Chapter 7
Quantitative Process Analysis

It is better to be approximately right than precisely wrong.
Warren Buffett (1930–)

Qualitative analysis is a valuable tool to gain systematic insights into a process. However, the results obtained from qualitative analysis are sometimes not detailed enough to provide a solid basis for decision making. Think of the process owner of BuildIT's equipment rental process who wants to convince the *Chief Operations Officer* (COO) that every site engineer should be given a tablet computer with wireless access in order to query the suppliers' catalogs and to create or modify rental requests from any construction site. The process owner will be asked to substantiate the benefits of this investment in quantitative terms by providing estimates of how the performance of the process will be measurably improved. To make such estimates, we need to go beyond qualitative analysis.

This chapter introduces techniques for analyzing business processes quantitatively in terms of process performance measures such as cycle time, waiting time, cost, and other measures we already discussed in Section 2.3.2. Specifically, the chapter focuses on three techniques: flow analysis, queueing analysis and simulation.

7.1 Flow Analysis

Flow analysis is a family of techniques to estimate the overall performance of a process given some knowledge about the performance of its tasks. For example, using flow analysis we can calculate the average cycle time of an entire process if we know the average cycle time of each task and the probability of taking each flow stemming from a decision gateway. A decision gateway is either an XOR-split or an OR-split. Similarly, we can use flow analysis to calculate the average cost of a process instance knowing the cost-per-execution of each task or to calculate the error rate of a process given the error rate of each task.

M. Dumas et al., *Fundamentals of Business Process Management*,
https://doi.org/10.1007/978-3-662-56509-4_7

In order to understand the scope and applicability of flow analysis, we start by showing how flow analysis can be used to calculate the average cycle time of a process. As a shorthand, we will use the term *cycle time* to refer to *average cycle time* in the rest of this chapter.

7.1.1 Calculating Cycle Time Using Flow Analysis

We recall that the cycle time of a process is the average time it takes between the moment the process starts and the moment it completes. By extension, we say that the cycle time of a task is the average time it takes between the moment the task starts and the moment it completes.

To understand how flow analysis works it is useful to start with an example of a purely sequential process as in Figure 7.1. The cycle time of each task is indicated between brackets. Since the two tasks in this process are performed one after the other, we can conclude that the cycle time of this process is $20 + 10 = 30$ h.

More generally, we can state that the cycle time of a sequential fragment of a process is the sum of the cycle times of the tasks in the fragment. We use T as the set of tasks with an index i and define:

$$CT = \sum_{i=1}^{n} T_i \tag{7.1}$$

When a process model or a fragment of a model contains gateways, the cycle time is on average no longer the sum of the task cycle times. Let us consider the example shown in Figure 7.2. Here, it is clear that the cycle time of the process is

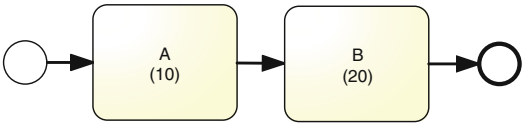

Fig. 7.1 Fully sequential process model (durations of tasks in hours are shown between brackets)

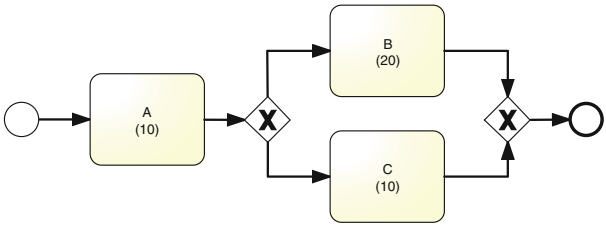

Fig. 7.2 Process model with XOR-block

not 40 (the sum of the task cycle times). Indeed, in a given instance of this process, either task B or task C is performed. If B is performed, the cycle time is 30 h, while if C is performed, the cycle time is 20 h. So we can conclude that the cycle time must be lower than 30 h.

Whether the cycle time of this process is closer to 20 h or closer to 30 h depends on how frequently each branch of the XOR-split is taken. For instance, if in 50% of instances the upper branch is taken and the remaining 50% of instances the lower branch is taken, the overall cycle time of the process is 25 h. On the other hand, if the upper branch is taken 90% of the times and the lower branch is taken 10% of the times, the cycle time should be intuitively closer to 30 h. Generally speaking, the cycle time of the fragment of the process between the XOR-split and the XOR-join is the weighted average of the cycle times of the branches in-between. Thus, if the upper branch has a frequency of 90% and the lower branch a frequency of 10%, the cycle time of the fragment between the XOR-split and the XOR-join is: $0.9 \times 20 + 0.1 \times 10 = 19$ h. We then need to add the cycle time of task A in order to obtain the total cycle time, that is, $10 + 19 = 29$ h. In the rest of this chapter, we will use the term *branching probability* to denote the frequency with which a given branch of a decision gateway is taken.

In more general terms, the cycle time of a fragment of a process model with the structure shown in Figure 7.3 is:

$$CT = \sum_{i=1}^{n} p_i \times T_i \tag{7.2}$$

In Figure 7.3, p_1, p_2, etc. are the branching probabilities. Each cloud represents a fragment that has a single entry flow and a single exit flow. The cycle times of these nested fragments are T_1, T_2, etc. This type of fragment is called a *XOR-block*.

Let us now consider the case where parallel gateways are involved as illustrated in Figure 7.4. Again, we can observe that the cycle time of this process cannot be 40 (the sum of the task cycle times). Instead, since tasks B and C are executed in parallel, their combined cycle time is determined by the slowest of the two tasks, that is, by B. Thus, the cycle time of the process shown in Figure 7.4 is $10 + 20 = 30$ h.

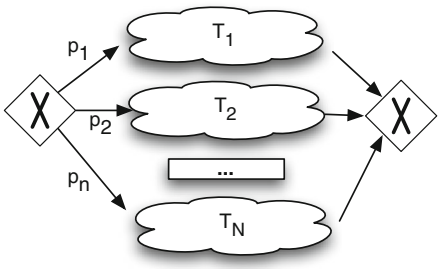

Fig. 7.3 XOR-block pattern

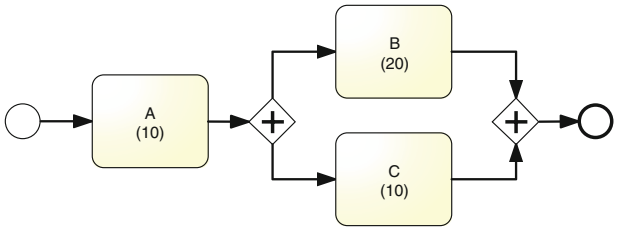

Fig. 7.4 Process model with AND-block

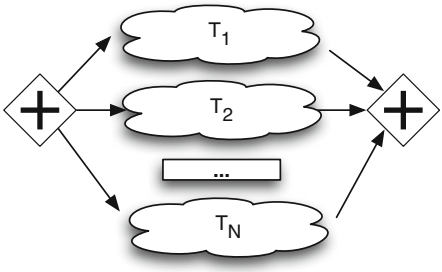

Fig. 7.5 AND-block pattern

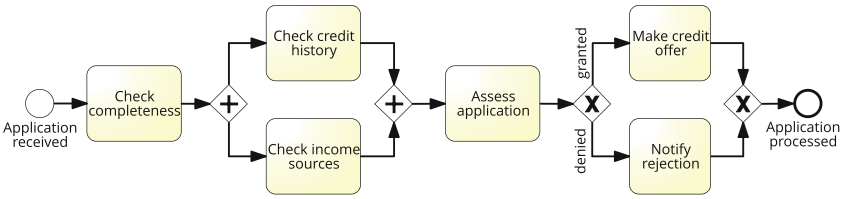

Fig. 7.6 Credit application process

More generally, the cycle time of an *AND-block* such as the one shown in Figure 7.5 is:

$$CT = Max(T_1, T_2, \ldots, T_n) \tag{7.3}$$

Example 7.1 Let us consider the credit application process model in Figure 7.6 and the task cycle times given in Table 7.1. Let us also assume that in 60% of the cases the credit is granted.

To calculate the cycle time of this process, we first note that the cycle time of the AND-block is 3 days (the cycle time of the slowest branch). Next, we calculate the cycle time of the fragment between the XOR-block using Eq. (7.2), that is, $0.6 \times 1 + 0.4 \times 2 = 1.4$ days. The total cycle time is then: $1 + 3 + 3 + 1.4 = 8.4$ days. □

Table 7.1 Cycle times for credit application process

Task	Cycle time
Check completeness	1 day
Check credit history	1 day
Check income sources	3 days
Assess application	3 days
Make credit offer	1 day
Notify rejection	2 days

Exercise 7.1 Consider the process model given in Figure 3.8 (page 86). Calculate the cycle time under the following assumptions:

- Each task in the process takes 1 h on average.
- In 40% of the cases the order contains only Amsterdam products.
- In 40% of the cases the order contains only Hamburg products.
- In 20% of the cases the order contains products from both warehouses.

Compare the process model in Figure 3.8 (page 86) with the one in Figure 3.10 (page 88). Does this comparison give you an idea of how to calculate cycle times for process models with OR gateways?

Another recurrent pattern is the one where a fragment of a process is repeated any number of times. This pattern is illustrated in Figure 7.7. In this figure, the decimal numbers attached to the flows denote the probability that the flow will be taken whenever the XOR-split gateway is reached. Looking at the figure, we can say for sure that task B will be executed once. Next, we can say that task B may be repeated once (i.e., executed a second time) with a probability of 20% (i.e., 0.2), which is the probability of going back from the XOR-split gateway to the XOR-join gateway. If we continue this reasoning, we can conclude that the probability that task B is repeated twice (in addition to the first execution) is $0.2 \times 0.2 = 0.04$. More generally, the probability that task B is repeated N times (in addition to the first execution) is 0.2^N.

If we sum up the cycle time of the first execution of B, plus the cycle times of the cases where B is repeated once, twice, three times, etc., we get the following summation: $\sum_{n=0}^{\infty} 0.2^n$. This is the expected number of executions of task B. If we replace 0.2 with a variable r, this summation is a well-known series, known as the *geometric series*, and it can be shown that this series is equivalent to $1/(1-r)$.

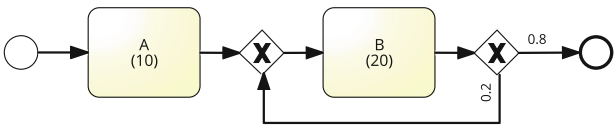

Fig. 7.7 Example of a rework block

Fig. 7.8 Rework pattern

Hence, the average number of times that B is expected to be executed is $1/(1 - 0.2) = 1.25$. Now, if we multiply this expected number of instances of B times the cycle time of task B, we get $1.25 \times 20 = 25$. Thus the total cycle time of the process in Figure 7.7 is $10 + 25 = 35$.

More generally, the cycle time of a fragment with the structure shown in Figure 7.8 is:

$$CT = \frac{T}{1 - r}. \tag{7.4}$$

In this formula, the parameter r is called the *rework probability*, that is, the probability that the fragment inside the cycle will need to be reworked. This type of block is called a *rework block* or *repetition block*.

In some scenarios, a task is reworked at most once. This situation would be modeled as shown in Figure 7.9. Using what we have seen, we can already calculate the cycle time of this example. First, we observe that the cycle time of the fragment between the XOR-split and the XOR-join is $0.2 \times 20 + 0.8 \times 0 = 4$. Here, the zero comes from the fact that one of the branches between the XOR-split and the XOR-join is empty and, therefore, does not contribute to the cycle time. To this, we have to add the cycle time of the preceding tasks, giving us a total cycle time of 34.

In summary, we have seen that the cycle time of a process can be calculated using the following four equations:

- The cycle time CT of a sequence of fragments with cycle times $CT_1, \ldots CT_n$ is the sum of the cycle times of these fragments: $CT = \Sigma_{i=1}^{n} CT_i$.
- The cycle time CT of an XOR-block is the weighted average of the cycle times of its branches (CT_i), using the branching probabilities p_i as the weights: $CT = \Sigma_{i=1}^{n} p_i \times CT_i$.

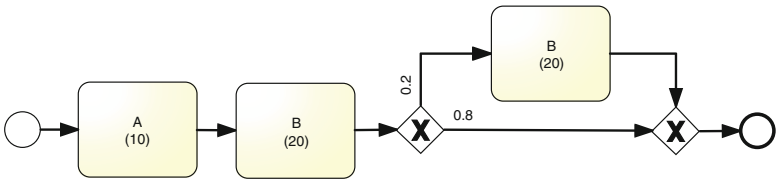

Fig. 7.9 Situation where a fragment (task) that is reworked at most once

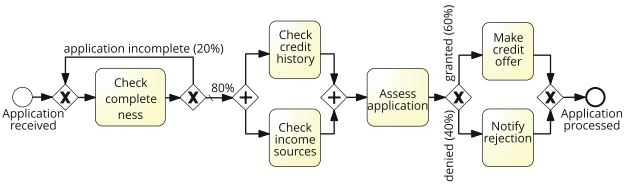

Fig. 7.10 Credit application process with rework

- The cycle time CT of an AND-Block is the cycle time of its slowest branch or, in other words, the maximum of the cycle times of the branches: $CT = Max(CT_1, \ldots CT_n)$.
- The cycle time CT of a rework block with a rework probability r is the cycle time of each iteration of the loop (let us call it CT_b) divided by $(1 - r)$: $CT = CT_b/(1 - r)$.

Example 7.2 Let us consider the credit application process model in Figure 7.10 and the cycle times previously given in Table 7.1. Let us assume that after each execution of "Check completeness", in 20% of the cases the application is incomplete. And let us also assume that in 60% of the cases the credit is granted.

The cycle time of the rework block is $1/(1 - 0.2) = 1.25$ days. The cycle time of the AND-block is 3 days and that of the XOR-block is 1.4 days as discussed in Example 7.1. Thus the total cycle time is $1.25 + 3 + 3 + 1.4 = 8.65$ days. □

7.1.2 Cycle Time Efficiency

The cycle time of a task or of a process can be divided into *waiting time* and *processing time*. Waiting time is the portion of the cycle time where no work is being done to advance the process. Processing time, on the other hand, refers to the time that participants spend doing actual work. In many, if not most processes, a considerable proportion of the overall cycle time is waiting time. Waiting time typically arises when there is a handoff between two participants. In this case, there is usually a waiting time between the moment the first participant finishes its task, and the moment when the next participant starts the next task. This waiting time may become relatively long if the process participants perform their work in batches. For example, in a purchase requisition process, the supervisor responsible for purchase approvals might choose to *batch* all purchase requisitions that arrive during a given day and approve them all at once at the end of the working day. Also, sometimes time is spent waiting for an external party to provide input for a task. For example, in the context of fulfilling a medical prescription, a pharmacist may require a clarification from the doctor. To do so, the pharmacist would try to call the doctor. But the doctor might be unavailable such that the pharmacist needs to put the prescription aside and wait until the doctor returns the call.

When analyzing a process with the aim of addressing issues related to cycle time, it may be useful to start by evaluating the ratio of overall processing time relative to the overall cycle time. This ratio is called *cycle time efficiency*. A cycle time efficiency close to 1 indicates that there is little room for improving the cycle time unless relatively radical changes are introduced in the process. A ratio close to zero indicates that there is a significant amount of room for improving cycle time by reducing the waiting time.

Concretely, the cycle time efficiency of a process is calculated as follows. First, we need to determine the cycle time and the processing time of each task. Given this information, we then calculate the cycle time CT of the process using the four equations we saw above. Next, we calculate the so-called the *Theoretical Cycle Time* (TCT) of the process. The TCT is the average amount of time that a case would take if there was no waiting time at all. The TCT is calculated using the same four equations we introduced above, but instead of using the cycle time of each task, we must use instead the processing time of each task. Once we have calculated the *TCT*, we calculate the Cycle Time Efficiency (CTE) as follows:

$$CTE = \frac{TCT}{CT} \tag{7.5}$$

Example 7.3 Let us consider the credit application process model in Figure 7.10 and the processing times given in Table 7.2. The task cycle times (including both waiting and processing time) are those previously given in Table 7.1. We assume again that in 20% of the cases the application is incomplete and in 60% of the cases the credit is granted. Let us additionally assume that 1 day is equal to 8 working hours.

We have seen in Example 7.2 that the total cycle time of this process is 8.65 days, which translates to 69.2 working hours. We now calculate the theoretical cycle time using the processing times given in Table 7.2. This gives us: $2/(1-0.2)+3+2+0.6\times 2+0.4\times 0.5 = 8.9$ working hours. The cycle time efficiency is thus $8.9/69.2 = 12.9\%$. □

Exercise 7.2 Calculate the overall cycle time, theoretical cycle time, and cycle time efficiency of the ministerial enquiry process introduced in Example 3.7 (page 90). Assume that the rework probability is 0.2 and the waiting times and processing times are those given in Table 7.3.

Table 7.2 Processing times for credit application process

Task	Processing time
Check completeness	2 h
Check credit history	30 min
Check income sources	3 h
Assess application	2 h
Make credit offer	2 h
Notify rejection	30 min

Table 7.3 Task cycle times and processing times for ministerial enquiry process

Task	Cycle time	Processing time
Register ministerial enquiry	2 days	30 min
Investigate ministerial enquiry	8 days	12 h
Prepare ministerial response	4 days	4 h
Review ministerial response	4 days	2 h

Table 7.4 Analysis of cycle times in white-collar processes [21]

Industry	Process	CT	TCT	CTE
Life Insurance	New policy application	72 h	7 min	0.16%
Consumer Packaging	New graphic design	18 days	2 h	0.14%
Commercial Bank	Consumer Loan	24 h	34 min	2.36%
Hospital	Patient Billing	10 days	3 h	3.75%
Automobile Manufacture	Financial Closing	11 days	5 h	5.60%

An important question is in what range we would typically observe cycle time efficiency in practice. Actual measurements are reported in a study by Blackburn from 1992 on white-collar processes, see Table 7.4 and [21]. These measurements appear to be around 5% or lower, which indicates that there are substantial waiting times in many business processes. This is a valuable observation. It implies that cycle times and cycle time efficiency can often be improved by reducing these waiting times. Process-Aware Information Systems and specifically Business Process Management Systems provide several features that are helpful in this regard, as we will see in Chapter 9.

Exercise 7.3 The measurements reported in Table 7.4 stem from 1992. How do you expect that these measurements have changed since then? What role do information systems play in this context?

7.1.3 Critical Path Method

A low cycle time efficiency raises the question of which parts of the process should be improved. In order to answer that question, we need to more precisely understand which tasks contribute to the theoretical cycle time. The *Critical Path Method* (CPM) is a well-known method for addressing this question in the context of project planning. This method can be applied to process models that do not contain decision gateways. This means that if the process model contains XOR or OR gateways, we need to simplify it by removing all such gateways, before we can apply the CPM method. We can do this by either replacing every XOR, OR, and loop block with a single task, or by considering only specific paths of our process and focusing on those ones. For example, we can eliminate the branches of an XOR-split that lead to an early completion of the case as well as those gateways associated with rework loops (like the rework loop in Figure 7.10). Indeed, if we optimize the theoretical

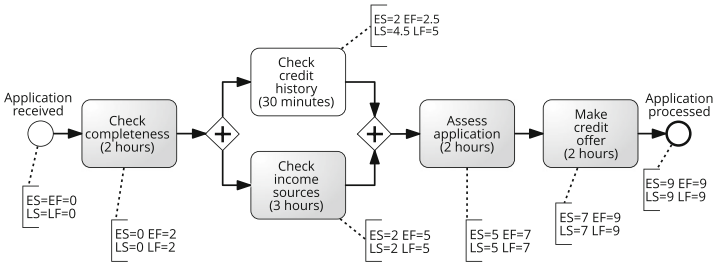

Fig. 7.11 Credit application process without XOR gateways

cycle time of the process without its rework loops, this would contribute to also optimizing the process with the rework loop.

Let us consider again the credit application process without repetition and with a positive credit assessment as shown in Figure 7.11. The theoretical cycle time is determined by the processing time of tasks "Check completeness", "Check income sources", "Assess Application", and "Make credit offer", which all take 2, 3, 2 and 2 h, respectively. These tasks are part of the *critical path* of this process (highlighted in grey). The critical path of a process is the sequence of tasks that determines the theoretical cycle time of the process. When optimizing a process with respect to theoretical cycle time, one should focus the attention on the processing times of the tasks that belong to the critical path.

CPM identifies the critical path based on the notions of *early start (ES)*, *early finish (EF)*, *late start (LS)*, and *late finish (LF)* of each task of the process. Early start and early finish are determined in a forward pass over the process. We start with time zero at the start event. Each task is assigned as early start the early finish time of its predecessor. Its early finish is its early start time plus its processing time. If this predecessor is the entry (split) gateway of an AND-block, it is assigned the early finish time of the preceding fragment. If it is the exit (join) gateway of an AND-block, it is assigned the maximum of the early finish from all its parallel branches. Using this procedure, we can determine at what time each task has to start and finish such that the cycle time is equal to the theoretical cycle time.

Not all tasks are equally critical for finishing the process within the theoretical cycle time. Consider the two verification (check) tasks of the credit application process. They are part of an AND-block. If the more time-consuming task "Check income sources" (3 h) gets delayed, this will delay the process altogether. If "Check credit history" (taking 30 min) is delayed, this will only delay the process if it takes longer than the processing time of the more time-consuming task "Check income sources". For this reason, we also have to determine the late start and late finish of all tasks in a backward pass over the process. Now, we start from the end event with the time set to the theoretical cycle time. For each task, its late finish is assigned the late start of its successor. Its late start is the late finish minus its processing time. We continue our pass from right to left of the process, now taking the earlier time at the entry split an AND-block.

Example 7.4 Let us now apply these calculation steps for the credit application process shown in Figure 7.11 and the processing times given in Table 7.2. We start with calculating the early start and early finish times (ES and EF).

- The start event "Application received" gets zero assigned ($ES = EF = 0$).
- Early start of "Check completeness" is the same as the early finish of its predecessor. This means $ES = 0$. We calculate $EF = ES + processingtime = 2$.
- Each task after the AND-split gets the same ES as the preceding task. Its EF is this ES plus the respective processing time. This means, for "Check credit history" we get $ES = 2$ and $EF = 2.5$ and for "Check income sources" we have $ES = 2$ and $EF = 5$.
- At the AND-split, we have to determine the maximum EF of its preceding tasks. This is $Max(2.5, 5) = 5$.
- The subsequent task "Assess application" gets this maximum as its $ES = 5$. Considering the processing time, its $EF = 7$.
- For "Make credit offer" we get $ES = 7$ and $EF = 9$.
- Therefore, $ES = EF = 9$ holds for the end event "Application processed" and for the overall process.

With this value of $EF = 9$, we start our pass backwards to calculate late start and late finish (LS and LF).

- For the task "Make credit offer", we assign the late finish time from the end event ($LF = 9$) and subtract the processing time to get $LS = 7$.
- In the same way, we first obtain $LF = 7$ and then $LS = 5$ for "Assess application".
- For the tasks preceding the AND-join, we obtain their late finish from the late start of the task after it. Therefore, both "Check credit history" and "Check income sources" have $LF = 5$. We subtract their respective processing times to get $LS = 4.5$ and $LS = 2$.
- At the AND-split, we determine the minimum LS of its successor tasks. This is $Min(4.5, 2) = 2$.
- The preceding task "Check completeness" gets this minimum as $LF = 2$. Considering its processing time, we get $LS = 0$.
- Therefore, we also have $LS = LF = 0$ for the start event.

□

In this example, we observe that the early start and finish times are the same for most tasks. The *critical path* is the set of tasks for which these two values are equal. Those tasks with a late start greater than the early start ($LS > ES$) or late finish greater than early finish ($LF > EF$) have *slack*. This means that even when they start or complete later, it might still be possible to finish the process without delay. This is the case of "Check credit history" in our example. Slack tasks are typically the less time-consuming tasks in AND-blocks.

Exercise 7.4 Consider the process model shown in Figure 7.4 with the processing times indicated in the brackets of the tasks. What is its critical path? How much slack is there on the task that is not on the critical path?

7.1.4 Little's Law

Cycle time is directly related to two measures that play an important role when analyzing a process, namely arrival rate and Work-In-Process (WIP).

The *arrival rate* of a process is the average number of new instances of the process that are created per time unit. For example, in a credit application process, the arrival rate is the number of credit applications received per day (or any other time unit we choose). Similarly, in an order-to-cash process, the arrival rate is the average number of new orders that arrive per day. Traditionally, the symbol λ (lambda) is used to refer to the arrival rate.[1]

The *Work-In-Process* (WIP) is the average number of instances of a process that are active at a given point in time, meaning the average number of instances that have not yet completed. For example, in a credit application process, the WIP is the average number of credit applications that have been submitted and not yet granted or rejected. Similarly, in an order-to-cash process, the WIP is the average number of orders that have been received, but not yet delivered and paid for.

Cycle time (CT), arrival rate (λ), and WIP are related by a fundamental law known as Little's law, which states that:

$$WIP = \lambda \times CT \tag{7.6}$$

In essence, what this law tells us is that:

- WIP increases if the cycle time increases or if the arrival rate increases. In other words, if the process slows down—meaning that its cycle time increases—there will be more instances of the process active at the same time. Also, the faster new instances are created, the higher will be the number of instances in an active state.
- If the arrival rate increases and we want to keep the WIP at current levels, the cycle time must decrease.

Little's law holds for any stable process. By stable, we mean that the number of active instances is not increasing infinitely. In other words, in a stable process, the amount of work waiting to be performed is not growing beyond control.

[1] A related concept is that of *throughout*, which is the average number of instances completed per time unit. In a stable system and over long periods of time, the throughput should be equal to the arrival rate (otherwise it means that we are not able to handle all the workload).

Although simple, Little's law can be an interesting tool for what-if analysis. We can also use Little's law as an alternative way of calculating the total cycle time of a process if we know the arrival rate and WIP. This is useful because determining the arrival rate and WIP is sometimes easier than determining the cycle time. For example, in the case of the credit application process, the arrival rate can be easily calculated if we know the total number of applications processed over a period of time. For example, if we assume there are 250 business days per year and we know the total number of credit applications over the last year is 2,500, we can infer that the average number of applications per business day is 10. WIP on the other hand can be calculated by means of sampling. We can ask how many applications are active at a given point in time, then ask this question again one week later and again two weeks later. Let us assume that on average we observe that 200 applications are active at the same time. The cycle time is then $WIP/\lambda = 200/10 = 20$ business days.

Exercise 7.5 A restaurant receives on average 1,200 customers per day (between 10 a.m. and 10 p.m.). During peak times (12 p.m. to 3 p.m. and 6 p.m. to 9 p.m.), the restaurant receives around 900 customers in total and, on average, 90 customers can be found in the restaurant at a given point in time. At non-peak times, the restaurant receives 300 customers in total and, on average, 30 customers can be found in the restaurant at a given point in time.

- What is the average time that a customer spends in the restaurant during peak times?
- What is the average time that a customer spends in the restaurant during non-peak times?
- The restaurant's premises have a maximum capacity of 110 customers. This maximum capacity is sometimes reached during peak times. The restaurant manager expects that the number of customers during peak times will increase slightly in the coming months. What can the restaurant do to address this issue without investing in extending its building?

7.1.5 Capacity and Bottlenecks

The calculations of Little's law rely on the assumption that the process is stable. In order to assess whether or not this assumption is applicable, we have to know the *theoretical capacity* of the process and the *resource utilization* of the resources involved in the process.[2]

The *theoretical capacity* of a process is the maximum amount of instances that can be completed per time unit given a set of resources. The theoretical capacity is reached when a subset of the resources are working at full capacity (no idle time)

[2]The term *occupation rate* is sometimes used a synonym for resource utilization.

and the other resources cannot help them due to the existing division of labor in the process. When this limit is reached, the resources who are at full capacity cannot handle more work per time unit.

To better understand the notion of theoretical capacity, we need to introduce the notion of *resource pool*. A resource pool R is a set of interchangeable resources who are responsible for executing a set of tasks in a process. For example, let us assume that in the loan application process, tasks "Check completeness", "Check credit history", and "Check income sources" are performed by *clerks*, whereas "Assess application", "Make credit offer", and "Notify rejection" are performed by *credit officers*. Therefore, this process has two resource pools (clerks and credit officers). Most likely, there are multiple clerks and multiple credit officers. This means that each resource pool has a given size.

Each instance of the process demands a given amount of time (on average) from each resource pool. The amount of time that a resource pool p needs to spend on one instance of the process is called the pool's *unit load* (ul). Each task assigned to a resource pool adds up to its unit load. And the more times a task is executed per instance, the more this task adds up to the load of its resource pool. For example, if two tasks a_1 and a_2 have the same processing time, but a_1 is executed on average 0.5 times per instance, while a_2 is executed on average twice per instance, then a_2 contributes four times more to the unit load than a_1. Hence, to calculate the unit load, we add up the processing times of each task a assigned to resource pool p, taking into account how many times each task is executed per instance of the process.

Below we present a two-step method to calculate ul (for a given pool p) using equations similar to the ones we used to calculate cycle time and theoretical cycle time. In the first step, we assign a unit load to each task with respect to pool p as follows:

- For each task assigned to resource pool p, its unit load is equal to its processing time.
- For each task not assigned to resource pool p, its unit load is zero.

In the second step, we use the following equations to calculate ul for a pool:

- The unit load ul of a sequence of fragments with unit loads $ul_1, \ldots ul_n$ is the sum of the unit loads of the fragments: $ul = \Sigma_{i=1}^{n} ul_i$.
- The unit load ul of an XOR-block is the weighted average of the unit loads of its branches (ul_i), using the branching probabilities p_i as the weights: $ul = \Sigma_{i=1}^{n} p_i \times ul_i$.
- The unit load ul of an AND-Block is the sum of the unit loads of its branches: $ul = \Sigma_{i=1}^{n} ul_i$. This is the same equation for sequences of fragments. The reason is that if a resource pool is involved in multiple branches of an AND-Block, each of these branches adds up to the load of this resource pool (i.e., each branch requires some effort and these efforts have to be added up).
- The unit load ul of a rework block with a rework probability r is the unit load of each iteration of the loop (let us call it ul_b) divided by $1 - r$: $ul = ul_b/(1 - r)$.

Example 7.5 Let us consider the credit application process model in Figure 7.10 and the processing times given in Table 7.2. Tasks "Check completeness", "Check credit history", and "Check income sources" are performed by *clerks*, whereas "Assess application", "Make credit offer", and "Notify rejection" are performed by *credit officers*.

Using the above equations, the unit loads of the three tasks assigned to the clerk are 2 h ("Check completeness"), 3 h ("Check credit history"), and 0.5 h ("Check income sources"). The remaining tasks after the AND-join gateway do not contribute to the unit load of the clerk pool. Hence, the unit load pool is $2/(1-0.2)+3+0.5=6$ working hours. This means that each loan application takes 6 h of time from the clerk pool.

Meanwhile, the unit loads of the tasks assigned to the credit officer are 2 h for both "Assess application" and for "Make credit offer", but 0.5 h for "Notify rejection". The first three tasks do not contribute to the unit load of credit officers. Taking into account the branching probabilities of the XOR-split, the unit load of the credit office pool is $2+0.6\times 2+0.4\times 0.5=5.2$ working hours. Again, this means that each loan application takes on average 5.2 h from the credit officers pool. □

We have seen how to calculate the unit load ul of a resource pool p, which means the amount of time that a resource pool p spends per instance of the process. To calculate the theoretical capacity, we now need to determine how much time each resource pool can deliver per time unit. This is called the *unit capacity* of the pool. To do this, it is convenient to lift the temporal granularity by one level. Since we have been working in terms of hours above, we will now move to the next higher granularity, which is the working day. Let us assume that a working day is 8 h. This means that one resource in a pool can deliver 8 h of work per day. This is called the unit capacity of a resource. By extension, the unit capacity of a resource pool uc is the size of the pool times the capacity of one resource.

Given the above, the theoretical capacity μ_p[3] of pool p is:

$$\mu_p = \frac{uc}{ul} \tag{7.7}$$

Example 7.6 Continuing the previous example, let us assume that the size of the clerk resource pool is 3, while the same holds for the credit officer pool. Let us also assume that 1 day is equal to 8 working hours. The unit capacity of a clerk (and same for a credit officer) is 8 h/day. The unit capacity of the clerk pool is 24 h/day (same for the credit officer pool). This means that the clerks can dedicate up to 24 h of effort per business day. Since each instance takes 6 h from them, they can collectively handle $24/6=4$ applications per day (i.e., $\mu=4$ instances/day for the clerk pool). Similarly, three credit officers can dedicate up to 24 h/day. And since each instance requires 5.2 h from them, their theoretical capacity is $24/5.2=4.62$ loan applications per day (i.e., $\mu=4.62$ for the credit officer pool). □

[3]Letter μ is pronounced mu.

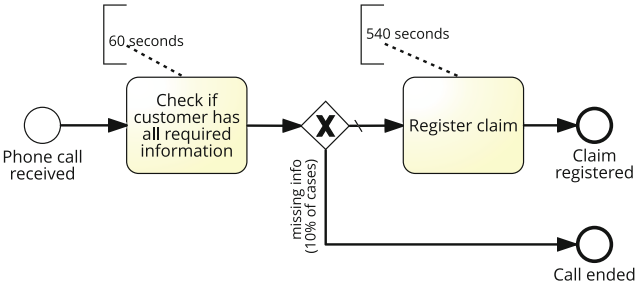

Fig. 7.12 Process model of a call center

We see that the clerks can handle less loan applications per day than the credit officers (4 versus 4.62). So, if we start receiving a lot of loan applications, the clerks will be the first ones to reach their theoretical capacity. We say that the clerk pool is the *bottleneck* of the process. More generally, the bottleneck is the resource pool with the minimum theoretical capacity among all pools in a process.

The *theoretical capacity of a process* is the theoretical capacity of its bottleneck pool. In our example, this is four instances per day. In the long term, there is no way we can deliver more instances per day unless something is changed, like for example if we add more resources to the resource pool or reduce the processing time of the tasks in which they are involved.

Another useful and related concept is that of *resource utilization*. The resource utilization ρ_p[4] of a pool p is the arrival rate λ of instances of the process (which we saw in Little's law) divided by the theoretical capacity μ_p of the pool, i.e.

$$\rho_p = \lambda/\mu_p \tag{7.8}$$

When there is no ambiguity, we will omit the subscript of ρ_p, meaning that we will simply write ρ if it is clear which pool we are referring to.

Example 7.7 Continuing from the previous example, and assuming that 3 loan applications arrive per day (i.e., $\lambda = 3$), the resource utilization of the clerk pool is $3/4 = 0.75$ and that of the credit officer pool is $3/4.62 \sim 0.65$. This means the pools are running at 75% and 65% of their theoretical capacity, respectively. □

Exercise 7.6 An insurance company receives 220 calls per day from customers who want to lodge an insurance claim. All calls are handled by 7 call center agents who work from 8 a.m. to 5 p.m. every day. The way calls are handled is captured in Figure 7.12. The process model in this figure also shows the processing times and branching probabilities.

[4]Letter ρ is pronounced rho.

Based on this information, answer the following questions:

- What is the unit load of the "Call center agent" pool?
- What is the unit capacity of the "Call center agent" pool? Specifically, how many seconds per hour can the agents collectively dedicate to the process?
- What is the theoretical capacity of the "Call center agent" pool?
- What is the resource utilization of the "Call center agent" pool?

7.1.6 Flow Analysis for Cost

As mentioned earlier, flow analysis can also be used to calculate other performance measures besides cycle time. For example, assuming we know the average cost of each task, we can calculate the cost of a process more or less in the same way as we calculate cycle time. In particular, the cost of a sequence of tasks is the sum of the costs of these tasks. Similarly, the cost of an XOR-block is the weighted average of the cost of the branches of the XOR-block and the cost of a rework pattern, such as the one shown in Figure 7.8, is the cost of the body of the loop divided by $1 - r$. The only difference between calculating cycle time and calculating cost relates to the treatment of AND-blocks. The cost of an AND-block, such as the one shown in Figure 7.5, is not the maximum of the cost of the branches of the AND-block. Instead, the cost of such a block is the sum of the costs of the branches. This is because after the AND-split is traversed, every branch in the AND-join is executed. Therefore the costs of these branches add up to one another.

Example 7.8 Let us consider again the credit application process model in Figure 7.10 and the processing times given in Table 7.2. As previously, we assume that in 20% of cases the application is incomplete and in 60% of cases the credit is granted. We further assume that the tasks "Check completeness", "Check credit history" and "Check income sources" are performed by a clerk, while "Assess application", "Make credit offer" and "Notify rejection" are performed by a credit officer. The hourly cost of a clerk is € 25 while the hourly cost of a credit officer is € 50. Performing a credit history requires that the bank submit a query to an external system. The bank is charged € 1 per query by the provider of this external system.

From this scenario, we can see that the cost of each task can be split into two components: the *labor cost* and *other costs*. The labor cost is the cost of the human resource that performs the task. This can be calculated as the product of the hourly cost of the resource and the processing time (in hours) of the task. Other costs correspond to costs that are incurred by an execution of a task, but are not related to the time spent by human resources on the task. In this example, the cost per query to the external system would be classified as "other costs" for the task "Check credit history". The remaining tasks do not have an "other costs" component. For the example at hand, the breakdown of resource cost, other cost and total cost per task is

Table 7.5 Cost calculation table for credit application process

Task	Resource cost	Other cost	Total cost
Check completeness	2× € 25 = € 50	€ 0	€ 50
Check credit history	0.5× € 25 = € 12.5	€ 1	€ 13.5
Check income sources	3× € 25 = € 75	€ 0	€ 75
Assess application	2× € 50 = € 100	€ 0	€ 100
Make credit offer	2× € 50 = € 100	€ 0	€ 100
Notify rejection	0.5× € 50 = € 25	€ 0	€ 25

given in Table 7.5. Given this input, we can calculate the total cost-per-execution of the process as follows: $50/(1-0.2)+13.5+75+100+0.6\times 100+0.4\times 25 = 321$. □

Exercise 7.7 Calculate the cost-per-execution of the ministerial enquiry process introduced in Exercise 3.7 (page 90). Assume that the rework probability is 0.2 and the times as given in Table 7.3. The task "Register ministerial enquiry" is performed by a clerk, task "Investigate ministerial enquiry" is performed by an adviser, "Prepare ministerial response" is performed by a senior adviser, and "Review ministerial response" is performed by a minister counselor. The hourly resource cost of a clerk, adviser, senior adviser and minister counselor are € 25, € 50, € 75, and € 100, respectively. There are no other costs attached to these tasks besides the resource costs.

7.1.7 Limitations of Flow Analysis

Before closing the discussion on flow analysis, it is important to highlight some of its pitfalls and limitations. First of all, we should note that the equations presented in Section 7.1.1 do not allow us to calculate the cycle time of any given process model. In fact, these equations only work in the case of *block-structured* process models. In particular, we cannot use these equations to calculate the cycle time of an unstructured process model such as the one shown in Exercise 3.12 (page 112). Indeed, this example does not fit into any of the patterns we have seen above. Also, if the model contains other modeling constructs besides AND and XOR gateways, the method for calculating cycle time becomes more complicated.

Fortunately, this is not a fundamental limitation of flow analysis, but only a limitation of the equations discussed in Section 7.1.1. There are more sophisticated flow analysis techniques that can be used for any process model. The maths can get a bit more complex. But this is generally not a problem given that several modern process modeling tools include functionality for calculating cycle time, cost and other performance measures of a process model using flow analysis.

A more fundamental roadblock faced by analysts when applying flow analysis is the fact that they first need to estimate the average cycle time of each task in the process model. In fact, this is a typical obstacle when applying any quantitative process analysis technique. There are at least two approaches to address this obstacle. The first one is based on interviews or observation. In this approach, analysts interview the stakeholders involved in each task or they observe how the stakeholders work during a given day or period of time. This allows analysts to at least make an informed guess regarding the average time a case spends in each task, in terms of both waiting time and processing time. In practice, the collection of data using interviews and observation should best be integrated with the process discovery as described in Section 5.2. A second approach is to collect logs from the information systems used in the process. For example, if a task "Approve purchase requisition" is performed via a Web portal, the portal's administrators may be able to extract execution logs to estimate the average time that a requisition spends in the "waiting for approval" state and the average time between the moment the supervisor opens a requisition for approval and the moment they approve it.

A more fundamental limitation of flow analysis is that it does not take into account the fact that a process behaves differently depending on the load. Intuitively, the cycle time of a process for handling insurance claims would be much slower if the insurance company is handling thousands of claims at once, due for example to a recent natural disaster such as a storm, versus the case where the load is low and the insurance company is only handling a hundred claims at once. When the load goes up and the number of resources (e.g., claim handlers) remains relatively constant, it is clear that the waiting times are going to be longer. This is due to a phenomenon known as *resource contention*. Resource contention occurs when there is more work to be done than resources available to perform the work, like for example more claims than insurance claim handlers. In such scenarios, some tasks will be in waiting mode until one of the necessary resources is freed up. Flow analysis does not directly inform us about the effects of increased resource contention. Instead, the estimates obtained from flow analysis are only applicable if the level of resource contention remains relatively stable over the long run.

7.2 Queues

Queueing theory is a collection of mathematical techniques to analyze systems that have resource contention. Resource contention inevitably leads to queues as we all probably have experienced in supermarket check-out counters, at a bank branch, post office, or government agency. Queueing theory gives us techniques to analyze important parameters of a queue such as the expected length of the queue or the expected waiting time of an individual case in a queue.

7.2.1 Basics of Queueing Theory

In basic queueing theory, a *queueing system* consists of one or multiple *queues* and a *service* that is provided by one or multiple *servers*. The elements inside a queue are called *jobs* or *customers*, depending on the specific context. In the following, we stick to the generic term *process instance*. For example, in the case of a supermarket, the service is that of checking out. This service is provided by multiple cashiers (the servers). Meanwhile, in the case of a bank office, the service is to perform a banking transaction, the servers are tellers, and there is generally a single queue that leads to multiple servers (the tellers). These two examples illustrate an important distinction between multi-line (i.e., multi-queue) queueing systems (like the supermarket) and single-line queueing systems (like the bank office).

Queueing theory provides a very broad set of techniques. Instead of trying to present everything that queueing theory has to offer, we will present two queueing theory models that are relatively simple, yet useful when analyzing business processes or tasks within a process.

In the two models we will be presenting there is a single queue (single-line queueing system). Instances arrive at a given average arrival rate λ. This is the same concept of arrival rate that we discussed above when presenting Little's law. For example, we can say that customers arrive to the bank office at a mean rate of 20 per hour. This implies that, on average, one customer arrives every 3 min ($\frac{1}{20}$ h). This latter number is called the mean *inter-arrival time*. We observe that if λ is the arrival rate per time unit, then $1/\lambda$ is the mean inter-arrival time.

It would be illusory to think that the time between the arrival of two customers at the bank office is always 3 min. This is just the mean value. In practice, customers arrive independently from one another, so the time between the arrival of one customer and the arrival of the next customer is completely random. Moreover, let us say the time between the arrival of the first customer and the arrival of the second customer is 1 min. This observation does not tell us anything about the time between the arrival of the second customer and the arrival of the third customer. It might be that the third customer arrives 1 min after the second, 5 min, or 10 min. We will not know until the third customer arrives.

Such an arrival process is called a *Poisson process*. In this case, the distribution of arrivals follows a so-called *exponential distribution* (specifically a *negative exponential distribution*) with a mean of $1/\lambda$. In a nutshell, this means that the probability that the inter-arrival time is exactly equal to t (where t is a positive number) decreases in an exponential manner when t increases. For instance, the probability of an inter-arrival time of 10 min is considerably smaller than the probability of the inter-arrival time being 1 min. Hence, shorter inter-arrival times are much more probable than longer ones, but there is always a probability (perhaps a very small one) that the inter-arrival time will be large.

In practice, the Poisson process and the exponential distribution describe a large class of arrival processes. So, we will be using them to capture the arrival of jobs or customers into a business process or a task in a business process. The Poisson

process can also be observed when we examine how often cars enter a given segment of a highway or how often calls go through a telephone exchange.

Having said this, one must always cross-check that cases arrive at a given process or task in an exponentially distributed manner. This cross-check can be done by recording the inter-arrival times for a given period of time and then feeding these numbers into a statistical tool such as R, Mathworks's Statistical Toolbox, and EasyFit. These tools use the input of a set of observed inter-arrival times to check if it follows a negative exponential distribution.

Exponential distributions are not only useful when modeling the inter-arrival time. They are also in some cases useful when describing the processing time of a task. In queueing theory, the term *service time* is often used instead of processing time. In the case of tasks that require a diagnosis, a non-trivial verification, or some non-trivial decision making, it is often the case that the processing time is exponentially distributed. Take, for example, the amount of time it takes for a mechanic to make a repair on a car. Most repairs are fairly standard and the mechanics might take 1 h to do them. However, some repairs are complex and in such cases it can take the mechanic several hours to complete. A similar remark can be made of a doctor receiving patients in an emergency room. A large number of emergencies are quite standard and can be dispatched in less than an hour, but some emergencies are extremely complicated and can take hours to deal with. So, it is likely that such tasks will follow an exponential distribution. As mentioned above, when making such a hypothesis, it is important to verify it by taking a random sample of processing times and feeding them to a statistical tool.

In the queueing theory field, a single-queue system is called an *M/M/1 queue*[5] if the inter-arrival times of customers follow an exponential distribution, the processing times follow an exponential distribution, there is one single server and instances are served on a First-In-First-Out (FIFO) basis. In the case of M/M/1 queue, we also assume that when an instance arrives it enters the queue and it stays there until it is taken on by the server.

If the above conditions are satisfied, but there are multiple servers instead of a single server, the queueing system is said to be *M/M/c*, where c is the number of servers. For example, a system is M/M/5 if the inter-arrival times of customers follow an exponential distribution, the processing times follow an exponential distribution, and there are 5 servers at the end of the queue. The "M" in this denomination stands for "Markovian", which is the name given to the assumptions that inter-arrival times and processing times follow an exponential distribution. Other queueing models exist that make different assumptions. Each such model is different, so the results we will obtain for an M/M/1 or M/M/c queue are quite different from those we would obtain from other distributions.

[5]This notation is commonly known as Kendall's notation.

7.2.2 M/M/1 and M/M/c Models

The previous discussion, an M/M/1 queue or M/M/c queue can be defined by means of the following parameters:

- λ is the mean arrival rate per time unit. The mean inter-arrival time is then $1/\lambda$. For example, $\lambda = 5$ means that there are 5 arrivals per hour and this entails that the mean inter-arrival time between two consecutive instances is $1/5$ h, that is 12 min.
- μ is the theoretical capacity per server (i.e., theoretical capacity per resource) or in other words, the number of instances that a server can execute per time unit. For example, $\mu = 6$ means that 6 instances are served per hour, which means that one instance is served in 10 min (on average).[6]
- In the case of M/M/c, the number of servers is c.

Given parameters λ and μ, we defined in Section 7.1.5 the *resource utilization* $\rho = \lambda/\mu$. In the above example, the resource utilization is 5/6 = 83.34%. It should be noted that this is a relatively high resource utilization. A system with a resource utilization of more than 100% is unstable, which means that the queue will become longer and longer forever because the server cannot cope with all the demand. In fact, even a system with a resource utilization close to 100% is unstable because of the randomness with which new instances arrive and the variability in the processing times per instance. To understand why this is the case, just imagine a doctor receiving patients at a rate of 6 per hour for 8 h, knowing that every patient takes 10 min on average to be treated (sometimes less but sometimes more). Without any slack, the doctor will end up with a tremendous backlog at the end of the day.

In the case of an M/M/c system, the resource utilization is $\frac{\lambda}{c\mu}$ since the system consists of a pool of resources that can collectively handle instances at a rate of $c\mu$. For example, if the system has 2 servers and each server can handle 2 instances per hour, the system can handle 4 instances per hour. If instances arrive at a mean rate of 3 per hour, the resource utilization of the system is $3/4 = 75\%$.

Given an M/M/1 or M/M/c system, queueing theory allows us to calculate the following parameters:

- L_q is the average number of instances in the queue.
- W_q is the average time one instance spends in the queue.
- W is the average time one instance spends in the entire system. This includes both the time the instance spends in the queue but also the time it spends being serviced.
- L is the average number of instances in the system (i.e., the Work-In-Process referenced in Little's law).

[6] In Section 7.1.5, we used symbol μ to refer to the theoretical capacity of a resource pool, while here we are using symbol μ to refer to the theoretical capacity of each individual resource (or server) in a pool. This is because the size of the pool is handled separately using parameter c.

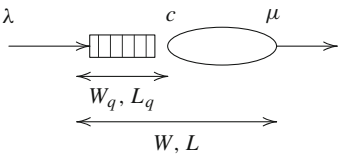

Fig. 7.13 Structure of an M/M/1 or M/M/c system, input parameters and computable parameters

To summarize, the general structure of a single-queue system, which consists of one queue and one or many servers, is depicted in Figure 7.13. The parameters of the queue (λ, c and μ) are shown at the top. The parameters that can be computed from these three input parameters are shown under the queue and the server. The average time an instance waits in the queue is W_q, while the average length of the queue is L_q. Eventually, an instance goes into the server and in there it spends on average $1/\mu$ time units.[7] The average time between the moment an instance enters the system and the moment it exits is W, while the average number of instances inside the system (in the queue or in a server) is L.

Queueing theory gives us the following equations for calculating the above parameters for M/M/1 models:

$$L_q = \rho^2/(1-\rho) \tag{7.9}$$

$$W_q = \frac{L_q}{\lambda} \tag{7.10}$$

$$W = W_q + \frac{1}{\mu} \tag{7.11}$$

$$L = \lambda W \tag{7.12}$$

Formulas (7.10), (7.11) and (7.12) can be applied to M/M/c models as well. The only parameter that needs to be calculated differently in the case of M/M/c models is L_q. For M/M/c models, L_q is given by the following formula:

$$L_q = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2 \left(\frac{(\lambda/\mu)^c}{c!(1-\rho)} + \sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} \right)} \tag{7.13}$$

This formula is particularly complicated because of the summations and factorials. Fortunately, there are tools that can do this for us. For example, the Queueing Toolpack[8] supports calculations for M/M/c systems (called *M/M/s* in the Queueing Toolpack) as well as M/M/c/k systems, where k is the maximum number of instances allowed in the queue. Instances that arrive when the length of the queue

[7] This is equivalent to what we called the unit load in Section 7.1.5.

[8] http://queueingtoolpak.org.

is k are rejected (and may come back later). Other tools for analyzing queueing systems include QSim[9] and PDQ.[10]

Example 7.9 A company designs customized electronic hardware for a range of customers in the high-tech electronics industry. The company receives orders for designing a new circuit every 20 working days on average. It takes a team of engineers on average 10 working days to design a hardware device.

This problem can be mapped to an M/M/1 model, assuming that the arrival of designs follows a Poisson process, that the distribution of times for designing a circuit follows an exponential distribution, and that new design requests are handled in a FIFO manner. Note that even though the team includes several people, they act as a monolithic entity and therefore should be treated as a single server. Let us take the working day as a time unit. On average, 0.05 orders are received per day ($\lambda = 0.05$), and 0.1 orders are fulfilled per day ($\mu = 0.1$). Thus, the resource utilization of this system $\rho = 0.05/0.1 = 0.5$. Using the formulas for M/M/1 models, we can deduce that the average length of the queue L_q is $0.5^2/(1 - 0.5) = 0.5$ orders. From there we can conclude that the average time an order spends in the queue is $W_q = 0.5/0.05 = 10$ days. Thus, it takes on average $W = 10 + 1/0.1 = 20$ working days for an order to be fulfilled. □

Exercise 7.8 Consider now the case where the engineering team in the previous example requires 16 working days to design a hardware device. What is then the average amount of time an order takes to be fulfilled?

Exercise 7.9 An insurance company receives 220 calls per day from customers who lodge insurance claims. The call center is open from 8 a.m. to 5 p.m. The arrival of calls follows a Poisson process. Looking at the intensity of arrival of calls, we can distinguish three periods during the day: the period 8 a.m. to 11 a.m., the period 11 a.m. to 2 p.m. and the period 2 p.m. to 5 p.m. During the first period, around 60 calls are received. During the 11 a.m. to 2 p.m. period, 120 calls are received, and during the 2 p.m. to 5 p.m. period, 40 calls are received. A customer survey has shown that customers tend to call between 11 a.m. and 2 p.m. because during this time they have a break at work.

Statistical analysis shows that the durations of calls follow an exponential distribution. According to the company's customer service charter, customers should not wait more than 1 min on average for their call to be answered.

- Assume that the call center can handle 70 calls per hour using 7 call center agents. Is this enough to meet the 1-min constraint set in the customer service charter? Please explain your answer by showing how you calculate the average length of the queue and the average waiting time.
- What happens if the call center's capacity is increased so that it can handle 80 calls per hour (using 8 call center agents)?

[9] http://www.stat.auckland.ac.nz/~stats255/qsim/qsim.html.

[10] http://www.perfdynamics.com/Tools/PDQ.html.

- The call center manager has a mandate to cut costs by at least 20%. Give at least two ideas to achieve this cut without reducing the salaries of the call center agents and while keeping an average waiting time below or close to 1 min.

7.2.3 Limitations of Basic Queueing Theory

The basic queueing analysis techniques presented above allow us to estimate waiting times and queue lengths based on the assumptions that inter-arrival times and processing times follow an exponential distribution. When these parameters follow different distributions, one needs to use different queueing models. Fortunately, queueing theory tools nowadays support a broad range of queueing models and of course they can do the calculations for us. The discussion above must be seen as an overview of single-queue models, with the aim of providing a starting point from where you can learn more about this family of techniques.

A more fundamental limitation of the techniques introduced in this section is that they only deal with one task at a time. When we have to analyze an entire process that involves several tasks, events, and resources, these basic techniques are not sufficient. There are many other queueing analysis techniques that could be used for this purpose, like for example queueing networks. Essentially, queueing networks are systems consisting of multiple inter-connected queues. However, the maths behind queueing networks can become quite complex, especially when the process includes concurrent tasks. A more popular approach for quantitative analysis of process models under varying levels of resource contention is process simulation, as discussed below.

7.3 Simulation

Process simulation is arguably the most popular and widely supported technique for quantitative analysis of process models. The essential idea underpinning process simulation is to use the process simulator for generating a large number of hypothetical instances of a process, executing these instances step-by-step, and recording each step in this execution. The output of a simulator then includes the logs of the simulation as well as statistics of cycle times, average waiting times, and average resource utilization.

7.3.1 Anatomy of a Process Simulation

During a process simulation, the tasks in the process are not actually executed. Instead, the simulation of a task proceeds as follows. When a task is ready to be executed, a so-called *work item* is created and the simulator first tries to find a

resource to which it can assign this work item. If no resource able to perform the work item is available, the simulator puts the work item in waiting mode until a suitable resource becomes available. Once a resource is assigned to a work item, the simulator determines the duration of the work item by drawing a random number according to the probability distribution of the task processing time. This probability distribution and the corresponding parameters need to be defined in the simulation model.

Once the simulator has determined the duration of a work item, it puts the work item in sleeping mode for that duration. This sleeping mode simulates the fact that the task is being executed. Once the time interval has passed (according to the simulation clock), the work item is declared to be completed and the resource that was assigned to it becomes available.

In reality, the simulator does not effectively wait for tasks to come back from their sleeping mode. For example, if the simulator determines that the duration of a work item is 2 days and 2 h, it will not wait for this amount of time to pass by. You can imagine how long a simulation would take if that was the case. Instead, simulators use smart algorithms to complete the simulation as fast as possible. Modern business process simulators can effectively simulate thousands of process instances and tens of thousands of work items in a matter of seconds.

For each work item created during a simulation, the simulator records the identifier of the resource that was assigned to this instance as well as three timestamps:

- The time when the task was ready to be executed.
- The time when the task was started, meaning that it was assigned to a resource.
- The time when the task completed.

Using the collected data, the simulator can compute the average waiting time for each task. These measures are quite important when we try to identify bottlenecks in the process. Indeed, if a task has a high average waiting time, it means that there is a bottleneck at the level of this task. The analyst can then consider several options for addressing this bottleneck.

Additionally, since the simulator records which resources perform which work items and it knows how long each work item takes, the simulator can find out the total amount of time during which a given resource is busy handling work items. By dividing the amount of time that a resource was busy during a simulation by the total duration of the simulation, we obtain the *resource utilization*, that is, the percentage of time that the resource is busy on average.

7.3.2 Input for Process Simulation

From the above description of how a simulation works, we can see that the following information needs to be specified for each task in the process model in order to simulate it:

- The probability distribution for the processing time of each task.
- Other performance attributes for the task such as cost and added-value produced by the task.
- The *resource pool* that is responsible for performing the task. In the loan application process, there are three resource pools: the claim handlers, the clerks and the managers. For each resource pool, we need to specify its size (e.g., the number of claim handlers or the number of clerks) and optionally their cost per time unit (e.g., the hourly cost of a claims handler). If we specify the cost per time unit for every resource pool, the simulation will calculate the mean labor cost per case in addition to calculating cycle times and waiting times.

Common probability distributions for task durations in the context of process simulation include:

- *Fixed.* This is the case where the processing time of the task is the same for all executions of this task. It is rare to find such tasks because most tasks, especially those involving human resources, would exhibit some variability in their processing time. Examples of tasks with fixed processing time can be found among automated tasks such as for example a task that generates a report from a database. Such a task would take a relatively constant amount of time, say for example 5 s.
- *Exponential distribution*. As discussed in Section 7.2, the exponential distribution may be applicable when the processing time of the task is most often around a given mean value, but sometimes it is considerably longer. For example, consider a task "Assess insurance claims" in an insurance claims handling process. For normal cases, the claim is assessed in an hour, or perhaps less. However, some insurance claims require special treatment, for example because the assessor considers that there is a risk that the claim is fraudulent. In this case, the assessor might spend several hours or even an entire day assessing a single claim. A similar observation can be made of diagnostics tasks, such as diagnosing a problem in an IT infrastructure or diagnosing a problem during a car repair process.
- *Normal distribution*. This distribution is used when the processing time of the task is around a given average and the deviation around this value is symmetric, which means that the actual processing time can be above or below the mean with the same probability. Simple checks, such as for example checking whether or not a paper form has been fully completed might follow this distribution. Indeed, it generally takes about 3 min to make such a check. In such cases, this time can be lower because for example the form is clearly incomplete or clearly complete. In other cases, it can take a bit longer, because a couple of fields have been left empty and it is unclear if these fields are relevant or not for the specific customer who submitted the form. Some simulators also support the *half-normal distribution*, which is similar to the normal distribution but it only allows for positive values. Negative values do not make sense when applied to processing times or costs.

When assigning an exponential distribution to a task duration the analyst has to specify the mean value. Meanwhile, when assigning a normal distribution, the analyst has to specify two parameters: mean value and standard deviation. These values are determined based on an informed guess (based on interviews with the relevant stakeholders), but preferably by means of sampling (the analyst collects data for a sample of task executions) or by analyzing execution logs of relevant information systems. Some simulation tools allow the analyst to import logs into the simulation tool and assist the analyst in selecting the right probability distribution for task durations based on these logs. This functionality is called *simulation input analysis*.

In addition to the above per-task simulation data, a branching probability needs to be specified for every flow coming out of a decision gateway. These probabilities may be determined by interviewing relevant stakeholders, observing the process during a period of time, or collecting logs from relevant information systems.

Finally, in order to run a simulation, the analyst additionally needs to specify at least the following:

- The mean inter-arrival time and its associated probability distribution. As explained above, a very frequent distribution of inter-arrival times is the exponential distribution and this is usually the default distribution supported by business process simulators. It may happen however that the inter-arrival times follow a different distribution such as for example a *normal distribution*. By feeding a sample of inter-arrival times during a certain period of time to a statistical tool, we can find out which distribution best matches the data. Some simulators provide a module for selecting a distribution for the inter-arrival times and for computing the mean inter-arrival time from a data sample.
- The starting date and time of the simulation (e.g., "11 Nov. 2017 at 8:00").
- One of the following:
  - The end date and time of the simulation. If this option is selected, the simulation will stop producing more process instances once the simulation clock reaches the end time.
  - The real-time duration of the simulation (e.g., 7 days, 14 days). In this way, the end time of the simulation can be derived by adding this duration to the starting time.
  - The required number of process instances to be simulated (e.g., 1,000). If this option is selected, the simulator generates process instances according to the arrival rate until it reaches the required number of process instances. At this point, the simulation stops. Some simulators will not stop immediately, but will allow the active process instances to complete before stopping the simulation.

Example 7.10 We consider the process for loan application approval modeled in Figure 4.2 (page 118). We simulate this model using the BIMP simulator.[11] This

[11] http://bimp.cs.ut.ee.

simulator takes as input a BPMN process model. We provide the following inputs for the simulation.

- Three loan applications arrive per hour on average, meaning an inter-arrival time of 20 min. Loan applications arrive only from 9 a.m. to 5 p.m. during weekdays.
- The tasks "Check credit history" and "Check income sources" are performed by clerks.
- The tasks "Notify rejection", "Make credit offer", and "Assess application" are performed by credit officers.
- The task "Receive customer feedback" is in fact an event. It takes zero time and it only involves the credit information system (no human resources involved). To capture this, the task is assigned to a special "System" role.
- There are two clerks and two credit officers. The hourly cost of a clerk is € 25 while that of a credit officer is € 50.
- Clerks and credit officers work from 9 a.m. to 5 p.m. during weekdays.
- The cycle time of the task "Assess application" follows an exponential distribution with a mean of 20 min.
- Cycle times of all other tasks follow a normal distribution. The tasks "Check credit history", "Notify rejection", and "Make credit offer" have a mean cycle time of 10 min with a 20% standard deviation, while "Check income sources" has a cycle time of 20 min with a 20% standard deviation as well.
- The probability that an application is accepted is 80%.
- The probability that a customer, whose application was rejected, asks that the application be re-assessed is 20%.

We run a simulation with 2,400 instances, which means 100 working days given that 24 loan applications arrive per day. The simulation gives an average cycle time of around 7.5 h if we count the time outside working hours (*cycle time including off-timetable hours* in BIMP). If we count only working hours, the cycle time is 2 h. The latter is called the *cycle time excluding off-timetable hours* in BIMP. These cycle time measurements may vary by about ± 10% when we run the simulation multiple times. These variations are expected due to the stochastic nature of the simulation. For this reason, we recommend running the simulation multiple times and to take averages of the simulation results.

Figure 7.14 shows the histograms for process cycle times (both including and excluding off-timetable hours), waiting times (excluding off-timetable costs), and costs. It can be seen that the waiting times are relatively low. This is because the resource utilization of clerks and credit officers is around 76–80%. □

Exercise 7.10 The insurance company called Cetera is facing the following problem: Whenever there is a major event (e.g., a storm), their claim-to-resolution process is unable to cope with the ensuing spike in demand. During normal times, the insurance company receives about 9,000 calls per week, but during a storm scenario the number of calls per week doubles.

The claim-to-resolution process model of Cetera is presented in Figure 7.15. The process starts when a call related to lodging a claim is received. The call is routed

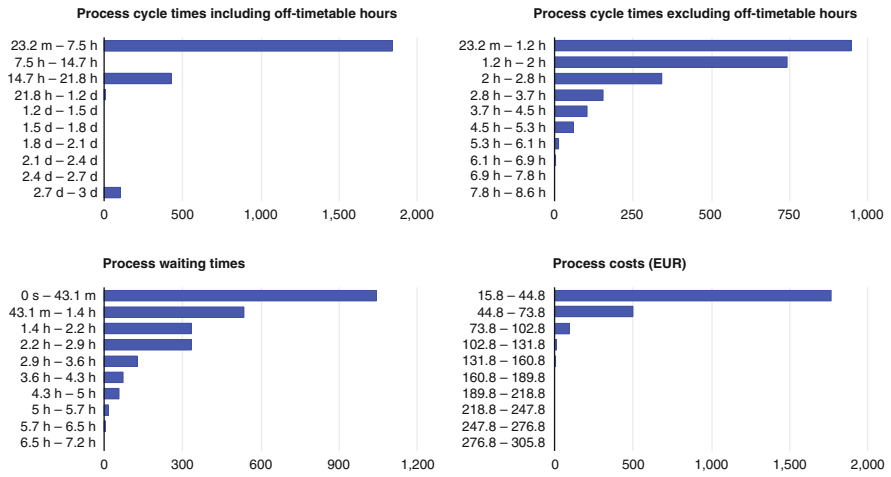

Fig. 7.14 Histograms produced by simulating the credit application process with BIMP

to one of two call centers depending on the location of the caller. Each call center receives approximately the same amount of calls (50–50) and has the same number of operators (40 per call center). The process for handling calls is identical across both call centers. When a call is received at a call center, the call is picked up by a call center operator. The call center operator starts by asking a standard set of questions to the customer to determine if the customer has the minimum information required to lodge a claim (e.g., insurance policy number). If the customer has enough information, the operator then goes through a questionnaire with the customer, enters all relevant details, checks the completeness of the claim, and registers the claim.

Once a claim has been registered, it is routed to the claims handling office, where all remaining steps are performed. There is one single claims handling office, so regardless of the call center agent where the claim is registered, the claim is routed to the same office. In this office, the claim goes through a two-stage evaluation process. First of all, the liability of the customer is determined. Secondly, the claim is assessed in order to determine if the insurance company has to cover this liability and to what extent. If the claim is accepted, payment is initiated and the customer is advised of the amount to be paid. The tasks of the claims handling department are performed by *claims handlers*. There are 150 claims handlers in total.

The mean cycle time of each task (in seconds) is indicated in Figure 7.15. For every task, the cycle time follows an exponential distribution. The hourly cost of a call center agent is € 30, while the hourly cost of a claims handler is € 50.

Describe the input that should be given to a simulator in order to simulate this process in the normal scenario and in the storm scenario. Using a simulation tool, encode the normal and the storm scenarios, and run a simulation in order to compare these two scenarios.

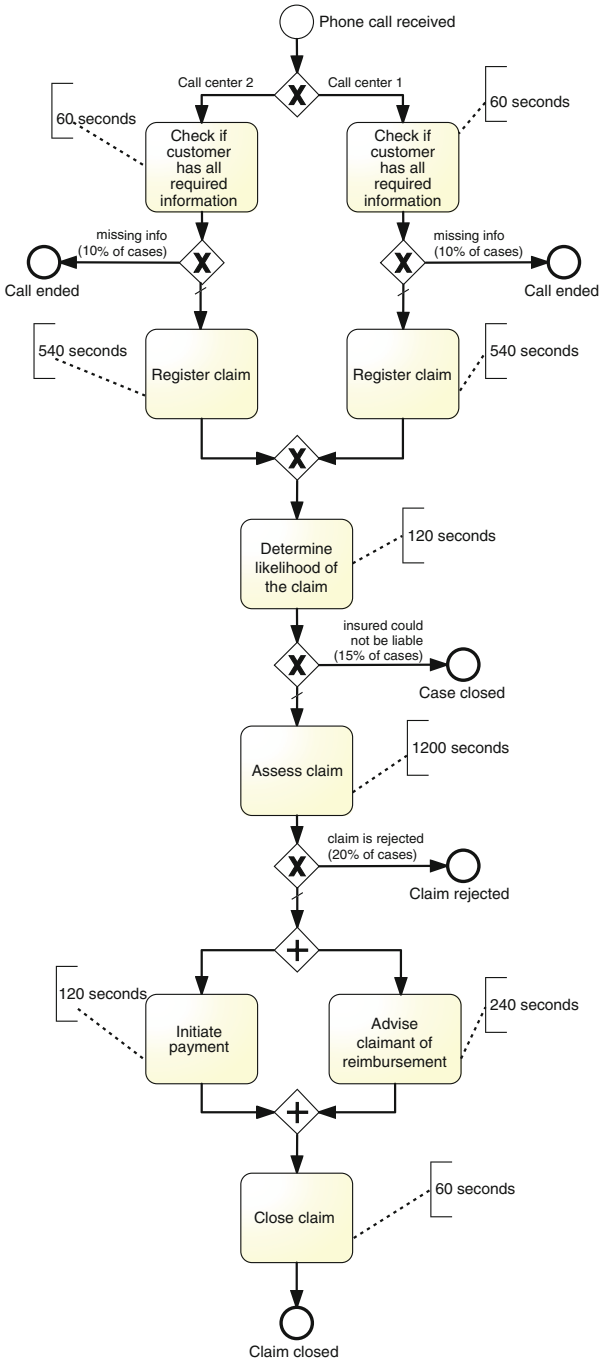

Fig. 7.15 Cetera's claim-to-resolution process

7.3.3 Simulation Tools

Nowadays, most business process modeling tools provide simulation capabilities. Examples of such tools with simulation support include Appian, ARIS, IBM BPM, Logizian, Oracle Business Process Analysis Suite, and Signavio Process Manager. The landscape of tools evolves continuously. Thus, it is important to understand the fundamental concepts of process simulation before trying to grasp the specific features of a given tool.

In general, the provided functionality varies from one tool to another. For example, some tools offer the functionality to specify that resources do not work continuously, but only during specific periods of time. This is specified by attaching a calendar to each resource pool. Some tools additionally allow one to specify that new process instances are created only during certain periods of time, for example only during business hours. Again, this is specified by means of a calendar.

Some of the more sophisticated tools capture not only branching conditions, but also actual boolean expressions that use attributes attached to data objects in the process model. In this way, we can specify, for example, that a branch coming out of an XOR-split should be taken when the attribute *loanAmount* of a data object called "loan application" is greater than € 10,000, whereas another branch should be taken when this amount is up to € 10,000. When the simulator generates objects of type loan, it gives them a value according to a probability distribution attached to this attribute.

There are minor differences in the way parameters are specified across simulation tools. Some tools require one to specify the mean arrival rate, that is the number of cases that start during one time unit (e.g., 50 cases per day), while other tools require one to specify the mean inter-arrival time between cases (e.g., 2 min until a new case arrives). Recall that the distinction between mean arrival rate (written λ in queueing theory) and mean inter-arrival time ($1/\lambda$) was discussed in Section 7.2.1. Other tools go further by allowing one to specify not only the inter-arrival time, but how many cases are created every time. By default, cases arrive one by one, but in some business processes, cases may arrive in batches.

Example 7.11 An example of a process with batch arrivals is an archival process at the Macau Historical Archives. At the beginning of each year, transfer lists are sent to the Historical Archives by various organizations. Each transfer list contains approximately 225 historical records. On average, two transfer lists are received each year. Each record in a transfer list needs to go through a process that includes appraisal, classification, annotation, backup, and re-binding. If we consider that each record is a case of this archival process, then we can say that cases arrive in batches of $225 \times 2 = 450$ cases. Moreover, these batches arrive at a fixed inter-arrival time of one year. □

Finally, process simulation tools typically differ in terms of how resource pools and resource costs are specified. Some tools restrict the specification to a resource pool and its number of resources. A single cost per time unit is then attached to the

entire resource pool. Other tools support a more fine-grained specification of the resources of a pool one by one with specific cost rates for each created resource (e.g., create 10 clerks one by one, each with its name and hourly cost).

The above discussion illustrates some of the nuances found across simulation tools. In order to avoid diving straight away into the numerous details of a tool, it may be useful for beginners to take their first steps using the BIMP simulator referred to in Example 7.10. BIMP is a rather simple BPMN process model simulator that provides the core functionality found in commercial business process simulation tools.

7.3.4 A Word of Caution

One should keep in mind that the quantitative analysis techniques we have seen in this chapter, and simulation in particular, are based on models and on simplifying assumptions. The reliability of the output produced by these techniques largely depends on the accuracy of the numbers that are given as input. Additionally, simulation assumes that process participants work mechanically. However, process participants are not robots. They are subject to unforeseen interruptions, they display varying performance depending on various factors, and they may adapt differently to new ways of working.

It is good practice whenever possible to derive the input parameters of a simulation from actual observations, meaning from historical process execution data. This is possible when simulating an as-is process that is being executed in the company, but not necessarily when simulating a to-be process. In a similar spirit, it is recommended to cross-check simulation outputs against expert advice. This can be achieved by presenting the simulation results to process stakeholders (including process participants). The process stakeholders are usually able to provide feedback on the credibility of the resource utilization levels calculated via simulation and the bottlenecks put into evidence by the simulation. For instance, if the simulation points to a bottleneck in a given task, while the stakeholders and participants perceive this task to be uncritical, there is an indication that incorrect assumptions have been made. Feedback from stakeholders and participants helps to reconfigure the parameters such that the results are closer to matching the actual behavior. In other words, process simulation is an iterative analysis technique.

Finally, it is advisable to perform sensitivity analysis of the simulation. Concretely, this means observing how the output of the simulation changes when adding one resource to or removing one resource from a resource pool, or when changing the processing times by $\pm$ 10%, for example. If such small changes in the simulation input parameters significantly affect the conclusions drawn from the simulation outputs, one must be careful when interpreting the simulation results.

7.4 Recap

In this chapter we saw three quantitative process analysis techniques, namely flow analysis, queueing theory, and simulation. These techniques allow us to derive process performance measures, such as cycle time or cost, and to understand how different tasks and resource pools contribute to the overall performance of a process.

Flow analysis allows us to calculate performance measures from a process model and performance data pertaining to each task in the model. We also analyzed the critical path of a process using the Critical Path Method. Finally, we studied the capacity of a process and defined the notion of resource utilization. The waiting times of a process are highly dependent on resource utilization—the busier the resources are, the longer the waiting times.

Basic queueing theory models, such as the M/M/1 model, allow us to calculate waiting times for individual tasks given data about the number of resources and their processing times. Other queueing theory models such as queueing networks can be used to perform fine-grained analysis at the level of entire processes. However, in practice it is convenient to use process simulation for fine-grained analysis. Process simulation allows us to derive process performance measures (e.g., cycle time or cost) given data about the tasks (e.g., processing times) and data about the resources involved in the process. Process simulation is a versatile technique supported by a range of process modeling and analysis tools.

7.5 Solutions to Exercises

Solution 7.1 First we observe that the cycle time of the AND-block is 1. Next, we calculate the cycle time of the XOR-block as follows: $0.4 \times 1 + 0.4 \times 1 + 0.2 \times 1$ h. The total cycle time is thus: $1 + 1 + 1 = 3$ h.

Solution 7.2 The cycle time of the process is $2 + 8 + \frac{4+4}{1-0.2} = 20$ days. Assuming 8 working hours per day, this translates to 160 working hours. The theoretical cycle time is $0.5 + 12 + \frac{4+2}{1-0.2} = 20$ h. Hence, cycle time efficiency is 12.5%.

Solution 7.3 It can be expected that the average cycle times of the reported process have generally improved. Since 1992, several technological advancements have drastically improved white-collar work productivity. These relate to a better coordination and routing of tasks using information technology including office applications, enterprise systems, and Internet technology. In Chapter 9, we will discuss how Business Process Management Systems and different types of Process-Aware Information Systems have contributed to better coordination and task automation. These advancements have likely reduced waiting times in many business processes. Therefore, also cycle time efficiency should have improved since 1992.

Solution 7.4 The process model shown in Figure 7.4 has three tasks with the following ES, EF, LS, and LF:

- Start event: $ES = EF = LS = LF = 0$.
- Task A: $ES = LS = 0$ and $EF = LF = 10$.
- Task B: $ES = LS = 10$ and $EF = LF = 30$.
- Task C: $ES = 10$ and $EF = 20$. Here is slack, because $LS = 20$ and $LF = 30$.
- End event: $ES = EF = LS = LF = 30$.

Task B has slack of 10. The critical path includes all tasks except B.

Solution 7.5 Little's law tells us that $CT = WIP/\lambda$. At peak time, there are 900 customers distributed across 6 h, so the mean arrival rate $\lambda = 150$ customers per hour. On the other hand, $WIP = 90$ during peak time. Thus, $CT = 90/150 = 0.6$ h (i.e., 36 min). During non-peak time, $\lambda = 300/6 = 50$ customer per hour while $WIP = 30$, thus $CT = 30/50 = 0.6$ h (again 36 min). If the number of customers per hour during peak times is expected to go up, but the WIP has to remain constant, we need to reduce the cycle time per customer. This may be achieved by shortening the serving time, the interval between the moment a customer enters the restaurant and the moment he or she places an order, or the time it takes for the customer to pay. In other words, the process for order taking and payment may need to be redesigned.

Solution 7.6

- A call center agent spends 60 + 0.9*540 = 546 s per instance.
- One call center agent can deliver 3,600 s per hour, hence 7 agents can deliver 25,200 s per hour.
- $\mu = 25,200/546 = 46.15$ calls per hour.
- For convenience, we use the hour as the time unit. Hence, $\lambda = 24.44$ and $\mu = 46.15$, and therefore $\rho = 24.44/46.15 = 0.53$

Solution 7.7 Given that there are no other costs, we calculate the cost of the process by aggregating the resource costs as follows: $0.5 \times €25 + 12 \times €50 + (4 \times €75 + 2 \times €100)/(1 - 0.2) = €1,237.50$.

Solution 7.8 On average, 0.05 orders are received per day ($\lambda = 0.05$), and 0.0625 orders are fulfilled per day ($\mu = 0.0625$). Thus, the resource utilization of this system $\rho = 0.05/0.0625 = 0.8$. Using the formulas for M/M/1 models, we can deduce that the average length of the queue L_q is: $0.8^2/(1 - 0.8) = 3.2$ orders. From this, we can conclude that the average time an order spends on the queue is $W_q = 3.2/0.05 = 64$ days. Thus, it takes on average $W = 64 + 16 = 80$ working days for an order to be fulfilled.

Solution 7.9 Strictly speaking, we should analyze this problem using an M/M/c queueing model. However, the formulas for M/M/c are quite complex to show the calculations in detail. For this reason, we will assume in this solution that the entire call center behaves as a single monolithic team, so that we can use an M/M/1 queueing model to analyse the problem. Because of this assumption, the results will not be exact.

If we only had 7 call center agents, then the resource utilization $\rho = 40/70 = 0.57$, $L_q = \rho^2/(1-\rho) = 0.57^2/(1-0.57) = 0.76$, and $W_q = L_q/\lambda = 0.76/40 = 0.0189$ h = 1.13 min. So we cannot meet the customer service charter.

If we can handle 80 calls per hour (8 call center agents), then the resource utilization $\rho = 40/80 = 0.5$, $L_q = \rho^2/(1-\rho) = 0.5^2/(1-0.5) = 0.5$, and $W_q = L_q/\lambda = 0.5/40 = 0.0125$ h = 45 s, so we meet the customer service charter.

Ways to reduce costs while staying as close as possible to the customer service charter are:

- We could reduce the number of call center agents to 7 and still have an average waiting time of 1.13 min. That reduces costs by 12.5% (one call center agent less).
- We could introduce a self-service system, whereby people lodge their application online (at least for simple claims).
- We could extend the call center working times (e.g., work until 6 p.m. or 7 p.m. instead of 5 p.m.), so that people can call after work. In this way, we might ease the call center load during its peak time.
- Reduce the time of each call by providing better training to call center agents.

Solution 7.10 For this problem, we will reason exclusively in terms of working hours as a unit of time, as opposed to calendar hours. We assume that a week consists of 40 working hours. Calls arrive only during these 40 working hours and call center operators and claims handlers work only during these 40 h. By taking working hours as a time unit, we avoid the need to attach calendars resources.

In the normal scenario (no storm), the arrival rate is 9,000 cases per week, that is one case every 16 s (this is the inter-arrival time). In the storm scenario the inter-arrival time is 8 s. In both cases, we use an exponential distribution for the inter-arrival time. We run simulations corresponding to 1 week of work, which means 9,000 cases for the normal scenario and 18,000 cases for the storm scenario.

In order to distinguish between the two call centers, we define two separate resource pools called "Call Center Operator 1" and "Call Center Operator 2" each one with 40 resources at an hourly cost of 30, plus a resource pool "Claims Handler" with 150 resources. We assign tasks to resource pools as indicated in the scenario and we use the cycle times indicated in the process model as input for the simulation. Running the simulation using the BIMP simulator gives us the following outputs. In the normal scenario, we obtain a resource utilization of around 48% for claims handlers and 34–36% for call center operators. The average cycle time (excluding off-timetable hours) is around 0.5 working hours and the maximum observed cycle time is around 3.3 working hours. In other words, the resources are under-utilized and, thus, the cycle time is low.

In the storm season, resource utilization of claims handlers is above 95% and around 78% for the call center agents. The average cycle time is 2 h while the maximum is around 7.5 h (excluding off-timetable time). The high resource utilization indicates that the claims handling office is overloaded during storm season. On the other hand, the call center has sufficient capacity. The average waiting time in the call center is in the order of seconds.

A BPMN model of this process together with the simulation parameters (in the format required by BIMP) can be found in the book's companion website.[12]

7.6 Further Exercises

Exercise 7.11 Calculate the cycle time, cycle time efficiency, and cost of the university admission process described in Exercise 1.1 (page 5), assuming that:

- The process starts when an online application is submitted.
- It takes on average 2 weeks (after the online application is submitted) for the documents to arrive to the students service by post.
- The check for completeness of documents takes about 10 min. In 20% of the cases, the completeness check reveals that some documents are missing. In this case, an email is sent to the student automatically by the university admission management system based on the input provided by the international students officer during the completeness check.
- A student services officer spends on average 10 min to put the degrees and transcripts in an envelope and to send them to the academic recognition agency. The time it takes to send the degrees and transcripts to the academic recognition agency and to receive back a response is 2 weeks on average.
- About 10% of applications are rejected after the academic recognition assessment.
- The university pays a fee of € 5 each time it requests the academic recognition agency to accept an application.
- Checking the English language test results takes 1 day on average, but the officer who performs the check only spends 10 min on average per check. This language test check is free.
- About 10% of applications are rejected after the English language test.
- It takes on average 2 weeks between the time students service sends the copy of an application to the committee members and the moment the committee makes a decision (accept or reject). On average, the committee spends 1 h examining each application.
- It takes on average 2 days (after the decision is made by the academic committee) for the students service to record the academic committee's decision in the university admission management system. Recording a decision takes on average 2 min. Once a decision is recorded, a notification is automatically sent to the student.
- The hourly cost of the officers at the international students office is € 50.
- The hourly cost of the academic committee (as a whole) is € 200.

[12] http://fundamentals-of-bpm.org/supplementary-material/.

Exercise 7.12 Let us consider the following process performed by an IT helpdesk that handles requests from clients. The clients are employees of a company. There are about 500 employees in total. A request may be an IT-related problem of a client or an access request (e.g., requesting rights to access a system). Requests need to be handled according to their type and their priority. There are three priority levels: "critical", "urgent", or "normal". The current process works as follows.

> A client calls the help desk or sends an email in order to make a request. The help desk is staffed with 5 Level-1 support staff who, typically, are junior people with less than 12 months experience, but are capable of resolving known problems and simple requests. The hourly cost of a Level-1 staff member is € 40.
> When the Level-1 employee does not know the resolution to a request, the request is forwarded to a more experienced Level-2 support staff. There are 3 Level-2 staff members and their hourly cost is € 60. When a Level-2 employee receives a new request, he or she evaluates it in order to assign a priority level. The ticketing system that tracks the process will later assign the request to the same or to another Level-2 staff depending on the assigned priority level and the backlog of requests.
> Once the request is assigned to a Level-2 staff member, the request is researched by the Level-2 employee and a resolution is developed and sent back to the Level-1 employee. Eventually, the Level-1 employee forwards the resolution to the client who tests the resolution. The client notifies the outcome of the test to the Level-1 employee via email. If the client states that the request is fixed, it is marked as complete and the process ends. If the request is not fixed, it is resent to Level-2 support for further action and goes through the process again.
> Requests are registered in a ticketing system. The ticketing system allows help desk employees to record the details of the request, the priority level and the name of the client who generated the request. When a request is registered, it is marked as "open". When it is moved to Level-2, it is marked as "forwarded to Level-2". When the resolution is sent back to Level-1, the request is marked as "returned to Level-1". Finally, when a request is resolved, it is marked as "closed". Every request has a unique identifier. When a request is registered, the ticketing system sends an email to the client. The email includes a so-called request reference number that the client needs to quote when asking questions about the request.

Calculate the cycle time efficiency and the cost-per-execution of the as-is process assuming that:

- Submitting and registering a new request takes 5 min on average.
- Requests spend on average 1 h waiting for a Level-1 staff member to check them. This applies both to new requests and to resubmitted requests.
- Checking if a new request is known takes on average 10 min. In 20% of the cases, the request is known. In this case, it takes between 2 and 10 min (average 5 min) for the Level-1 staff member to communicate the resolution to the client. Once this is done, the request is marked as "closed". On the other hand, if the request is not known, the request is automatically forwarded to Level-2.
- New requests spend on average 2 h waiting for a Level-2 staff member to evaluate them. Level-2 staff take on average 20 min to evaluate a new request.
- Level-2 staff take 5 min to prioritize a request.
- The time between the moment a request has been prioritized and the moment the request is picked up by a Level-2 staff member is 20 h.
- The time required to research and resolve a request is on average 2 h.

- The time to write the resolution to a request is on average 20 min.
- Once a Level-2 staff member has written the resolution of a request, it takes on average 20 h before a the request is fetched from the ticketing system by a Level-1 staff member.
- It takes on average 20 min for a Level-1 staff member to send to the client a problem resolution previously written by a Level-2 staff member.
- It takes on average 20 h between the moment a resolution is sent by the Level-1 staff member and the moment the resolution is tested by the client.
- It takes the client around 10 min to email the test results to the Level-1 staff.
- In 20% of the cases, the request is not resolved and it needs to be forwarded to Level-2 again. In this latter case, it takes about 2 min for the Level-1 staff to forward the request to the Level-2 staff. Unresolved requests that are forwarded in this way are automatically marked as prioritized, since they have already been prioritized in the previous iteration.
- There are no other costs besides the resource costs.

Hint To calculate theoretical cycle time and cost, only take into consideration time spent doing actual work, excluding waiting times and handoffs.

Acknowledgement This exercise is inspired by an example found in [28].

Exercise 7.13 We consider a simplified process for handling a Request for Quote (RFQ) for custom-made metal products at a company called MetalWorks. The process model including processing times and branching probabilities is shown in Figure 7.16. There are two production sales engineers dedicated to this process and one production manager. The production sales engineers work can dedicate up to 32 h per week to this process (each) while the production manager can only dedicate up to 18 h per week to this process. Calculate the theoretical capacity of each of these two resource pools. Which of the pools is the bottleneck pool?

Exercise 7.14 Consider the scenario described in Exercise 7.8 (page 278). The company in question is being pressed by several of its customers to fulfill their orders faster. The company's management estimates that the company stands to lose

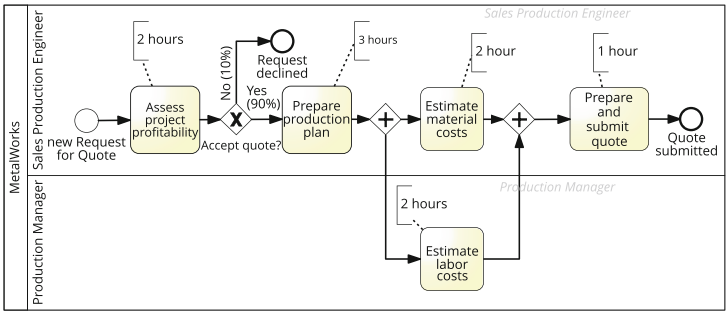

Fig. 7.16 Request for handling a request for quote at MetalWorks

€ 250,000 in revenue, if they do not reduce their order fulfillment time below 40 working days. Adding one engineer to the existing team would reduce the time to design a hardware down to 14 working days (from 16 days). An additional engineer would cost the company € 50,000. On the other hand, hiring a second engineering team would cost € 250,000. Analyze these two scenarios and formulate a recommendation to the company.

Exercise 7.15 We consider a Level-2 IT service desk with 2 staff members. Each staff member can handle one service request in 4 working hours on average. Service times are exponentially distributed. Requests arrive at a mean rate of one request every 3 h according to a Poisson process. What is the average time between the moment a service request arrives to this desk and the moment it is fulfilled?

Exercise 7.16 Consider the Level-2 IT service desk described in Exercise 7.15. Let us assume that the number of requests is one per hour. How many level-2 staff members are required in order to ensure that the mean waiting time of a request is less than two working hours?

Exercise 7.17 Consider again the IT helpdesk process described in Exercise 7.12 (page 291). Model and simulate it assuming that cases arrive at a rate of 50 per day according to an exponential distribution. Assume that all the task cycle times follow an exponential distribution with the average given in Exercise 7.12.

Note When modeling the process, do not model the waiting times between tasks, only the tasks themselves.

Exercise 7.18 Consider the process model in Figure 7.17. This model captures a simplified process for handling mortgage applications. There are two checks involved. CT1 deals with a check of the financial coverage of the mortgage application. The second check CT2 concerns the verification of the property that is to be mortgaged. If the result of both checks is positive, the application is accepted (task AC). On average, after the execution of task CT1, 20% of all applications

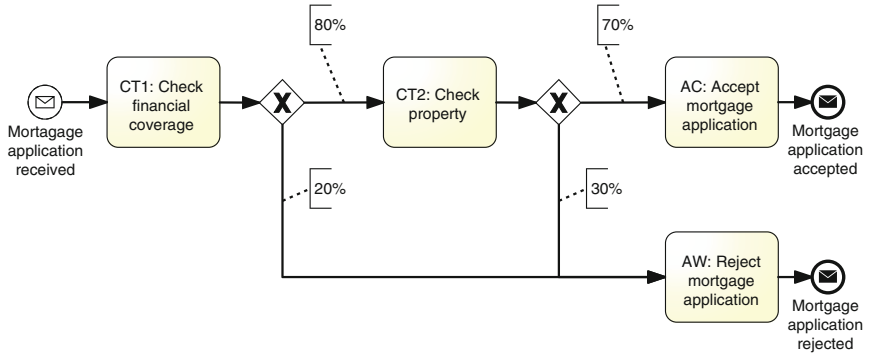

Fig. 7.17 Mortgage process model

are rejected. Meanwhile, task CT2 leads to 30% of further rejections. If either of the checks has an unsatisfactory result, the application is rejected (task AW). The arrival process is Poisson with an average arrival of 5 cases per hour during business hours. For each task, exactly one dedicated resource is available. The processing time of every task follows an exponential distribution. The mean processing times for tasks CT1, CT2, AC, and AW are respectively 5, 4, 3, and 3 min. The wage of each resource is € 20 per hour. Business hours are from Monday to Friday from 9 a.m. to 5 p.m. Resources are only available during these hours.

a. Determine the resource utilization of each resource.
b. Determine the average cycle time of the process.
c. Determine the cycle time efficiency of the process.
d. Determine the average number of mortgage applications that are being handled at any given point in time.

Hint For this exercise, it might be convenient to use a combination of process simulation, Little's law, and flow analysis.

7.7 Further Readings

In Section 7.1, we showed how flow analysis techniques can be used to calculate cycle time and cost. Laguna & Marklund [85] discuss flow analysis in detail. Another possible application of flow analysis is to estimate the error rate of the process, meaning the number of cases that will end up in a negative outcome. This latter application of flow analysis is discussed for example by Yang et al. [196]. Yang et al. also present a technique for flow analysis that is applicable not only to block-structured process models but also to a much broader class of process models.

As mentioned in Section 7.2, the formula for determining the average queue length in the context of the M/M/c model is particularly complicated. Laguna & Marklund [85, Chapter 6] analyze the M/M/c model (including the formula for average queue length) and its application to process analysis. They also analyze the M/M/c/K model, where an upper-bound to the length of the queue is imposed (this is parameter K in the model). The M/M/c/K model is suitable for example when there is a maximum length of queue beyond which customers are rejected from the queue. Adan & Resing [3] give detailed introductions to M/M/1, M/M/c, M/M/c/K and other queueing theory models.

As stated in Section 7.3, business process simulation is a versatile approach for quantitative process analysis. Numerous case studies illustrating the use of process simulation in various domains can be found in the literature. For example, Greasley [58] illustrates the use of business process simulation for redesigning a process for road traffic accident reporting. In a similar vein, Van der Aalst et al. [178] discuss the use of business process simulation to evaluate different strategies to avoid or to mitigate deadline violations in the context of an insurance claims handling process in an insurance company. Exercise 7.10 is based on this latter paper.

Current tools for business process simulation have various limitations. Several of these limitations are discussed at length by Van der Aalst et al. [177]. Research by Martin et al. discusses how data about previous executions of the process can be used to build more accurate simulation models [104, 105]. Van der Aalst et al. [177] propose to use more sophisticated tools for process simulation, namely Discrete-Event Simulation (DES) tools. They specifically put forward *CPN Tools* as a possible DES that can be used for business process simulation. CPN Tools is based on *Colored Petri Nets*—a language that extends Petri nets. Other DES tools that can be used for business process simulation include ExtendSim [85] and Arena [75]. For example, Arena is used in the aforementioned case study of a road traffic reporting process [58]. DES tools are clearly more powerful than specialized business process simulation tools. However, the choice of a DES tool means that one cannot directly use a BPMN model for simulation. Instead the model has to be re-encoded in another notation. Moreover, the use of DES tools requires more technical background from the analyst. These trade-offs should be considered when choosing between DES tools and specialized business process simulation tools based for example on BPMN.

We saw throughout the chapter that quantitative analysis techniques allow us to identify critical paths and bottlenecks. These are essentially paths in the process that require special attention if the goal is to reduce cycle time. Anupindi et al. [9] offer detailed advice on how to deal with critical paths and bottlenecks in business processes as well as how to reduce waste and repetition. The following chapter will discuss some of these insights.