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 preparing to make a case to the company's COO that every site engineer should be given a tablet computer with wireless access in order to query suppliers' catalogs and to make rental requests from any construction site. The process owner will be asked to substantiate this investment in quantitative terms and specifically to estimate how much time and money would be saved by doing this investment. To make such estimates, we need to go beyond qualitative analysis.

This chapter introduces a range of techniques for analyzing business processes quantitatively, in terms of performance measures such as cycle time, total waiting time and cost. Specifically, the chapter focuses on three techniques: flow analysis, queueing analysis and simulation. All these techniques have in common that they allow us to calculate performance measures of a process, given data about the performance of individual activities and resources in the process.

7.1 Performance Measures

7.1.1 Process Performance Dimensions

Any company would ideally like to make its processes faster, cheaper, and better. This simple observation leads us already to identifying three *process performance dimensions*: time, cost and quality. A fourth dimension gets involved in the equation once we consider the issue of change. A process might perform extremely well under normal circumstances, but then perform poorly in other circumstances which are perhaps equally or more important. For example, van der Aalst et al. [98] report the story of a business process for handling claims at an Australian insurance

DOI 10.1007/978-3-642-33143-5_7, © Springer-Verlag Berlin Heidelberg 2013

company. Under normal, everyday conditions, the process was performing to the entire satisfaction of all managers concerned (including the process owner). However, Australia is prone to storms and some of these storms cause serial damages to different types of properties (e.g. houses and cars), leading to numerous claims being lodged in a short period of time. The call center agents and backoffice workers involved in the process were over-flooded with claims and the performance of the process degraded—precisely at the time when the customers were most sensitive to this performance. What was needed was not to make the process faster, cheaper or better during normal periods. Rather, it was needed to make the process more flexible to sudden changes in the amount of claims. This observation leads us to identify a fourth dimension of process performance, namely flexibility.

Each of the four performance dimensions mentioned above (time, cost, quality, and flexibility) can be refined into a number of *process performance measures* (also called *key performance indicators* or *KPIs*). A process performance measure is a quantity that can be unambiguously determined for a given business process—assuming of course that the data to calculate this performance measure is available.

For example, there are several types of cost such as cost of production, cost of delivery or cost of human resources. Each of these types of cost can be further refined into specific performance measures. To do so, one needs to select an aggregation function, such as count, average, variance, percentile, minimum, maximum, or ratios of these aggregation functions. A specific example of a cost performance measure is the average delivery cost per item.

Below, we briefly discuss each of the four dimensions and how they are typically refined into specific performance measures.

Time Often the first performance dimension that comes to mind when analyzing processes is time. Specifically, a very common performance measure for processes is *cycle time* (also called *throughput time*). Cycle time is the time that it takes to handle one case from start to end. Although it is usually the aim of a redesign effort to reduce cycle time, there are many different ways of further specifying this aim. For example, one can aim at a reduction of the average cycle time or the maximal cycle time. It is also possible to focus on the ability to meet cycle times that are agreed upon with a client at run time. Yet another way of looking at cycle time is to focus on its variation, which is notably behind approaches like Six Sigma (cf. Chap. 1). Other aspects of the time dimension come into view when we consider the constituents of cycle time, namely:

1. *Processing time* (also called *service time*): the time that resources (e.g. process participants or software applications invoked by the process) spend on actually handling the case.
2. *Waiting time*: the time that a case spends in idle mode. Waiting time includes *queueing time*—waiting time due to the fact that no resources available to handle the case—and other waiting time, for example because synchronization must take place with another process or because input is expected from a customer or from another external actor.

Cost Another common performance dimension when analyzing and redesigning a business process has a financial nature. While we refer to cost here, it would also have been possible to put the emphasis on turnover, yield, or revenue. Obviously, a yield increase may have the same effect on an organization's profit as a decrease of cost. However, process redesign is more often associated with reducing cost. There are different perspectives on cost. In the first place, it is possible to distinguish between fixed and variable cost. Fixed costs are overhead costs which are (nearly) not affected by the intensity of processing. Typical fixed costs follow from the use of infrastructure and the maintenance of information systems. Variable cost is positively correlated with some variable quantity, such as the level of sales, the number of purchased goods, the number of new hires, etc. A cost notion which is closely related to productivity is *operational cost*. Operational costs can be directly related to the outputs of a business process. A substantial part of operational cost is usually labor cost, the cost related to human resources in producing a good or delivering a service. Within process redesign efforts, it is very common to focus on reducing operation cost, particularly labor cost. The automation of tasks is often seen as an alternative for labor. Obviously, although automation may reduce labor cost, it may cause incidental cost involved with developing the respective application and fixed maintenance cost for the life time of the application.

Quality The quality of a business process can be viewed from at least two different angles: from the client's side and from the process participant's side. This is also known as the distinction between external quality and internal quality. The *external quality* can be measured as the client's satisfaction with either the product or the process. Satisfaction with the product can be expressed as the extent to which a client feels that the specifications or expectations are met by the delivered product. On the other hand, a client's satisfaction with the process concerns the way how it is executed. A typical issue is the amount, relevance, quality, and timeliness of the information that a client receives during execution on the progress being made. On the other hand, the *internal quality* of a business process related to the process participants' viewpoint. Typical internal quality concerns are: the level that a process participants feels in control of the work performed, the level of variation experienced, and whether working within the context of the business process is felt as challenging. It is interesting to note that there are various direct relations between the quality and other dimensions. For example, the external process quality is often measured in terms of time, e.g., the average cycle time or the percentage of cases where deadlines are missed. In this book, we make the choice that whenever a performance measure refers to time, it is classified under the time dimension even if the measure is also related to quality.

Flexibility The criterion that is least noted to measure the effect of a redesign measure is the flexibility of a business process. Flexibility can be defined in general terms as the ability to react to changes. These changes may concern various parts of the business process, for example:

- The ability of resources to execute different tasks within a business process setting.
- The ability of a business process as a whole to handle various cases and changing workloads.
- The ability of the management in charge to change the used structure and allocation rules.
- The organization's ability to change the structure and responsiveness of the business process to wishes of the market and business partners.

Another way of approaching the performance dimension of flexibility is to distinguish between run time and build time flexibility. *Run time flexibility* concerns the opportunities to handle changes and variations while executing a specific business process. *Build time flexibility* concerns the possibility to change the business process structure. It is increasingly important to distinguish the flexibility of a business process from the other dimensions.

Example 7.1 Let us consider the following scenario.

> A restaurant has recently lost many customers due to poor customer service. The management team has decided to address this issue first of all by focusing on the delivery of meals. The team gathered data by asking customers about how quickly they liked to receive their meals and what they considered as an acceptable wait. The data suggested that half of the customers would prefer their meals to be served in 15 minutes or less. All customers agreed that a waiting time of 30 minutes or more is unacceptable.

In this scenario, it appears that the most relevant performance dimension is time, specifically serving time. One objective that distills from the scenario is to completely avoid waiting times above 30 minutes. In other words, the percentage of customers served in less than 30 minutes should be as close as possible to 100 %. Thus, "percentage of customers served in less than 30 minutes" is a relevant performance measure. Another threshold mentioned in the scenario is 15 minutes. There is a choice between aiming to have an average meal serving time below 15 minutes or again, minimizing the number of meals served above 15 minutes. In other words, there is a choice between two performance measures: "average meal delivery time" or "percentage of customers served in 15 minutes".

This example illustrates that the definition of performance measures is tightly connected to the definition of *performance objectives*. In this respect, one possible method for deriving performance measures for a given process is the following:

1. Formulate performance objectives of the process at a high level, in the form of a desirable state that the process should ideally reach, e.g. "customers should be served in less than 30 minutes".
2. For each performance objective, identify the relevant performance dimension(s) and aggregation function(s), and from there, define one or more performance measures for the objective in question, e.g. "percentage of customers served in less than 30 minutes". Let us call this measure ST_{30}.
3. Define a more refined objective based on this performance measure, such as $ST_{30} \geq 99\ \%$.

During the redesign and implementation phases, a possible additional step is to attach a timeframe to the refined performance objective. For example, one can state that the above performance objective should be achieved in 12 months time. A performance objective with a timeframe associated to it is usually called a *performance target*. At the end of the chosen timeframe, one can assess to what extent the redesigned process has attained its targets.

Exercise 7.1 Consider the travel agency scenario described in Exercise 6.6 (p. 208).

1. Which business processes should the travel agency improve?
2. For each of the business processes you identified above, indicate which performance measure should the travel agency improve.

7.1.2 Balanced Scorecard

Another way of classifying and defining performance measures is given by the concept of *Balanced Scorecard*. The Balanced Scorecard is essentially an approach to align the goals and measures that are used to evaluate the work of managers. The main argument behind the Balanced Scorecard is that it is not sufficient to use financial metrics, such as Return-On-Investment (ROI) or operating margin, when evaluating managers. An extreme focus on these measures is in the long-term detriment to the company as it neglects fundamental sources of value, namely the customer, the company's internal structure and the company's employees. Accordingly, the Balanced Scorecard is based on four performance dimensions—each one covering a fundamental concern of a company:

- Financial Measures, e.g. cash flow, to ensure survival, operating margin to ensure shareholder satisfaction.
- Internal Business Measures, e.g. cycle time, to ensure efficiency and low levels of inventory in the case of manufacturing organizations.
- Innovation and Learning Measures, e.g. technology leadership, to ensure competitive advantage and to attract and retain talent.
- Customer Measures, e.g. on-time delivery, to ensure customer satisfaction and loyalty.

The classical way to implement the Balanced Scorecard follows a top-down procedure. It begins with a corporate scorecard, followed by departmental ones with an emphasis on goals and metrics directly affected by the specific department. Process-related measures tend to appear only at the level of heads of units or their subordinates. This classical implementation of the Balanced Scorecard overemphasizes the functional division of organizations not paying enough attention to processes view. Companies implementing the Balanced Scorecard in conjunction with BPM need to carefully consider the relation between the measures in the Balanced Scorecard—both at the corporate level, departmental level and lower levels—and the performance measures associated with their business processes. One way to ensure this alignment is to implement a Balanced Scorecard structured according to the company's process architecture (cf. Chap. 2). This process-oriented Balanced Scorecard may co-exist with a Balanced Scorecard that is traditionally associated to the company's functional architecture.

In any case, we observe that the Balanced Scorecard is a useful tool for identifying process performance measures across an entire organization. This is in contrast with the method based on performance dimensions outlined in Sect. 7.1.1 is geared towards identifying performance measures for one given process. Thus, this latter method and the Balanced Scorecard are complementary.

7.1.3 Reference Models and Industry Benchmarks

Reference process models—previously mentioned in Chap. 2—provide another basis to identify process performance measures. For instance, within the Supply Chain Operations Reference Model (SCOR), processes and activities in the process hierarchy are linked to performance measures. An example of a performance measure in SCOR is the "Purchase Order Cycle Time", defined as the "average amount of time (e.g. days) between the moment an intention to purchase is declared and the moment the corresponding purchase order is received by the relevant vendor". This is basically the average cycle time of a fragment of a procure-to-pay process. Other measures in SCOR deal with inventory management or out-of-stock events. In addition to defining performance measures, SCOR also provides threshold values for each measure that allow a company to compare itself against peers within its industry and to determine whether they are in the top-10 %, top-50 % or bottom-50 % with respect to other companies in their industry sector.

Another relevant framework mentioned in Chap. 2 is APQC's Process Classification Framework (PCF). The primary aim of this framework is to provide a standardized decomposition of processes in an organization together with standardized names and definitions for these processes. As a complement to PCF, APQC has also developed a set of performance measures for the processes included in PCF. This is also a potentially useful tool for performance measure identification.

Yet another example of a reference model that provides a catalog of process performance measures is the *IT Infrastructure Library*—ITIL. ITIL's performance measures include, for example, "Incidents due to Capacity Shortages" defined as the "number of incidents occurring because of insufficient service or component capacity". This performance measure is linked to ITIL's Capacity Management process area, which includes a number of inter-related processes to manage the capacity of IT processes or components of an IT system.

Other reference models that provide catalogs of process performance measures include DCOR (Design Chain Operations Reference model) and eTOM (Enhanced Telecom Operations Map).

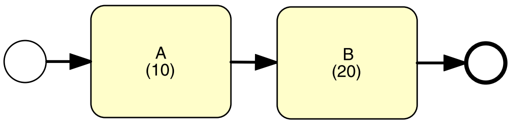


Fig. 7.1 Fully sequential process model

7.2 Flow Analysis

Flow analysis is a family of techniques that allow us to estimate the overall performance of a process given some knowledge about the performance of its activities. 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 activity. We can also use flow analysis to calculate the average cost of a process instance knowing the cost-per-execution of each activity, or calculate the error rate of a process given the error rate of each activity.

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.

7.2.1 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 an activity is the average time it takes between the moment the activity is ready to be executed 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 Fig. 7.1. The cycle time of each activity is indicated between brackets. Since the two activities in this process are performed one after the other, we can intuitively conclude that the cycle time of this process is $20 + 10 = 30$. More generally, it is quite intuitive that the cycle time of a purely sequential fragment of a process is the sum of the cycle times of the activities in the fragment.

When a process model or a fragment of a model contains gateways, the cycle time is no longer the sum of the activity cycle times. Let us consider the example shown in Fig. 7.2. Here, it is clear that the cycle time of the process is not 40 (the sum of the activity cycle times). Indeed, in a given instance of this process, either activity B or activity C is performed. If B is performed, the cycle time is 30, while if C is performed, the cycle time is 20.

Whether the cycle time of this process is closer to 20 or closer to 30 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. However, if the

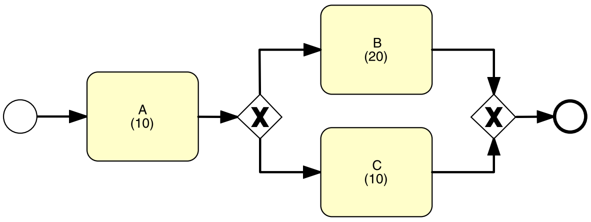


Fig. 7.2 Process model with XOR-block

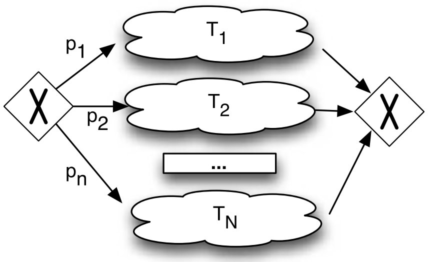


Fig. 7.3 XOR-block pattern

upper branch is taken only 10 % of the times and the lower branch is taken 90 % of the times, the cycle time should be intuitively closer to 30. 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 lower branch has a frequency of 10 % and the upper branch has a frequency of 90 %, the cycle time of the fragment between the XOR-split and the XOR-join is: $0.1 \times 10 + 0.9 \times 20 = 19$. We then need to add the cycle time of activity A (which is always executed) in order to obtain the total cycle time, that is, $10 + 19 = 29$. 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 Fig. 7.3 is

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

In Fig. 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. Hereon, this type of fragment is called a *XOR-block*.

Let us now consider the case where parallel gateways are involved as illustrated in Fig. 7.4.

Again, we can observe that the cycle time of this process cannot be 40 (the sum of the activity cycle times). Instead, since tasks B and C are executed in parallel, their combined cycle time is determined by the slowest of the two activities, that is,

Fig. 7.4 Process model with AND-block

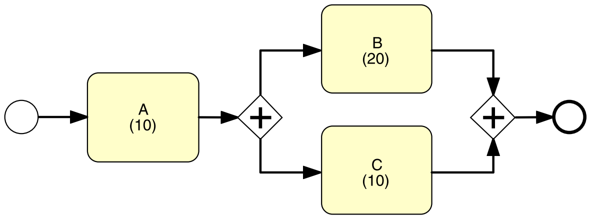
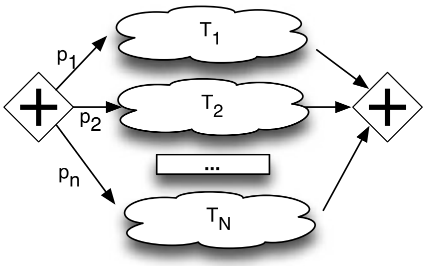


Fig. 7.5 AND-block pattern



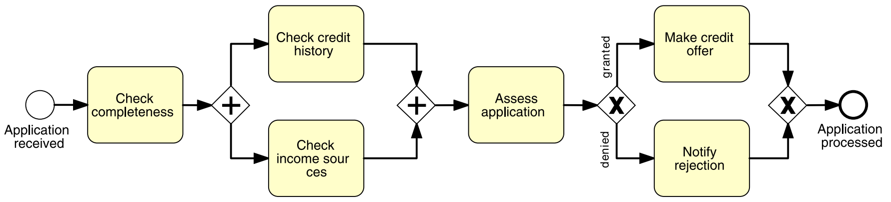


Fig. 7.6 Credit application process

by C. Thus, the cycle time of the process shown in Fig. 7.4 is $10 + 20 = 30$. More generally, the cycle time of an *AND-block* such as the one shown in Fig. 7.5 is

$$CT = Max(T_1, T_2, \ldots, T_n) \tag{7.2}$$

Example 7.2 Let us consider the credit application process model in Fig. 7.6 and the activity cycle times given in Table 7.1. Let us also assume that in 60 % of 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 (slowest activity). Next, we calculate the cycle time of the fragment between the XOR-block using (7.1), 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.

In this example the cycle time is in great part determined by task “Check income sources”, which is the one that determines the cycle time of the fragment between the AND-split and the AND-join. In this case, we say that this task is part of the *critical path* of the process. The critical path of a process is the sequence of tasks that determines the cycle time of the process, meaning that the cycle time of any instance of the process is never lower than the sum of the cycle times of this sequence of

Table 7.1 Cycle times for credit application process

Activity	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

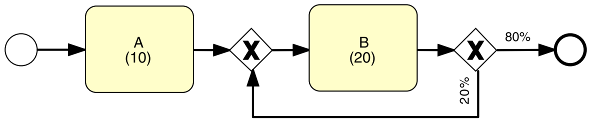


Fig. 7.7 Example of a rework loop

tasks. When optimizing a process with respect to cycle time, one should focus the attention on tasks that belong to the critical path.

Exercise 7.2 Consider the process model given in Fig. 3.8 (p. 73). Calculate the cycle time under the following assumptions:

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

Compare the process model in Fig. 3.8 (p. 73) with the one in Fig. 3.10 (p. 74). Does this comparison give you an idea of how to calculate cycle times for process models with OR gateways?

Another recurrent case worth considering is the case where a fragment of a process may be repeated multiple times. This situation is called *rework* and is illustrated in Fig. 7.7. Here the decimal numbers attached to the arcs denote the probability that the corresponding arc will be taken. For sure, we can say that activity B will be executed once. Next, we can say that activity B may be executed twice 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 find out that the probability that task B is executed three times is $0.2 \times 0.2 = 0.04$, and more generally, the probability that task B is executed N times is 0.2^N.

If we sum up the cycle times in the cases where B is executed once, twice, three times, etc., we get the following summation $\sum_{i=0}^{\infty} 0.2^i$. In essence, this is the number of times that B is expected to be executed. 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 $1/(1-r)$. Hence, the average number of times that B is expected to be executed is $1/(1-0.2) = 1.25$. Now, if we multiply

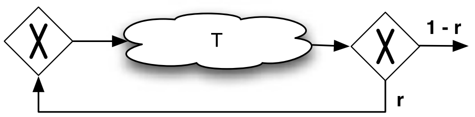


Fig. 7.8 Rework pattern

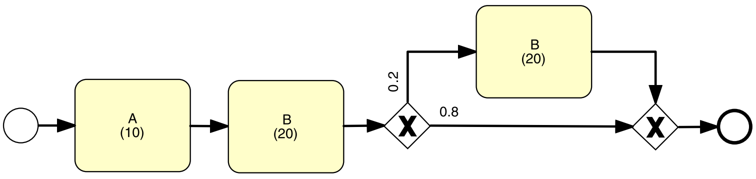


Fig. 7.9 Activity that is reworked at most once

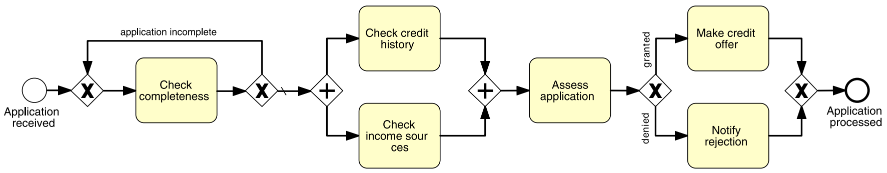


Fig. 7.10 Credit application process with rework

this expected number of instances of B times the cycle time of activity B, we get $1.25 \times 20 = 25$. Thus the total cycle time of the process in Fig. 7.7 is $10 + 25 = 35$.

More generally, the cycle time of the fragment with the structure shown in Fig. 7.8 is

$$CT = \frac{T}{1 - r}. \tag{7.3}$$

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

In some scenarios, an activity is reworked at most once. This situation would be modeled as shown in Fig. 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 complete this, we have to add the cycle time of the preceding activities, giving us a total cycle time of 34.

Example 7.3 Let us consider the credit application process model in Fig. 7.10 and the cycle times previously given in Table 7.1. Let us also assume that in 20 % of the cases, the application is incomplete and in 60 % of cases the credit is granted.

The cycle time of the rework block is $10/(1 - 0.2) = 1.25$ 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 7.2. Thus the total cycle time is $1.25 + 3 + 3 + 1.4 = 8.65$ days.

7.2.2 Cycle Time Efficiency

As previously mentioned, the cycle time of an activity 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. This includes time spent in transferring information about the case between process participants, like for example when documents are exchanged by post, as well as time when the case is waiting for an actor to process it. Processing time on the other hand refers to the time that actors spend doing actual work. In many if not most processes, the waiting time makes up a considerable proportion of the overall cycle time. There are a variety of reasons for this phenomenon. For example, this situation may happen because work is performed in batches. In a process related to the approval of purchase requisitions at a company, the supervisor responsible for such approvals in a business unit might choose to batch all applications and check them only once at the start or the end of a working day. Also, sometimes waiting time is spent waiting for an external actor to provide some 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 and so 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, for example due to handovers between participants.

The cycle time efficiency of a process can be calculated as follows. First, we need to determine the cycle time and the processing time of each activity. Given this information, we can then calculate the overall cycle time of the process using the same formulas we saw above. Let us call this amount CT. Next, using the same formulas, we can calculate the overall amount of time that is spent doing actual work. This is called the *theoretical cycle time* of the process. Essentially, this is the amount of time that an instance of the process would take if there was no waiting time at all. To calculate the theoretical cycle time, we apply the same method as for calculating cycle time, but instead of using the cycle time of each activity, we use the processing time of each activity. Let us call the theoretical cycle time TCT. The cycle time efficiency (CTE) is then calculated as follows:

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

Example 7.4 Let us consider the credit application process model in Fig. 7.10 and the processing times given in Table 7.2. The activity cycle times (including both waiting and processing time) are those previously given in Table 7.1. Let us assume

Table 7.2 Processing times for credit application process

Activity	Cycle time
Check completeness	2 hours
Check credit history	30 minutes
Check income sources	3 hours
Assess application	2 hours
Make credit offer	2 hours
Notify rejection	30 minutes

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

Activity	Cycle time	Processing time
Register ministerial enquiry	2 days	30 mins
Investigate ministerial enquiry	8 days	12 hours
Prepare ministerial response	4 days	4 hours
Review ministerial response	4 days	2 hour

that in 20 % of cases, the application is incomplete and in 60 % of cases the credit is "granted". Let us additionally assume that one day is equal to 8 working hours.

We have seen in Example 7.3 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 in the same way as the total cycle time but using the processing times given in Table 7.2. This gives us: $2/(1-0.2)+3+2+0.6\times 2+0.4\times 0.5=9.9$ working hours. The cycle time efficiency is thus $8.9/69.2=12.9$ %.

Exercise 7.3 Calculate the overall cycle time, theoretical cycle time and cycle time efficiency of the ministerial enquiry process introduced in Exercise 3.7 (p. 77). Assume that the rework probability is 0.2 and that the waiting times and processing times are those given in Table 7.3.

7.2.3 Cycle Time and Work-In-Process

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 is used to refer to the arrival rate.

Meanwhile, 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.

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

$$WIP = \lambda \times CT$$

Basically 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 concurrently. 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.

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 total cycle time of a process if we know the arrival rate and WIP. This can be useful sometimes 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 2500, 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 concurrently. The cycle time is then $WIP/\lambda = 200/10 = 20$ business days.

Exercise 7.4 A restaurant receives on average 1,200 customers per day (between 10am and 10pm). During peak times (12pm to 3pm and 6pm to 9pm), 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?

7.2.4 Other Applications and Limitations of Flow Analysis

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 activity, 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 activities is the sum of the costs of these activities. 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 Fig. 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 Fig. 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 and therefore the costs of these branches add up to one another.

Example 7.5 Let us consider again the credit application process model in Fig. 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 activities "Check completeness", "Check credit history" and "Check income sources" are performed by a clerk, while activity "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 submits 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 *human resource cost* and *other costs*. The human resource cost is the cost of the human resource(s) 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 in the task. In this example, the cost per query to the external system would be classified as "other costs" for task "Check credit history". The remaining tasks do not have an "other cost" component. For the example at hand, the breakdown of resource cost, other cost and total cost per task is given in Table 7.4. Given this input, we can calculate the total cost-per-execution of the process as follows: $50/(1 - 0.2) + 13.5 + 75 + 100 + 0.6 \times 100 + 0.4 \times 25 = 321$.

Table 7.4 Cost calculation table for credit application process

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

Exercise 7.5 Calculate the cost-per-execution of the ministerial enquiry process introduced in Exercise 3.7 (p. 77). Assume that the rework probability is 0.2 and that the times are those given in Table 7.3. Activity "Register ministerial enquiry" is performed by a clerk, activity "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 activities besides the resource costs.

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 Sect. 7.2.1 do not allow us to calculate the cycle time of any 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 3.9 (p. 93). Indeed, this example does not fit into any of the patterns we have seen above. Calculating the cycle time in this case is trickier. 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 specific set of equations discussed in Sect. 7.2.1. There are other more sophisticated flow analysis techniques that allow us to calculate the cycle time of virtually any process model. The maths can get a bit more complex and in practice, one would not do such calculations by hand. 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 activity in the process model. In fact, this obstacle is typical 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 activity, both in terms of waiting time and processing time. A second approach is to collect logs from the information systems used in the process. For example, if a given activity such as approving a purchase requisition is performed by means of an internal Web portal (an Intranet), the administrators of the portal should be able to extract logs from this portal that would allow the analyst to estimate the average amount of time that a requisition form spends in "waiting for approval" mode and also the average time between the moment the supervisor opens a purchase requisition for approval and the time they approve it. With careful analysis, these logs can provide a wealth of information that can be combined via flow analysis to get an overall picture of which parts of the process consume the most time.

A major limitation of flow analysis is that it does not take into account the fact that a process behaves differently depending on the load, that is, depending on the amount of instances of the process that are running concurrently. 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 are freed up. Flow analysis does not take into account 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.

7.3 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's office, 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.

7.3.1 Basics of Queueing Theory

In basic queueing theory, a *queueing system* is seen as consisting 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. 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. It would be unrealistic to introduce all these techniques in this chapter. So 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 activities within a process.

In the two models we will be presenting, there is a single queue (single-line queueing system). Customers come at a given mean arrival rate that we will call 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 at the bank office at a mean rate of 20 per hour. This implies that, on average, one customer arrives every 5 minutes ($\frac{1}{20}$ hour). 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 post office is always 5 minutes. 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 that the time between the arrival of the first customer and the arrival of the second customer is 1 minute. This observation does not tell us absolutely 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 minute after the second, or 5 minutes or 10 minutes. 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 between any two consecutive customers 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 the time of inter-arrival time being 10 is considerably smaller than the probability of the inter-arrival time being 1. 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 a large number.

In practice, the Poisson process and the exponential distribution describe a large class of arrival processes that can be found in business processes, so we will be using them to capture the arrival of jobs or customers into a business process or an activity in a business process. The Poisson process can also be observed for example 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 to a given process or activity 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 for example R, Mathworks's Statistical Toolbox or EasyFit. These tools allow one to input a set of observed inter-arrival times and 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 an activity.[1] In the case of activities that require a diagnosis, a non-trivial verification or some non-trivial decision making, it is often the case that the activity's 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 for example one hour to do them. However, some repairs are very 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 activities will follow an exponential distribution. As mentioned above, when making such a hypothesis, it is important that you first check 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* if the inter-arrival times of customers follow an exponential distribution, the processing times follow an exponential distribution, there is one single server and jobs are served on a First-In-First-Out (FIFO) basis. In the case of M/M/1 queue, we also assume that when a job 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 number of servers. For example, a queue 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 five 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.

[1] In queueing theory, the term service time is used instead of processing time. For uniformity purposes, here we use the term processing time.

7.3.2 M/M/1 and M/M/c Models

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

- λ—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 jobs is $1/5$ hours, that is 12 minutes.
- μ—the mean number of customers that can be served per time unit. The mean processing time per job is then $1/\mu$. For example, $\mu = 6$ means six jobs are served per hour, that is, one job is served in 10 minutes (on average).
- In the case of M/M/c, the number of servers (c).

Given parameters λ and μ, we can already state how busy a server is. This is called the occupation rate ρ and is equal to λ/μ. In the above example, the occupation rate is $5/6 = 83.34$ %. It should be noted that this is a relatively high occupation rate. A system with an occupation rate of more than 100 % is unstable, meaning that the queue will become longer and longer forever because the server cannot cope with all the demand. In fact, even a system at close to 100 % of occupation rate is unstable because of the randomness at which new jobs arrive and the variability in the processing times per job. To understand why this is the case, just imagine if you were a doctor receiving patients at a rate of 6 per hour for 8 hours, knowing that every patient takes 10 minutes on average to be treated (sometimes less but sometimes more). Without any slack, most likely you will end up with a tremendous backlog at the end of the day.

In the case of an M/M/c system, the occupancy rate is $\frac{\lambda}{c\mu}$ since the system can handle jobs at a rate of $c\mu$. For example, if the system has two servers and each server can handle two jobs per hour, the system can handle four jobs per hour. If jobs arrive at a mean rate of 3 per hour, the occupancy rate 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—The average number of jobs (e.g. customers) in the queue.
- W_q—The average time one job spends in the queue.
- W—The average time one job spends in the system. This includes both the time the customer spends in the queue but also the time the customer spends being serviced.
- L—The average number of jobs in the system (i.e. the Work-in-Progress referenced in Little's law).

To summarize, the general structure of a single-queue system—consisting of one queue and one or many servers—is depicted in Fig. 7.11. 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 a job waits in the queue is W_q, while the average length of the queue is L_q. Eventually, a job goes into the server and in there it spends on average $1/\mu$

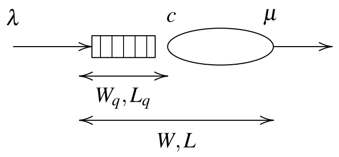


Fig. 7.11 Structure of an M/M/1 or M/M/c system, input parameters and computable parameters

time units. The average time between the moment a job enters the system and the moment it exits is W, while the average number of jobs inside the system (in the queue or in a server) is L.

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

$$L_q = \rho^2/(1-\rho) \tag{7.4}$$

$$W_q = \frac{L_q}{\lambda} \tag{7.5}$$

$$W = W_q + \frac{1}{\mu} \tag{7.6}$$

$$L = \lambda W \tag{7.7}$$

Formulas (7.5), (7.6), and (7.7) 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:

$$L_q = \frac{(\lambda/\mu)^c \rho}{c!(1-\rho)^2(\frac{(\lambda/\mu)^c}{c!(1-\rho)} + \sum_{n=0}^{c-1}\frac{(\lambda/\mu)^n}{n!})} \tag{7.8}$$

The formula for computing L_q in the case of M/M/c models is particularly complicated because of the summations and factorials. Fortunately, there are tools that can do this for us. For example, the Queueing Toolpack[2] 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 jobs allowed in the queue. Jobs that arrive when the length of the queue is k are rejected (and may come back later). Other tools for analyzing queueing systems include QSim[3] and PDQ.[4]

Example 7.6 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.

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

[2]http://apps.business.ualberta.ca/aingolfsson/qtp/.

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

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

on a FIFO manner. Note that even though the team includes several people, they act as a monolithic entity and therefore it should be treated as a single server.

We will hereby take the working day as a time unit. On average, 0.05 orders are received per day ($\lambda = 0.05$), and 0.2 orders are fulfilled per day ($\mu = 0.1$). Thus, the occupation rate 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 on the queue is $W_q = 0.5/0.05 = 10$ days. Thus, it takes on average order $W = 10 + 1/0.1 = 20$ working days for an order to be fulfilled.

Exercise 7.6 Consider now the case where the engineering team in the previous example takes 16 working days to design a hardware. What is then the average amount of time an order takes to be fulfilled?

Exercise 7.7 An insurance company receives 220 calls per day from customers who want to lodge an insurance claim. The call center is open from 8am to 5pm. 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 8am to 11am, the period 11am to 2pm and the period 2pm to 5pm. During the first period, around 60 calls are received. During the 11am–2pm period, 120 calls are received, and during the 2pm–5pm period, 40 calls are received. A customer survey has shown that customers tend to call between 11am and 2pm because during this time they have a break at work and they take advantage of their break to make their personal calls.

Statistical analysis shows that the durations of calls follow an exponential distribution.

According to the company's customer service charter, customers should wait no more than one minute on average for their call to be answered.

- Assume that the call center can handle 70 calls per hour using seven call center agents. Is this enough to meet the 1-minute 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 eight call center agents)?
- The call center manager has been given 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 one minute.

7.3.3 Limitations of Basic Queueing Theory

The basic queueing analysis techniques presented above allow us to estimate waiting times and queue length 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 very 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 was intended 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 activity at a time. When we have to analyze an entire process that involves several activities, 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 activities. A more popular approach for quantitative analysis of process models under varying levels of resource contention is process simulation, as discussed below.

7.4 Simulation

Process simulation is arguably the most popular and most widely supported technique for quantitative analysis of process models. The basic idea underpinning process simulation is quite simple. In essence, a process simulator generates a large number of hypothetical instances of a process, executes these instances step-by-step, and records each step in this execution. The output of a simulator typically includes the logs of the simulation as well as some statistics related to cycle times, average waiting times and average resource utilization.

7.4.1 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 found, the simulator puts the work item in waiting mode until a suitable resource is freed up. 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's 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's 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 hours, 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. Luckily, 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 time stamps:

- 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 very 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.[5]

7.4.2 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:

- 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 set of resources that are able to perform the task. This set is usually called a *resource pool*. For example, a possible resource pool could be the "Claim Handlers" or "Clerks" or "Managers". Separately, the analyst needs to specify for each resource pool the number of resources in this pool (e.g. the number of claim handlers or the number of clerks) and other attributes of these resources such as the hourly cost (e.g. the hourly cost of a claims handler).

[5]Note that when discussing queueing theory above, we used the term occupation rate instead of resource utilization. These two terms are synonyms.

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 seconds.
- *Exponential distribution.* As discussed in Sect. 7.3, 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. You can imagine that in most cases, the insurance claims fall within very standard cases. In such 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, meaning 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 minutes to make such a check. In some cases, this time can be lower because for example the form is clearly incomplete or clearly complete, and 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.

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 derived through an informed guess (based on interviews with the relevant stakeholders), but preferably by means of sampling (the analyst collects data for a sample of tasks executions) or by analyzing 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 arc stemming from a decision gateway. These probabilities are determined by interviewing relevant stakeholders, observing executions of the process during a certain 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 inter-arrival times and the mean arrival rate. 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. 2012 at 8:00").
- One of the following:
  - The end date and time of the simulation. If this option is taken, 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 taken, 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 7.7 We consider the process for loan application approval modeled in Fig. 4.6 (p. 104). We simulate this model using the BIMP simulator available at: http://bimp.cs.ut.ee. This simulator takes as input BPMN process models in XML format produced by other process modeling tools such as Signavio Process Editor or OpenText Provision. We provide the following inputs for the simulation.

- Two loan applications per hour, meaning an inter-arrival time of 30.
- Tasks "Check credit history" and "Check income sources" are performed by clerks.
- Tasks "Notify rejection", "Make credit offer" and "Assess application" are performed by credit officers.
- Task "Receive customer feedback" is in fact an event. It takes zero time and it only involves the credit information system (no human actors involved). To capture this, the task is assigned to a special "System" role.
- There are three clerks and three credit officers. The hourly cost of a clerk is € 25 while that of a credit officer is € 50.
- Clerks and credit officers work from 9am to 5pm during weekdays.
- The cycle time of task "Assess application" follows an exponential distribution with a mean of 20 minutes.

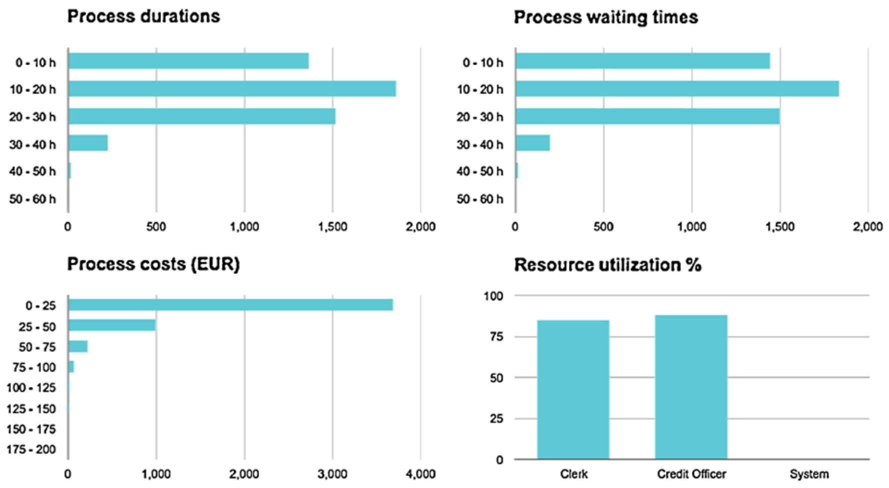


Fig. 7.12 Histograms produced by simulation of the credit application process

- Cycle times of all other tasks follow a normal distribution. Tasks “Check credit history”, “Notify rejection” and “Make credit offer” have a mean cycle time of 10 minutes with a 20 % standard deviation, while “Check income sources” has a cycle time of 20 minutes 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 5,000 instances, which means around 104 days of loan applications arrivals assuming that applications arrive 24 hours a day, 7 days a week.[6] The simulation gives an average cycle time of around 17 hours. A variance of ± 2 hours can be observed when running the simulation multiple times. This variance is expected due to the stochastic nature of the simulation. Accordingly, it is recommended to run the simulation multiple times and to take averages of the simulation results. Figure 7.12 shows the histograms for process cycle time (called process duration in BIMP), waiting time (time a case spends waiting for resources to become available), cost (of resources), and resource utilization. It can be seen that applications spend most of the time in waiting mode, waiting for resources to become available. Resource utilization of clerks and credit officers is at 85 % and 88.5 %, respectively, meaning that there is some overload. As a rule of thumb, a resource utilization above 80 % means that one can expect long queues and high waiting times. If we add two clerks and two credit officers to the simulation we obtain an average cycle time of around 8 hours (compared to 17 hours) and an utilization rate of around 80 % for clerks and 50 % for credit officers.

[6]Some simulators additionally allow one to specify that new cases are only created during certain times of the day and certain days of the week, or according to a given calendar.

Exercise 7.8 An insurance company, namely 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 Fig. 7.13. The process starts when a call related to lodging a claim is received. The call is routed 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 by 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 activities 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 Fig. 7.13. For every task, the cycle time follows an exponential distribution. The hourly cost of a call center agent is 30, while hourly cost of a claims handler is 50.

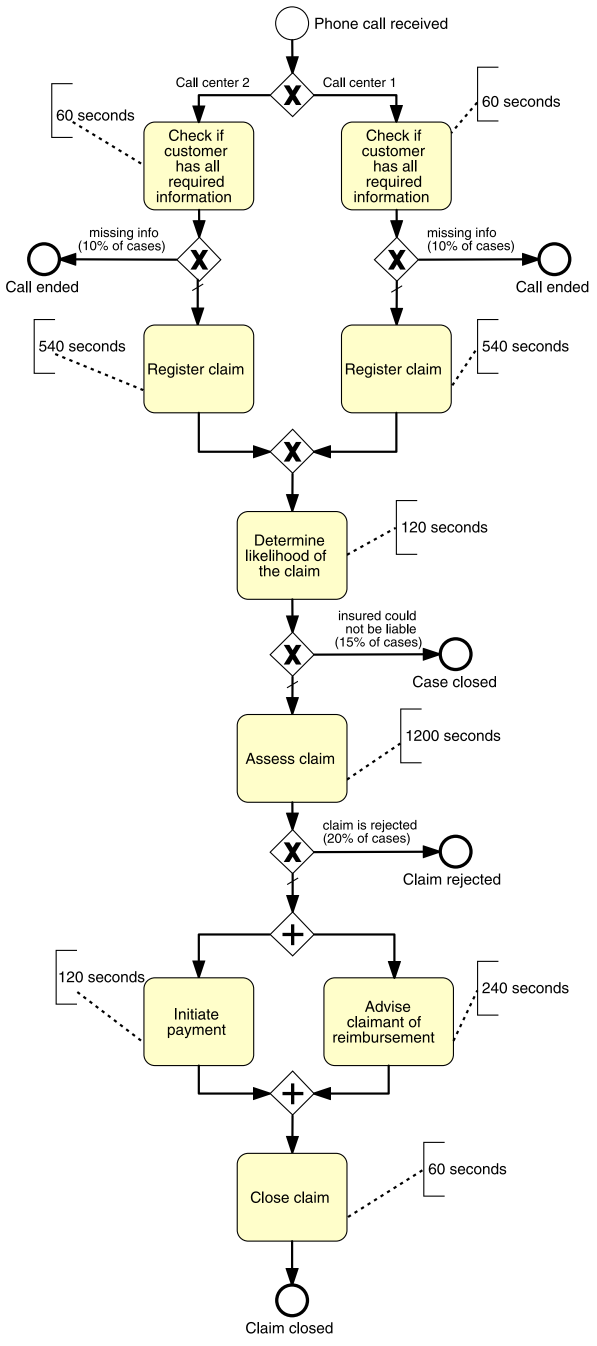
Describe the input that should be given to a simulator in order to simulate this process in the normal scenario and in the storm scenario. Using a simulation tool, encode the normal and the storm scenarios and run a simulation in order to compare these two scenarios.

7.4.3 Simulation Tools

Nowadays, most business process modeling tools provide simulation capabilities. Examples of tools with simulation support include: ADONIS, ARIS Business Designer, IBM Websphere Business Modeler, OpenText ProVision, Oracle Business Process Analysis (BPA) Suite, Savvion Process Modeler, Signavio Process Editor and TIBCO Business Studio. The landscape of tools evolves continuously, and thus it is very useful 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 visibly from one tool to another. For example, some tools allow one to capture the fact that resources do not work

Fig. 7.13 Cetera's claim-to-resolution process


Phone call received
Call center 2
Call center 1
60 seconds
60 seconds
Check if customer has all required information
Check if customer has all required information
missing info (10% of cases)
missing info (10% of cases)
Call ended
Call ended
540 seconds
540 seconds
Register claim
Register claim
120 seconds
Determine likelihood of the claim
insured could not be liable (15% of cases)
Case closed
1200 seconds
Assess claim
claim is rejected (20% of cases)
Claim rejected
120 seconds
240 seconds
Initiate payment
Advise claimant of reimbursement
60 seconds
Close claim
Claim closed

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 attaching a calendar to the process model.

Some of the more sophisticated tools allow one to specify not only branching conditions, but also actual boolean expressions that make use of 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 lower than 10,000. In this case, the probabilistic distribution of values for the attribute "loanAmount" needs to be specified. When the simulator generates objects of type loan, it will give them a value according to the probability distribution attached to that attribute.

There are also small nuances between tools. For example 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. one case every 2 minutes). Recall that the distinction between mean arrival rate (written λ in queueing theory) and mean inter-arrival time ($1/\lambda$) was discussed in Sect. 7.3.1. 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 as illustrated by the following scenario extracted from a description of 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 among other tasks.
>
> 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 would only allow one to define a resource pool and define the number of resources in the pool. A single cost per time unit is then attached to the entire resource pool. Other tools would allow one to create the resources of a pool one by one and to assign a cost to 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 7.7. BIMP is a rather simple BPMN process model simulator that provides the core functionality found in commercial business process simulation tools.

7.4.4 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 continuously on the process being simulated. In practice though, process participants are not robots. They get distracted due to interruptions, they display varying performance depending on various factors, and they may adapt differently to new ways of working.

In this respect, 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 actual manifestation of the bottlenecks shown in 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 a clear indication that incorrect assumptions have been made. Feedback from stakeholders and participants helps to reconfigure the parameters such that the results come closer to matching the actual behavior. In other words, process simulation is an iterative analysis technique with potentially multiple validation loops.

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 ± 10 % for example. If such small changes in the simulation input parameters significantly affect the conclusions drawn from the simulation outputs, one can put a question mark on these conclusions.

7.5 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 activities 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 activity in the model. However, flow analysis does not take into account the level of busyness of the resources involved in the process, i.e. their level of resource utilization. Yet, waiting times 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 activities given data about the number of resources and their processing times. Other queueing theory models such as queueing networks allow us 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 activities (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.

7.6 Solutions to Exercises

Solution 7.1

1. There are at least two business processes that need improvement: the quote-to-booking process—which starts from the moment a quote is received to the moment that a booking is made—and the process for modifying bookings.
2. The quote-to-book process needs to be improved with respect to cycle time, and with respect to error rate. The booking modification process needs improvement with respect to error rate.

Solution 7.2 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: $0.4 \times 1 + 0.4 \times 1 + 0.2 \times 1$ hour. The total cycle time is thus: $1 + 1 + 1 = 3$ hours.

Solution 7.3 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 $0.5 + 12 + \frac{4+2}{1-0.2} = 20$ hours. Hence, cycle time efficiency is 12.5 %.

Solution 7.4 Little's law tells us that: $CT = WIP/\lambda$. At peak time, there are 900 customers distributed across 6 hours, so the mean arrival rate $\lambda = 150$ customers per hour. On the other hand, $WIP = 90$ during peak time. Thus, $CT = 90/150 = 0.6$ hours (i.e. 36 minutes). During non-peak time, $\lambda = 300/6 = 50$ customer per hour while $WIP = 30$, thus $CT = 30/50 = 0.6$ hours (again 36 minutes). 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 they place 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 7.5 Given that there are no other costs, we calculate the cost of the process by aggregating the resource costs as follows: $0.5 \times 25 + 12 \times 50 + (4 \times 75 + 2 \times 100)/(1 - 0.2) = 1237.50$.

Solution 7.6 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 occupation rate 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 there 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 order $W = 64 + 16 = 80$ working days for an order to be fulfilled.

Solution 7.7 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. Accordingly, 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 analyze the problem. Because of this assumption, the results will not be exact.

If we only had seven call center agents, then the occupation rate $\rho = 40/70 = 0.57$, $L_q = \rho^2/(1 - \rho) = 0.57^2/(1 - 0.57) = 0.76$, and $W_q = L_q/\lambda = 0.76/40 = 0.0189$ hours $= 1.13$ minutes. So we cannot meet the customer service charter.

If we can handle 80 calls per hour (eight call center agents), then the occupation rate $\rho = 40/80 = 0.5$, $L_q = \rho^2/(1 - \rho) = 0.5^2/(1 - 0.5) = 0.5$, and $W_q = L_q/\lambda = 0.5/40 = 0.0125$ hours $= 45$ seconds, so we meet the customer service charter.

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

- We could reduce the number of call center agents to 7 and still have an average waiting time of 1.13 minutes. 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 6pm or 7pm instead of 5pm) so that people can call after work, therefore easing the call center load during its peak time.
- Reduce the time of each call by providing better training to call center agents.

Solution 7.8 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 hours. 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 seconds (this is the inter-arrival time). In the storm scenario the inter-arrival time is 8 seconds. In both cases we use an exponential distribution for the inter-arrival time. We run simulations corresponding to 5 weeks of work, meaning 45,000 cases for the normal scenario and 90,000 cases for the storm scenario.

In order to distinguish between the two call centers, we define two separate resource pools, namely "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. Under the normal scenario, we obtain a resource utilization of around 53 % for claims handlers and 41 % for call center operators. The average cycle time is around 0.5 working hours and the maximum observed cycle time is around 4.4 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 close to 100 % and that of claims handlers (in both claims handling centers) is around 79 %. The average cycle time is 12 working hours while the maximum cycle time is around 33 hours, that is approximately 4 working days. The high resource utilization indicates that, in practice, the claims handling office is over-flooded during storm season and additional staff are required. On the other hand, the call center has sufficient capacity for storm season. The average waiting time for the tasks in the call center is around 20 seconds.

7.7 Further Exercises

Exercise 7.9 Calculate the cycle time, cycle time efficiency and cost of the university admission process described in Exercise 1.1, 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 minutes. In 20 % of cases, the completeness check that some documents are missing. In this cases an e-mail 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 minutes to put the degrees and transcripts in an envelope and send them to the academic recognition agency. The time it takes to send the degrees/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 in reality the officer who performs the check only spends 10 minutes on average per check. This language test check 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 hour 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 minutes. 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.

Exercise 7.10 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 that a client has, 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 e-mail in order to make a request. The help desk is staffed with five "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 three Level-2 staff members and their hourly cost is € 60. When a Level-2 employee receives a new request, they evaluate it in order to assign a priority level. The job tracking system 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 e-mail. 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 job tracking system. The job tracking 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" and 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 job tracking system sends an e-mail to the client. The e-mail includes a "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 minutes on average.

- Requests spend on average 1 hour waiting for a Level-1 staff to check them. This applies both to new requests and to re-submitted requests.
- Checking if a new request is "known" takes on average 10 minutes. In 20 % of cases the request is known. In this case, it takes between 2 and 10 minutes (average 5 minutes) for the Level-1 staff 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 hours waiting for a Level-2 staff to evaluate them. Level-2 staff take on average 20 minutes to evaluate a new request.
- Level-2 staff take 5 minutes 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 hours.
- The time required to research and resolve a request is on average 2 hours.
- The time to write the resolution to a request is on average 20 minutes.
- Once a Level-2 staff has written the resolution of a request, it takes on average 20 hours before a the request is fetched from the job tracking system by a Level-1 staff.
- It takes on average 20 minutes for a Level-1 staff to send to the client a problem resolution previously written by a Level-2 staff.
- It takes on average 20 hours between the moment a resolution is sent by the Level-1 staff, and the moment the resolution is tested by the client.
- It takes the client around 10 minutes to e-mail the test results to the Level-1 staff.
- In 20 % of cases the request is not resolved, and it needs to be forwarded to Level-2 again. In this latter case, it takes about 2 minutes for the Level-1 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 handovers.

Acknowledgement This exercise is inspired by an example developed by Sue Conger [8].

Exercise 7.11 Consider the scenario described in Exercise 7.6. 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 € 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 recommend an option to the company.

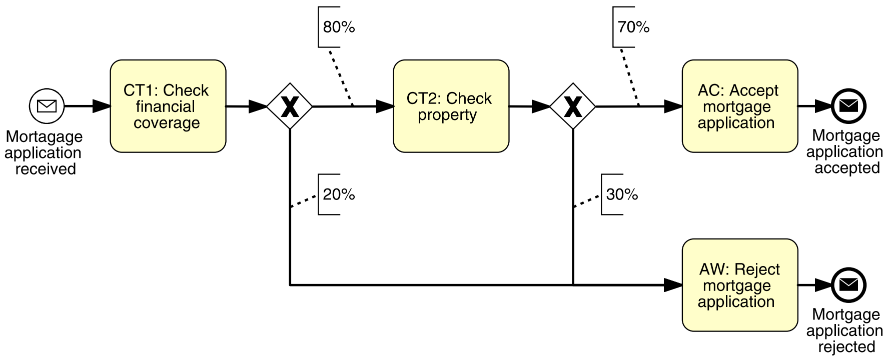

80%
70%
CT1: Check financial coverage
CT2: Check property
AC: Accept mortgage application
Mortagage application received
Mortgage application accepted
20%
30%
AW: Reject mortgage application
Mortgage application rejected

Fig. 7.14 Mortgage process

Exercise 7.12 We consider a Level-2 IT service desk with two 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 hours according to a Poisson process. What is the average time between the moment a service request arrives at this desk and the moment it is fulfilled?

Exercise 7.13 Consider again the IT helpdesk process described in Exercise 7.10. Model and simulate it assuming that cases arrive at a rate of 50 per day according to an exponential distribution. Assume that all the activity cycle times follow an exponential distribution with the mean (average) given in Exercise 7.10.

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

Exercise 7.14 Consider the process model in Fig. 7.14. This model captures a simplified process for handling applications for mortgages. 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 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 five 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 minutes. The wage of each resource is € 20 per hour. Business hours are from Monday to Friday from 9am to 5pm. 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.

7.8 Further Reading

The Balanced Scorecard concept alluded to in Sect. 7.1.2 was proposed by Kaplan and Norton in 1992 [39] and quickly gained popularity thereafter as a tool to define organizational strategy and performance measures. Harmon [31] argues that the traditional approach to apply the Balanced Scorecard leads to a bias towards functional units (i.e. performance measures are defined for company departments). To address this bias, he elaborates an approach to apply the Balanced Scorecard along the process architecture rather than the functional architecture. Fürstenau [21] gives a more detailed overview of approaches to process performance measurement all the way from the identification of performance measures using the Balanced Scorecard, to their implementation in the context of IT-enabled processes.

In Sect. 7.2, we showed how flow analysis techniques can be used to calculate cycle time and cost. Laguna and Marklund [43] additionally show how to use flow analysis to calculate *process capacity*. Process capacity is the maximum number of cases per time unit (e.g. cases per hour) that can be theoretically handled. 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. [109]. Yang et al. also present a technique for flow analysis that is applicable not only to block-structured process models but to a much broader class of process models.

As mentioned in Sect. 7.3, the formula for determining the average queue length in the context of the M/M/c model is particularly complicated. Laguna and Marklund [43, Chap. 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 and Resing [1] give detailed introductions to M/M/1, M/M/c, M/M/c/K and other queueing theory models.

As stated in Sect. 7.4, 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 [24] 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. [98] 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 7.8 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. [99]. One such limitation has to do with batching of cases. For example, consider the University admissions process described in Exercise 1.1 (p. 4). In this process, each individual application progresses independently of other applications, up the point where the admissions committee has to assess the application. The committee does not meet to handle each individual application, but rather meets at certain times and examines a batch of applications. This batching is necessary because it is not practical for the committee to meet too often since the committee involves several members with busy diaries. Also, the committee needs to compare applications with respect to one another, so it only makes sense to meet when either all applications have arrived, or a sufficient number of applications have arrived to make comparisons between candidates. Let us imagine specifically that the committee meets every 2 weeks but only if there are at least 10 applications to be examined. If there are less than 10 applications, the meeting is skipped and the assessment of pending applications is postponed until the next scheduled meeting. Many business process simulation tools simply cannot capture this batching behavior.

To address this and other limitations, Van der Aalst et al. [99] 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 [43] and Arena [40]. For example, Arena is used in the aforementioned case study of a road traffic reporting process [24]. 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 and activities in the process that require special attention if the goal is to reduce cycle time. Anupindi et al. [4] offers 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.