Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Fast and accurate quantitative business process analysis using feature complete queueing models

Sander Peters[a], Yoav Kerner[b], Remco Dijkman[a,*], Ivo Adan[a], Paul Grefen[a]

[a] *Eindhoven University of Technology, PO Box 513, 5600MB Eindhoven, The Netherlands*
[b] *Ben-Gurion University of the Negev, PO Box 653, Beer-Sheva 8410501, Israel*

ARTICLE INFO

Article history:
Received 11 January 2021
Received in revised form 15 September 2021
Accepted 16 September 2021
Available online 25 September 2021
Recommended by Marlon Dumas

Keywords:
Quantitative business process analysis
Queueing models

ABSTRACT

Quantitative business process analysis is a powerful approach for analyzing timing properties of a business process, such as the expected waiting time of customers or the utilization rate of resources. Multiple techniques are available for quantitative business process analysis, which all have their own advantages and disadvantages. This paper presents a novel technique, based on queueing models, that combines the advantages of existing techniques, in that it leads to accurate analyses, is computationally inexpensive, and feature complete with respect to its support for basic process modeling constructs. An extensive quantitative evaluation has been performed that compares the presented queueing model to existing queueing models from literature. This evaluation shows that the presented model outperforms existing models with one order of magnitude on accuracy. The resulting queueing model can be used for fast and accurate timing predictions of business process models. These properties are useful in optimization scenarios.

1. Introduction

One of the goals of modeling business processes is to facilitate quantitative analysis of their properties. Quantitative analysis techniques can be applied to answer questions, such as: what is the waiting time for a case if it would arrive now; how busy are my resources expected to be; and which task is the bottleneck in the process? They perform their calculations on the properties of individual tasks, cases, and roles, including the arrival rate of cases, the distributions of the processing times of tasks, and the number of resources available per role. In this paper a new method for translating business process models to queueing models is introduced to facilitate quantitative analysis of these business process models. The method is based on previous work by the same authors [1], which is extended with a new approximation for the parallel construct.

Fig. 1 shows an example of a business process model in which timing and resource properties are modeled and that can consequently be analyzed quantitatively. The business process is modeled in the Business Process Model and Notation (BPMN). This notation can be used to model the 'flow' of cases through an organization, as tasks are performed on them. Cases arrive at a start event, denoted by a circle with a thin outline. They then follow the flow of the arrows in the system and at each activity, denoted by a rounded rectangle they are served by a resource of the type associated with the lane that the activity is in. For example, the task 'register application' must be performed by a 'back office' resource. When a case arrives at a choice gateway, denoted by a diamond with an $\times$ in it, a choice must be made between alternative paths (in case there are multiple outgoing arrows), or alternative paths are merged again (in case there are multiple incoming arrows). When a case arrives at a parallel gateway, denoted by a diamond with a $+$ in it, multiple paths continue in parallel (in case there are multiple outgoing arrows), or parallel paths are merged again (in case there are multiple incoming arrows). We annotate the BPMN models with the number of resources c per resource type, the arrival rate λ of customer cases, the expected processing times $E(P)$ of activities, and choice probabilities. These parameters are defined precisely in Section 3.

Three types of techniques exist for quantitative analysis of business processes: flow analysis, queueing model analysis, and simulation [1,2]. Flow analysis uses a set of simple equations to model the process, where queueing model analysis generates a queueing model to analyze the process. In simulation a simulation model is generated and ran multiple times in a simulation engine to obtain the quantitative results for the business process. Each of these techniques has its advantages and disadvantages especially regarding duration of the computations, accuracy and support for

* Corresponding author.
E-mail addresses: s.p.f.peters@tue.nl (S. Peters), kerneryo@bgu.ac.il (Y. Kerner), r.m.dijkman@tue.nl (R. Dijkman), i.adan@tue.nl (I. Adan), p.w.p.j.grefen@tue.nl (P. Grefen).

https://doi.org/10.1016/j.is.2021.101892

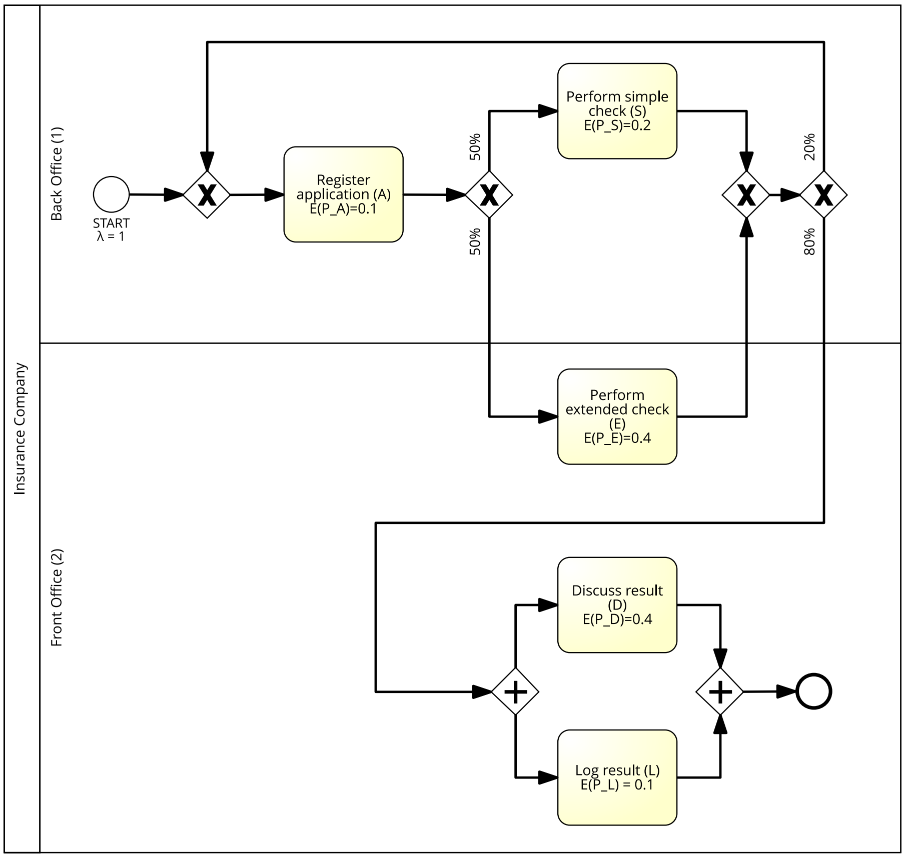
Insurance Company
Back Office (1)
Front Office (2)
START
λ = 1
Register application (A) E(P_A)=0.1
50%
50%
Perform simple check (S) E(P_S)=0.2
Perform extended check (E) E(P_E)=0.4
20%
80%
Discuss result (D) E(P_D)=0.4
Log result (L) E(P_L) = 0.1

Fig. 1. Process model with parameters for quantitative analysis.

different process modeling constructs and probability distributions. These advantages and disadvantages for flow analysis and simulation are discussed in detail in Section 2.

This paper focuses on queueing model analysis, since it has the benefit of being fast and accurate at the same time, while flow analysis is less accurate and accurate simulation is slow. While the benefits of fast computation times are limited in traditional what-if analysis, they are important when real-time decisions must be made by finding an optimum among a large number of predicted future scenarios. An example of such a decision is deciding if a temp agency should be called now to hire temporary resources to be able to complete the work in progress on time. This combination of prediction and optimization is called prescriptive analytics. The optimal parameters for the process model are set by executing the model with different parameters given a certain optimization function (for example minimizing costs or minimizing throughput time) [3]. Prescriptive analytics has been proposed for different fields, for example to optimize capacity planning [4] and to optimize resource allocation [5].

However, while queueing model analysis has the benefits of both being fast and accurate, it has limited support for typical business process modeling patterns (see for example [6]). Of particular concern is the limited support for parallelism and role-based resource assignment. From a structured literature review three queueing models are obtained that support parallelism [7–9], but these queueing models work with approximations and are consequently less accurate.

Therefore, this paper presents a queueing model for business process model analysis that is feature complete with respect to the basic process modeling patterns, including parallelism and role-based resource assignment. In case of exponential arrivals and processing times the results are even exact. In an extensive evaluation we show that for common business process modeling patterns, our technique increases the accuracy of the calculations by up to one order of magnitude over the currently most promising existing queueing model. This paper extends previous work by the same authors [1] by adding support for parallelism (including role-based resource assignment) and providing an extensive evaluation.

To analyze business process models in terms of a queueing model, we propose a three step approach. First, compute the arrival rate of each individual task. Second, transform each role in the business process into a queue, in which items can arrive for each of the tasks that are handled by the role — with the corresponding arrival rate. Third, compute the overall resource utilization, processing time, waiting time and sojourn time for each of the queues and then for the business process model as a whole.

Against this background, the remainder of this paper is structured as follows. In Section 2 discusses related work on quantitative analysis of business process models and the state-of-the-art literature on queueing analysis. Section 3 explains how basic queueing model analysis can be performed on individual tasks (step 1 of the method). Section 4.1 explains how multiple tasks that are assigned to a single role can be transformed into a queue (step 2 of the method), and Section 4.2 explains how multiple resources in a role can be considered. Section 5 presents the extension for the parallel construct. Finally, Section 6 shows how these elements can be combined to transform a complete business process model into a queueing model. Section 7 evaluates the proposed method by comparing its performance to existing queueing models for quantitative analysis. Section 8 presents the conclusions of this research and future work.

Table 1
Comparison of quantitative analysis techniques.

	Flow analysis	Simulation	Queueing analysis
Probability Distributions	–	+	+
Computation Time	+	–	+
Sequence	+	+	+
Choice	+	+	+
Loop	+	+	+
Parallelism	+	+	–

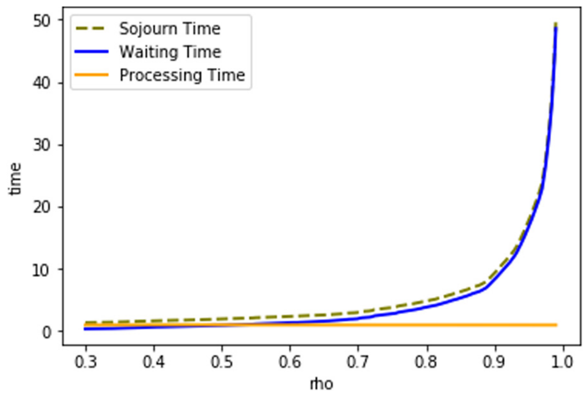

Fig. 2. Simulated values for single task.

2. Related work

This section discusses the differences between the three techniques for quantitative analysis of business process models and related work on queueing models. An overview of the advantages and disadvantages for all techniques is presented in Table 1, the techniques are compared on their support for probability distributions, their computation time and their support for the main constructs in business process modeling [6,10,11]: Sequence, Choice, Loop and Parallelism.

2.1. Flow analysis

Flow analysis [12] is a technique which uses the order of activities in a process, the so-called process structure and a set of mathematical formulas to compute the performance of a process. Since the formulas presented are linear over the number of activities in a process model, the computation time is fast. In the process model all main business process modeling constructs are supported, but there is a major drawback. Flow analysis uses the average waiting times and processing times per task to perform the analysis. There is no support for probability distributions, this has severe impact on the validity of the analysis. If there is a (high) variation in service times the actual performance of the business process can deviate significantly from the predicted value using flow analysis. For example, if the arrival rate in a process has on average an arrival every time unit, so $\lambda = 1$ and the processing time in the only task in the model is on average 0.99 time units, so $E(P) = 0.99$, flow analysis would conclude that the model would perform flawlessly. However, if we simulate this model with an Poisson arrival process and an exponential processing time, we see that the waiting time in the model increases very rapidly towards infinity, as shown in Fig. 2, therefore – in reality – the process model does not perform flawlessly at all and customers would in practice not be happy with this process. Consequently, flow analysis is the least precise form of quantitative business process analysis.

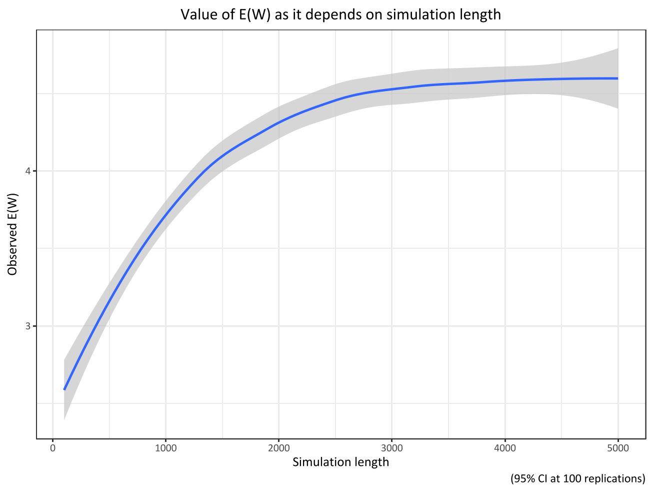

Fig. 3. Simulated values for expected waiting time.

2.2. Simulation

Simulation [13] is a technique which creates a simulation model of the existing process model and runs the simulation in a simulation engine. Simulation has extensive support for business process modeling patterns. Therefore, it is the technique which has the least limitations on analyzing the business process. The downside of simulation is that every simulation requires a warm-up time and sufficient number of replications [14–17], which leads to long computation times. It must be noted that many business process simulation tools do not include replications and warm-up times, making them much faster, but at a serious penalty to accuracy [18].

If warm-up time is not considered in a simulation, the results of the simulation do not represent the values from reality, because the system is not (yet) fully loaded. Fig. 3 illustrates how the lack of warm-up time could affect the observed expected waiting time $E(W)$. The expected waiting time only becomes stable around 4,000 simulated hours. Furthermore the number of replications should be sufficient, the gray area in the figure is the confidence interval at 100 replications. If the number of replications is lowered the gray area would increase, representing that the error of the simulation model would increase, this could lead to incorrect conclusions about the business process [18]. The warm-up time and replications make the simulation model relatively slow compared to flow analysis and queueing models. Decreasing the warm-up time or reducing the amount of replications would speed up the simulation, but decrease the accuracy of the results obtained. Therefore if a large number of alternative settings needs to be explored quickly, simulation is too slow to be a valid option. Also see Section 7 for a comparison of the time performance of simulation and queueing analysis.

2.3. Queueing analysis

Similar to flow analysis, queueing analysis is an analytical method to analyze business processes. Therefore it has the benefit of having very short computation times. However, queueing analysis does support probability distributions, making it accurate for computing processing times and waiting times. This is a major advantage over flow analysis. In existing literature we identified three existing queueing models [7–9].

The paper of Xie et al. [7] describes a business process as a control structure diagram. The four basic control structures defined are a sequence of activities, a loop, a parallel set of activities

Table 2
Qualitative evaluation of existing techniques.

	Xie et al. [7]	Sheng et al. [8]	Ha et al. [9]	This paper
Shared Resource	+/–	–	+/–	+
General Distributions	–	–	–	+[c]
Sequence	+[a]	+[a]	+[b]	+
Choice	+[a]	+[a]	+[b]	+
Loop	+[a]	+[a]	+[b]	+
Parallel	+[a]	+[a]	+[b]	+[a]

[a]Only exponential distributions supported.
[b]Only average service rates supported.
[c]Except for the parallel construct.

and a choice between activities. The control structure diagram can be broken down into these basic control structures. The paper presents two assumptions: all process instances arrive according to a Poisson process and the processing time of each task follows an exponential distribution. And the second assumption is that the queue size is infinite and the tasks are performed in the order as they arrive. For each of the different basic control structures a formula is presented to compute the main cycle time of an activity. In order to compute the overall mean cycle time of the process each basic control structure is analyzed separately. The mean cycle time of the basic control structure is computed and it is replaced by a single activity with that mean cycle time. This process is repeated until there is one queue left which provides the mean cycle time.

The paper of Sheng et al. [8] describes a business process as a network of $M/M/c$ queues. Using the control structures of each activity in the process the arrival rate into the queue is computed based on the initial arrival rate in the system. This technique uses the probability density function of the cycle time, waiting time and service time for each activity. In order to combine the different probability density functions from different activities they are aggregated per control structure. Here also the four control structures: sequence, loop, parallel and choice are supported. The probability density functions are then aggregated following specific formulas to compute the probability density function in each control structure. Combining all control structures the probability density functions can be computed for the complete workflow. The probability density functions obtained represents the probability that the process takes a certain amount of cycle time, waiting time and processing time.

The paper of Ha et al. [9] describes a process as a set of activities, grouped into control structure blocks, connected by a set of links. A control structure block can represent a sequence, a loop or so called repeat block and a parallel set of activities. Choice is handled by indicating a probability on an arc in the process. The activities are handled in a first in first out manner. The cases are equally distributed over the different resources, based on the number of resources available. The expected cycle time is computed differently per block type, but involves the expected cycle time of the activities, furthermore the variance of the cycle time is determined per block type. Aggregating over all the different blocks the average total cycle time can be computed. A comparison of these techniques is presented in Table 2, comparing their support for common business process modeling constructs, including shared resources, general probability distributions, and the main business process patterns.

As shown in Table 2 the queueing models all support the common business process modeling constructs. However, existing queueing models only support the exponential distribution or even only average service rates in case of the work of Ha et al. [9]. The queueing model proposed in this paper supports general distributions for all constructs except the parallel construct, but in the quantitative evaluation section it is shown that the approximation is quite robust for general distributions. The paper of Sheng et al. [8] assumes dedicated resources per task, where the paper of Xie et al. [7] and the paper of Ha et al. [9] have separate queues per resource where work items are allocated to specific resources. Therefore it supports resource sharing partially since resources can work on the same activity, but do not share the same work queues. An extensive quantitative analysis is provided in Section 7.

Another way of analyzing business processes via queueing analysis is by using queue mining [19,20]. This technique uses an event log to mine a queueing network. The work on queue mining differs in focus from the work described in this manuscript: it focuses on mining a queueing network from an event log, while the work described in this manuscript focuses on analyzing a queueing network after it has been mined. Because of this focus, the work on queue mining also requires an event log to work, while the work described in this manuscript works based on a business process model that can be obtained through (queue) mining but also by other means. Consequently, queue mining and the analysis techniques described in this paper complement each other to facilitate quantitative analysis based on real-world data (event logs).

Congestion graphs are related to queue mining and can also be used for analyzing business processes [21]. Like queue mining, congestion graphs also use an event log to generate a process model, which indicates the congestion in the system. If there is more congestion at a certain activity there are more cases which must be served in that queue, so the waiting time increases. This technique uses the congestion properties to predict the waiting time in a system.

Since queue mining and congestion graphs differ from the other techniques for quantitative business process analysis, presented ere, in that they require an event log to work, they are not taken into account in the evaluation section of this paper.

2.4. Queueing theory

As presented in the previous section, queueing analysis is an advanced analytics tool for analyzing business processes. It is performed by transforming a business process to a network of queues [22] and subsequently using queueing theory to analyze that network. The state-of-the-art in queueing theory regarding networks of queues served by a multitude of resources are generalized Jackson Networks [23,24]. A generalized Jackson network is a system of queues which all have independent Poisson arrival processes and general processing times, where each queue is served by a single resource [24]. To cover for the situation in which there are multiple resources for a single queue, the processing time is divided by the number of resources available for a single resource. These Jackson networks are used in operations research to predict job shop scheduling in a factory, see e.g. [25], which can be considered similar to analyzing resource occupancy in a business process model. In job shop scheduling each station where jobs can be processed is modeled as a separate queue, for which the order in which the jobs need to be scheduled is determined. Furthermore, these networks are used in telecommunications and computer systems [26]. In this paper we will transform a BPMN process into a queueing network, which can be used to analyze the process performance.

For the support of parallelism, fork-join networks are commonly used in queueing theory [27–29]. A fork-join network consist out of a set of parallel queues, each with their own servers. The arrival at the so called fork initiates all queues simultaneously, where at the join all paths need to synchronize [27]. Approximations for computing a fork-join network under a First

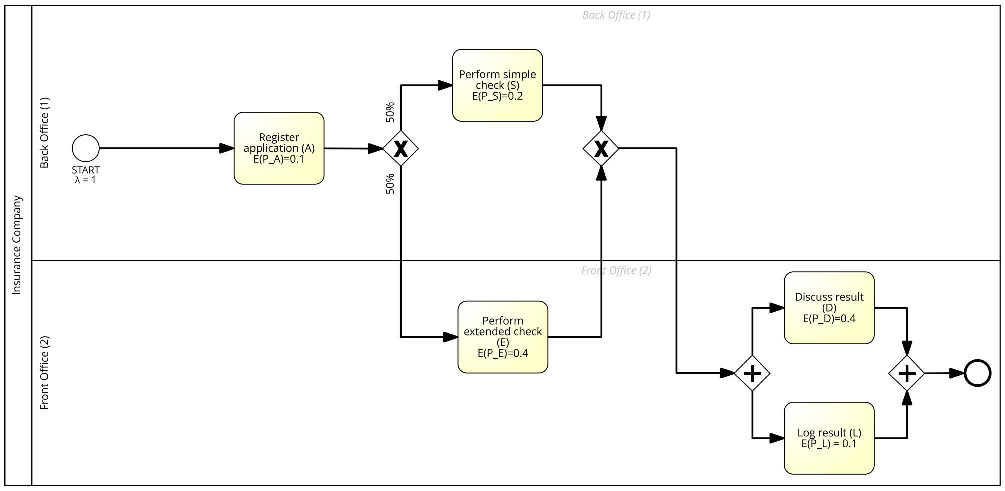
Insurance Company
Back Office (1)
Front Office (2)
START
λ = 1
Register application (A) E(P_A)=0.1
50%
50%
Perform simple check (S) E(P_S)=0.2
Perform extended check (E) E(P_E)=0.4
Discuss result (D) E(P_D)=0.4
Log result (L) E(P_L) = 0.1

Fig. 4. Example process.

Come First Served policy are described by Nguyen [30,31]. Downside of these fork-join networks is that they require dedicated servers for each parallel task. Since business processes very often have shared resources in parallel tasks, a new approximation for parallel tasks with shared resources is presented in this paper.

3. Basic queueing models

A way to simply translate business processes to queueing models is to replace each activity in a business process by a queue [2]. Each queue has an arrival rate and a processing rate. The arrival rate is the interval at which cases arrive in the queue. The processing rate is how fast a resource can process these cases. According to Kendall et al. [32]. a queue can be denoted by its arrival rate (A), processing rate (S) and number of servers (c) as $A/S/c$. For each different probability distribution a different letter is used to indicate what type of queue it is. The M stands for an exponential distribution, the D stands for a deterministic value and the G stands for a general distribution. Furthermore, these queues have no limit in terms of the maximum number of cases in a queue. The service discipline, i.e. the order in which the queued cases are served, is for these queues First In, First Out (FIFO). The assumption of a FIFO serving discipline may not always hold in practice [33]. The technique presented can still be used to transform BPMN models into queueing networks and analyze the performance and approximate the actual queueing discipline with a FIFO discipline. However, the results of the analysis may then differ from the actual performance of the process. The extent to which this approximation impairs on the conclusions, depends on model various model properties. A queue which has an exponential arrival rate and an exponential processing rate with only one server is denoted as a $M/M/1$ queue. For these type of queues mathematical formulas for the expected waiting time and expected processing time are available.

According to Kleinrock [34] the expected processing time and expected waiting time of an $M/M/1$ queue can be computed when the arrival rate (λ) and the processing rate (μ) are known. For the expected processing time of each case, denoted as $E(P)$, it holds that $E(P) = \frac{1}{\mu}$. Based on this expected processing time the expected waiting time, denoted as $E(W)$, can be computed. By obtaining both the expected processing time and the expected waiting time, also the expected sojourn time, denoted as $E(S)$, can be computed. This is the total time a case spends in the queue, combining the time it is waiting and the time it is being processed. Furthermore, the fraction of the time the resource serving the queue is occupied, the utilization rate denoted by ρ, can be computed as shown in Definition 1.

Definition 1. Utilization rate, expected waiting time, and expected sojourn time in an M/M/1 queue

$$\rho = \lambda \cdot E(P)$$
$$E(W) = \frac{\rho}{1-\rho} \cdot E(P)$$
$$E(S) = E(W) + E(P)$$

By using Little's Law [35] the expected number of cases in the queue can be computed. Little's Law describes the number of cases in the queue, denoted as $E(L)$, as the time it spends in the queue multiplied by the arrival rate. Furthermore the number of cases *waiting* in the queue is denoted as $E(L_q)$ is computed by the expected waiting time multiplied by the arrival rate. These formulas are shown in Definition 2.

Definition 2. Little's Law

$$E(L) = E(S) \cdot \lambda$$
$$E(L_q) = E(W) \cdot \lambda$$

As an example for calculating the waiting time of a process, consider the process model in Fig. 4, which is extended from the running example in Dijkman et al. [1]. Fig. 4 describes an insurance process, where applications arrive with $\lambda = 1$. Each application is first registered with an expected processing time of $E(P_A) = 0.1$. Fifty percent of the cases are determined to be simple and are handled by the 'Back Office' by a simple check ($E(P_S) = 0.2$). The other fifty percent are complex cases and therefore are handled by the 'Front Office'. An extended check is performed for the application ($E(P_E) = 0.4$). After the checks the result is shared with the customer who filled the application ($E(P_D) = 0.4$). In parallel to that the result is logged in the system ($E(P_L) = 0.1$). The 'Back Office' has one employee assigned to it, the 'Front Office' has two employees assigned to it.

In order to transform a process, as shown in Fig. 4, into a queueing model it can be assumed that each task is transformed into a $M/M/1$ queue with a dedicated resource. For the first task in the process it holds that $\rho = 2 \cdot \frac{25}{60} = \frac{5}{6}$, such that is calculated $E(W) = \frac{25}{12}$ and $E(S) = \frac{5}{2}$. Since it is common that there are more resources deployed for single activity, but also that a resource is deployed for a set of activities, more advanced queueing models are needed. Furthermore, there are multiple other type of queues, since in practice processing times often are not exponentially distributed [34,36]. Using these advanced queueing models, discussed in Section 4, a new method for transforming business processes is introduced in Section 6.

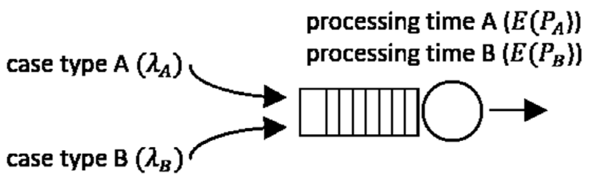
processing time A $(E(P_A))$
processing time B $(E(P_B))$
case type A (λ_A)
case type B (λ_B)

Fig. 5. Queue with two case types A and B.

4. Advanced queueing models

In order to represent business processes two types of queueing models are needed compared to standard Jackson networks: queueing models with different type of cases and queueing models with multiple resources per role. In this section both extensions are discussed.

4.1. Different types of cases in a queue

In a business process it is common to have different type of cases. For example, an insurance company can have three type of insurance claims running the same process (e.g. fire-, car- and travel-insurance claims). Therefore, to compute the performance of a process these different types of cases should be accommodated. It is assumed that the type of case is known for all cases in the process and is known before the process starts. Furthermore, when translating a business process into a queueing network the roles in a business process can be considered a queue and each individual activity assigned to that role a different case type entering that queue [1]. This allows resources to be shared between tasks, while if all resources are dedicated to a specific task there would be no different type of cases in a queue, and consequently no sharing of resources.

Fig. 5 shows a queue with two type of cases [1]. Both types of cases, respectively A and B, have their own processing time $E(P_A)$ and $E(P_B)$ and an arrival rate λ_A and λ_B. In order to compute the expected waiting time and expected sojourn time an adaptation needs to be made to the queueing formulas as presented in Section 3. Since the processing time distribution also is no longer exponential, because the two different case types both have a different processing time, the queue type is changed to one with a general processing time distribution. The arrival rate of cases in the queue is equal to the sum of both arrival rates. For the expected processing time ($E(P)$) and the variance of the processing time ($E(P^2)$) a weighted approach is taken. Let λ be the total arrival rate, then λ_A and λ_B are respectively the arrival rates of cases of type A and B. According to Kleinrock [37], the fraction of cases of type A is equal to $\frac{\lambda_A}{\lambda}$, and the fraction of cases of type B is equal to $\frac{\lambda_B}{\lambda}$. Therefore the fraction of the expected processing time of cases of type A is $E(P_A)$ multiplied by $\frac{\lambda_A}{\lambda}$. Generalizing, the set T holds all case types and case types are denoted by t, where $t \in T$. The arrival rate for case type t is λ_t, the expected processing time is $E(P_t)$ and the expected squared processing time is $E(P_t^2)$. The overall arrival rate, expected processing time and squared processing time are computed as defined in Definition 3 [37].

Definition 3. Arrival rate and expected processing time in a multi-case type M/G/1 queue

$$\lambda = \sum_{t \in T} \lambda_t$$
$$E(P) = \sum_{t \in T} \frac{\lambda_t}{\lambda} E(P_t)$$
$$E(P^2) = \sum_{t \in T} \frac{\lambda_t}{\lambda} E(P_t^2)$$

Using the formulas from Definition 3 the expected waiting time $E(W)$ can be computed. In order to make this computation an assumption needs to be made on the joint inter-arrival time. The joint inter-arrival time is assumed to follow a Poisson process. Therefore, they are exponentially distributed. For multiple case types a Poisson process is a good representation when multiple arrival processes are joined together [38]. Since it cannot be assumed that the new distribution for arrivals is exponentially distributed, as assumed in Section 3, new formulas need to be applied [1]. The new formula for the expected waiting time $E(W)$ uses the expected remaining processing time $E(R)$ for a case. By using the joint expected processing time $E(P)$, the joint expected squared processing time $E(P^2)$, and ρ from Definition 1 the following formulas are defined:

Definition 4. Expected waiting time in a multi-case type M/G/1 queue

$$E(R) = \frac{E(P^2)}{2 \cdot E(P)}$$
$$E(W) = \frac{\rho}{1-\rho} \cdot E(R)$$

The formula for $E(W)$ in Definition 4 is based on the assumption that cases arrive in the queue and see an average of $E(L_q)$ (see Definition 2) cases in front of it, based on the PASTA property [39]. Each case in front of the arriving case is processed with the expected processing time $E(P)$. Furthermore, in addition to that the server might be already working on a case, with probability ρ. This case only needs to complete the remaining processing time $E(R)$. Therefore the formula becomes $E(W) = E(L_q) \cdot E(P) + \rho \cdot E(R)$. From Little's Law, see definition 2, the formula for $E(L_q)$ can be expressed as $\lambda \cdot E(W)$. By substitution we obtain the formula as presented in Definition 4 for $E(W)$. For details on the proof of both formulas a reference to general queueing theory is made [34,36].

When applying these formulas to the example process as displayed in Fig. 4 the following analysis can be made. First the role, in this example the 'Back Office' role, is transformed into a queue. Both activities associated with the role are then transformed into case types for this queue: A 'Register application' and S 'Perform simple check'. For these case types, let $\lambda_A = \frac{5}{4}$, $\lambda_S = \frac{5}{8}$, and let the expected processing time be exponentially distributed with $E(P_A) = \frac{1}{10}$, and $E(P_S) = \frac{2}{10}$, such that it can be computed $E(P_A^2) = \frac{1}{50}$ and $E(P_S^2) = \frac{2}{25}$. Using these values, $E(P)$ and $E(P^2)$ can be computed:

$$\lambda = \lambda_A + \lambda_S = \frac{15}{8}$$
$$E(P) = \frac{\lambda_A}{\lambda} \cdot E(P_A) + \frac{\lambda_S}{\lambda} \cdot E(P_S) = \frac{2}{15}$$
$$E(P^2) = \frac{\lambda_A}{\lambda} \cdot E(P_A^2) + \frac{\lambda_S}{\lambda} \cdot E(P_S^2) = \frac{1}{25}$$

Using the values obtained for $E(P)$ and $E(P^2)$ the utilization rate, expected waiting time and expected sojourn time can be computed:

$$\rho = \lambda \cdot E(P) = \frac{1}{4}$$
$$E(R) = \frac{E(P^2)}{2 \cdot E(P)} = \frac{3}{20}$$
$$E(W) = \frac{\rho}{1-\rho} \cdot E(R) = \frac{1}{20}$$
$$E(S) = E(W) + E(P) = \frac{1}{20} + \frac{2}{15} = \frac{11}{60}$$

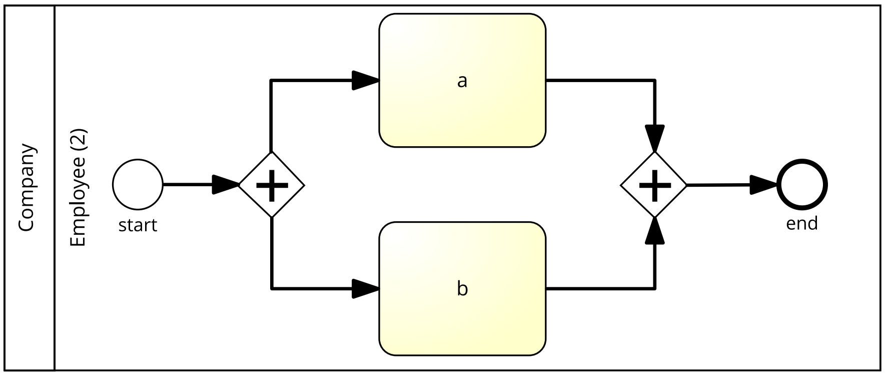

Fig. 6. Parallel construct with two tasks *a* and *b*.

4.2. Multiple server queues

Until this point the assumption has been that there is only a single resource available per queue. Commonly in a business process there are multiple resources available to serve the cases in a queue. In order to account for multiple resources serving a queue the formulas for ρ and $E(W)$ need to be adjusted to a $M/G/c$ queue, where c is the number of resources serving the queue as defined according to Buzacott [40].

Definition 5. Utilization rate and expected waiting time in a multi-server M/G/c queue

$$\rho = \frac{\lambda \cdot E(P)}{c}$$

$$E(W) = \Pi_W \cdot \frac{1}{1-\rho} \cdot \frac{E(R)}{c}$$

$$\Pi_W = \frac{(c\rho)^c}{c!}\left((1-\rho)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!}\right)^{-1}$$

The formula of ρ changes since the work is now done by c resources instead of only 1, therefore the formula for ρ from Definition 1 is divided by c. The formula of $E(W)$ is adjusted for the fact that cases in the queue are now processed c times as fast. The formula is exact when an exponentially distributed processing time is in place and it is an excellent approximation for general processing times [40]. For an extensive proof of these formula's see [38]. Suppose that the 'Back Office', from the example in Fig. 4, now has three resources instead of only a single resource. Therefore, the value of c increases to 3, which results in a utilization rate which is reduced to $\rho = \frac{1}{12}$. Also Π_W can be computed:

$$\frac{(\frac{3}{12})^3}{3!}\left((1-\frac{1}{12})(\frac{(\frac{3}{12})^0}{0!} + \frac{(\frac{3}{12})^1}{1!} + \frac{(\frac{3}{12})^2}{2!}) + \frac{(\frac{3}{12})^3}{3!}\right)^{-1} = \frac{1}{452}$$

5. Parallel tasks

The queueing models presented in the previous section can be used to transform business process models with sequence, choice and loop constructs. In order to represent the parallel construct in a single queue a new approximation needs to be made. This section first presents exact formulas for the $E(P)$ and $E(W)$ for a parallel construct with two tasks, a and b, exponential processing times, an exponential arrival rate, and two shared servers, as illustrated in Fig. 6. It then shows how these formulas can be extended to the general case.

For ease of reasoning and readability, the formulas for the parallel construct with two tasks are developed in two steps. First, the formulas are presented under the condition that the number of jobs that is waiting when a new job arrives is fixed. Second, the formulas for the unconditional case are presented, i.e. the case in which an arriving job sees an arbitrary number of other jobs waiting in front of it.

Against this background, the remainder of this section in structured as follows. Section 5.1 presents the conditional formulas for the expected processing time and waiting time, $E(P)$ and $E(W)$, for the case with two parallel tasks and two shared resources. Section 5.2 presents the unconditional formulas for this case. In general, parallel constructs with more than two resources or more than two tasks also exist and Section 5.3 shows how the formulas for $E(P)$ and $E(W)$ can be adapted for this. Finally, if there are dedicated resources within a parallel construct the formulas for the exact solution with shared resources need to be adapted as well. This adaptation is described in Section 5.4.

5.1. Conditional formulas

To compute the expected processing time and expected waiting time, $E(P)$ and $E(W)$, of the parallel construct, a Markov chain is constructed for the quasi birth death process (QBD) [41] of the parallel construct. For the parallel construct, with two tasks, a and b, jobs arrive according to a Poisson process with rate λ. Each job is a pair (b, a), with processing times that are exponentially distributed with rates μ_a and μ_b. It is assumed that a is always the first task of the job that will be served. Typically, the first task a to be served would be the task with the longest $E(P)$, because that leaves the most flexibility for the parallel construct. Furthermore, there are two servers, which can both process task a and b.

The Markov chain is illustrated in Fig. 7. A state in the Markov chain is denoted as (n, i), where n represents the number of pairs of (b, a) in the queue, and the i denotes the state of the servers. For $n > 0$, the state i can be one of:

1. Both servers serve type a (and hence the next first in the queue is b)
2. There is one a and one b in service and the first in the queue is a
3. There is one a and one b in service and the first in the queue is b
4. Both servers serve type b (and hence the next first in the queue is a)

Consider, for example, the state $(2, 1)$, which means that there are two pairs (b, a) in the queue, and both servers are currently busy performing a task a, meaning that there must be an (additional) item b in the queue, which is the next one to be served. From this state, a new pair (b, a) can arrive, with rate λ, leading to the state $(3, 1)$, also an a can be completed, and since both servers are currently performing a task a, this happens at twice the processing rate μ_a of that task.

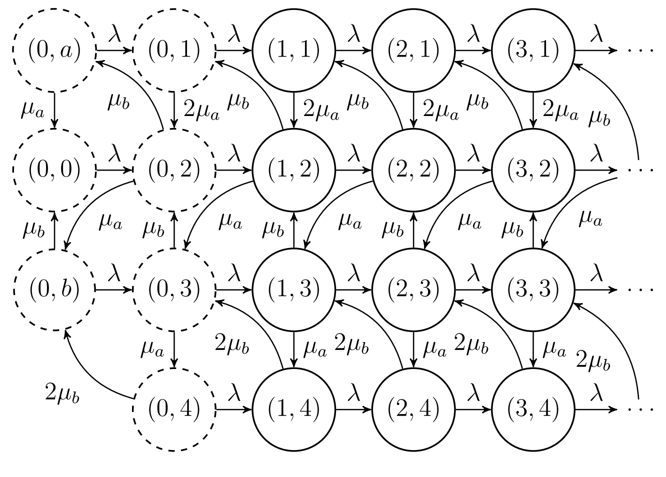

Fig. 7. The Markov chain for the parallel construct.

Besides the states with $n > 0$, there are a seven 'boundary states' with $n = 0$, which represent the situation in which there is no complete pairs in the queue:

- the state $(0, 0)$ represents the empty system;
- the state $(0, a)$ represents an empty queue, one active server that serves a;
- the state $(0, b)$ represents an empty queue, one active server that serves b;
- the state $(0, 1)$ represents a queue with a single b, both servers serving a;
- the state $(0, 2)$ represents an empty queue, one server that serves a, and one that serves b;
- the state $(0, 3)$ represents a queue with a single b, one server that serves a, and one that serves b.
- the state $(0, 4)$ represents an empty queue and both servers serving b;

For this Markov chain, formulas can be derived for the waiting time W_n and for the processing time P_n, conditional on the number of jobs (b, a) in the queue. The recursive formula for $W_{n,i}$, the waiting time of a job that arrives and find the system in state (n, i), is:

$$W_{n,1} = \frac{1}{2 \cdot \mu_a} + W_{n,2} \tag{1}$$

$$W_{n,2} = \frac{1}{\mu_a + \mu_b} + \frac{\mu_a}{\mu_a + \mu_b} \cdot W_{n-1,3} + \frac{\mu_b}{\mu_a + \mu_b} \cdot W_{n-1,1} \tag{2}$$

$$W_{n,3} = \frac{1}{\mu_a + \mu_b} + \frac{\mu_a}{\mu_a + \mu_b} \cdot W_{n,4} + \frac{\mu_b}{\mu_a + \mu_b} \cdot W_{n,2} \tag{3}$$

$$W_{n,4} = \frac{1}{2 \cdot \mu_b} + W_{n-1,3} \tag{4}$$

Where the boundary conditions can be written as:

$$W_{0,1} = \frac{1}{2\mu_a} + \frac{1}{\mu_a + \mu_b} \tag{5}$$

$$W_{0,2} = \frac{4\mu_a^2 + 3\mu_a\mu_b + \mu_b^2}{2\mu_a \cdot (\mu_a + \mu_b)^2} \tag{6}$$

$$W_{0,3} = \frac{1}{\mu_a + \mu_b} \cdot \left(1 + \frac{\mu_a}{2\mu_b} + \frac{1}{\mu_a + \mu_b}\right) \tag{7}$$

$$W_{0,4} = \frac{1}{2\mu_b} + \frac{1}{\mu_a + \mu_b} \tag{8}$$

In order to get the values for the waiting time in each state, W_n, the following computation is made:

$$W_n = \begin{pmatrix} W_{n,1} \\ W_{n,2} \\ W_{n,3}, \\ W_{n,4} \end{pmatrix} = \left(\sum_{i=1}^{n} A^i\right) \cdot W_0 \tag{9}$$

Where the matrix A is the transition matrix between the different phases of the waiting time W_n, with the following rates:

$$A = \begin{pmatrix} \frac{\mu_b}{\mu_a + \mu_b} & 0 & \frac{\mu_a}{\mu_a + \mu_b} & 0 \\ 0 & \frac{\mu_b \cdot (2\mu_a + \mu_b)}{(\mu_a + \mu_b)^2} & 0 & \left(\frac{\mu_a}{\mu_a + \mu_b}\right)^2 \\ \left(\frac{\mu_b}{\mu_a + \mu_b}\right)^2 & 0 & \frac{\mu_a \cdot (\mu_a + 2\mu_b)}{(\mu_a + \mu_b)^2} & 0 \\ 0 & \frac{\mu_b}{\mu_a + \mu_b} & 0 & \frac{\mu_a}{\mu_a + \mu_b} \end{pmatrix} \tag{10}$$

For the processing time there are the same four states possible as identified for the waiting time for the system to be in. These are expressed by $P_{n,i}$, which is the probability that a batch will start service with the other server serving a job, given the state upon arrival is (n, i):

$$P_{n,1} = P_{n,2} = \frac{\mu_a}{\mu_a + \mu_b} \cdot P_{n-1,3} + \frac{\mu_b}{\mu_a + \mu_b} \cdot P_{n-1,4} \tag{11}$$

$$P_{n,3} = \frac{\mu_a}{\mu_a + \mu_b} \cdot P_{n,4} + \frac{\mu_b}{\mu_a + \mu_b} \cdot P_{n,2} \tag{12}$$

$$P_{n,4} = P_{n-1,3} \tag{13}$$

These result in an expression for P_n which is equal to:

$$P_n = \begin{pmatrix} P_{n,1} \\ P_{n,2} \\ P_{n,3} \\ P_{n,4} \end{pmatrix} \tag{14}$$

$$P_n = \begin{pmatrix} \frac{\mu_b \cdot (2\mu_a + \mu_b)}{(\mu_a + \mu_b)^2} & 0 & 0 & \left(\frac{\mu_a}{\mu_a + \mu_b}\right)^2 \\ 0 & \frac{\mu_b}{\mu_a + \mu_b} & \frac{\mu_a}{\mu_a + \mu_b} & 0 \\ 0 & \frac{\mu_b^2 \cdot (2\mu_a + \mu_b)}{(\mu_a + \mu_b)^3} & \frac{\mu_a}{\mu_a + \mu_b} & \frac{\mu_b * (\mu_a)^2}{(\mu_a + \mu_b)^3} \\ \frac{\mu_b}{\mu_a + \mu_b} & 0 & 0 & \frac{\mu_a}{\mu_a + \mu_b} \end{pmatrix} \cdot P_{n-1} \tag{15}$$

Or in a compact way:

$$P_n = \overline{P} \cdot P_{n-1} = \overline{P}^n \cdot P_0 \tag{16}$$

Where P_0 are the three possible boundary states the system can be in. The first state is the empty system, where there are no jobs in progress. The second state is the case where there is only one server busy processing an a task, where the third state is where only one server is busy with processing a b task. This results in the following value for P_0:

$$P_0 = \begin{pmatrix} P_{(0,1)} \\ P_{(0,2)} \\ P_{(0,3)} \\ P_{(0,4)} \end{pmatrix} = \begin{pmatrix} \frac{\mu_b}{\mu_a + \mu_b} \\ \frac{\mu_b}{\mu_a + \mu_b} \\ (\frac{\mu_b}{\mu_a + \mu_b})^2 \\ 0 \end{pmatrix} \tag{17}$$

5.2. Unconditional formulas

In order to obtain the unconditional expression for the waiting time and the processing time for the parallel construct the QBD that is represented by the Markov chain from Fig. 7 should be solved to find the steady state. To solve the QBD three matrices are defined that make up the QBD process [41]: B is the matrix of a state transition backwards in the Markov chain from Fig. 7 (i.e. a transition where a full job is processed and leaves the queue), F is the matrix for a state transition forwards (i.e. a transition where a job arrives in the queue), and L is the matrix that contains the transition rates of events that does not change the number of full jobs in the queue (and its diagonal makes the row sum to 0). These are defined as follows:

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \mu_b & 0 & \mu_a & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2\mu_b & 0 \end{pmatrix}, F = \begin{pmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{pmatrix} \tag{18}$$

$$L = \begin{pmatrix} -(\lambda + 2\mu_a) & 2\mu_a & 0 & 0 \\ 0 & -(\lambda + \mu_a + \mu_b) & 0 & 0 \\ 0 & \mu_b & -(\lambda + \mu_a + \mu_b) & \mu_a \\ 0 & 0 & 0 & -(\lambda + 2\mu_b) \end{pmatrix} \tag{19}$$

Using an iterative procedure the matrices will be used to obtain the R matrix, which solves the equation $F + R \cdot L + R^2 \cdot B = 0$ with all elements being non-negative and the spectral radius smaller than 1. In order to obtain the steady state $\underline{\pi}_n = (\pi_{n,1}, \pi_{n,2}, \pi_{n,3}, \pi_{n,4})$, the relation between the R matrix and the steady state will be used: $\underline{\pi}_n = \underline{\pi}_0 \cdot R^n$. Only unknown variable in this equation is $\underline{\pi}_0$, which can be computed by solving the following system of linear equations:

$$\begin{aligned} \lambda \cdot \pi_{0,0} &= \mu_a \cdot \pi_{0,a} + \mu_b \cdot \pi_{0,b} \\ (\lambda + \mu_a) \cdot \pi_{0,a} &= \mu_b \cdot \pi_{0,2} \\ (\lambda + \mu_b) \cdot \pi_{0,b} &= \mu_a \cdot \pi_{0,2} + 2\mu_b \cdot \pi_{0,4} \\ (\lambda + 2\mu_a) \cdot \pi_{0,1} &= \lambda \cdot \pi_{0,a} + \mu_b \cdot \pi_{1,2} \\ (\lambda + \mu_a + \mu_b) \cdot \pi_{0,2} &= \lambda \cdot \pi_{0,0} + 2\mu_a \cdot \pi_{0,1} + \mu_b \cdot \pi_{0,3} \\ (\lambda + \mu_a + \mu_b) \cdot \pi_{0,3} &= \lambda \cdot \pi_{0,b} + \mu_a \cdot \pi_{1,2} + 2\mu_b \cdot \pi_{1,4} \\ (\lambda + 2\mu_b) \cdot \pi_{0,4} &= \mu_a \cdot \pi_{0,3} \end{aligned}$$

$$\pi_{0,0} + \pi_{0,a} + \pi_{0,b} + \underline{\pi}_0 \cdot (I - R)^{-1} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = 1$$

Using the solution obtained from the system of linear equations $\underline{\pi}_0$ the value of the expected waiting time until the batch is served can be computed by:

$$E(W) = \underline{\pi}_0 \cdot (I - R)^{-1} \cdot (I - RA)^{-1} \cdot W_0 \tag{20}$$

For the expected processing time there are three scenarios: the first is when the batch arrives in an empty system and start service immediately, the second is when the batch waits and starts service with the other server is serving an a job, this scenario has probability θ and the third is when the batch waits and starts service with the other server is serving a b job, with probability $1 - \theta$. In order to obtain the expected service time the boundary conditions $((0,0), (0,a), (0,b))$ are multiplied by the steady state probabilities for the first scenario.

$$\theta = \underline{\pi}_0 \cdot (I - R \cdot \overline{P}^n)^{-1} \cdot P_0 \tag{21}$$

In order to construct the final formula for the expected processing time $E(P)$, there are three scenario's to take into account as described, which will be denoted by $E(P_{(0,0)})$, $E(P_{(0,a)})$, $E(P_{(0,b)})$. Where S_a and S_b are the exponential service time distributions of a and b with rates μ_a and μ_b, this gives:

$$E(P_{(0,0)}) = E(max\{S_a, S_b\}) = \frac{1}{\mu_a} + \frac{1}{\mu_b} - \frac{1}{\mu_a + \mu_b} \tag{22}$$

$$E(P_{(0,a)}) = \frac{1}{2 \cdot \mu_a} + \frac{1}{2} \cdot \frac{1}{\mu_b} + \frac{1}{2} \cdot E(P_{(0,0)}) \tag{23}$$

$$E(P_{(0,b)}) = \frac{1}{\mu_a + \mu_b} + \frac{\mu_a}{\mu_a + \mu_b} \cdot \frac{1}{\mu_b} + \frac{\mu_b}{\mu_a + \mu_b} \cdot E(P_{(0,0)}) \tag{24}$$

$$E(P) = \pi_{0,0} \cdot E(P_{(0,0)}) + (\pi_{0,a} + \theta) \cdot E(P_{(0,a)}) + (1 - \pi_{0,0} - \pi_{0,a} - \theta) \cdot E(P_{(0,b)}) \tag{25}$$

5.3. Generalization of the parallel construct

The general case for our model has k identical servers, where the parallel construct has m tasks, with the processing times of task j follow an exponential distribution with μ_j. So for each server there are m options of the type it is serving, plus there are m options for the type of task that is the first in line. Therefore the number of phases in the Markov model is equal to m^{k+1}, but some phases are transient. This means that there are no state transitions from the current state which lead to these states, for example if there is a task a in front of the queue the states where task b is in front of the queue are unreachable. For the QBD this is not a major issue, but the R matrix will include a lot of states which have a value of zero. These transient phases can be pruned of the model to simplify computations. Adding more tasks or more resources will only increase the state space, but the method of computing the $E(W)$ and $E(P)$ remains equal.

5.4. Dedicated resources

In case there are dedicated resources in a parallel construct for both tasks a couple of changes have to be made to model this situation. The stability condition changes from $\frac{\lambda}{2} \cdot \left(\frac{1}{\mu_a} + \frac{1}{\mu_b}\right)$ to $\lambda \cdot \min\left\{\frac{1}{\mu_a}, \frac{1}{\mu_b}\right\}$ and therefore it can handle less cases than a system where resources are shared between tasks. The situation with dedicated resources can be modeled in a two dimensional Markov chain with the states being the pairs (n_a, n_b), respectively the number of tasks of type a and b. The state space is unbounded in two variables, therefore numerical approximation needs to be used to truncate the system using a maximum number of items in the system K thereafter cases are rejected from the system.

In order to compute the sojourn time of the parallel construct with dedicated resources the non-zero transition rates and the steady state are defined. The non-zero transition rates between two states:

$$\begin{aligned} q_{(n_a,n_b),(n_{a+1},n_{b+1})} &= \lambda \\ q_{(n_a,n_b),(n_{a-1},n_b)} &= \mu_a \\ q_{(n_a,n_b),(n_{a-1},n_{b-1})} &= \mu_b \end{aligned}$$

Since the two jobs in a parallel process need to wait for each other the expected sojourn time is equal to the maximum of the sojourn times of both two tasks. Both tasks have an Erlang distribution distributed with $(n_a + 1, \mu_a)$ and $(n_b + 1, \mu_b)$. Using the fact that $\min\{S_a, S_b\}$ + $\max\{S_a, S_b\} = S_a + S_b$:

$$E(min\{S_a, S_b\}|n_a, n_b) = \sum_{i=0}^{n_a} \sum_{j=0}^{n_b} \frac{\mu_a^i \cdot \mu_b^j}{(\mu_a + \mu_b)^{i+j+1}} \cdot \binom{i+j}{i}$$

$$E(max\{S_a, S_b\}|n_a, n_b) = \frac{n_a+1}{\mu_a} + \frac{n_b+1}{\mu_b} - \sum_{i=0}^{n_a}\sum_{j=0}^{n_b} \frac{\mu_a^i \cdot \mu_b^j}{(\mu_a+\mu_b)^{i+j+1}} \cdot \binom{i+j}{i}$$

The unconditional expected sojourn time is the weighted average of the formulas presented above, where the weights are the steady state probabilities of the Markov chain. The steady state probability of observing a state n_a, n_b is derived using the equations $\pi \cdot P = 0$. The formula for the expected sojourn time is:

$$E(S) = \sum_{n_a=0}^{K}\sum_{n_b=0}^{K} \pi_{(n_a,n_b)} \cdot E(max\{S_a, S_b\}|n_a, n_b) \qquad (26)$$

6. From BPMN model to queueing model

By using the results from the previous section, now also parallel constructs can be analyzed in business processes. In order to analyze a business process, in the form of a BPMN model, using queueing models a conversion between the two is necessary. The procedure to convert a BPMN model into a queueing model is the following [1]:

1. Calculate the arrival rate for each task in the BPMN model.
2. Create a queue for each role in the BPMN model and calculate the arrival rate, processing time, utilization rate, waiting time, and sojourn time for that queue.
3. Calculate the overall processing time, waiting time, and sojourn time.

This procedure to convert BPMN models into queueing models and compute the overall processing time, waiting time and sojourn time has been implemented in a simulator [42].

6.1. Calculate arrival rate for each task

First, for each task in the BPMN model their arrival rate should be computed. The arrival rate of the start event in a business process should be specified in the process. If there are no loops in a business process, the computation of the arrival rate of the other tasks is fairly straightforward. The total arrival rate in a task is the sum of all incoming arrival rates. Furthermore, the total incoming flow of a tasks is equal to the total outgoing flow of a task. For gateways in the process the following applies:

- For a choice in a process the fraction of the arrival rate, indicated by the percentage on the arc given in the process model, is added to the arrival rate of the activity that the branch leads to.
- For a join of a choice in the process all incoming arrival rates are added together.
- For a parallel split all activities in the parallel split have the same arrival rate.
- For a parallel join the arrival rate is the same as one of the incoming arcs.

If we apply this to the example in Fig. 4 all cases flowing into task A ('Register application'), denoted by λ_A, are split between task S ('Perform simple check') and task E ('Perform extended check') both with a fraction of 50%. Consequently, the arrival rate for task S is equal to $\lambda_S = 0.5 \cdot \lambda_A$ and the arrival rate for task E is equal to $\lambda_E = 0.5 \cdot \lambda_A$. The process then is split between the two parallel tasks D ('Discuss result') and L ('Log result'). Since these two tasks are in parallel, both have the same arrival rate, which is the flow from task S and task E combined, resulting in $\lambda_D = \lambda_S + \lambda_E$ and $\lambda_L = \lambda_S + \lambda_E$. When there are loops in the process, the arrival rate in the process partially depends on its own arrival rate. To illustrate this the example from Fig. 4 is extended with a loop as shown in Fig. 8.

In the example shown in Fig. 8 a loop is added, after the check on the application of the customer it might be incorrectly registered, which happens in 20% of the cases. These cases are redirected to the start of the process to be registered again. Since there is now a loop in the process the arrival rate of task A depends on the arrival rates of tasks S and E. Since these two already depended on the arrival rate of task A, the arrival rate at task A is indirectly dependent on itself. In order to compute the arrival rate at task A a system of linear equations is solved, these are defined as follows.

Definition 6 (*Arrival Rate into Task*). Let λ be the arrival rate into the BPMN process that consist of a set of tasks T. Furthermore, let for each $t, t' \in T$: $p_{t,t'}$ be the probability that after executing task t for a case, task t' will be executed for that case, and $p_{\odot,t}$ be the probability that t is executed after the start event, denoted by $\odot$ ($\odot \notin T$). For each task $t \in T$, the arrival rate λ_t can now be defined as [1]:

$$\lambda_t = \sum_{t' \in T} p_{t,t'} \cdot \lambda_{t'} + p_{\odot,t} \cdot \lambda$$

Rewriting the formulas by transferring the sum part to the left side of the equation, a system of linear equations is obtained which can be solved to an unique solution, since every case will at some point leave the business process. When applying this to the running example in Fig. 8, the system of linear equations is:

$$\begin{aligned}
\lambda_A - 0\cdot\lambda_A - 0.2\cdot\lambda_S - 0.2\cdot\lambda_E - 0\cdot\lambda_D - 0\cdot\lambda_L &= 1\cdot 1\\
\lambda_S - 0.5\cdot\lambda_A - 0\cdot\lambda_S - 0\cdot\lambda_E - 0\cdot\lambda_D - 0\cdot\lambda_L &= 0\cdot 1\\
\lambda_E - 0.5\cdot\lambda_A - 0\cdot\lambda_S - 0\cdot\lambda_E - 0\cdot\lambda_D - 0\cdot\lambda_L &= 0\cdot 1\\
\lambda_D - 0\cdot\lambda_A - 0.8\cdot\lambda_S - 0.8\cdot\lambda_E - 0\cdot\lambda_D - 0\cdot\lambda_L &= 0\cdot 1\\
\lambda_L - 0\cdot\lambda_A - 0.8\cdot\lambda_S - 0.8\cdot\lambda_E - 0\cdot\lambda_D - 0\cdot\lambda_L &= 0\cdot 1
\end{aligned}$$

Solving this yields $\lambda_A = \frac{5}{4}, \lambda_S = \frac{5}{8}, \lambda_E = \frac{5}{8}, \lambda_D = 1$ and $\lambda_L = 1$.

6.2. Create a queue for each role

The logical translation of a business process model to a queueing model is to create a queue for each role, as already discussed in Section 4.1. Therefore, in the second step for each role in the business process a queue is created, where each task associated with that role is a case type and the number of servers is the number of resources in that role. Based on Dijkman et al. [1], a role queue can be defined as follows:

Definition 7 (*Role Queue*). A role r, with tasks T_r associated with that role, behaves like a multi-case type queue with c_r servers and case types T_r, where each case type $t \in T_r$ has arrival rate λ_t, expected processing time $E(P_t)$ and expected processing time squared $E(P_t^2)$.

Using this definition the formulas from Sections Section 4 can be used to compute the waiting time for this role ($E(W_r)$). The utilization rate for this role (ρ_r). Using these formulas the assumption is made that the combined arrival process of the tasks can be represented by a Poisson process. If we apply this to the example in Fig. 8 for the role of 'Back Office' a multi-case queue with a single server and two case types A and S is created. The two case types have the following properties: $\lambda_A = \frac{5}{4}$, $\lambda_S = \frac{5}{8}$, $E(P_A) = 0.1$, $E(P_S) = 0.2$, $E(P_A^2) = 0.02$, $E(P_S^2) = 0.08$. Using these properties the $E(W_{BackOffice}) = \frac{1}{20}$ and $\rho_{BackOffice} = \frac{1}{4}$ can be computed. If this technique is applied to all roles in the business

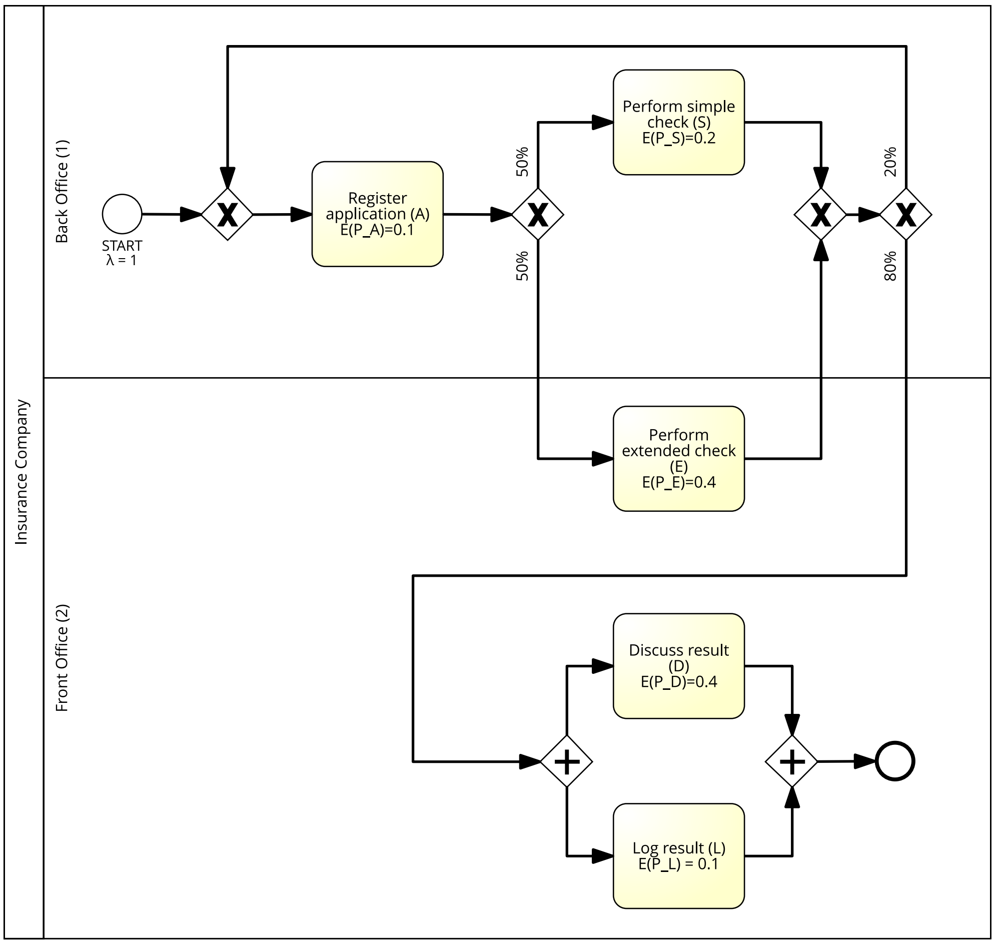
Insurance Company
Back Office (1)
Front Office (2)
START
λ = 1
Register application (A) E(P_A)=0.1
Perform simple check (S) E(P_S)=0.2
Perform extended check (E) E(P_E)=0.4
50%
50%
20%
80%
Discuss result (D) E(P_D)=0.4
Log result (L) E(P_L) = 0.1

Fig. 8. Example process with loop.

process, from Fig. 8, the queueing network as shown in Fig. 9 is obtained. Similarly, for the 'Front Office' with two servers, it can be computed that $E(W_{FrontOffice}) \approx \frac{1}{40}$ and $\rho_{FrontOffice} = \frac{3}{8}$.

6.3. Calculate the overall timing properties

Finally, the overall waiting time, processing time and sojourn time of the process as a whole can be computed. Tasks within the same role have the same expected waiting time, since they are in the same queue. For the example from Fig. 8 tasks A and S for the 'Back Office' have the same expected waiting time, for the 'Front Office' the tasks E, D and L also have the same expected waiting time. The overall times can be computed by analyzing all the possible paths through the process and adding up all the times of the different tasks. Each path is multiplied with the probability that it is taken. The only issue here is that loops in the system create an infinite set of paths through the process. This is solved by the restriction to only analyze block-structured BPMN models [43,44]. This restriction limits the process models suitable for analysis with this method, but is not considered to be a major drawback since it is often used in different process related algorithms [45–47] and some business process modeling languages are block structured by definition, such as the Business Process Execution Language (BPEL) [48], which is a business process modeling language focused on modeling processes with web services [49]. For more information on the algorithms that can be used to translate a BPMN model to a block structure a reference is made to their definitions [43,44] and to the open source library that implements an algorithm to generate a block structure from a BPMN model [50]. A block in a block structure is a sub-graph which has one entry and one exit point, multiple of these subgraphs connected can form a block structured process. Due to the block structure of the process each part of the process can be computed individually. A block structured BPMN model can be described by a process structure tree as defined in Dijkman et al. [1]. Let t be a task. Furthermore, let $p, p_1, p_2, \ldots, p_n$ be probabilities.

Definition 8 (*Process Structure Tree*)**.**

- t is a process structure tree.
- if all $Pt, Pt_1, Pt_2, \ldots, Pt_n$ are process structure trees, then so are $\circlearrowleft(Pt, p)$, $\rightarrow(Pt_1, Pt_2, \ldots, Pt_n)$, $\otimes(Pt_1, Pt_2, \ldots, Pt_n, p_1, p_2, \ldots, p_n)$, and $\oplus(Pt_1, Pt_2, \ldots, Pt_n)$.

The various types of subtrees, indicated by ($\circlearrowleft$, $\rightarrow$, $\otimes$, $\oplus$) are the loop, sequence, choice, and parallelism subtrees, respectively. The semantics of each type of subtree can be defined as shown in Definition 9, which is an extension on Definition 4 in Dijkman et al. [1].

Definition 9 (*Process Structure Tree Semantics*)**.** In a process structure tree of type:

- t, an arriving item will be processed and leave the queue associated with task t according to the usual queueing semantics.
- $\circlearrowleft(Pt, p)$, an arriving item is sent as an arriving item to subtree Pt, an item that leaves subtree Pt, is sent as an

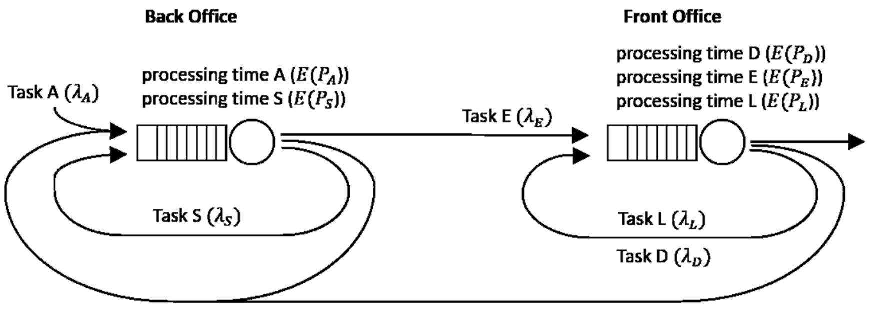
Back Office
Front Office
Task A (λ_A)
processing time A ($E(P_A)$)
processing time S ($E(P_S)$)
Task E (λ_E)
processing time D ($E(P_D)$)
processing time E ($E(P_E)$)
processing time L ($E(P_L)$)
Task S (λ_S)
Task L (λ_L)
Task D (λ_D)

Fig. 9. Queueing model for running example.

arriving item to subtree Pt with probability p, or leaves the tree with probability $1-p$.

- $\rightarrow(Pt_1, Pt_2, \ldots, Pt_n)$, an arriving item is sent as an arriving item to subtree Pt_1, an item that leaves subtree Pt_i for $1 \leq i < n$ is as an arriving item to subtree Pt_{i+1}, and an item that leaves subtree Pt_n leaves the tree.
- $\otimes(Pt_1, Pt_2, \ldots, Pt_n, p_1, p_2, \ldots, p_n)$, an arriving item is sent to subtree Pt_i with probability p_i, when it leaves the subtree it leaves the tree.
- $\oplus(Pt_1, Pt_2, \ldots, Pt_n)$, an arriving item is sent to all subtrees $(Pt_1, Pt_2, \ldots, Pt_n)$, when the items in all subtrees leave it leaves the tree.

By taking into account these semantics the waiting time, processing time and sojourn time of a process structure tree can be computed. By combining the different process structure trees of a business process the waiting time, processing time and sojourn time of a block structured BPMN model can be computed. Proposition 1 is an extension of Proposition 6 in Dijkman et al. [1].

Proposition 1 ($E(W)$, $E(P)$, $E(S)$ *of a Process Structure Tree*)**.** *Let r be a role and T_r be the tasks associated with that role. Furthermore, let $E(P_x)$, $E(W_x)$, $E(S_x)$ be the processing time, waiting time, and sojourn time of a component x, which can be either a task, a role, or a process structure tree. In a process structure tree Q of type:*

- *t, the expected waiting time $E(W_Q)$ of the tree is $E(W_r)$ for the role r with $t \in T_r$; the expected processing time $E(P_Q)$ of the tree is $E(P_t)$; and the expected sojourn time $E(S_Q)$ of the tree is $E(W_Q) + E(P_Q)$.*
- *$\circlearrowleft(Pt, p)$, the expected waiting time $E(W_Q)$ of the tree is $\frac{1}{1-p} \cdot E(W_{Pt})$. $E(P_Q)$ and $E(S_Q)$ are defined analogously.*
- *$\rightarrow(Pt_1, Pt_2, \ldots, Pt_n)$, the expected waiting time $E(W_Q)$ of the tree is $\sum_{i=1}^{n} E(W_{Pt_i})$. $E(P_Q)$ and $E(S_Q)$ are defined analogously.*
- *$\otimes(Pt_1, Pt_2, \ldots, Pt_n, p_1, p_2, \ldots, p_n)$, the expected waiting time $E(W_Q)$ of the tree is $\sum_{i=1}^{n} p_i \cdot E(W_{Pt_i})$. $E(P_Q)$ and $E(S_Q)$ are defined analogously.*
- *$\oplus(Pt_1, Pt_2, \ldots, Pt_n)$, the expected waiting time $E(W_Q)$ of the tree is computed by Eq. (20) and the expected processing time $E(P_Q)$ is computed by Eq. (25). $E(S_Q)$ is defined analogously.*

The process structure tree of type t is very straightforward. The expected waiting time for a task is $E(W_t)$, where the processing time is equal to $E(P_t)$. For the process structure tree of type $\circlearrowleft$ it involves the probability of the loop p to compute the expected waiting time and expected processing time. The loop is always executed one time, the probability that it is executed again is equal to p. The probability that the loop is executed for a third time is equal to $p{\cdot}p$. This continues to be multiplied by p for every possible iteration of the loop. Consequently, the probability that a loop is executed n times is equal to $\sum_{i=0}^{n} p^i$ and the expected number of times the loop is executed is equal to $\sum_{i=0}^{\infty} p^i$. This is a geometric series which is equal to $\frac{1}{1-p}$ [2]. Therefore the expected waiting time of a loop is $\frac{1}{1-p} \cdot E(W_{Pt})$ and the expected processing time of the loop is $\frac{1}{1-p} {\cdot} E(P_{Pt})$. For the process structure tree of type $\rightarrow$, the sequence of n tasks, the expected waiting time is equal to the sum of the waiting times of the n tasks $\sum_{i=1}^{n} E(W_{Pt_i})$ and the expected processing time is equal to the sum of the processing times of the n tasks $\sum_{i=1}^{n} E(P_{Pt_i})$. For the process structure tree of type $\otimes$, the choice between n tasks, the expected waiting time is equal to $\sum_{i=1}^{n} p_i \cdot E(W_{Pt_i})$ and the expected processing time is equal to $\sum_{i=1}^{n} p_i \cdot E(P_{Pt_i})$. For the process structure tree $\oplus$, the parallelism of n tasks, the formulas as presented in Section 5 need to be applied to compute $E(W)$ and $E(P)$.

For the running example from Fig. 8, the process structure tree is:

$$\rightarrow(\circlearrowleft(\rightarrow(A, \otimes(S, E, \tfrac{1}{2}, \tfrac{1}{2})), \tfrac{1}{5}), \oplus(D, L))$$

From the previous section it is known that $E(W_A) = E(W_S) = \frac{1}{20}$ and $E(W_E) = E(W_D) = E(W_L) \approx \frac{1}{40}$. Now $E(W)$ of the process as a whole can be recursively calculated:

$$E(W_{\otimes(S,E,\frac{1}{2},\frac{1}{2})}) = \frac{1}{2} \cdot \frac{1}{20} + \frac{1}{2} \cdot \frac{1}{40} = \frac{3}{80};$$
$$E(W_{\rightarrow(A,\otimes(S,E,\frac{1}{2},\frac{1}{2}))}) = \frac{1}{20} + \frac{3}{80} = \frac{7}{80};$$
$$E(W_{\circlearrowleft(\rightarrow(A,\otimes(S,E,\frac{1}{2},\frac{1}{2})),\frac{1}{5})}) = \frac{1}{1-\frac{1}{5}} \cdot \frac{7}{80} = \frac{7}{64};$$
$$E(W_{\oplus(D,L)}) \approx 0.046;$$
$$E(W_{\rightarrow(\circlearrowleft(\rightarrow(A,\otimes(S,E,\frac{1}{2},\frac{1}{2})),\frac{1}{5}),\oplus(D,L))}) \approx 0.046 + \frac{7}{64} \approx 0.155$$

This leads to $E(W) \approx 0.155$.

7. Evaluation of accuracy of queueing models

In this section an evaluation is performed regarding the accuracy of the queueing models as presented in this paper. Section 7.1 discusses the evaluation of the queueing models compared to each other for the various process constructs. Section 7.2 discusses the evaluation of the developed method on a real-life size process model.

7.1. Evaluation of queueing models

To analyze if the method in this paper analyzes the expected waiting time, expected processing time and sojourn time correctly it is evaluated. Furthermore, it is compared to existing queueing models for quantitative business process analysis [7–9], as they are presented in Section 3. In the paper of Ha et al. [9] some inconsistencies where found in the formulas for the parallel construct. These inconsistencies exist with respect to the definitions of k_A and d_b in the paper. k_A is a fraction of which

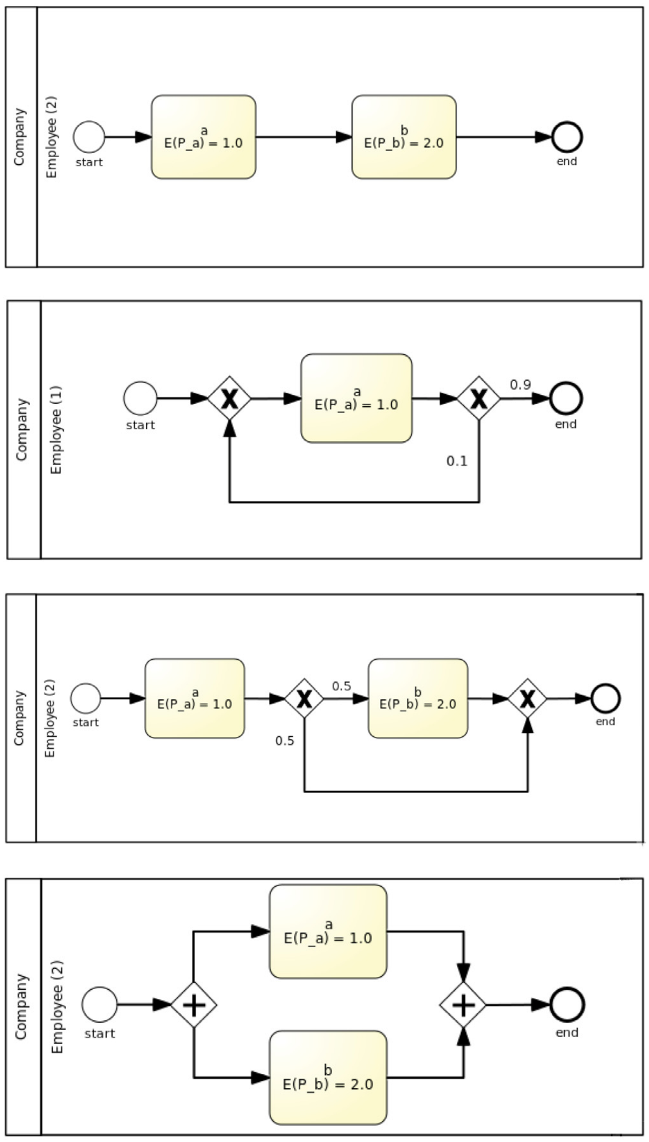

Fig. 10. Simulated constructs.

the denominator can be zero in case A consists on only one task, which will frequently be the case because A is a powerset of a set of tasks. d_b relies on the cycle time of the task or tasks b. However, it is undefined how the cycle time is calculated if b is a set of multiple tasks. The authors were contacted to obtain more insight to implement these formulas, but until the time of writing this paper no response was obtained. Therefore this queueing model is not taken into account in the evaluation.

In the evaluation the four main business process modeling constructs [6,10,11] are investigated with exponential arrival- and processing times. In order to also test the validity of the newly developed approximation for the parallel construct, it is also evaluated under a normal distribution. Since this distribution is not supported by the approximation, it can be seen whether the approximation still provides a good enough estimate of the process performance. For the reference values a simulation model is ran in the simulator presented in the Peters et al. [51] (see Fig. 10).

Table 3 shows the simulation parameters that are used per construct. Note that the choice for a particular processing time is arbitrary, because the expected waiting time depends on the

Table 3
Quantitative evaluation setups.

Construct	# Tasks	# Resources	P_a	P_b
Sequential	2	2	Exp(1.0)	Exp(2.0)
Choice ($p = 0.5$)	2	2	Exp(1.0)	Exp(2.0)
Loop ($p = 0.1$)	1	1	Exp(1.0)	–
Parallel (Exp.)	2	2	Exp(1.0)	Exp(2.0)
Parallel (Norm.)	2	2	N(1.0, 0.125)	N(2.0, 0.25)

Table 4
Evaluation results.

	Model	Sequential	Choice	Loop	Par (Exp.)	Par (Norm.)
MAE	Xie et al. [7]	5.80	3.89	0.28	4.04	5.50
	Sheng et al. [8]	5.80	3.89	0.56	39.18	40.64
	This paper	**1.15**	**0.70**	**0.23**	**0.37**	**1.83**
MSE	Xie et al. [7]	74.48	35.07	**0.16**	38.19	67.46
	Sheng et al. [8]	74.48	35.07	0.97	6148.03	6342.32
	This paper	**2.84**	**0.91**	0.24	**0.18**	**5.96**
%Err	Xie et al. [7]	74.48%	74.55%	5.77%	58.92%	111.58%
	Sheng et al. [8]	74.48%	74.55%	10.43%	567.57%	802.61%
	This paper	**14.26%**	**13.67%**	**3.18%**	**6.81%**	**40.50%**

utilization rate, which is determined by the processing time in combination with the arrival rate and the number of servers. For that reason, a fixed processing time is used and vary the utilization rate with a minimum of 0.30 and a maximum of 0.95 with increments of 0.001. For each of the parameter settings and a utilization rate, a simulation model is executed with a warm-up time of 1000 h and a total duration of 5000 h. This in order to ensure that the system is completely loaded before the performance of the techniques is measured for a long enough period. In each model resources are shared between tasks that are performed by the same role, as is common in business processes. Note that he models obtained from literature do not support shared resources and dedicated resources are used instead to approximate the results for those models.

In Table 4 the results are presented for each of the setups. The table reports the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the difference in percentage between the queueing model and the simulation result (%Err). The Mean Absolute Error is computed using the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{27}$$

In Eq. (27) each value $\hat{y}_i$ is compared against the value obtained in the simulator y_i, the sum over these absolute differences is taken and divided by the number of observations. The Mean Squared Error is computed using the following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{28}$$

In Eq. (28) each value $\hat{y}_i$ is compared against the value obtained in the simulator y and squared, the sum over these absolute differences is taken and divided by the number of observations. In Table 4 the best value between the three models compared is marked bold, from this it is clear that the queueing model as presented in this paper outperforms the other queueing models in most cases. Only for the setup of the loop all values are almost identical, this is because there was only one resource involved and therefore the other models did not suffer from the lack of support for resource sharing. In the next paragraphs the individual results of the different setups are discussed in more detail.

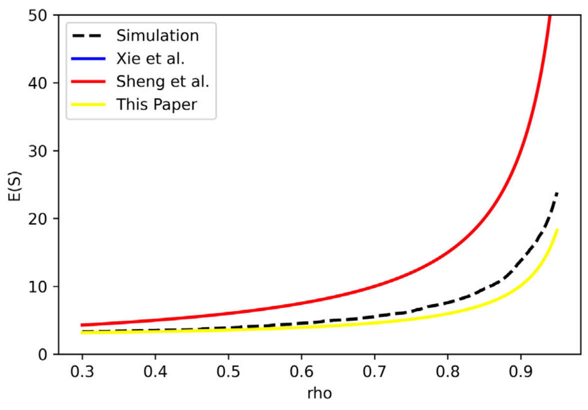

Fig. 11. Sequential construct.

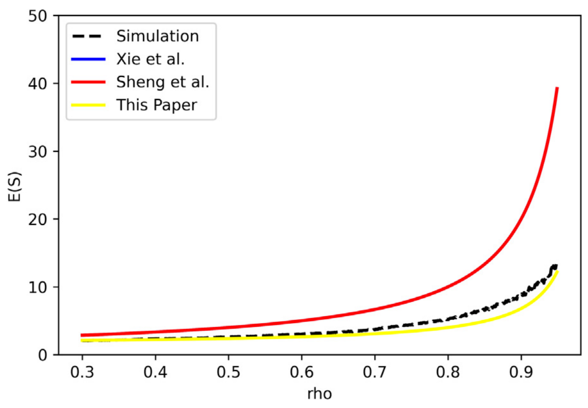

Fig. 12. Choice construct.

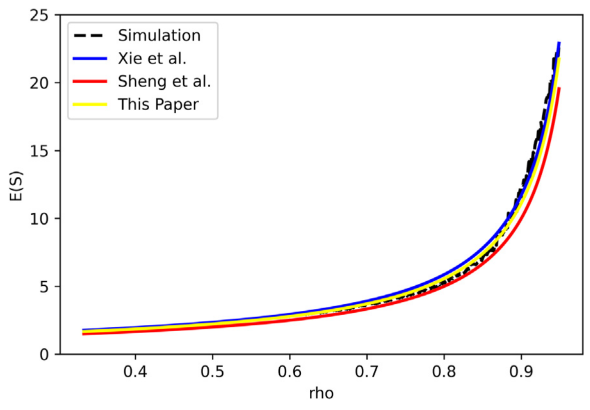

Fig. 13. Loop construct.

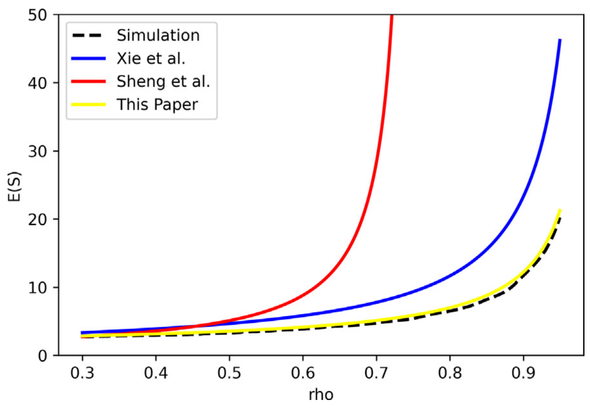

Fig. 14. Exponential parallel construct.

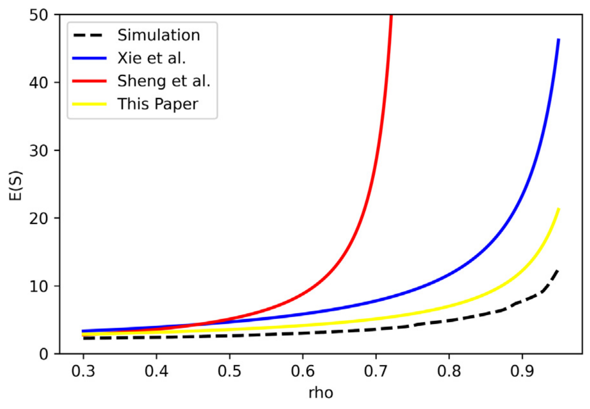

Fig. 15. General parallel construct.

For the sequential construct, where task *a* is directly followed by task *b*, the comparison between the queueing models is provided in Fig. 11. Both models from the literature assume that there are dedicated resources for both tasks, therefore this results in the same performance and an overlapping line in the graph. The model presented in this paper approaches the black line significantly closer than the other two methods and therefore is considered better to use for predicting sequential tasks with shared resources.

For the choice construct again two tasks are present, but after task *a* there is a choice whether the process continues with task *b* with probability 0.5 or the process terminates directly with probability 0.5. From Fig. 12 it can be observed that the models from literature are more close to the simulation and to the method proposed in this paper, this can be explained by less cases being processed in task *b* and therefore minimizing the effect of shared resources as identified in the sequential pattern.

For the loop construct a single task *a* has after completion of the task the probability of 0.1 to be performed again and a probability of 0.9 that the process is terminated. The model of Sheng et al. only takes into account the first three iterations of the loop, so a difference might be expected in the sojourn time, but with a probability of 0.1 of re-entering the loop, the fourth iteration has a probability of 0.1^4 of occurrence, which does not influence the results on the sojourn time. All models perform equally well for the loop setup as can be seen in Fig. 13, provided that there is no difference now between the models due to shared resources, since there is only a single activity.

For the parallel construct two tasks are put in parallel, where the waiting time ends as soon as the first tasks starts service and the processing time ends when the last task is finished. From Fig. 14 it can be clearly observed that the solution presented in this paper indeed conforms to the simulation results. The models from literature deviate significantly from the real result for the parallel construct, since they do not support the shared resources between the two parallel tasks. Therefore the sojourn time is significantly higher at a specific utilization rate, since the cases need to wait until the dedicated resource becomes available, even when the other resource might be idle.

In order to check whether the proposed exact solution for the parallel construct in this paper is also valid in cases where general distributions are used a comparison is made for a model with normally distributed processing times and the formulas as

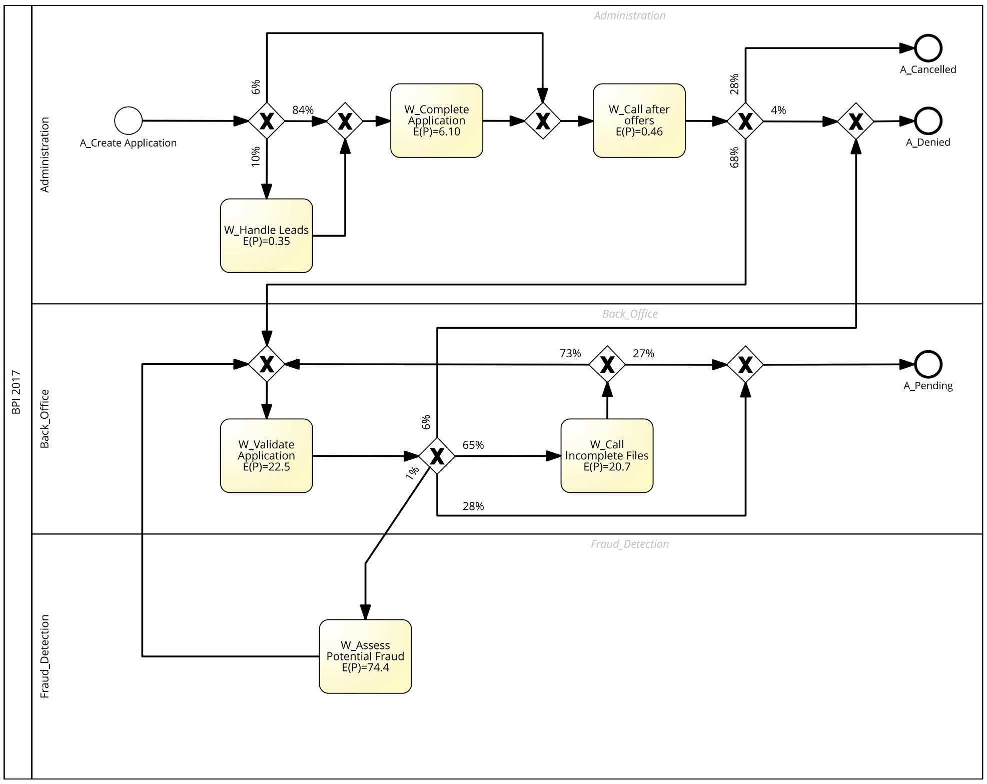
BPI 2017
Administration
Back_Office
Fraud_Detection
A_Create Application
6%
84%
10%
W_Complete Application E(P)=6.10
W_Call after offers E(P)=0.46
W_Handle Leads E(P)=0.35
28%
4%
68%
A_Cancelled
A_Denied
73%
27%
A_Pending
W_Validate Application E(P)=22.5
6%
65%
1%
28%
W_Call Incomplete Files E(P)=20.7
W_Assess Potential Fraud E(P)=74.4

Fig. 16. BPI 2017 process model.

presented in this paper. In Fig. 15 it can be seen that the proposed exact solution still outperforms the existing models from literature for the parallel construct. Since there is only a small deviation, relatively compared to the other techniques, from the simulated result it is a good indication that the presented solution is robust for general distributions.

Table 5
Evaluation results BPI 2017.

Model	$\lambda = 0.2$	$\lambda = 0.4$
Xie et al. [7]	60.05	102.30
Sheng et al. [8]	59.66	83.30
This paper	54.17	89.85

7.2. Real-life process

In order to verify if the presented queueing analysis method also work for a real-life size business process the BPI 2017 dataset [52] is used as a basis for constructing a business process model to analyze using the method described in this paper. The BPI 2017 dataset is chosen because it contains a set of activities in the workflow where the processing time is precisely recorded. We constructed a business process model for the BPI 2017 dataset by using the Disco process mining toolkit [53] and focusing only on the tasks involved in the workflow, indicated by the letter W in the dataset (i.e. the tasks that involved human activity). Using a filter in Disco to filter out the other elements of the data the process model that is shown in Fig. 16, is obtained. The process is simplified by using only 10 percent of all paths to improve readability of the process. It is important to note that the simplification of paths does not influence the performance of the queueing analysis method.

The process described in the BPI 2017 dataset and illustrated in Fig. 16 is an application process for a loan. In the process three different roles are identified, the first role is the application administration, the second role is the back office and the third role is the fraud detection department. All tasks are assumed to have an exponentially distributed processing time with the mean value as their parameter. Furthermore, in order to obtain the arrival rate, the number of cases is divided by the total time the dataset is recorded over, this results in a cases arriving on average every 18 min. The interarrival time is again be assumed to be exponentially distributed.

In order to validate the results of the queueing analysis in a real-life size process model a simulation of the model is executed. The simulation can be considered a more accurate representation of reality than the queueing analysis, because it makes fewer assumptions than the queueing model. Therefore, a trade-off can be made between the accuracy of the simulation versus the speed of the queueing analysis in analyzing business processes. For this case study, a simulation was run with a warm-up time of 1000 h, a total duration of 5000 h and 30 replications. Running the simulation on a computer equipped with an Intel i7-6700 K running at 4.4GhZ with 16GB RAM, the simulation took 4 min to complete. Compared to the queueing analysis method, which takes only a few milliseconds to complete, it is substantially slower. This is no issue if there is only a single simulation to be executed. But for optimization purposes, where thousands of scenarios need to be analyzed, the difference between the execution times of both techniques becomes relevant.

For comparison of the different methods on this case a simulation study is performed at the obtained λ of 0.2. Given that the waiting time for that scenario is minimal since the utilization rate is relatively low, another scenario is also investigated with the λ of 0.4, where the utilization rate is high.

The results of the comparison are summarized in Table 5. In the first scenario with a λ equal to 0.2 we obtain from the simulation model an expected sojourn time of 54.28 h is obtained with a 95-percent confidence interval ranging from 53.30 h to 55.26 h. The expected sojourn time from the queueing analysis is 54.17 h. For the technique from Sheng et al. [8] we obtain an expected sojourn time of 59.66 h. For the technique from Xie et al. [7] we obtain an expected sojourn time of 60.05 h. Both other techniques are outside of the confidence interval, where the queueing analysis is within the confidence interval.

In the second scenario with a λ equal to 0.4 we obtain from the simulation model an expected sojourn time of 89.62 h with a 95-percent confidence interval ranging from 83.59 h to 95.64 h. The expected sojourn time from the queueing analysis is 89.85 h. This approaches the expected value for the sojourn time of the simulation model. For the technique from Sheng et al. [8] we obtain an expected sojourn time of 83.30 h. This time is lower than expected since it is outside the confidence interval, this is caused by the lack of incorporation of the frequent loop in the back office. For the technique from Xie et al. [7] we obtain an expected sojourn time of 102.30 h. This time is higher than expected, where it is outside of the confidence interval. Therefore we can conclude that the proposed technique works best in a scenario with a high, and more realistic, utilization rate.

Concluding, for this single case study the computational efficiency gain of the queueing analysis is multiple orders of magnitude over the simulation model. At the same time the accuracy of the queueing analysis is in line with that of the simulation model with a 95% level of confidence. These results are in line with the findings of the theoretical evaluation above.

8. Conclusion and future work

In this paper a new method for translating BPMN models to queueing models is introduced. This method is based on Dijkman et al. [1], which is extended with a new approximation for the parallel construct. The new method for translating BPMN models translates every role into a queue and analyzes the queueing model based on the possible paths through the process. In order to make this method suitable for business process analysis a suitable approximation for the parallel construct needed to be developed. The new approximation is evaluated and compared to the state-of-the-art literature. From this evaluation it is clear that the new approximation for the parallel construct performs significantly better than the state-of-the-art in predicting the sojourn time under different utilization rates. Therefore this new method for analyzing business processes using queueing models is an improvement over the current state-of-the-art literature and is well suited for business process analysis, because it supports sequence, choice, loop and parallel constructs. A single case study with a real-world dataset confirmed the findings from the theoretical evaluation, showing that the analysis results of the queueing model are in line with those of a simulation model (with a 95% level of confidence), while being several orders of magnitude faster. In future work, these results – and in particular the accuracy results – can be studied further in more case studies. In future work we also aim to develop methods in which the proposed model is used to optimize business processes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The research described in this paper is supported by the Dutch Organization for Scientific Research as part of the DATAS project with grant number 438-15-507.

References

[1] R.M. Dijkman, I. Adan, S.P.F. Peters, Advanced queueing models for quantitative business process analysis, in: Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, 2018, pp. 260–267.
[2] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, Fundamentals of Business Process Management, Springer Berlin Heidelberg, 2013.
[3] C. Gröger, H. Schwarz, B. Mitschang, Prescriptive analytics for recommendation-based business process optimization, in: International Conference on Business Information Systems, Springer, 2014, pp. 25–37.
[4] J. Eckert, S. Schulte, N. Repp, R. Berbner, R. Steinmetz, Queuing-based capacity planning approach for web service workflows using optimization algorithms, in: Digital Ecosystems and Technologies, 2008. DEST 2008. 2nd IEEE International Conference on, IEEE, 2008, pp. 313–318.
[5] Y. Xie, C.-F. Chien, R.-Z. Tang, A dynamic task assignment approach based on individual worklists for minimizing the cycle time of business processes, Comput. Ind. Eng. 99 (2016) 401–414.
[6] W.M. van Der Aalst, Workflow patterns, in: Encyclopedia of Database Systems, 2009, pp. 3557–3558.
[7] Y. Xie, C.-F. Chien, R.-Z. Tang, A method for estimating the cycle time of business processes with many-to-many relationships among the resources and activities based on individual worklists, Comput. Ind. Eng. 65 (2) (2013) 194–206.
[8] L. Sheng, F. Yushun, L. Huiping, Dwelling time probability density distribution of instances in a workflow model, Comput. Ind. Eng. 57 (3) (2009) 874–879.
[9] B.-H. Ha, H.A. Reijers, J. Bae, H. Bae, An approximate analysis of expected cycle time in business process execution, in: International Conference on Business Process Management, Springer, 2006, pp. 65–74.
[10] W.M.P. Van der Aalst, A.H.M. Ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distrib. Parallel Databases 14 (1) (2003) 05–51.
[11] M. Zur Muehlen, J. Recker, How much language is enough? theoretical and practical use of the business process modeling notation, in: Seminal Contributions to Information Systems Engineering, Springer, 2013, pp. 429–443.
[12] J.H. Son, J.S. Kim, M.H. Kim, Extracting the workflow critical path from the extended well-formed workflow schema, J. Comput. System Sci. 70 (1) (2005) 86–106.
[13] A.M. Law, W.D. Kelton, Simulation Modeling and Analysis, Vol. 2, McGraw-Hill, New York, 1991.
[14] W.M.P. Van der Aalst, J. Nakatumba, A. Rozinat, N. Russell, Business Process Simulation, Springer, Berlin Heidelberg, 2010.
[15] M.A. Centeno, M.F. Reyes, So you have your model: What to do next, in: Proc. of the 30th Winter Simulation Conference, IEEE Computer Society Press, 1998, pp. 23–30.
[16] R.J. Paul, G.M. Giaglis, V. Hlupic, Simulation of business processes, Am. Behav. Sci. 42 (10) (1999) 1551–1576.
[17] J.P. Kleijnen, W. van Groenendaal, Simulation: A Statistical Perspective, John Wiley & Sons, Inc., 1992.
[18] S.P.F. Peters, R.M. Dijkman, P.W.P.J Grefen, Quantitative effects of advanced resource constructs in business process simulation, in: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC), IEEE, 2018, pp. 115–122.
[19] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining – predicting delays in service processes, in: Advanced Information Systems Engineering, Springer Berlin Heidelberg, 2014, pp. 42–57.
[20] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining for delay prediction in multi-class service processes, Inf. Syst. 53 (2015) 278–295.
[21] A. Senderovich, J.C. Beck, A. Gal, M. Weidlich, Congestion graphs for automated time predictions, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4854–4861.
[22] S. Balsamo, A. Marin, Queueing networks, in: International School on Formal Methods for the Design of Computer, Communication and Software Systems, Springer, 2007, pp. 34–82.
[23] H. Chen, D.D. Yao, Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization, Vol. 4, Springer, 2001.
[24] D. Gamarnik, A. Zeevi, et al., Validity of heavy traffic steady-state approximations in generalized jackson networks, Ann. Appl. Probab. 16 (1) (2006) 56–90.
[25] S. Qu, J. Wang, J. Jasperneite, Dynamic scheduling in large-scale stochastic processing networks for demand-driven manufacturing using distributed reinforcement learning, in: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Vol. 1, IEEE, 2018, pp. 433–440.
[26] J. Sommer, J. Berkhout, H. Daduna, B. Heidergott, Analysis of jackson networks with infinite supply and unreliable nodes, Queueing Syst. 87 (1) (2017) 181–207.
[27] F. Baccelli, W.A. Massey, D. Towsley, Acyclic fork-join queuing networks, J. ACM 36 (3) (1989) 615–642.

[28] Y. Zeng, A. Chaintreau, D. Towsley, C.H. Xia, Throughput scalability analysis of fork-join queueing networks, Oper. Res. 66 (6) (2018) 1728–1743.
[29] E. Özkan, A.R. Ward, On the control of fork-join networks, Math. Oper. Res. 44 (2) (2019) 532–564.
[30] V. Nguyen, Processing networks with parallel and sequential tasks: Heavy traffic analysis and brownian limits, Ann. Appl. Probab. (1993) 28–55.
[31] V. Nguyen, The trouble with diversity: Fork-join networks with heterogeneous customer population, Ann. Appl. Probab. (1994) 1–25.
[32] D.G. Kendall, Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain, Ann. Math. Stat. (1953) 338–354.
[33] G. Berkenstadt, A. Gal, A. Senderovich, R. Shraga, M. Weidlich, Queueing inference for process performance analysis with missing life-cycle data, in: 2020 2nd International Conference on Process Mining (ICPM), IEEE, 2020, pp. 57–64.
[34] L. Kleinrock, Theory, Volume 1, Queueing Systems, Wiley-Interscience, 1975.
[35] J.D. Little, A proof for the queuing formula: L=λ w, Oper. Res. 9 (3) (1961) 383–387.
[36] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, New York, NY, USA, 2013.
[37] L. Kleinrock, Queueing Systems, Volume 2: Computer Applications, Vol. 66, wiley, New York, 1976.
[38] S. Ross, Introduction to Probability Models, Elsevier, 2014.
[39] R.W. Wolff, Poisson arrivals see time averages, Oper. Res. 30 (2) (1982) 223–231.
[40] J.A. Buzacott, J.G. Shanthikumar, Stochastic Models of Manufacturing Systems, Prentice Hall, 1993.
[41] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, SIAM, 1999.
[42] S.P.F. Peters, Analysis and Optimization of Resources in Business Processes, Eindhoven University of Technology, The Netherlands, 2021.
[43] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, in: Business Process Management, Springer Berlin Heidelberg, 2008, pp. 100–115.
[44] A. Polyvyanyy, J. Vanhatalo, H. Völzer, Simplified computation and generalization of the refined process structure tree, in: Web Services and Formal Methods, Springer Berlin Heidelberg, 2011, pp. 25–41.
[45] S.J.J. Leemans, D. Fahland, W.M.P. van der Aalst, Discovering block-structured process models from event logs - a constructive approach, in: Application and Theory of Petri Nets and Concurrency, Springer Berlin Heidelberg, 2013, pp. 311–329.
[46] M. La Rosa, M. Dumas, R. Kaarik, R.M. Dijkman, Business process model merging: An approach to business process consolidation, ACM Trans. Softw. Eng. Methodol. 22 (2) (2013) 11:1–11:42, http://dx.doi.org/10.1145/2430545.2430547.
[47] J. Vanhatalo, H. Völzer, F. Leymann, Faster and more focused control-flow analysis for business process models through SESE decomposition, in: Service-Oriented Computing, Springer Berlin Heidelberg, 2007, pp. 43–55.
[48] C. Ouyang, M. Dumas, A.H. Ter Hofstede, W.M. Van der Aalst, From bpmn process models to bpel web services, in: 2006 IEEE International Conference on Web Services (ICWS'06), IEEE, 2006, pp. 285–292.
[49] X. Fu, T. Bultan, J. Su, Analysis of interacting bpel web services, in: Proceedings of the 13th International Conference on World Wide Web, 2004, pp. 621–630.
[50] A. Polyvyanyy, M. Weidlich, Towards a compendium of process technologies : the jBPT library for process model analysis, in: International Conference on Advanced Information Systems Engineering, 2013, pp. 106–113.
[51] S.P. Peters, R.M. Dijkman, P.W. Grefen, Advanced simulation of resource constructs in business process models, in: International Conference on Business Process Management, Springer, 2018, pp. 159–175.
[52] B.F. Van Dongen, Bpi challenge 2017, 2017.
[53] C.W. Günther, A. Rozinat, Disco: Discover your processes, BPM (Demos) 940 (2012) 40–44.