathrm{L}}^2}{2} \cdot \frac{u_\mathrm{L}}{1-u_\mathrm{L}} \cdot T_{0,\mathrm{L}} + \frac{k-1}{2} \cdot t_{0,\mathrm{L}} + t_{0,\mathrm{L}} && (10.3)\\
&= \frac{k \cdot C_{\mathrm{d,F}}^2 + c_{0,\mathrm{L}}^2}{2} \cdot \frac{u_\mathrm{L}}{1-u_\mathrm{L}} \cdot t_{0,\mathrm{L}} + \frac{k-1}{2} \cdot t_{0,\mathrm{L}} + t_{0,\mathrm{L}} && (10.4)\\
&= \frac{C_{\mathrm{d,F}}^2 + c_{0,\mathrm{L}}^2/k}{2} \cdot \frac{u_\mathrm{L}}{1-u_\mathrm{L}} \cdot k \cdot t_{0,\mathrm{L}} + \frac{k-1}{2} \cdot t_{0,\mathrm{L}} + t_{0,\mathrm{L}} \quad \text{with } u_\mathrm{L} = \frac{t_{0,\mathrm{L}}}{t_\mathrm{a}} && (10.5)
\end{aligned}$$

The first term represents the queueing time of the batches. The coefficient of variation on the process time for a single lot $c_{0,\mathrm{L}}$ can be related to the process time for k lots: $C_{0,\mathrm{L}}^2 = c_{0,\mathrm{L}}^2/k$. Moreover, the mean process time for k lots $T_{0,\mathrm{L}} = k \cdot t_{0,\mathrm{L}}$. The second term represents the breaking down of the batch. When a batch is first in line, not all its lots are immediately processed. The first lot in the batch is processed right away, the litho machine starts processing the second lot after $t_{0,\mathrm{L}}$, the third lot after $2 \cdot t_{0,\mathrm{L}}$, and the k-th lot after $(k-1) \cdot t_{0,\mathrm{L}}$. The average wait-for-batch-breakdown time is $(k-1)/2 \cdot t_{0,\mathrm{L}}$. The third, and last term, is the process time for the lot on the litho machine.

For the given specifications ($t_\mathrm{a} = 3.0, c_\mathrm{a} = 1.0, T_{0,\mathrm{F}} = 5.5, C_{0,\mathrm{F}} = 0.25, t_{0,\mathrm{L}} = 2.7$, and $c_{0,\mathrm{L}} = 1.0$) we investigate the influence of batch size k on flowtime φ and coefficient variation on the inter-departure time $C_{\mathrm{d,F}}$ for the furnace area. Figure 10.10 shows how the flowtime and the coefficient of variation on the inter-departure time vary with k.

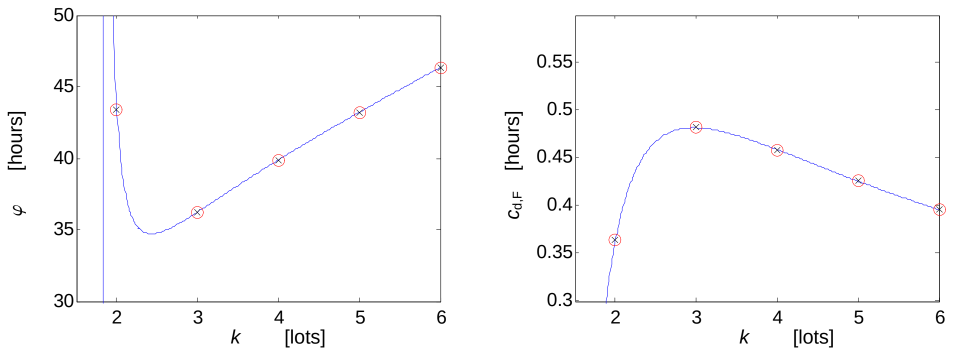


Figure 10.10: (a) Mean flowtime and (b) coefficient of variation on inter-departure time for different batch sizes

We see that for batch size 1 the flowtime grows to infinity: the capacity is insufficient for this batch size. Small batches have a longer queueing time. Large batches require a larger wait-to-batch-time. For the given parameters, we see that the flowtime is minimal for 3 lots. However, we see that the variation on the inter-departure time for $k = 3$ lots is significantly higher, leading to less predictable delivery dates.

10.5 Simulation-based analysis

In the previous section we have applied the analytical approaches from Chapter 3 to investigate the influence of the batch size on the flowtime. In this section we will reinvestigate the same minifab, but with simulation.

Figure 10.11 shows a graphical representation of the minifab system.

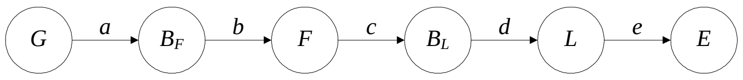

$$\begin{tikzcd} G \arrow[r, "a"] & B_F \arrow[r, "b"] & F \arrow[r, "c"] & B_L \arrow[r, "d"] & L \arrow[r, "e"] & E \end{tikzcd}$$

Figure 10.11: Graphic representation of model Minifab

We define two types: lots and batches.

```
type lot = real,
     bat = list lot;
```

Chi-10.1

Lots are represented by an id number and the time at which they are released in the line. Batches are represented by a list of lots.

Process G generates lots according to the predefined arrival pattern. We specify t_a and c_a as parameters.

```
proc G(chan! lot a; real ta,ca2):
   dist real d = gamma(1.0 / ca2, ca2 * ta);
   while true:
      a!time; delay sample d
   end
end
```

Chi-10.2

Buffer B_F receives lots and sends batches of fixed size k.

```
proc BF(chan? lot a; chan! bat b; int k):
   bat xs; lot x;
   while true:
      select
         a?x:
            xs = xs + [x]
      alt
         size(xs) >= k, b!xs[:k]:
            xs = xs[k:]
      end
   end
end
```

Chi-10.3

The buffer can always receive new lots and puts them in a list. If this list contains at least one complete batch (len(xs) $>=$ k), the buffer tries to send a batch. After sending the batch, the batch is removed from the list of lots.

Furnace F processes batches. The process time is gamma distributed with mean t_0 and coefficient of variation c_0.

```
proc F(chan? bat a; chan! bat b; real T, C2):
   dist real d = gamma(1 / C2, C2 * T);
   bat xs;
   while true:
      a?xs; delay sample d; b!xs
   end
end
```

Chi-10.4

Buffer B_L in front of the litho machine, receives batches and sends single lots to the litho machine.

```
proc BL(chan? bat a; chan! lot b):
   bat xs, ys;
   while true:
      select
         a?ys:
            xs = xs +  ys
      alt
         not empty(xs), b!xs[0]:
            xs = xs[1:]
      end
   end
end
```

Chi-10.5

Note that adding the batch xs to the total list of lots xss requires no square brackets: both xs and xss are lists. The litho machine is a regular machine with a gamma distributed process time.

```
proc L(chan? lot a; chan! lot b; real t, c2):
   dist real d = gamma(1 / c2, c2 * t);
   lot x;
   while true:                                        Chi-10.6
      a?x; delay sample d; b!x
   end
end
```

Exit E receives finished lots and calculates the mean throughput and mean flowtime. These measures are printed to screen every 10000-th lot.

```
proc E (chan? lot a):
   int i; real mft; lot x;
   while true:
      a?x; i = i + 1; mft = (i - 1) / i * mft + (time - x) / i;
      if i mod 10000 == 0:                            Chi-10.7
         write("%10.2f\tmpt = %5.2f\tmft = %5.2f\n", time, i / time, mft)
      end
   end
end
```

Finally, we define parametrized model $Minifab$. In this way we can easily adapt parameters of the model.

```
model Minifab(real ta, ca, TOF, COF, tOL, cOL; int k):
   chan lot a;
   chan bat b, c;
   chan lot d, e;
   run G(a, ta, ca * ca),                             Chi-10.8
       BF(a, b, k), F(b, c, TOF, COF * COF),
       BL(c, d)   , L(d, e, tOL, cOL * cOL),
       E(e)
end
```

Simulating for the given parameters (see previous section) and $k = 3$ gives:

```
30185.8 mtp = 0.331282  mft = 41.8421
60203.9 mtp = 0.332204  mft = 38.2607
90001.4 mtp = 0.333328  mft = 38.5445
...
1.37959e+06     mtp = 0.333433  mft = 38.0518
1.40924e+06     mtp = 0.333514  mft = 38.0234
...
2.10035e+07     mtp = 0.333278  mft = 37.4706
2.10336e+07     mtp = 0.333276  mft = 37.4739
```

We see that the mean throughput is 0.333 lots/hour, which means we have a stable system, where all lots entering the system ($t_a = 3.0$ hours) leave the system. The mean flowtime converges around 37.1 hours. Simulating for the same settings again, yields a slightly different outcome: 37.5 hours. Figure 10.12 shows how the analytically determined flowtime and coefficient of variation on the inter-departure time (line marked with o's) along with the outcome of a number of simulation runs (2 million lots) marked with x's.

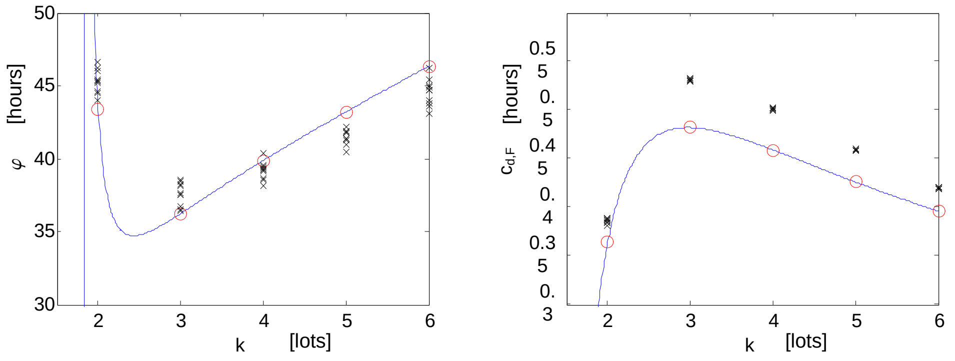
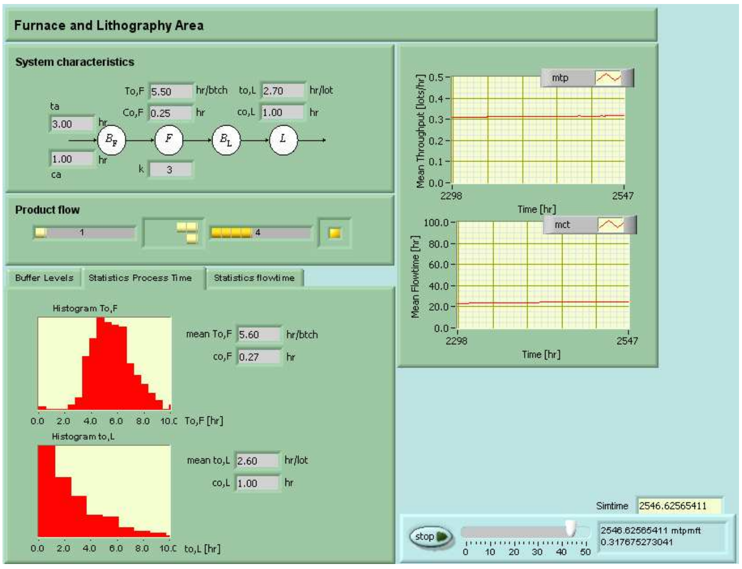


Figure 10.12: (a) Mean flowtime and (b) coefficient of variation on inter-departure time for different batch sizes determined analytically (line marked by o's) and by simulation (marked by x's)

Using Python [Pyt04] we can visualize the simulation output in for example Labview [Nat04, Ver02]. Figure 10.13 shows a screendump of the visualisation in Labview.


Furnace and Lithography Area

System characteristics

ta 3.00 hr

ca 1.00 hr

To,F 5.50 hr/btch

Co,F 0.25 hr

to,L 2.70 hr/lot

co,L 1.00 hr

k 3

Product flow

Buffer Levels | Statistics Process Time | Statistics flowtime

mean To,F 5.60 hr/btch

co,F 0.27 hr

mean to,L 2.60 hr/lot

co,L 1.00 hr

Simtime 2546.62565411

2546.62565411 mtpmft
0.317675273041

Figure 10.13: Labview visualisation of simulation output

Bibliography

[AHR12] I.J.B.F. Adan, A.T. Hofkamp, and J.E. Rooda. *Chi 3 tutorial*, 2012.

[AN98] W.T.M. Alberts and G. Naumoski. *A Discrete-Event Simulator for Systems Engineering.* PhD thesis, Eindhoven University of Technology, Department of Mechanical Engineering, The Netherlands, 1998.

[AR01] I. Adan and J. Resing. Queueing theory. Lecture notes, Eindhoven University of Technology, Department of Mathematics and Computing Science, Eindhoven, The Netherlands, 2001.

[Are96] N.W.A. Arends. *A Systems Engineering Specification Formalism.* PhD thesis, Eindhoven University of Technology, Department of Mechanical Engineering, The Netherlands, 1996.

[Ban98] J. Banks. *Handbook of Simulation.* John Wiley and Sons, New York, 1998.

[BFS87] P. Bratley, B.L. Fox, and L.E. Schrage. *A guide to simulation.* Springer-Verlag, New York, 1987.

[BK00] V. Bos and J.J.T. Kleijn. Formalisation of a production systems modelling language: the operational semantics of χ core. *Fundamenta Informaticae*, 41(4):367–392, 2000.

[Bro03] Brooks-PRI Automation. *Automod*, 2003. `http://www.autosim.com`.

[Bru97] G. Bruns. *Distributed Systems Analysis with CCS.* International series in computer science. Prentice Hall, New York, 1997.

[BS93] J.A. Buzacott and J.G. Shantikumar. *Stochastic models of manufacturing systems.* Prentice Hall, New York, 1993.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, Cambridge, 1990.

[Cac02] Caci Products Company. *Simscript*, 2002. `http://www.caciasl.com`.

[CAJ01] R.B. Chase, N.J. Aquilano, and F.R. Jacobs. *Production and Operations Management.* Irwin, Chicago, 2001.

[Fle02] Flexsim. *Flexsim*, 2002. `http://www.flexsim.com`.

[Fok00] W.J. Fokkink. *Introduction to Process Algebra.* Springer-Verlag, 2000.

[Gan19] H.L Gantt. Organizing for work. *Industrial Management*, 58:89 – 93, August 1919.

[Ger94] S.B. Gershwin. *Manufacturing Systems Engineering.* Prentice Hall, Englewood Cliffs, 1994.

[GH85] D. Gross and C.M. Harris. *Fundementals of queueing theory.* John Wiley and Sons, New York, second edition, 1985.

[Hof01] A.T. Hofkamp. *Reactive machine control. A simulation approach using χ.* PhD thesis, Eindhoven University of Technology, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2001.

[HR12] A.T. Hofkamp and J.E. Rooda. *Chi 3 Reference Manual*, 2012. http://chi.se.wtb.tue.nl/reference_manual/reference_manual.html.

[HS01] W.J. Hopp and M.L. Spearman. *Factory Physics.* McGraw-Hill, Boston, second edition, 2001.

[Ima02] Imagine that. *Extend*, 2002. `http://www.imaginethatinc.com`.

[Int04] Intel. How chips are made. educational website, 2004.

[Ken53] D.G. Kendall. Stochastic processes occuring in the theory of queues and their analysis by the method of the imbedded markov chain. *Ann. Math. Stat*, pages 338–354, 1953.

[Khi32] A. Khinchine. Mathematical theory of stationary queues. In *Mat. Sbornik*, volume 39, pages 73–84, 1932.

[Kin61] J.F.C. Kingman. The single server queue in heavy traffic. In *Proc. of the Cambridge Philosophical Society*, volume 57, page 902, 1961.

[Kol33] A.N. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione (in italian). In *Giornale dell Istituto Italiano degli Attuari*, volume 4, pages 83–91, 1933.

[Kue79] P.J. Kuehn. Approximate analysis of general queueing networks by decomposition. *IEEE Trans. Comm.*, 27:113–126, 1979.

[Lan02] Lanner Group. *WITNESS*, 2002. `http://www.lannergroup.com`.

[Lee99] L. Leemis. Simulation input modelling. In P.A. Farrington, H.B. Nembhand, D.T. Sturrock, and G.W. Evans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 14–23, 1999.

[Lef03] A.A.J. Lefeber. Iteratieve berekening van μ en σ (in dutch). Personal communication, 2003.

[Lew95] E.E. Lewis. *Introduction to reliability engineering.* John Wiley and Sons, New York, second edition, 1995.

[Lit61] J.D.C. Little. A proof of the queueing formula $l = \lambda w$. *Operations Research*, 9:383–387, 1961.

[LK00] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, 2000.

[Man03] Manugistics Inc. *Statgraphics*, 2003. `http://www.statgraphics.com`.

[Mat03] Mathworks inc. *Matlab release 13*, 2003. `http://www.mathworks.com`.

[MR02] D.C. Montgomery and G.C. Runger. *Applied Statistics and Probability for Engineers.* Wiley, New York, third edition, 2002.

[Nat04] National Instruments. Labview, 2004. `http://www.ni.com`.

[Pea00] K. Pearson. On a criterion that a given system of deviations from the probable in the case of a correlated system of variables in such that it can be reasonably supposed to have arisen in random sampling. In *Phil. Mag. (5)*, volume 50, pages 157–175, 1900.

[Pol30] F. Pollaczek. Uber eine Aufgabe der Wahrscheinlichkeitstheorie. In *I-II Math. Zeitschrift*, volume 32, pages 64–100, 729–750, 1930. (in German).

[Pyt04] Python website, 2004. `http://www.python.org/`.

[Ros00] S.M. Ross. *Introduction to probability models.* Academic press, London, 2000.

[RT98] R.S. Russel and B.W. Taylor. *Operations management.* Prentice Hall, Upper Saddle River, 1998.

[SAS03] SAS. *SAS*, 2003. `http://www.sas.com`.

[Smi33] N. Smirnov. Estimate of deviation between empirical distribution functions in two independent samples. In *Bulletin Moscow University*, volume 2, pages 3–16, 1933.

[SPS03] SPSS inc. *SPSS*, 2003. `http://www.spss.com`.

[Ste74] M.A. Stephens. Edf statistics for goodness of fit and some comparisons. In *Journal American Statistical Association*, volume 69, pages 730–737, 1974.

[Tij94] H.C. Tijms. *Stochastic Models, An algorithmic approach.* John Wiley & Sons, Chichester, 1994.

[vdMFR95] J.M. van de Mortel-Fronczak and J.E. Rooda. Application of concurrent programming to specification of industrial systems. In *Proceedings of INCOM'95*, pages 421–426, Beijing, China, October 1995.

[Ver02] J. Vervoort. Visualisation of chi simulation output using labview. Technical report, Eindhoven University of Technology, Eindhoven, 2002.

[Wol90] S. Wolf. *Silicon processing.* Lattice press, Sunset Beach, California, 1990.

Selected answers

	Section 2.3
1a)	$\delta_{M_1} = 0.9\lambda$, $\delta_{M_3} = 0.9\lambda$, $\delta_{M_4} = \lambda$
b)	$\delta_{M_2} = 0.225\lambda$
c)	$u_1 = 1.8\lambda$, $u_2 = 1.35\lambda$
	$u_3 = 1.62\lambda$, $u_4 = 1.6\lambda$
d)	$\lambda_{\max} = 0.56$ lots/hour
2a)	$\delta_{\max} = 1.0$ lots/hour
b)	$\varphi = 5.8$ hours
c)	$w = 5.8$ lots
3a)	$\varphi = 0.31$ hours
b)	6,9, and 4 machines
c)	workstation 1 and 2
4a)	$\delta_{\max} = \frac{1}{3}$ lots/hour
b)	B_2 fills up
c)	$\delta_{\max} = \frac{1}{3}$ lots/hour
d)	B_1 fills up
5a)	$u_1 = 4\lambda$, $\delta_{\max} = \frac{1}{4}$ lots/hour
b)	$\overline{\varphi} = \frac{28}{3}$ hours
c)	$\overline{w} = 28/15 = 1.87$ lots
6a)	$u_{\text{SL}} = 2\lambda$, $u_{\text{B}} = \frac{8}{5}\lambda$
	$\delta_{\max} = \frac{1}{2}$ lots/hour
b,c)	$\overline{\varphi} = 14$ hours
d)	$\overline{w} = 7$ lots
e)	No
f)	$\delta_{\max} = 0.4$ lots/hour, $u_{\text{SL}} = 0.8$
g,h)	$\overline{\varphi} = 15.75$ hours
7a)	$u_1 = 2.4\lambda$, $u_2 = 2.5\lambda$
b)	$\delta_{\max,\text{A}} = 0.16$ prodA/hr
	$\delta_{\max,\text{B}} = 0.24$ prodB/hr
d)	$\varphi = 6$ hours
e)	$\overline{w} = 2.4$ prod
f)	$\overline{w}_{\text{A}} = 0.96$ prodA, $\overline{w}_{\text{B}} = 1.44$ prodB
	Section 2.4
1a)	$\delta_{W_2} = \lambda/0.8$, $\delta_{W_3} = \lambda/0.8$
	$\delta_{W_5} = \lambda/4$
b)	$\delta_{W_4W_5} = \lambda/4$, $\delta_{W_4W_6} = \lambda$
2a)	$\delta_{W_1} = \lambda$, $\delta_{W_2} = 0.8\lambda$
	$\delta_{W_3} = 0.9\lambda$, $\delta_{W_4} = \lambda$
3c)	$\delta_{W_4} = \lambda/0.7 = 1.43\lambda$
d)	$\delta_{W_1} = 1.34\lambda$, $\delta_{W_2} = 1.70\lambda$
	$\delta_{W_3} = 1.79\lambda$
f)	$\lambda_{\max} = 0.56$ lots/hour
4d)	$\delta_{W_1} = 1.46\lambda$, $\delta_{W_2} = 1.05\lambda$
	$\delta_{W_3} = 2.50\lambda$, $\delta_{W_4} = 2.0\lambda$
4f)	$u_{W_1} = 1.46\lambda$, $u_{W_2} = 1.37\lambda$
	$u_{W_3} = 1.75\lambda$, $u_{W_4} = 1.60\lambda$
	Section 3.7
1a)	$u_1 = u_3 = 0.75$, $u_2 = 0.625$
b)	$\varphi_{B_1} = 5.91$ min
c)	$c^2_{a,2} = 1$
d)	$\varphi_{B_2} = 0.97$ min
e)	$\varphi_{B_3} = 9.0$ min
f)	$\overline{w}_{B_1} = 1.48$ lots, $\overline{w}_{B_2} = 0.24$ lots
	$\overline{w}_{B_3} = 2.25$ lots
h)	$\varphi_{\text{tot}} = 63.88$ min
i)	$\overline{w}_{\text{tot}} = 15.97$ lots
2a)	$u_1 = u_3 = \lambda$, $u_2 = 4\lambda$
c)	$u_1 = u_3 = 0.225$, $u_2 = 0.9$
d)	$\varphi_{\text{tot}} = 63.0$ hours
e)	$\overline{w}_{B_1} = 0.20$ lots, $\overline{w}_{B_2} = 12.56$ lots
	$\overline{w}_{B_3} = 0.07$ lots
g)	$\lambda_{\max} = 1.0$ lots/hour
h)	$\varphi_{\text{tot}} = 86.47$ hours
i)	$\varphi_{\text{tot}} = 62.63$ hours
j)	$\overline{w}_I = 77.82$ lots, $\overline{w}_{II} = 56.27$ lots
3a)	$\varphi(k) = 5k + 4.75$
b)	$k_{\varphi_{\min}} = 1$
4a)	$u = 5/6$

b) $\varphi_{\mathrm{bk}} = 0.75$ hours
c) $\varphi_{\mathrm{bq}} = 2.69$ hours
d) $\varphi_{\mathrm{tot}} = 8.44$ hours

Section 4.5

1b) $\delta = 5.0$ [lots/hour]
d) $\delta = 5.0$ [lots/hour]

3b) $\delta = 15.0$ [lots/hour]
c) $\delta_0 = \delta_1 = \delta_2 = 5.0$ [lots/hour]
d) $\delta_0 = \delta_1 = \delta_2 = 5.0$ [lots/hour]

Section 5.9

2 $\overline{\delta} = 0.2$ lots/minute
$\overline{\varphi} = 6.24$ minute
$\overline{u} = 0.988$
$\overline{w} = 1.263$ lots

4 $\overline{\delta} = 5.0$ lots/hour
$\overline{\varphi} = 0.2$ hour
$\overline{u} = 1.0$
$\overline{w} = 1.0$ lots

5a) $\overline{\delta} = 0.25$ lots/min
b) $\overline{\varphi} = 5.69$ min
c) $\overline{\varphi}_1 = 3.63$ min
$\overline{\varphi}_2 = 5.36$ min
$\overline{\varphi}_3 = 8.35$ min
d) $n_B = 0.59$ lots

6a) $\overline{\delta} = 0.24$ lots/hour
b) $\overline{\delta} = 0.249$ lots/hour

Section 6.5

1 T, T, F, T, F
2 T, F, F, F
3 F, F, T, F, F, T

4b) $n = 2.353$ [lots/hour]
c) n(N=1) = 2.490 [lots/hour]
n(N=2) = 2.499 [lots/hour]
n(N=3) = 2.500 [lots/hour]

5a) seconds
b) $\overline{t_0} = 11.4$ s

Section 7.4

2a,b)

machine	δ [lots/hour]	t [hour]
0	0.6	1.5
1	0.2	4.5
2	0.2	4.5
3	0.2	4.5
4	0.4	2.25
5	0.4	2.25
6	0.3	3.0
7	0.3	3.0
8	0.4	2.25

c) $\varphi_{\mathrm{min},0,1,4,5} = 9.0$ hours
$\varphi_{\mathrm{min},2,3} = 6.75$ hours

Section 8.5

1a) $u_{\mathrm{AP}} = 0.8$
d) $\overline{b}_0 = 0.6$ lots, $\overline{b}_1 = 0.7$ lots
$\overline{b}_2 = 1.05$ lots, $\overline{b}_3 = 1.2$ lots

2a) $u_0 = u_2 = 0.75$, $u_1 = 0.625$

3a) $r_{\mathrm{a}} = 2.0$ lots/hour