Service Analysis and Simulation In Process Mining

ניתוח וסימולציה מבוססי שירות בכריית תהליכים

Ph.D. Research Proposal

Arik Senderovich

Adviser: Professor Avigdor Gal

Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology

February 2014

Contents

1 Introduction

Modern business processes are supported by information systems that record process-related events into event logs. *Process mining* is a maturing research field that aims at discovering useful information about the business process from its corresponding event logs [76]. Recently, several works considered process mining in an operational setting; specifically, performance-oriented models were discovered from event data and later applied to operational analysis and support.

A central argument in the our research claims that service operations must be modeled and analyzed via custom-tailored techniques, due to their unique characteristics. For instance, the main concern in service operations is the balance between quality-of-service and resource efficiency (utilization). Consequently, a natural theoretical framework for service modeling and analysis is Queueing Theory, since it accommodates the quality-efficiency trade-off. To date, studies on operational process mining did not consider queueing models as candidates for process discovery.

The primary goal of the proposed research is to bridge the gap between operational process mining and service analysis; concretely, the research aims at discovering queueing models from event logs and applying them to operational problems. The secondary objective of this research is to provide (data-based) validity for established theoretical results in Queueing Theory. This step is achieved by validating the discovered queueing models against their originating event logs.

A preliminary study serves as a proof-of-concept for the proposed approach. As a first step, the work introduces the *queueing perspective* for process mining and defines *queue mining* as the set of techniques that extracts queueing models from event logs. Then, two queue mining techniques are used to obtain solutions to a specific operational problem, which is *online delay prediction*. The queue mining predictors are compared to state-of-the-art benchmark techniques from process mining with respect to well-established prediction measures. Empirical evaluation against real-life event logs provides evidence that queue mining predictors are superior to other prediction techniques. Moreover, theoretical results that were (so far) well-known in Queueing Theory gain validity through these experiments.

Another work-in-progress is the discovery of *resource networks*, which is motivated by service systems with ample servers. For example, a large call center could employ 1000's of agents, and a complex outpatient clinic has 100's of physicians, nurses and administrators. Resource networks focus on *server-flow* within a service system, which is in contrast to the traditional *customer-centric* view of Queueing Theory. Resource networks can then address operational questions on resource scheduling and utilization profiles. Furthermore, fitting queueing networks to resource paths is an uncharted territory in operational service analysis and Queueing Theory; therefore, validating these models against data will contribute to these research disciplines as well.

Future research directions beyond those mentioned above include: customer-resource models,

simulation mining, completing missing information in event logs and the foundation of a unified approach modeling business processes from both control flow and queueing perspectives.

The proposal is structured as follows. We start by providing a general overview of process mining (Section 2). In Section 3 we propose an operational paradigm to process mining that would be the basis for our future techniques. Section 4 comprises unique characteristics of services and operational models that account for these characteristics. The section provides a bridge between service analysis and operational process mining. The section ends with the statement of our goals and future research plans that follow from these goals. Section 5 describes an important preliminary to the proposal, namely *queueing models*. In Section 6 we define the service log to be an event log that comprises service events and paths. At the end of the section we discuss a research direction for completing missing data in service logs. Section 7 presents our preliminary work on *queue mining* that is based on [69], whereas Section 8 introduces the queueing perspective to operational resource analysis. The concluding section (Section 9) describes the automatic extraction of simulation models from service logs.

2 Process Mining

This section starts with a brief introduction to the rapidly developing field of *process mining*. The idea behind process mining is to extract non-trivial information on *business processes* from *event data*. Therefore, process mining is often viewed as a bridge that connects *process* modeling and analysis with *data* mining [77]. Consequently, a core assumption is that information systems record process-related events, e.g. activity executions, into so called *event logs* [76, Ch. 1]. Traditionally, process mining tasks are categorized into three types: *process discovery*, *conformance checking* and *model enhancement*. The three types of process mining tasks can be either performed independently of each other [76, Ch. 1], or alternatively, they can be considered as part of a broader integrated view [76, Ch. 8]. Below, we briefly describe the three types of tasks, then complete the introduction with an integrated approach that unites all three types under a common roof.

2.1 Process Discovery

Conceptually, process discovery techniques aim at transforming an event log into a useful representation of the underlying business process, without imposing an a priory model [76, Ch. 1]. Practically, every discovery technique considers at least one family of modeling languages to represent the business process[1].

[1] See [76, Ch. 6.5] for a historical overview of modeling languages that are relevant to business processes.

In early process mining literature, the goal of discovery algorithms was to extract models from the control flow perspective (e.g. Petri nets) [76, Ch. 5]. Specifically, the most basic discovery algorithm, the α-algorithm [79], maps event logs into workflow-nets (WF-Nets) [73], which is a special case of the Petri net formalism [10]. The α-algorithm and other early algorithms were limited in their applicability to real-life event logs due to several unrealistic assumptions. For example, event logs were assumed to be free of noise and complete, which is rarely the case for real logs (see [76, Ch. 5] for the full list of limitations and the definitions of noise and completeness). Consequently, advanced process discovery algorithms were developed to overcome these limitations. A representative selection of advanced discovery algorithms includes the *heuristic miner* [87, 88], the *fuzzy mining* algorithm [29] and the *genetic mining* algorithm [20]. For further reading on process discovery see [76, Ch. 5-6].

2.2 Conformance Checking

Given a process model (data-driven or conceptual), the question of its relevance to real business processes can be raised. Process mining literature distinguishes between four model-quality notions: *fitness*, *simplicity*, *precision* and *generalization* [76, Ch. 5]. For example, the notion of fitness indicates how much of the observed behavior (in the event log) is feasible in the process model. Conformance checking is a quality assurance procedure that provides measures of the distance between a process model and its originating event log. Concretely, the goal of conformance checking algorithms is to provide a measure for the 'distance' between the model and the event log with respect to one of the quality measures [76, Ch. 7].

A common technique for conformance checking is *replay* [61]. The idea behind replay is that the event log is 'replayed' on top of the process model (e.g. consider the token game for Petri nets, with event data being the tokens); correspondences between moves in the log and possible moves in the model indicate conformance, whereas discrepancies between the two provide evidence for non-conformance. There are two discrepancy types that can be captured by replay: (1) sequences in the log that cannot be executed in the model, i.e. the model is under-fitting the event log and (2) behavior that is possible in the model and was not observed in the event log, i.e. the model is over-fitting the event log. The replay technique was extended to an optimization problem with two types of costs that correspond to the two types of discrepancies [2]. Moreover, several types of conformance can be checked via replay, depending on the information available in the event log [19].

Another type of conformance checking that applies the replay technique is referred to as *compliance checking* or *process verification* [86]. Suppose that we consider a *normative model*, i.e. a model that represents a set of regulations or rules that must be enforced in the real process. Replaying the log 'on' the model enables one to check for compliance of the real process to these rules [3, 55, 80].

For further applications of conformance checking techniques, see [76, Ch. 7].

2.3 Model Enhancement

Model enhancement procedures transform process models (data-driven or conceptual) into enhanced models, based on additional information extracted from the event log. One special case of the enhancement procedure is *model repair*. Suppose that the result of a conformance checking algorithm is that the data-driven model is not close enough to the event log, in terms of fitness. Then, the non-conforming model can be modified (iteratively) until it converges to a conforming model [14, 23].

Another type of model enhancement is referred to as *model extension* [76, Ch. 1]. As we already mentioned, early discovery algorithms mainly considered control flow models. Model extension was later proposed to mine event data for other perspectives, on top of the control flow perspective [76, Ch. 8]. Some of the additional perspectives that were considered in the process mining literature are: the time perspective (time prediction [56, 83, 84]), the organizational perspective (mining of social networks [81]), the case perspective (decision mining [60]) and the resource perspective (resource-aware process analysis [51]).

2.4 Bringing It All Together

In recent research on process mining, the three tasks (discovery, conformance and enhancement) are considered to be part of an integrated view [49, 76, 77]. The concept behind the holistic approach is as follows. An event log is used for the discovery of a control flow model that would already include several additional perspectives (this step can be viewed as a combination of discovery and extension). Then, the extracted model is compared to the event log (conformance checking) and the model is repaired accordingly (in case of a non-conforming model). Lastly, the conforming (multi-perspective) model is then applied to solve specific problems, e.g. configuring Workflow-Management Systems (WFM) and providing operational support [76, Ch. 8-9].

3 Operational Process Mining

We refer to operational analysis techniques that are based on event data as *operational process mining*[2]. Figure 1 presents our paradigm, which starts with a set of operational goals and an event log. Then, the event log is transformed into data-driven models that we validate against event data; the model, in turn, enables operational analysis and real-time support. Consequently, our methods throughout this proposal, follow the principles that we present in the framework.

[2] An analogous term, performance mining, was used in [58].

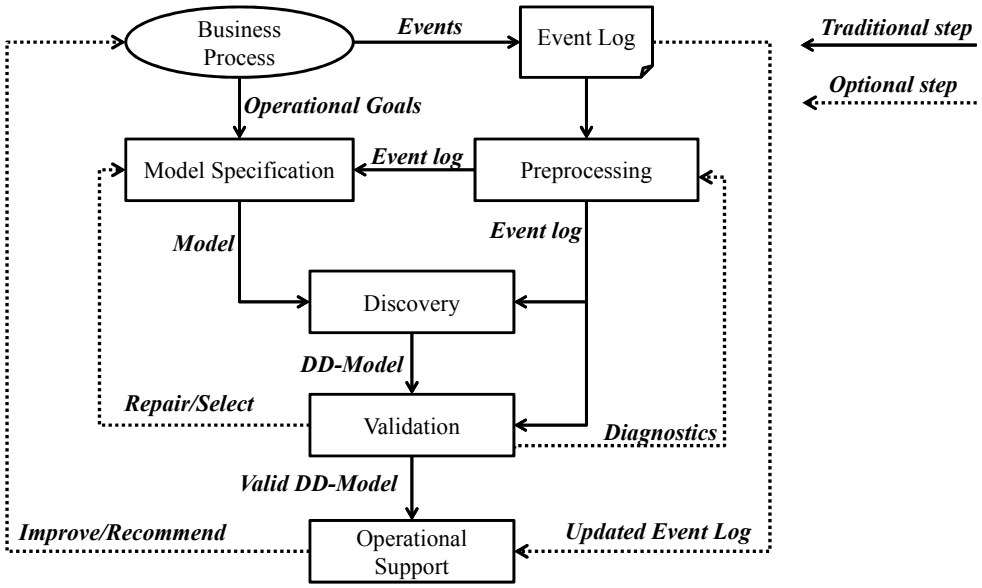

Figure 1: Our framework for operational process mining

The design of our framework has its roots in the holistic approaches that we introduced at the end of the previous section. Hence, we adjust it to operational process mining by extending current approaches, in order to accommodate operational process mining. Moreover, as we go over our framework, we demonstrate that recent research on operational process mining has sporadically applied steps that are similar to our approach; we consider this to be evidence for the timidness and value of the framework.

3.1 Event Log Extraction and Preprocessing

A central assumption in process mining is the existence of an event log that contains information regarding the underlying business process. Extracting event logs can be a challenging task, since information is often distributed and inaccessible. Even in cases of unlimited access to all available data, combining the databases into a single event log can be a difficult task [76, Ch. 4].

Once the event log is obtained, and as we now explain, a preprocessing step is required to make the log suitable for analysis. For example, Rogge-Solti et al. [57] propose a technique for repairing event logs with missing data via stochastic Petri nets and Bose and van der Aalst [14] consider a scenario in which event log traces are not aligned with the real process; therefore, the authors then develop a technique that repairs these misalignments according to some predefined logical rules.

3.2 Operational Goals, Model Specification and Discovery

The first prerequisite for operational analysis is to specify its *goals*. These goals are often formulated at a strategic level, e.g. 'improvement in service levels'; therefore, strategic goals are to be 'translated' into operational goals prior to model specification. Alternatively, the goals can be formed directly as operational goals, e.g. 'predict waiting times for arriving customers'. Once the set of operational goals is formulated and fed forward into the model specification phase, the actual process mining procedure begins.

Our framework of operational process mining is model-centered, in which both discovery procedure and operational analysis that follow are based upon operational models. These models must be aligned with the operational goals and the available event data. Moreover, an appropriate operational model reduces the need for the extension phase as it was presented in the previous section, since all relevant pieces of information are considered when the model is specified.

As an example for model specification in the literature, we mention a series of recent papers by Rogge-Solti et al. [56, 57, 58]. For operational support, they consider the stochastic Petri net (SPN) formalism to be their model of choice. This is explained by the SPN's ability to capture operational elements (execution times, routing probabilities) as well as the randomness of the underlying business processes. Moreover, SPNs can depict the control-flow perspective (e.g. conformance checking), on top of the operational one.

Model specification includes the conceptual phase of making assumptions which in turn, results in a set of building blocks (or model primitives). Then, the operational model is discovered from the log, i.e. its building blocks are mined according to the event data. The result of this single discovery procedure is a data-driven model (DD-Model). The discovery stage can be repeated for several candidate models. Thus, the input for the following stage, namely, model validation, can be a set of DD-Models.

3.3 Model Validation

Model validation, in our context, aims at comparing the discovered operational models with the event log. In our framework, we propose three types of model validity: conceptual, qualitative and operational. We now discuss these three types, based on a review paper by Sargent [64] that considers the validity of simulation models. Then, we propose a combined method for model validation that consists of the three types. Lastly, we review the notion of model validity in the literature of operational process mining and compare it to our approach.

3.3.1 Conceptual Validity

Conceptual validity is defined as analysis of the assumptions and theories that underlie the model [64]. In operational models we often make assumptions on: execution times, customer arrival rates and routing probabilities. Moreover, some operational models rely on approximations and therefore, should be tested for their applicability against data. To determine conceptual validity, we may use several techniques [64]. For example, we may apply statistical tests to verify model assumptions. Note that a model can be useful even if it fails in conceptual validity. However, if a model is not valid in one of the other two senses that we present below, we may test its conceptual validity in attempt to find a root-cause to this inappropriateness.

3.3.2 Qualitative Validity

To demonstrate qualitative validity we return to the *fitness* measure that we discussed in the previous part. The model is said to be valid with respect to the fitness measure if model behavior corresponds to the observed event data. Generally, fitness as well as other measures that are considered in conformance checking are qualitative measures; therefore, conformance checking, which is used for estimating these quality measures can be perceived as a qualitative validation procedure. Moreover, conformance checking may also be used to correct erroneous traces in the event log (as we already discussed), therefore, providing data validity (which is out of the scope of this discussion) [64].

We believe that qualitative validity is also essential from the operational perspective. As an example, consider an Emergency Department (ED) simulation model that is used for optimal planning of patient routing through the process; suppose, that the simulation outputs impossible patient paths. In such cases, a behavioral inconsistency between the reality and the simulation is observed. This is obviously an undesired situation that would cause the model to lose its validity.

Nonetheless, operational models can still be useful, even when they are not valid in both qualitative and conceptual senses. For example, in large systems, simple queueing models may capture complex realities [45, 91], however they are aggregated (i.e. may experience behavioral inconsistency with event logs) and assume unrealistic assumptions (e.g. exponential service times).

3.3.3 Operational Validity

As we already mentioned, when considering operational analysis, conceptual and qualitative validities do not suffice. To clarify this point, consider an operational model of a certain process. Suppose that the corresponding event log consists of traces with a single activity, ‘Service’. Clearly, any discovery technique would provide us with a model that would be correct in the qualitative sense: the model would also have a single activity. Moreover, suppose that we have used appropriate statistical techniques to fit the distribution and estimate the parameters of the time it takes to execute ‘Service’.

Unfortunately, the model may still end up being inaccurate in the operational sense, when compared to the event log. Specifically, execution times that would result from the model would not match their corresponding times from the event log. The phenomena could be caused by the fact that the activity 'Service' aggregates several activities that cause discrepancies in the analysis. For example, if the system is overloaded for the analyzed period in the log, queueing delays obviously influence the overall throughput time of the process. Therefore, when mining operational models, the most important notion of validity is the operational validity.

3.3.4 Model Validation: A Combined Approach

Based on the discussion of the three types of model validity, we propose the following approach for model validation in operational process mining. First, check the operational validity of the model. If it fails, test its conceptual validity to find possible causes for the discrepancies and repair the model (e.g. change its assumptions). In case that qualitative aspects are important, perform a conformance checking procedure and determine whether the model is qualitatively valid. Note that we do not consider the notion of validity to have a binary interpretation (valid/not valid); instead validity may satisfy a certain score function. Therefore, models can be accepted, rejected or repaired according to predefined thresholds for model validity.

3.3.5 Model Validation in Literature

In this part, we compare our combined approach to a technique that was applied in a paper on operational process mining. Rozinat et al. [62] proposed a method for discovering simulation models from event data. The validation technique that was presented in this work is referred to as *'second-pass' evaluation*. The method is rather simple: the simulation model runs for 1000 process instances (cases or customers); each case leaves its 'footprints' in a synthetic event log. Then, the synthetic log is used to discover the simulation model once again (rediscovery); the resulting model is then compared to the original model. The authors discussed the conceptual validity of several operational assumptions (e.g. execution times and case arrival rates). In addition, a behavioral alignment between the synthetic log and the original log was determined from several perspectives, i.e. some qualitative aspects were validated. However, the paper lacks the third type of model validity, the operational validity. Instead, the paper presents the conceptual validity as performance validation, therefore, confounding the two types.

3.4 Operational Support

Once a set of valid models is obtained from the previous stage, we are able to use them for operational support. We consider two types of operational support: offline 'What-if' analysis and real-time

support. When carrying out a 'What-if?' analysis we adjust model parameters to fit extreme or other interesting scenarios. Insights, improvements and recommendations that result from the offline analysis may then be applied to the real process. The offline approach does not require updated event data, and therefore, we consider it to be beyond our operational framework, i.e. process mining tasks end when offline analysis begins.

However, when considering real-time support, an event log that is up-to-date is required and process mining becomes relevant again. Provided with an updated event log we may, as a first step, tune model parameters based on new information, thus preventing the so-called concept drift [15]. As a second step, it is possible to use the updated event log to infer the current state of the process (e.g. by observing running cases) and apply the model to support operational decisions.

Some examples for operational support in process mining include: exploring alternative futures via simulation models [63, 75], predicting remaining execution times for running cases [56, 83, 84], online detection of deviations from normal executions [44] and decision support via a recommendation system [66].

4 Services Analysis and Simulation in Process Mining

Services constitute the center of today's western economics, and they include the financial, telecommunication and healthcare sectors [18]. In fact, the two data sets that we use for empirical evaluation of our theories and techniques are a bank's call center and a day-hospital. This section starts with an introduction to services and their operational analysis. Specifically, we show that due to their unique characteristics, services require custom-tailored modeling approaches. We conclude the section with our research objectives and an outline of the proposal.

4.1 Service Characteristics and Operational Goals

Services are economic interactions between customers and service providers (servers) that create (typically intangible) added value in return for the customer's time, money and effort [25, Ch. 1]. Furthermore, services can be characterized by several features that distinguish them from non-service activities ([25, Ch. 2]):

- Customer *participation* during service execution is often required, with several exceptions, e.g. back-office services.
- Services are consumed and created *simultaneously*, i.e. it is usually impossible to store services in an inventory.

- Inability to meet demand will cause the service to *perish*, i.e. resources that were not utilized are 'wasted' (cannot keep inventory of service providers for safety), while customers that did not receive service may never return (must allow customers to wait).
- Services are typically *intangible*, i.e. there is no actual product provided to the customer.
- Services are *heterogeneous*; customers may require different (customized) service when using the same service-system. Moreover, human service providers may serve similar customers in different fashions.

The operational goals of service analysis become apparent from these characteristics. On the one hand, the outcome of a service is not a product (intangibility), but rather, customer (dis)satisfaction; therefore, managers aim at maximizing the *Quality-of-Service* (QoS). On the other hand, resources are expensive and their idleness cannot be stored for later use; thus, managers aim at maximizing resource utilization. Clearly, these are competing goals, which implies that service models must enable the control and optimization of the balance between QoS and resource efficiency (utilization)[3].

4.2 Service Modeling and Analysis

The need to balance between QoS and efficiency in processes with waiting customers, points towards Queueing Theory as a natural discipline for modeling and analysis of service operations [45]. Correspondingly, our preliminary experiments showed that extracting queueing models from event data is indeed useful when considering operational problems [69]. Nonetheless, we do not limit the scope of service models to Queueing Theory; for example, we also consider stochastic Petri nets (SPN) to be appropriate for modeling services, since they capture both behavioral and operational aspects of complex systems[4] [31].

The unique characteristics of services directly affect model specifications. For example, the *participation* feature implies that customers may choose to wait for service or to abandon; therefore, operational models better account for *customer (im)patience*. To demonstrate the usefulness of service models, we provide a sample of possible analyses that can be accomplished via service models [45]:

- **Capacity Analysis**: Identifying resource-related bottlenecks in the current process and analyzing utilization profiles of resources.
- **Time Analysis**: Analyzing sojourn and waiting times of customers.

[3] Profitability of services is not discussed in this proposal; instead, we focus mainly on operational goals and assume that the two are strongly correlated.

[4] In Section 5 we provide a formal overview of queueing models and SPNs.

- **Sensitivity Analysis**: Testing non-existent operational scenarios by alternating model parameters and gathering future performance measures. The purpose of sensitivity analysis is to identify possible directions for process improvement.

- **Optimization**: Improving the process via some notion of optimality.

Operational service analysis can be either based on *analytical solutions* (if the service model is mathematically tractable) or on *simulation* (see [13, Ch. 11] for a thorough discussion on the advantages and disadvantages of the two types of analysis).

4.3 Research Overview

Modern business processes, which are the main target of process mining techniques, partially or entirely comprises service activities [22, Ch. 1]. Process mining studies consider the analysis of web services, which is referred to as *service mining* [78]. However, we are not aware of process mining techniques that account for the unique characteristics of services that we mentioned in the previous part. Moreover, the application of analytical service models, such as queueing networks, to operational process mining is non-existent.

The primary goal of this research is to integrate service analysis techniques into operational process mining. Specifically, as our step we introduce the *queueing perspective* that treats queues as first-class citizens to operational process mining [69]. Furthermore, we aim to modify existing simulation techniques in process mining, so that they accommodate service operations analysis and fit our paradigm for operational process mining.

Although the main contribution of this integration is expected to be in the field of process mining, we also aim to contribute to disciplines that provide conceptual service analysis (e.g. Queueing Theory). The latter objective is a by-product of the model validation step in Figure 1, i.e. we aim to test well-established theoretical results against real data, and identify new theoretical challenges as they emerge. We aim to achieve our objectives by pursuing the following research directions:

- **The Service Log** (Future Work; Section 6): The starting point of any process mining procedure is the event log. In Section 6, we introduce the definition of a *service log*, which is an event log that consists of service events. Service logs (and event logs in general) often suffer from data incompleteness, e.g. unobserved events and attributes. Therefore, at the end of Section 6, we provide a research perspective on completing missing data in service logs.

- **Queue Mining** (Preliminary work; Section 7): We define *queue mining* to be the specification, discovery, validation and operational support of services via queueing models. In a preliminary work on *queue mining*, we demonstrate the applicability of queueing models and theoretical

results in solving operational problems [69]. The work establishes the queueing perspective in business process mining and analysis, on top of the existing perspectives (e.g. control flow, time).

- **Resource Mining** (Work-in-progress; Section 8): Service models consider customers as dynamic entities, while resources are often regarded as static. For example, a central performance measure in queueing models is customer waiting time. We define resource mining as the discovery of service models from the resource perspective with servers being the flowing entities. In other words, resource mining is an extension to queue mining, since it introduces the queueing perspective to the world of resource analysis.

  Recent studies examined the resource perspective in process mining, yet their results were limited to estimating the availability and busyness of resources [50]. Moreover, we are not aware of operational service models that treat resources as dynamic entities. Therefore, this research direction can potentially become a two-way contribution, to both process mining and service analysis.

- **Simulation Mining** (Future Work; Section 9): The downside of analytical service models (e.g. queueing networks) lies in their high-level of abstraction. In complex systems, these models retain tractability at the expense of oversimplifying reality in a manner that reduces their behavioral and operational validity. In such scenarios, simulation models are useful for service analysis, since they are able to capture reality in detail and, if properly configured, provide valuable results. We refer to the extraction and configuration of simulation models from event data as *simulation mining*. Although the topic was already considered in previous work on operational process mining [62, 63, 82], we are hoping that the adjustment of these techniques to specifically accommodate services can be a potentially significant contribution.

Each of these research plans is described in a separate section and includes further future work that is directly related to the discussed topic.

5 Queueing Models

Before going further into some of the research directions, we take a detour to present *queueing models*. Specifically, we introduce single-station queues and their multi-station extension, namely, queueing networks (QNs). For both types we provide notation and a guiding example that will be used for mining queueing models. The discussion on queueing models is primarily based on [13, 32].

5.1 Single-Station Queues

Single-station queues are characterized by the following notation, known as Kendall's notation [40]:

$$\mathcal{A}/\mathcal{B}/\mathcal{C}/\mathcal{Y}/\mathcal{Z}.$$

The arrival process ($\mathcal{A}$) is defined by the joint distribution of the inter-arrival times (assuming that a single customer arrives at a time). Whenever no distributional assumption regarding the arrival process is made, $\mathcal{A}$ is replaced by G for *general* distribution. The processing duration of a single case ($\mathcal{B}$) is described by the distribution of service time. The total number of agents at the queueing station is denoted by $\mathcal{C}$. When a case arrives and all agents are busy, the new arrival is queued.

The maximum size of the system, $\mathcal{Y}$, can be finite, so that new customers are blocked if the number of running cases is larger than $\mathcal{Y}$. In large call centers, $\mathcal{Y}$ is practically infinite and can be omitted. On the other hand, healthcare operations are often limited in waiting places and therefore $\mathcal{Y}$ is naturally assumed finite. Once an agent becomes available and the queue is not empty, a customer is selected according to some routing policy $\mathcal{Z}$. The most common policy is the FCFS (First Come First Served) policy and in that case $\mathcal{Z}$ is omitted from the notation. For a discussion on policy types c.f. [13, Ch. 6]. An additional element that we consider is customer (im)patience. Customers that seek a service, unlike goods in manufacturing processes, may decide to abandon the queue. Therefore, the information on the distribution of customer (im)patience ($\mathcal{G}$), is added following a '$+$' sign at the end of Kendall's notation.

For mathematical tractability and sometimes backed up by practice, it is often assumed that sequences of inter-arrival times, service times and customer (im)patience are independent of each other, and each consists of i.i.d. elements that are exponentially distributed. (In particular, this is to say that the arrival process is a time-homogeneous Poisson process.) Then, $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{G}$ are replaced by M (which apparently stands for Markovian). Figure 2 depicts a queueing model that we consider in the queue mining section (Section 7). The model can be written as, $G/M/n + M$, i.e. inter-arrival times come from a general distribution, service and (im)patience times are exponential with n statistically identical servers, unlimited queueing capacity and an FCFS routing policy.

5.2 Queueing Networks

The skeleton of a queueing network is an undirected graph, with each node corresponding to a queueing station and each arc indicating whether the source and target stations are interconnected. Single-class networks assume that a single customer class is flowing through the different nodes[5]. For our purposes here, single-class queueing networks can be divided into *open* and *closed* networks.

[5]For the extension to mixed and multi-class networks, see [13, Ch. 7].

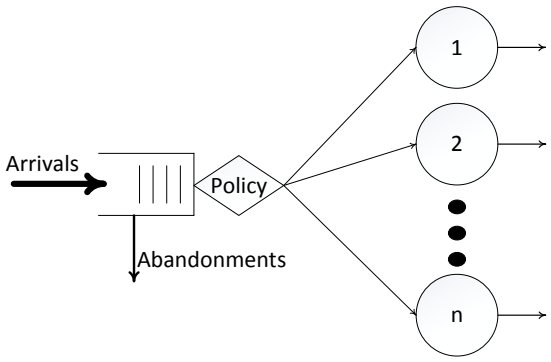

Figure 2: An illustration of the $G/M/n+M$ queueing system

In open networks, customers arrive, receive a series of services and eventually depart. In closed networks, a constant number of customers is circulating within the system. In this part, we introduce the notation for a (single-class) closed network, since this particular model is applied to resource mining in Section 8. A single-class closed queueing network is a tuple, $(K, N, \boldsymbol{n}, \boldsymbol{\mu}, R)$, where:

- K is the finite number of customers circulating the network.
- N is the number of queueing stations.
- $\boldsymbol{n} = (n_1, ..., n_N)$ is the number of servers in each station.
- $\boldsymbol{\mu} = (\mu_1, ..., \mu_N)$ is the vector of service rates per station: μ_i is the service rate at station i.
- $R = [R_{i,j}]$ is the routing probability matrix: $R_{i,j}$ represents the probability of routing to node j, after finishing service at node i.
- The service policy is assumed FCFS (unless stated otherwise) and the servers in each station are assumed i.i.d.

Figure 3 presents an example of a closed network, with node number 1 being a hub to which customers return after receiving service in nodes $2, ..., N$.

6 The Service Log

In this part, we define the *service log* (S-Log), which is an event log (c.f. [76, Ch. 4]) that consists of service events and paths. Then, we provide an example of mapping a database into an S-Log and lastly, we introduce a research direction for completing missing data in S-Logs.

6.1 Definition

As a first step to defining an S-Log, we make the following assumptions:

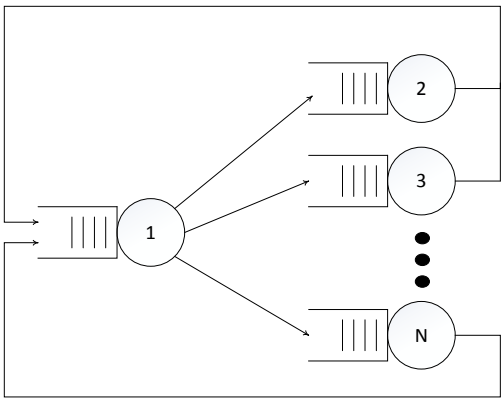

Figure 3: An illustration of a closed queueing network

- Service entities (e.g. customers, resources) go through *service paths* that consist of *service events*.
- Service events and paths must have unique identifiers (events and paths cannot have the same identifier).
- Service events and service paths both have attributes.

In order to define the service log, we shall now define service events, service paths, relate events to paths and define their attributes. We shall refer to service events as events and to service paths as paths.

Definition 1 (Service event, Service path)**.** *Denote by $\mathcal{S}$ the set of all possible service events, i.e. unique event identifiers. Let S^* be the set of all finite sequences over $\mathcal{S}$. We define $\Pi \subseteq S^*$ as the set of all* feasible *service paths, i.e. finite sequences of service events. We require that each service event appears at most once in some path.*

In other words, a path $p \in \Pi$, is a finite sequence of events such that $p_i \in \mathcal{S}, i = 1, ..., n$, with n being the length of the path[6].

Service events are associated with attributes, e.g., *timestamps*, *service activities*, *service locations* and *resources*. We denote by $\mathcal{A}_\mathcal{S}$ the set of all event attribute spaces: $\mathcal{A}_\mathcal{S} = \{A_i | i \in I\}$, with A_i being the ith attribute space and I being the event attribute indices.

Definition 2 (Event schema, Event attribute function (EAF))**.** *The event schema, α, is defined as a function $\alpha : \mathcal{S} \to \bigtimes_{i \in I} A_i$. Note that $\alpha = (\alpha_i, i \in I)$, where α_i is the ith event attribute function (EAF), namely $\alpha_i : \mathcal{S} \to A_i$.*

[6]In process mining, paths are referred to as *cases* (that are associated with customers); in the proposed research, we wish to capture resource paths as well, hence the extension of cases to paths.

For example, if A_i is the space of service activities (e.g. $\{$service start, service end$\}$), then for a service event $s \in S$, $\alpha_i(s)$ is the corresponding service activity.

Similarly, service paths are associated with attributes, such as *a unique path identifier* or *service entity* (e.g. customer, resource) and so on. We denote by $\mathcal{B}_{\Pi}$ the set of all event attribute spaces $\mathcal{B}_{\Pi} = \{B_i | j \in J\}$, with B_i being the jth attribute space and J the path attribute indices.

Definition 3 (Path schema, Path attribute function)**.** *The path schema, β, is defined as a function $\beta : \Pi \to \underset{j \in J}{\times} B_j$. Denoting $\beta = (\beta_j, j \in J)$, each β_j is the path attribute function: $\beta_j : \Pi \to B_j$.*

Service paths consist of events, with each event having its set of attributes. Therefore, we may also have path attributes that would be aggregations of event attributes. The definition of such aggregated path attributes remains out of the scope of the current proposal.

Only subsets of events and paths will actually be observed (i.e. recorded) in the S-Log. Therefore, we denote by $S \subseteq \mathcal{S}$ the set of observed events and by $P \subseteq \Pi$ the set of observed paths. We are now ready to define the service log (S-Log).

Definition 4 (S-Log)**.** *The service log is defined over $\mathcal{S}, \Pi, \mathcal{A}_{\mathcal{S}}, \mathcal{B}_{\Pi}$ as a tuple $(S, P, \alpha_{\mathcal{S}}, \beta_{\Pi})$, where*

- *$S \subseteq \mathcal{S}$ is the set of* observed *service events.*
- *$P \subseteq \Pi$ is the set of* observed *service paths.*
- *$\alpha_{\mathcal{S}} = \langle \alpha_i \rangle, i \in L \subseteq I$ is the event schema.*
- *$\beta_{\Pi} = \langle \beta_j \rangle, j \in K \subseteq J$ is the path schema.*

Note our definition of a service log generalizes the *functional* definition of an event log as it is defined in [57] and that of the *enriched log* as it is defined in [72].

6.2 Mapping a Database into an S-Log

In real-world problems, event data is often stored in databases and must be converted into S-Logs. In the following part we demonstrate a mapping from the database of a day-hospital into an S-Log[7]. Table 1 presents a sample of records from the database. The information was recorded by a real-time locating system (RTLS) on December 3rd, 2013. Customer 12358 started an $infusion$ procedure at time $12:33:00$ in room $705C$. The infusion end time for that customer was recorded at $17:41:04$ (note that time is depicted in a universal timestamping method that counts seconds since $1/1/1970$).

An infusion nurse 30395 monitored several patients in the same room, including patient number 12358. Thus, every time the nurse approached the patient, the RTLS system recorded an event into the database. One such monitoring session started at $12:57:45$ and ended at $12:58:16$. The last activity of the nurse, during that day was recorded in the staff lounge, as she was leaving the hospital.

[7] The datasets throughout this proposal come from the Technion laboratory for Service Enterprise Engineering (SEELab): http://ie.technion.ac.il/Labs/Serveng.

Table 1: Day-Hospital S-Log - Sample from Dec. 3rd, 2013

Service Path	Type	Event	Activity	Location	Timestamp	Transaction
12358	Patient	17201	Infusion	705C	1386074035	Start
12358	Patient	944871	Infusion	705C	1386092464	End
30395	Infusion Nurse	569764	Monitor	705C	1386075465	Start
30395	Infusion Nurse	569765	Monitor	705C	1386075496	End
30395	Infusion Nurse	570998	Leave	Staff Lounge	1386075465	Start

The mapping from the schema presented in Table 1 into an S-Log is straightforward. Service events are uniquely identified by the Event attribute, and therefore, $S = \{17201, 944871, ...\}$. The set of service paths, P, contains several types of elements—those of patients and those of various resources. The path schema comprises a single path attribute function that maps paths into their types (e.g. patient, infusion nurse). The event schema constitutes unique identifier, activity, location, timestamp and transaction attribute functions.

6.3 Service Logs with Missing Data

The first step of any mining technique (e.g. queue mining, resource mining) is to state assumptions on the available information in the S-Log, i.e. provide an event schema and a path schema. In practice, essential information for the mining task is often missing due to incomplete data recordings. In this part, we propose a future research direction on completing missing data in S-Logs; however, the proposed approaches are not specific to service logs and can be applied to general event logs as well.

For motivation, consider the task of estimating the queue length of a single-station queue at a certain point in time. (Queue length is a proxy of system load and is useful in operational analysis.) Now, suppose that the event log does not contain queueing activities (e.g. queue start and queue end); therefore, we are unable to obtain queue length directly from event data. However, it is possible to infer the queue length, based only on events that correspond to service activities (e.g. service start and service end). For single-station queues, under certain assumptions (e.g. Poisson arrivals), Larson [43] developed the so-called 'queue inference engine' (QIE). The QIE provides estimates for several queueing parameters, including the queue length, based solely on service events. Furthermore, in [46] the method was extended to queueing networks.

Missing information is strongly related to the mining question at hand. Suppose that the queueing data is missing, as before; however, the task is to estimate the number of busy servers at a certain point in time. In this case, service events provide sufficient information to complete the task, even though event data is missing. Therefore, the same S-Log with missing data can have complete or incomplete information, with respect to different mining tasks.

6.3.1 Completing Missing Data in the Literature

The general problem of completing missing values in data samples was thoroughly investigated in Statistics. State-of-the-art statistical techniques for completing missing data can be found in [65]. However, these techniques primarily target surveys and do not account for dependencies that often occur in event logs of business processes (e.g. activity precedence relation). In process mining, a recent work by Rogge-Solti et al. [57], used stochastic Petri nets to complete missing events and their attributes (e.g. timestamps). Another work on the subject by Bertoli et al. [12], considered a logic-based approach to event logs with missing events.

In the remainder of this section, we follow our definition of an S-Log to categorize missing data into three types: (1) *unobserved events and paths*, (2) *missing values* and (3) *missing attribute functions*. For each category, we consider solution strategies and do not provide concrete techniques for now. Our proposed strategies for data completion correspond to the predictive approach that is described in [90].

6.3.2 Unobserved Events and Paths

Consider an S-Log that contains a *service start* event for a certain customer. At the end of the day, we observe the log and discover that the *service end* event was not recorded. If such an error is prevalent throughout the S-Log then mining techniques that are applied to the log are bound to be inaccurate. For example, consider the online delay prediction problem from Section 7; the lack of service end time would bias the estimate for average service time, which is essential in both transition system methods and in queueing predictors.

Formally, we define the set of the observed events as $S \subseteq \mathcal{S}$ and the set of the events that actually occurred as $S' \subseteq \mathcal{S}$ (clearly, $S \subseteq S'$). Unfortunately, the S-Log is defined over observed events (and not over S'). The conceptual problem of completing unobserved events is to find a method that expands S with new elements from S' until we get, or estimate that $S = S'$.

We propose an approach to complete missing events, in the spirit of [12]. Suppose that there exists a set of implication functions, F, with each implication function $f \in F$ being a function $f : S^* \rightarrow S$. Formally, the occurrence of $s_1, ..., s_n \in \mathcal{S}$ implies the occurrence of $r \in \mathcal{S}$ with respect to F, if and only if there exists an implication function $f \in F$ such that $f(s_1, ..., s_n) = r$.

For example, a simple implication function would be the following. An activity with an observed start event must have an end event. Another option for deducing F is to use a normative model, i.e. a set of regulations. For example, in the day-hospital case data, a patient is to go through a series of tests. Suppose that each of the test results is required for the patient to proceed and receive treatment. If the treatment started (its start event is observable) then it is implied that all these tests were indeed completed. However, an alternative hypothesis of medical misconduct may arise, thus turning the set

of unobserved events ($S' \setminus S$) into the set of possible medical irregularities.

Once F is properly defined, we may use structural induction to obtain the set of the observed and implied events, S''. Note that $S \subseteq S'' \subseteq S'$ and ideally, $S'' = S'$. Formally, let S be the basis and F the set of implication functions. Then, we may define S'' as the closure of S over F.

A similar procedure can be performed for unobserved paths. Here, we only provide motivation for completing missing paths, but do not go into detail due to space limitation. Suppose that we only observe service events from the customer's perspective, yet we are interested in resource mining. We have only partial recordings for the resources, i.e. given a service, we know which resource handled the customer; however, we are missing the full service paths of these resources. Completion of resource paths, given traces of customer events, is a challenging task that often arises when mining real-life S-Logs.

6.3.3 Missing Attribute Values

After completing unobserved events, one must assign values to each of its EAFs (and similarly for paths and PAFs). Even when considering a complete set of events, i.e. every event that occurred was also observed, S-Logs often suffer from missing values for some of the event attributes. In practice, completing events without their relevant attributes could turn out not very useful. We are now ready to state the problem of completing event-attribute values.

Problem 1 (Completing Event-Attribute Values)**.** *Let $(S, P, \alpha_{\mathcal{S}}, \beta_{\Pi})$ be an S-Log. Given an event attribute function, α_i, and an event, $s \in S$, such that $\alpha_i(s) = null$, find a value $\hat{\alpha}_i(s)$, such that $\hat{\alpha}_i(s)$ accurately substitutes for the real value of $\alpha_i(s)$.*

A similar problem statement can be written for completing path-attribute values.

As a motivating example to Problem 1, consider a process in which only start times of services are recorded, while service completion timestamps are censored, e.g. we have only a lower bound on the service time. For example, when customers abandon from queue, the time until they would have to wait before entering service is censored. In order to analyze service duration, techniques from statistical Survival Analysis can be applied to complete the missing timestamps [39]. Moreover, other statistical techniques, such as missing data imputations [65], are applicable as well.

6.3.4 Missing Attribute Functions

The last type of missingness in the S-Log is that of an entire attribute function. This is a generalization of Problem 1, for all $s \in S$. As a guiding example, consider the location attribute in Table 1. In case of missing EAF, none of the service events contains information on its location. Therefore, if we

aim at mining a queueing network, where we ought to distinguish between service stations, we must complete the location attribute for every service event.

We propose the following definition of a composite EAF, as a possible approach to completing missing attribute functions.

Definition 5 (Composite EAF)**.** *Let $f : \mathcal{S} \to D$ be an EAF and let $g : D \to A$ be some function; then $f \circ g : \mathcal{S} \to A$ is a composite EAF.*

Therefore, it is enough to possess an EAF, $f : S \to D$, and a function g such that $g(f(s)) \in A$, in order to indirectly obtain the missing EAF function that maps events into attribute space A. Returning to our example, suppose that service names include locations in a unique form, e.g. $InfusionRoom705C$. Therefore, a transformation from the attribute service name into a location identifier (e.g. a string parsing operation) can serve as a composite function that provides the location of the service.

7 Queue Mining

This section is based on our preliminary work, in which we establish a queueing perspective in process mining [69]. We propose to consider queues as first-class citizens and refer to the various tasks of process mining that involve the queueing perspective as *queue mining*; specifically, queue mining is defined as specification, discovery and validation and analysis of queueing models based on event data.

To demonstrate the value of queue mining, we follow our operational mining paradigm (Figure 1), i.e. we state the operational problem of *online delay prediction*, then we specify our models and provide corresponding discovery techniques. Consequently, we validate the models against the S-Log and conclude the section with a discussion on our empirical insights into queue mining and future research directions.

7.1 Online Delay Prediction

The phenomena of delay has been a popular research subject in queueing theory, see [52]. The interest in delay prediction is motivated by psychological insights on the negative effect of waiting on customer satisfaction [37]. Field studies have found that seeing the line ahead moving and getting initial information on the expected delay, both have a positive effect on the waiting experience [17, 42]. Thus, announcing the expected delay in an online fashion improves customer satisfaction.

We refer to the customer, whose delay time we wish to predict as the *target-customer*. The target-customer is assumed to have infinite patience, i.e. the target customer will wait patiently for

service, without abandoning, otherwise our prediction becomes useless. However, the influence of abandonments of other customers on the delay time of the target-customer is taken into account. We are ready to state the *online delay prediction* problem.

Problem 2 (Online delay prediction)**.** *Let W be a random variable that measures the first delay time of a target-customer. Denote by $\widehat{W}$ the predictor for W. Then, the* online delay prediction *problem aims at identifying an accurate and precise predictor $\widehat{W}$.*

Accuracy and precision are common measures for prediction (see, for example, the use of precision and recall in information retrieval [47]). In our empirical evaluation, we use, as concrete measures, absolute bias for accuracy and root mean squared error (RMSE) for precision. Note that the problem statement phase corresponds to the stage when operational goals are passed to enact model specification, according to the operational mining paradigm (Figure 1).

7.2 The Service Log

Below, we define the (single-station) S-Log that corresponds to event data from single-station queues. Specifically, we provide the assumed attributes for both service events and service paths. Although the S-Log is adapted to single-station queueing models, its definition can be easily extended to more general queueing models (e.g. queueing networks).

Definition 6 (Single-Station S-Log)**.** *The single-station service log is the tuple $(S, P, \alpha_{\mathcal{S}}, \beta_{\Pi})$ over $\mathcal{S}, \Pi, \mathcal{A}_{\mathcal{S}}, \mathcal{B}_{\Pi}$, where*

- *$\alpha_{\mathcal{S}} = (\tau, \alpha)$ is the event schema.*
- *$\tau : \mathcal{S} \to \mathbb{N}^{+}$ is the timestamp EAF.*
- *$\alpha : \mathcal{S} \to A = \{qEntry, qAbandon, sStart, sEnd\}$ is the service activities EAF.*

Note that service paths in the S-Log are assumed to be structured, e.g. a $qAbandon$ event cannot occur before $qEntry$. In other words, we assume that the log corresponds to the logical order of events in a single-station queue.

7.3 Model Specification and Discovery Queries

In order to solve the delay prediction problem, we propose two queue mining techniques that have grounds in Queueing Theory: one is based on an exact analysis of queueing models, while the other is based on a result from heavy-traffic approximations.

7.3.1 Queueing Predictors

Idea We define two delay predictors based on the $G/M/n$ and the $G/M/n + M$ models, respectively. We refer to the first predictor as queue-length (based) predictor (QLP) and to the second as

queue-length (based) Markovian (abandonments) Predictor (QLMP) [38]. As their names imply, these predictors use the queue length (in front of the target customer) to predict its expected delay. We define the queue-length, $L(t)$, to be a random variable that quantifies the number of cases that are delayed at time t. The QLP for a target customer arriving at time t is:

$$\widehat{W}_{QLP}(L(t)) = \frac{(L(t)+1)}{n\mu} \tag{1}$$

with n being the number of agents and μ being the service rate of an individual agent. The QLMP predictor assumes finite patience and is defined as:

$$\widehat{W}_{QLMP}(L(t)) = \sum_{i=0}^{L(t)} \frac{1}{n\mu + i\theta}. \tag{2}$$

Intuitively, when a target-customer arrives, it may progress in queue only if customers that are ahead of him enter service (when an agent becomes available, at rate $n\mu$) or abandon (at rate $i\theta$ with i being the number of customers in queue). For the QLP, $\theta = 0$ and thus the QLMP predictor (Eq. 2) reduces to the QLP predictor (1).

Queue Mining Provided with an S-Log that is up-to-date, at time t, we extract the primitives required for calculating the QLP and QLMP. We denote by k_p the length of path p. We start with $L(t)$ and n:

$$\begin{aligned} \widehat{L(t)} &= |\{p \in P | \alpha(p_{k_p}) = qEntry\}|, \\ \hat{n} &= |\{p \in P | \alpha(p_{k_p}) = sStart\}|. \end{aligned}$$

In other words, the queue length is estimated by the number of paths that have experienced only a $qEntry$ event, while the number of servers online is estimated by the number of customers that are in service at time t. Note that the latter estimator can be inaccurate, when the queue is empty, since it does not account for idle servers. To obtain μ and θ we first define a predicate $Q(\cdot,\cdot)$ and three relations, namely, R_1, R_2, R_3:

$$\begin{aligned} Q(s_1,s_2) &= \exists p \in P, i \in \mathbb{N}^+ (s_1 = p_i \wedge s_2 = p_{i+1}), \\ R_1 &= \{(s_1,s_2) \in \mathcal{S} \times \mathcal{S} | \alpha(s_1) = sStart \wedge \alpha(s_2) = sEnd \wedge Q(s_1,s_2)\}, \\ R_2 &= \{(s_1,s_2) \in \mathcal{S} \times \mathcal{S} | \alpha(s_1) = qEntry \wedge (\alpha(s_2) = sStart \vee \alpha(s_2) = qAbandon) \wedge Q(s_1,s_2)\}, \\ R_3 &= \{p \in P | \alpha(p_{k_p}) = qAbandon\}. \end{aligned}$$

The predicate, Q, indicates that two events $s_1, s_2 \in S$ are sequential in the same path. The first relation, R_1, contains pairs of events from the same path that correspond to service start and end activities, respectively. Similarly, R_2 contains pairs of events that are sequential in the same path and indicate waiting in queue (until abandonment or service). Lastly, R_3 contains paths that ended with an abandonment. We use R_1 to estimate the average service time, m, as follows:

$$\hat{m} = \frac{\sum_{(s_1,s_2)\in R_1}(\tau(s_2) - \tau(s_1))}{|R_1|}, \tag{3}$$

and deduce a naive moment estimator for $\hat{\mu}$, $\hat{\mu} = 1/\hat{m}$ [67]. Lastly, we estimate θ based on a statistical result that relates it to the total number of abandonments and the total delay time for both served and abandoned customers, cf., [16]. Formally,

$$\hat{\theta} = \frac{\sum_{(s_1,s_2)\in R_2}(\tau(s_2) - \tau(s_1))}{|R_3|}. \tag{4}$$

7.3.2 Snapshot Predictors

Idea An important result in queueing theory is the *(heavy-traffic) snapshot principle* (see [89], p. 187). A heavy-traffic approximation refers to the behaviour of a queue model under limits of its parameters, as the workload converges to capacity. In the context of Problem 2, the snapshot principle implies that under the heavy-traffic approximation, delay times (of other customers) tend to change negligibly during the waiting time of a single customer [38]. We define two snapshot predictors: Last-customer-to-Enter-Service (LES or $\widehat{W}_{LES}$) and Head-Of-Line (HOL or $\widehat{W}_{HOL}$). The LES predictor is the delay of the most recent service entrant, while the HOL is the delay of the first customer in line.

In real-life settings, the heavy-traffic approximation is not always plausible and thus if the operational validity of the approximation is low then its conceptual validity is to be tested. The applicability of the snapshot principle predictors should be tested ad-hoc, when working with real data sets. Results of synthetic simulation runs, conducted in [38], show that the LES and HOL are indeed appropriate for predicting delays.

Queue Mining Given an S-Log that is up-to-date, at time t we mine the snapshot predictors as follows. Assuming the FCFS policy, we estimate HOL as follows:

$$\widehat{W}_{HOL} = \min_{s\in R_4} t - \tau(s),$$

where,

$$R_4 = \{s \in S | \alpha(s) = qEntry \wedge \exists p \in P(p_{k_p} = s)\},$$

which corresponds to customers that are currently waiting. The LES is estimated in two phases. First, we obtain the path that has $sStart$ as the most recent event:

$$\upsilon = \arg\max_{p \in R_5} \tau(p_{k_p}),$$

where,

$$R_5 = \{p \in P | \alpha(p_{k_p}) = sStart\}.$$

In other words, υ contains the path of the LES. (Here, we assume that every event has a unique timestamp; the difference can be in milliseconds.) Finally, in order to obtain the LES, we calculate the waiting time for υ:

$$\widehat{W}_{LES} = \tau(p_\upsilon) - \tau(p_{\upsilon-1}).$$

7.4 Model Validation

To test the operational validity of our delay predictors we ran a set of experiments on a real-life S-Log. We compared the results of our predictors against two benchmark predictors that were based on the transition system method proposed by van der Aalst, et al. [83]; first, we considered the plain transition system predictor (PTS), which performed poorly, in relation to our predictors. Then, we enhanced the state-space of the PTS with queueing information (queue length), to make the transition system method comparable to our queue mining techniques. We used the K-means algorithm to cluster the queue length into K classes and hence named the algorithm as K transition system, or KTS.

Our experiments show that the snapshot predictors outperform other predictors, in virtually every experimental setting considered. For the predictors based on transition systems we observe that the KTS leads to a significant improvement over the PTS. Both queueing predictors, in turn, performed worse than the snapshot predictors, since the queueing model assumptions suffered from low conceptual validity.

Below, we first describe the data that was used for our experiments (Section 7.4.1). Then, we define the evaluation's performance measures (Section 7.4.2) and provide our experimental setup (Section 7.4.3). Lastly, we report on the main results (Section 7.4.4).

7.4.1 Data Description

For our delay prediction experiments, we selected three months of data: January 2011–March 2011 (a queue log of 879591 records) from the *ILDUBank* data set, which contains operational event data of an Israeli bank's call center (see [68, pp. 149-157] for further description of the ILDUBank data set). This amount of data enables us to gain useful insights into the prediction problem, while easing the computational complexity (as opposed to analyzing the entire data set). The three months were selected since they are free of Israeli holidays. In our experiments, we focused only on cases that demanded 'general banking' services, which is the majority of calls arriving into the call center (89%). This case selection is appropriate, since our queueing models assume that customers are homogeneous.

We divided the experimental S-Log into two subsets: a training log and a test log. This is common practice when performing statistical model assessment [36]. The training log comprises all calls that arrived between January 1st, 2011 and February 28th, 2011; a total of 250488 delays and 247281 completed services. The test log consisted of delays that occurred during March 2011; a total of 117709 delays. We pretended that the test log customers are target-customers and predicted their expected delay. These predictions were then evaluated against the real delays via appropriate performance measures.

7.4.2 Performance Measures

To evaluate the quality of the delay predictors, we introduce two performance measures: *absolute bias*, for accuracy, and *root mean squared error* (RMSE), for precision. The absolute bias is defined as:

$$|Bias(\widehat{W})| = |E[\widehat{W}] - W|, \tag{5}$$

with W being the delay and $\widehat{W}$ being the delay predictor. We define a point estimate for the absolute bias as:

$$|\widehat{Bias}| = |\frac{1}{k}\sum_{i=1}^{k}(d_i - p_i)|, \tag{6}$$

with $i = 1, ..., k$ being the i-th test-log delay out of k delays, d_i the real duration of the i-th delay and p_i the corresponding predicted delay; $|Bias(\widehat{W})| > 0$ indicates a systematic error in our predictor, thus low accuracy.

For precision define RMSE as:

$$RMSE(\widehat{W}) = \sqrt{E[(W - \widehat{W})^2]}. \tag{7}$$

We consider a point estimate for the RMSE to be the *root average squared error* (RASE), namely,

$$RASE = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(d_i - p_i)^2}, \tag{8}$$

with d_i and p_i defined as before. Low RMSE indicates that the corresponding predictor is precise.

We consider the RMSE (and precision) to be more significant, penalizing for any type of deviation from the real delay. In contrast, the absolute bias may result in 0 (high accuracy), but deviate strongly from the delay predictor (*e.g.*, deviating strongly both above and below the real delay). We thus use accuracy as a 'compass' to detect systematic errors in model assumptions, but consider precision to be the indicator for quality of prediction.

7.4.3 Experimental Setup

The controlled variable in our experiments is the *prediction method* (or the delay predictor). Six various methods are used according to the predictors earlier defined, namely the QLP, QLMP, LES, HOL, PTS and KTS predictors. The uncontrolled variables are the two performance measures $|\widehat{Bias}|$ and RASE.

As a preliminary step, we mined the K-loads transition system (applied the KTS) from the training log with $K = 3$. The result was a clustering of the queue-length ($L(t)$) into 3 classes: 'heavy load', 'moderate load,' and 'typical load'. Given the partition into the three classes, we tested our predictors on four different experimental scenarios. Scenario I considered the entire test log and thus we refer to it as the 'all loads' scenario, while Scenarios II–IV relate to the three load-clusters and to delays that are associated with these clusters.

7.4.4 Results

Figure 4 presents the absolute bias for all six predictors under the four load scenarios. The PTS predictor presents a near-zero bias in Scenario I, but when observing its bias across scenarios we note a much larger bias. This originates in the insensitivity of the PTS predictor to system load. The KTS has a negligible bias in all scenarios, except for the one representing heavy load. This result hints at the existence of a finer partitioning of the heavy load scenario.

For the queue-length based predictors (QLP and QLMP), we observe that both predictors appear to be biased. This may point towards violations in the queueing model assumptions. The bias of the snapshot predictors (LES and HOL) is small across scenarios, indicating an absence of a systematic error in these predictors. This observation supports the applicability of the snapshot predictors to service processes in call centers.

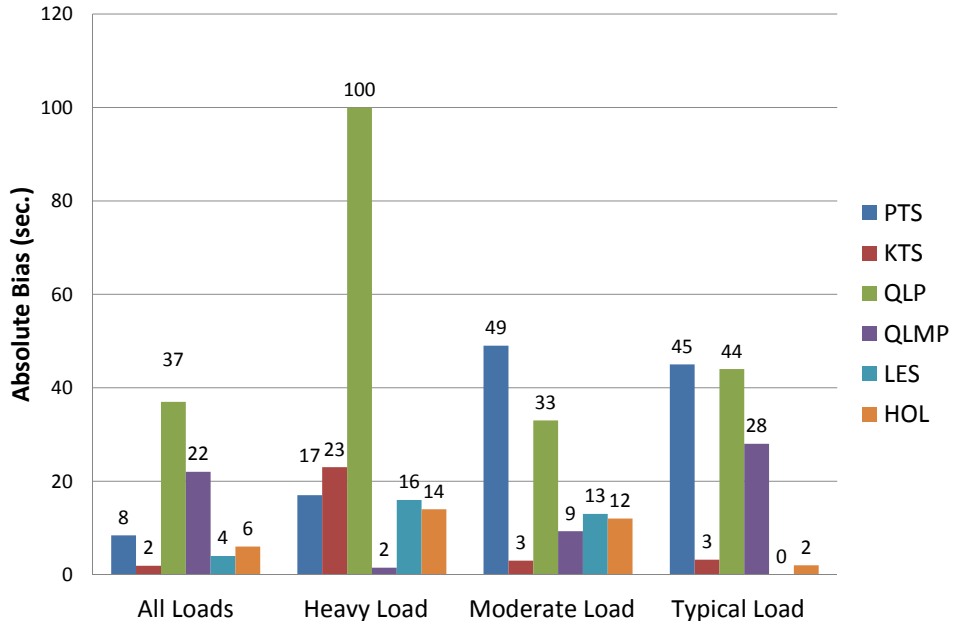

Figure 4: Sampled bias - Test set delays

Figure 5 presents the RASE in seconds. Snapshot predictors are superior across all scenarios, improving over the PTS by 34%–46%. Note that both snapshot predictors perform identically in terms of RASE. This empirical proximity between the LES and the HOL, under certain assumptions, has a theoretical background in [38] (Theorem 4.4).

The QLP performed worse than PTS across scenarios, except the moderate load scenario, while the QLMP outperformed the PTS in all scenarios except the typical load scenario. In the moderate load scenario, the QLMP performs almost as well as the snapshot predictors. The KTS outperforms both the PTS and queueing model predictors, in all scenarios, except the moderate load scenario.

7.5 Discussion

In this part we discuss the insights into queue mining that we gathered from our experiments.

7.5.1 Snapshot Principle Predictors: Recent History Dominates in Time-Varying Systems

Throughout our experiments, snapshot predictors have shown the largest improvement over the PTS method and outperformed the rest in terms of precision. Thus, we conclude that for the considered queueing process (of a call center), an adequate delay prediction for a newly enqueued customer would be the delay of either the current head-of-the-line (HOL) or the delay of the last-customer-to-enter-service (LES). Our main insight is that in time-varying systems, such as a call center, one must

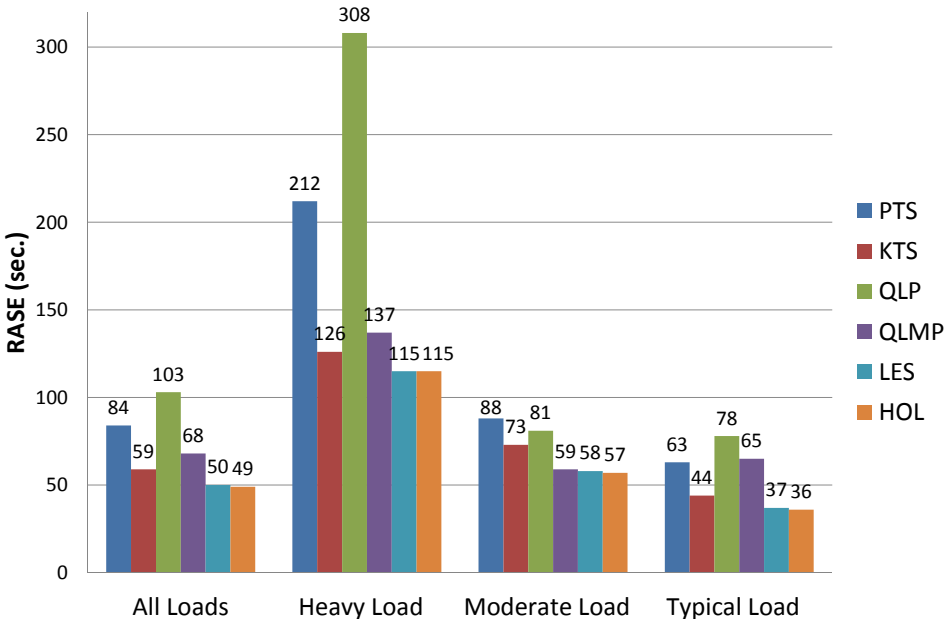

Figure 5: Root-average squared error in seconds

consider only recent delay history when inferring on arriving cases.

7.5.2 Queueing Predictors: Conceptual Validity Matters

The queueing predictors consider the time-varying behaviour of the system and attempt to quantify the system-state based on the number of delayed cases. The QLP fails in *accuracy and precision* for most scenarios, since it assumes that customers have infinite patience, which is seldom the case in call center processes. We presume that the QLP would perform better for processes with negligible abandonment rates.

On the other hand, the QLMP outperforms the PTS for most scenarios both in *accuracy and precision*. Therefore, accounting for customer (im)patience is indeed relevant in the context of call centers, and other processes in which abandonments occur [26]. In contrast, the QLMP is inferior when compared to snapshot predictors or the KTS predictor. This phenomena can be explained by deviations between model assumptions and reality. We demonstrate one possible deviation by conducting a short (descriptive) statistical analysis that is relevant for both the QLP and the QLMP. Figure 6 presents the mean service time over a single day (January 2nd, 2011, which is a typical Sunday in our training log).

The horizontal axis presents the time-of-day in a 30-minute resolution and the vertical axis presents the mean service time in seconds, during each of the 30 minutes. We see that the mean service time is mostly stable, but nonetheless violations do occur during several time points. This

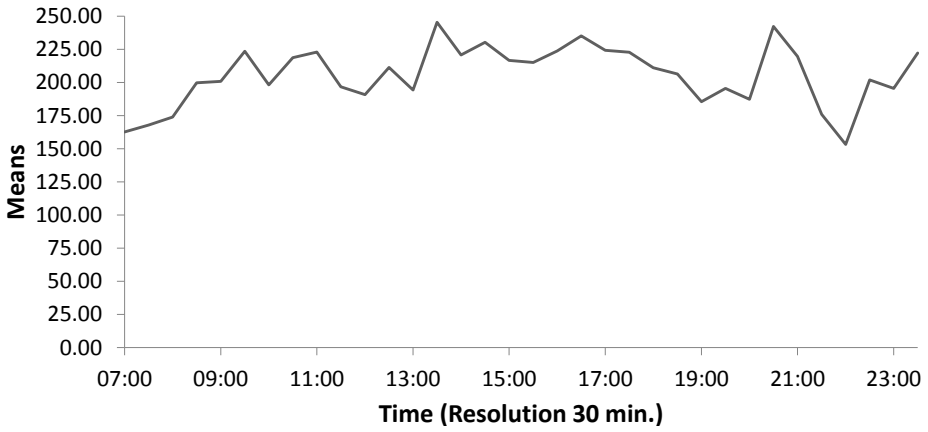

Figure 6: Mean service time during a typical Sunday (January 2nd, 2011)

fluctuation over the day may cause deterioration in overall performance of both QLP and QLMP, since these predictors assume constant mean service time. We have shown similar violations for both the constant (im)patience time and the exponential service times assumptions, but we do not present them in this paper, due to space considerations.

7.6 Future Work

The lack of conceptual validity of single-station queues and the empirical success of the snapshot principle, motivate several directions for future research. First, we aim at relaxing some of the model assumptions for the queueing predictors, e.g. exponential distribution of individual service times and (im)patience. This relaxation can cause the models to be intractable for exact analysis. Therefore, the analysis of these models often resorts to simulation (c.f. Section 9) or approximation methods in the spirit of the *snapshot principle*.

Most business processes actually consist of multiple service stations. Therefore, another research direction involves the extension of single-station models to queueing networks. Once discovered from data, queueing networks, similarly to single-station queues, can be then validated via exact analysis (e.g. product-form solutions), approximations (e.g. in heavy-traffic) and simulation. For example, in theory, the snapshot principle was shown to work well in queueing networks [53, 89]. Therefore, investigating the use of this principle to queueing networks with a complex underlying process may provide competent delay prediction.

Lastly, we aim at developing an algorithm that, given several candidate queueing models, would search for the one that is most fitting, according to measures that relate to the three validity types that we discussed in Section 4. This research direction can be viewed as a special type of *model selection* in statistical learning [36, Ch. 7].

8 Resource Mining

This section focuses on *resource mining*, which is the specification, discovery and validation of *resource networks*, i.e. queueing networks of resources[8]. In other words, we aim at integrating the *queueing perspective* in the analysis of resources in process mining. This can be viewed as the complementary side of queue mining, which brings the queueing perspective into the analysis of cases (or customers). Fitting queueing networks to resource paths is an uncharted territory in operational service analysis and Queueing Theory; therefore, validating these models against data will contribute to these research disciplines as well.

In operational process mining, resource-aware techniques were recently introduced; these studies analyze the performance of business processes from the resource perspective [50, 51]. In [51], for example, the authors used a regression analysis to show that speed of service is indeed affected by the workload. However, we are not aware of previous work in process mining that discovers server-flow from event data. In [68, Ch .2], server networks were defined as directed graphs that depicted server-flow through service activities or customer queues. Once these networks were discovered from event data they were used to estimate *resource absenteeism rate*, i.e. the rate at which servers become unavailable during their shifts. Consequently, the estimated absenteeism was used as an input to long and short-term workforce planning in call centers, i.e. as an input to *strategical* and *tactical* decisions for a service organization. We plan to continue this line of work by extracting queueing models for *operational* analysis of resources.

The remainder of the section follows our operational paradigm (c.f. Figure 1). First, we define the operational goals of resource mining; then, we mine the building blocks of a queueing network that serves as our model for the resource perspective. Consequently, we analyze server-paths on the basis of the transition matrix that corresponds to the queueing network. Furthermore, we demonstrate how a simple steady-state analysis to resource-flow can be useful to analyze time-varying performance measures. We conclude the section with an overview of future research on resource mining.

8.1 Operational Resource Mining

Before stating the operational goals of resource mining, we consider a guiding example that will serve us throughout this section. We think of the service process that is described in Figure 7. Several classes of customers arrive into class-dependent queues and wait until they are served by homogeneous servers (i.e. servers have an identical skills and service rates). In our example, customer paths are simple: customers arrive, wait in queue, receive service (or abandon) and depart. Servers however, handle several types of customers, perform 'back-office' work and consult with other

[8] Throughout the section we shall use resources and servers, interchangeably.

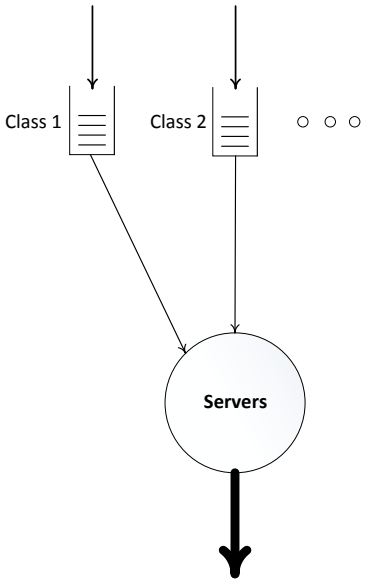

Figure 7: A multi-class queueing station with homogeneous servers

servers. Hence, in the described setting, server paths have more variety than customer paths (as our service log shall demonstrate later in the section).

Now, we return to defining the operational goals of resource mining. Operational resource mining is the discovery and analysis of (operational) service models from service logs with an emphasis on the queueing perspective. The analysis can be performed at several levels of detail. At the *aggregate level*, we consider flow-models, such as queueing networks to describe operational resource behavior. For example, such models enable us to perform *protocol mining*, which is the discovery of predefined rules for scheduling multi-skilled servers to multi-class customers. An analogous step, of mining routing protocols, can be considered for rules of matching customers to resources.

Protocols serve as an important input to service analysis techniques. For example, consider a simulation model that describes the system that we depicted in Figure 7. Suppose that a server finishes serving a class i customer and there are two customers waiting: one of type j ($\neq i$) and the other is of type i. The discovered protocol will enable us to generate a matching event between the server and a customer according to the discovered rules, thus allowing the simulator to execute a more realistic realization of scheduling decisions. Protocols can be used as input for both simulation and analytical service models (e.g. the transition matrix in a queueing network can be viewed as a simple routing protocol).

Service models such as queueing networks are appropriate to describe systems at the aggregate level, i.e. these models assume that the flowing entities are either homogeneous or can be divided

into several entity classes. However, operational questions that are related to the *individual level* of resource behavior are also relevant. For example, consider the task of *clustering* resources into several skill classes according to some performance parameter (e.g. the type of customers that these resources can serve). This task can be performed only via individual path analysis of resources. Clustering of resource paths is related to previous work in process mining on clustering cases of customers according to their recorded paths of activities, c.f. [24, 71]. Another task that is a generalization of resource clustering is resource *ranking*, i.e. inducing a total order over the discovered clusters according to predefined parameters, such as service speed (for the cluster) and individual resource protocols. Ranking of servers according to operational parameters can be viewed as an extension to the analysis of learning curves (for resources) in Industrial Engineering, c.f. [30]. The resulting operational analysis, at both the aggregate and individual levels, can be fed as input to both tactical decisions (e.g. utilization profiles) and strategic planning (e.g. long-term resource planning) [68].

8.2 Modeling Resource-Flow via Queueing Networks

We return to the service process in Figure 7, but this time we reverse the view and observe it from the resource perspective. Specifically, we consider a closed queueing network with servers being the flowing entities, c.f. Figure 3 in Section 5. We make the following set of assumptions:

- Arriving *customers* can be divided into $N-2$ classes; each customer-class requires a unique type of service.
- There are K *servers* circulating within the system.
- Customers and resources are *independent and identically distributed*. Moreover, they change roles with customers serving the resources.
- Resources can be in one of the following three states: (1) *serving* a customer (from one of the classes), (2) *ready* (i.e. waiting for customers to arrive) or (3) *unavailable* (e.g. on a break).
- When a service ends, the newly available resource is routed to serve the next customer class, based on a stochastic routing rule that is independent of time and system state.

The closed network that we consider is characterized by the tuple,

$$(K, N, \boldsymbol{n}(t), \boldsymbol{\mu}, R), \tag{9}$$

where K is the number of servers that flow through N stations, namely $N-2$ (service) *class* stations and 2 non-service stations, namely *ready* and *unavailable*. Individual service rates per station are denoted by $\boldsymbol{\mu} = (\mu_1, ..., \mu_N)$ and R denotes the individual transition matrix. The number of customers in node i at time t is denoted by $n_i(t)$ with $\boldsymbol{n}(t) = (n_1(t), ..., n_N(t))$, with $n_{N-1}(t) = n_N(t) = 0, \forall t$.

Remark 1. *The queueing model enables us to capture the individual transition matrix and service rates for each station. However, the model assumes that individual transition matrices, denoted R^k ($k = 1, ..., K$), are identical to R. Therefore, following our assumptions once R is discovered, both the aggregated and individual path analysis is complete.*

Remark 2. *The queueing model is inspired by the service process in Figure 7. However, it is general enough to capture resource-flow in any setting that lacks concurrent (parallel) activities. In other words, the model represents servers that perform a finite set of serial activities $\{1, ..., N\}$ without a predefined order (transitions between activities are stochastic).*

8.3 Resource Mining: Service Log and Discovery Queries

For resource mining, we assume the existence of an S-Log, $(S, P, \alpha_{\mathcal{S}}, \beta_{\Pi})$, that contains resource events and paths, where:

- $\alpha_{\mathcal{S}} = (\tau, \eta, \delta, \alpha)$ is the event schema and $\beta_{\Pi} = (\phi)$ is the path schema.
- $\phi : \Pi \rightarrow \mathbb{N}^+$ being a unique path identifier path attribute function (PAF).
- $\tau : S \rightarrow \mathbb{N}^+$ being a timestamp event attribute function (EAF).
- $\eta : S \rightarrow \mathbb{N}^+$ being a (customer) class identifier EAF.
- $\delta : S \rightarrow T$ being a transaction type EAF, with $T = \{Start, End\}$.
- $\alpha : S \rightarrow A$ being the activity EAF, with $A = \{Serving, Ready, Unavailable\}$.

We assume that for $s \in S$ with $\alpha(s) = Unavailable \Rightarrow \eta(s) = N$; similarly, for $\alpha(s) = Ready \Rightarrow \eta(s) = N - 1$. In other words, class N represents unavailability, while class $N - 1$ stands for readiness to serve customers.

To analyze the queueing network we must discover its building blocks, namely K, R, $\boldsymbol{\mu}$ and $\boldsymbol{n}$. However, for the preliminary analysis that we conduct further in the section, we require the discovery of R and K. For future research, in order to estimate $\boldsymbol{n}(t)$ (for future research) we can use the sum of $L(t)$ (number of delayed customers at time t) and $n(t)$ (number of served customers at time t) from the corresponding queries in Section 7.3.1. The service rates vector, $\boldsymbol{\mu}$, can be obtained in analogy to the estimation of μ in the single-station model presented in Section 7.3.1.

To discover R, we estimate each of its elements, $R_{i,j}$:

$$\widehat{R_{i,j}} = \frac{|\{s \in S | \eta(s) = i \wedge \exists p \in P, k \in \mathbb{N}^+ (p_k = s \wedge \eta(p_{k+1}) = j)\}|}{|\{s \in S | \eta(s) = i \wedge \delta(s) = Start\}|}. \tag{10}$$

In other words, the estimator is the number of times that servers moved from state i to j out of the total number of visits in state i. Note that for models that require an individual transition matrix per server ($R^k, k = 1, ..., K$) we can define a similar query and use $\phi(p)$ to distinguish between resource paths.

Now, we turn to estimating the number of servers (K) in our closed system. In practice, the number of servers is constant only over certain periods of time (e.g. busy hours). Therefore, in real service logs, we expect to observe a time-varying number of servers in the system. Denote $K(t)$ the number of servers at time t, with $K(t) = K$ in closed network settings. Then, given an up-to-date service log at time t, $K(t)$ can be estimated as,

$$\widehat{K(t)} = |\{p \in P | \delta(p_{k_p}) = Start\}, \tag{11}$$

with k_p being the length of path p. In other words, we assume that a server must be in one of the N positions at all times and that servers that left the system present $\delta(p_{k_p}) = End$ in their paths.

8.4 Resource Analysis: Preliminary Experiments

In this part we provide preliminary results of service analysis from the resource perspective. The discovery queries were applied to a single day of the *ILDUBank* data. We start the analysis with discovering the transition matrix (R). We consider R as a naive example for *resource path analysis*, since model assumptions imply that R represents both the aggregated and individual protocols. Then, we demonstrate how a *steady-state analysis* of the time spent in a station captures the predictable variability of the time-varying probability to visit that station.

8.4.1 Resource Paths Analysis

Figure 8 presents the discovered transition matrix R. The nodes correspond to the possible server states, namely: $(N-2)$ *customer classes* (blue), *ready* (green) and *unavailable* (orange). Every arc corresponds to a positive probability of going from the source node to the sink node. The number that is written next to the arcs is the estimated $R_{i,j}$, which is the transition probability from source i to sink j.

We observe that the dominant three states, with respect to probability estimators are *ready*, *unavailable* and *General Banking*. The latter is the prevalent customer class in terms of number of incoming calls (about 70% of customers are classified as General Banking). After completing a call, about 50% of calls become either ready or unavailable. From these two states, about 50% of calls go into the General Banking state. Servers that complete service in General Banking return to another General Banking call with probability of 0.44. This type of analysis is incomplete without information on length-of-stay in each state; the complementary analysis is provided in the next part.

The resulting transition matrix can feed the queueing model and serve as one of its building blocks. Moreover, as we already mentioned, the process that is depicted by the transition matrix can be observed as a (naive) protocol for server-flow: given that a server completed his visit in state i we

have an estimate for the 'jump' probability to state j ($i, j \in \{1, ..., N\}$). However, for an advanced individual analysis, we would apply resource mining queries to each server path.

8.4.2 Steady-State Analysis of Time-Varying Resources

Beyond understanding routing protocols and individual paths of resources, we are interested in an operational analysis of server-related performance measures. We demonstrate an approach to analyze resource operational behavior based on *time averages*, i.e. times that resources spend in various states (serving, ready, unavailable). The analysis corresponds to the transition process that is described in Figure 8; the formal technique is based on [34].

We consider a set of independent and identically distributed (i.id.) stochastic processes, $X_i = \{X_i(t); t \geq 0\}$ such that $X_i(t) \in \{1, ..., N\}$; each of the processes corresponds to a single server ($i = 1, ..., K$), while their values correspond to the N network nodes (or server states). We assume that all processes start at node N, which is the unavailable state (i.e. $X_i(0) = N$) and that the processes regenerate after time T (e.g. all servers return to the unavailable state). The semantics of $X_i(t) = j$ is that the ith server is occupying the jth station at time t. Note that the assumption of i.id. processes is strong, since X_i are dependent through the number of customers in the system, i.e. if there is a single type j customer in the system and $X_i = j$ then $P(X_k = j) = 0, \forall k : k \neq i$.

Denote by $p_j(t) = P(X_i(t) = j), i = 1, ..., K$ the probability of any of the processes (since they are i.id.) to be in state j at time t. Denote by τ_j the time that servers spend in the jth node between two consequent regenerations, and let $\tau = E(T) < \infty$. Then, it can be shown that

$$p_j(t) \rightarrow \frac{\tau_j}{\tau} \equiv \pi_j, \tag{12}$$

as $t \rightarrow \infty$ [34], with π_j ($j = 1, ..., N$) being the average proportion of time that servers spend in node j over the regeneration period. In other words, when observing the system in steady-state, the probability to find a process in the jth node converges to the average proportion of time that this process spends in state j.

We use this result to demonstrate that a steady-state (empirical) analysis of resource paths is useful to describe the time-varying behavior of resources. We estimate each π_j directly from event data as the total time servers spend in node j out of the total time recorded in the S-Log. The resulting vector of time averages complements the proportions that we analyzed after the discovery of R (Figure 9). Although we have seen that *ready* is a highly visited state, the average visit duration in ready is only 7 seconds. Moreover, servers spend in the ready state only 0.76% of their total time. However, visits to General Banking constitute 53% of the total time and servers are unavailable to receive calls for 20% of their work time. Such an analysis can serve as grounds to constructing a utilization profile that is useful for, e.g. workforce planning.

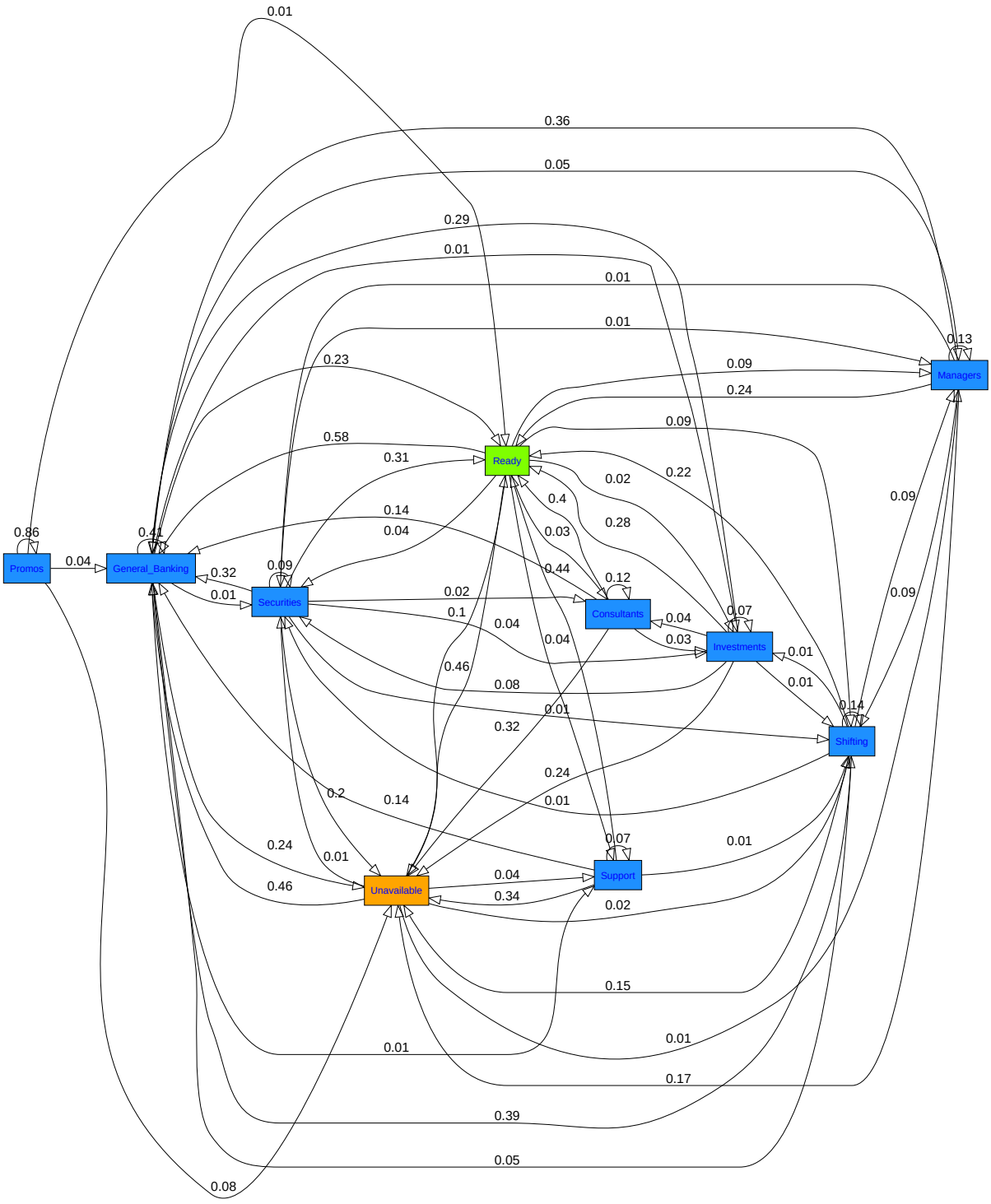

Figure 8: The discovered transition matrix

We now turn to compare the steady-state probabilities (π_j) to the time-varying probability of a server to visit node j at time t. The latter can obviously change over the day, while π_j remain constant. Denote $K_j(t)$ the number of servers at node j at time t, i.e. $K_N(t)$ the number of unavailable servers, $K_{N-1}(t)$ the number of servers that wait to serve customers and $K_1(t), ...K_{N-2}(t)$ the number that serve the respective customer class. We are now ready to present our preliminary results on operational resource analysis.

Figure 9 presents two empirical curves: one corresponds to $K_3(t)$ (blue), while the other corresponds to $\pi_3 \cdot K(t)$ (red). In other words, the blue line corresponds to the number of resources that serve the 3rd customer class (General Banking), whereas the red line corresponds to the number of servers that serve General Banking customers out of total number of servers (at time t). The vertical axis measures number of servers, while the horizontal axis depicts time between $07:00$ (beginning of the day) and $00:00$ (end of the day), in 15 minutes intervals.

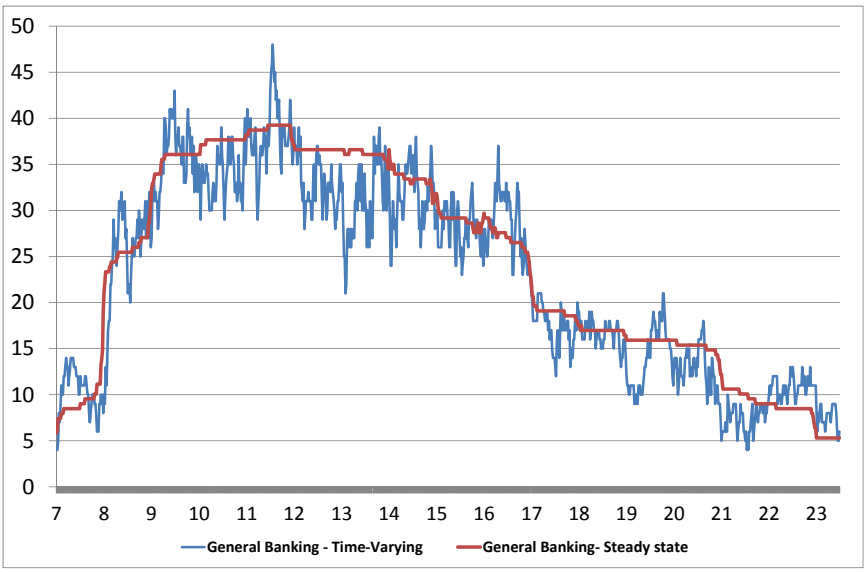

Figure 9: The discovered transition matrix

The result is somewhat surprising, since π_3 corresponds to the long-run probability to find a server at the 3rd node. However, we observe that for a time-varying $K_3(t)$, there is a small difference between $\pi_3 \cdot K(t)$ and $K_3(t)$ (mean absolute error of 3 servers, for 15 minute intervals). We observe that steady-state probability (π_j) that relies on strong assumptions account for the predictable variability of the time-varying probability ($p_j(t)$), i.e. the proportion of resources in the various states remains near-constant over the day. A similar phenomena is observed for the other customer classes and the two non-service nodes.

8.5 Future Work

In the concluding part of this section we consider several research directions for resource mining. First, we aim at providing further techniques for discovery and validation of models for operational service analysis. For example, the steady-state analysis that we presented did not rely directly on the structure of the queueing network. However, when one wishes to analyze advanced performance measures, (e.g. waiting times of servers to customers), the dependencies between queueing stations must be considered.

Another research direction is further developing protocol mining, i.e. the discovery of rules by which systems match customers to appropriate servers. These rules can be at the individual level (single customer/server) or at the aggregated level (customer classes/server skills). Moreover, these rules can be time and state dependent, e.g. they may vary over the day or depend on the number of customers in the different queues. At the individual level we wish to discover service patterns of resources. Consequently, the clustering and ranking of servers according to service patterns and other performance parameters (e.g. service times) can follow.

Resource mining introduces the queueing perspective into the world of servers, whereas queue mining considers queues of customers that wait for resources. We propose a third perspective that combines customers and servers under a single roof. In services, queues that form due to scarce resources are not the only cause for customer delay. For instance, synchronization delays often originate when customers are delayed until precedence constraints between several activities are fulfilled [5, 92]. We argue that by discovering the process from a combined (customer-service) perspective, we may gain insights into synchronizations between customers and resources.

For example, consider a scenario in which a resource becomes available and there is a customer waiting for service. However, the resource decides to work on back-office tasks, instead of serving that customer. This may reflect a problem in the scheduling protocol or that the specific customer was not important enough (i.e. the protocol was actually correct). Such behavioral aspects can be detected from event logs only via the hybrid perspective. Furthermore, understanding protocols that capture both customer routing and resource scheduling from service logs can turn out useful, possibly essential, when constructing data-based simulation (Section 9).

Lastly, we plan to contribute to the development of both statistical and queueing models for aggregated and individual analysis from the resource perspective. We aim to achieve this by providing statisticians and operations researchers with empirical insights into resource analysis that would originate from our mining experiments.

9 Simulation Mining

This section proposes a future research direction in *simulation mining*, which is the extraction of simulation models from event logs. The advantages and disadvantages of simulation with respect to analytical solutions to service models are well-known, c.f. [13, pp. 608]. Discovering simulation models from event data is the most challenging task in process mining; it was considered in several studies on operational process mining, c.f. [62, 75, 82]. Rozinat et al. [62] proposed a comprehensive approach that combines multiple process perspectives, e.g. control flow, time, resource, to deal with the complex discovery of simulation models from event data. However, as we mentioned earlier in the proposal, these works do not accommodate service characteristics, e.g. the queueing perspective is not considered. Our first potential contribution to simulation mining is the discovery of the queueing perspective (for both customers and resources). Then, we plan to integrate the queueing perspective into a single modeling framework, thus complementing the work of [62].

The section is structured as follows. We start with an introduction to the building blocks of simulation models and review the characteristics of these models across several dimensions. Then, we provide a birds-eye view of our plan for simulation mining from the queueing perspective. Lastly, business process simulation is discussed at a broader scope and we propose approaches that combine multiple process perspectives. This unified framework would provide the basis for business process simulation and for other types of operational process mining.

9.1 Simulation Models: Building Blocks and Characteristics

The building blocks of a simulation model comprise primitives and dynamics. The *primitives* are *system states* (referred to as states) and *timed events*. At each state, only a subset of all possible events are *active* and may cause a change in state. The *dynamics* of simulation models are defined by the distributions of remaining times-to-events and of *state transitions* that these events cause. The distribution of state transitions is typically a function of the current state and the event that influences the state.

To demonstrate simulation primitives and dynamics we consider a simple single-station queue with single-class customers and single server, i.e. $G/G/1$. The state can be represented by the number of customers in the service station; events can be either *service completions* (intrinsic) or *customer arrivals* (extrinsic). An arrival event increases the state by one, while a service completion event reduces the state by one (both with probability one). This influence of events on states is what we referred to as state transitions. The events are described by inter-arrival and service time distributions. Therefore, in order to mine a simulation model for a single-station queue we need to generate a trace of arrival and completion events.

Prior to the discussion on simulation mining from the queueing perspective, we list several dimensions that classify simulation models and their analysis. We demonstrate how each dimension is related to the theme of business process simulation, and specifically to simulating queueing models. Other dimensions for simulation model classification can be considered, c.f. [13, pp. 610], yet we choose to focus only on those listed below, due to their relevance to our agenda.

Time scale The time scale at which events are considered in simulations can be either *discrete* or *continuous*. Discrete simulation is appropriate for systems where events and thus, state transitions, occur at discrete points in time. Continuous simulation is used when a flow of events occur over time, without the ability to separate the time of consequent events. Business processes typically produce events at discrete points in time. However, continuous simulation can be relevant to services, since fluid and diffusion approximations of queueing models are continuous-time models.

Simulation horizon The running horizon of simulations is categorized into long-term (steady-state analysis) or short-term (transient analysis). Long-term analysis is appropriate for systems with a stable behavior (sometimes stable over certain time periods). This type of simulation models have a warm-up period (that is transient); then, once the steady-state is reached, a statistical analysis of the simulated sample paths may be used to estimate performance measures, e.g. waiting times and queue-lengths. Performance measures in long-term simulation are typically considered in terms of time averages.

Short-term simulation is suitable for analyzing systems that do not reach a steady-state due to time constraints (short-term simulation, no time to warm-up) or system dynamics (e.g. a time-varying system). This type of simulation can be useful for predicting system behavior in the 'near future'. Therefore, deciding on the simulation horizon depends mainly on the type of analysis that one wishes to perform. For example, when considering bottleneck identification in a process we may prefer long-term simulation, whereas for online operational support, e.g. time prediction, we shall adopt a transient approach.

Event generation The event schedule in simulation models is a *trace* of timestamped events. These traces can be distribution-driven, i.e. produced synthetically based on distributions of the time between events (e.g. Poisson arrivals correspond to exponentially distributed i.id. inter-arrival times). The other option is to schedule events from real data (e.g. service logs) as an input to the model; this technique is referred to as trace-driven simulation. The replay method for conformance checking (see Section 2) applies the latter approach based on the corresponding event log.

In our research on simulation mining, we plan to combine the two approaches. First, a trace-based approach would be used to test the conceptual validity of inter-event time distributions. For

example, the simulation will receive a real trace of arrival events from the service log and a trace of synthetic service-completion events that would come from a theoretical distribution (e.g. a log-normal distribution that is fit to the event data). Accurate performance measures would support the conceptual validity of the service time distribution. However, for operational support, the entire trace of events would be generated from distributions, thus avoiding the risk of over-fitting future events to historical event data.

Modeling orientation Simulation models can be either event-oriented or process-oriented. In an event-oriented setting, the system evolves over time by executing events from a single event trace, without awareness of processes that produce these events. On the other hand, process-oriented simulation considers process-related constraints, e.g. the precedence of activities in the process. Clearly, a process oriented approach is more suitable for business process simulation. However, for simulating the queueing perspective that underlies the business process, event-driven approaches can be simpler to implement, since they do not require the discovery of control-flow models.

Modeling formalism The primitives of the simulation model can be directly written as a computer program without the use of mathematical formalisms. However, there are good reasons to use such formalism to describe the building blocks of simulation models. For example, a formalism that is traditionally used to model discrete-event simulation of complex systems (e.g. queueing networks) is *generalized semi-Markov processes* (GSMP's) [28, 31]. The GSMP is mathematically tractable and provides with exact tools, such as variance reduction methods, improved sampling techniques and an exact analysis of simulation output. However, the GSMP is event-oriented and does not explicitly accommodate process-related constraints such as precedence relations, synchronized activities and concurrency.

Generalized stochastic Petri nets (GSPN, or SPN) are suitable for modeling processes from the control flow perspective; they explicitly and graphically account for the process-related constraints that we mentioned above [31, Ch. 1]. Moreover, they provide tools for a qualitative analysis and verification of the simulation model prior to its use for analysis [74]. In addition, many of the existing process mining techniques were developed on the basis of traditional Petri nets (which underlies the GSPN). The modeling power of these two formalisms, in terms of the underlying stochastic process, was proven to be equal in [31, Ch. 4].

Furthermore, as we show in the next part of this section, one can discover a queueing network from event data and transform it, either to a computer program or the one of the above formalisms, for simulation analysis. Moreover, a formalism that combines queueing networks and stochastic Petri nets, namely queueing Petri nets (QPN), was proposed by Bause in [9]. In [8], Bause demonstrated (by construction) that any queueing network and stochastic Petri net can be converted into a QPN.

This provides positive evidence for the expressive power of QPN. We consider the decision on a suitable formalism for simulation mining and operational process mining in general, as a future challenge of the proposed research.

9.2 Simulation Mining of Queueing Models

As we stated throughout the proposal, our first-choice modeling formalism for operational service analysis (from the queueing perspective) is queueing networks (with single-station queues as a special case). Once these models are discovered from service logs they can be solved analytically or simulated, when an analytic solution is not available.

However, simulating the discovered queueing network is not straightforward; a transformation step is required to convert a queueing network into a formalism (or a computer program) that would explicitly express the building blocks, i.e. primitives and dynamics. In other words, in order to simulate the queueing perspective from event data we must adopt one of the following approaches: (1) transform the discovered model into a lower-level formalism (e.g. GSPN) or (2) discover a low-level formalism directly from data. In our context, we prefer the former approach, since discovering processes from the queueing perspective is, in our view, an essential step towards operational analysis, regardless of the eventual technique of choice (analytic solution or simulation).

We consider the transformation step to be analogous to the *model specification* and *discovery* phases in Figure 1, since the transformation *specifies* the *data-driven* building blocks of the simulation model. The method for validating the simulation is not different than validation of the originating queueing network, c.f. Section 7.4. Below, we describe some of the issues that must be addressed when transforming queueing networks into simulation primitives and dynamics.

9.2.1 Primitives

When determining simulation primitives, i.e. states and events, we must consider the type of queueing model that we are facing. As an example, we return to the single-station queue with a single customer type served by homogeneous servers with FCFS routing policy. The state in such case can be represented by the number of customers in the system. However, in complex queueing stations, with skills-based routing mechanisms [27], multi-class customers seek service from multi-skilled servers. The policy in such stations is not FCFS and thus, enrichment of the state is required. For example, the state must distinguish between customer classes and septate delays from services. Correspondingly, the set of events in such complex stations is not limited to arrivals and departures of customers, e.g., the simulation model must distinguish between types of arrivals (multi-class customers). Moreover, resource-related events influence system state and thus, the corresponding resource perspective (e.g. mining resource paths) must be taken into account.

9.2.2 Dynamics

The specification of dynamics that correspond to a queueing network can be categorized into inter-node dynamics and intra-node dynamics. Inter-node dynamics are related to customer-flow between different service stations (assuming that servers do not switch stations). For example, inter-node dynamics can be represented by the transition matrix of the queueing network.

For intra-node dynamics both server events and customer events may trigger state transition. Intra-node dynamics require the specification of event schedules for arrivals, service-completions and abandons. Moreover, intra-node dynamics require protocols for customer routing and server scheduling for each service-station. Note that system dynamics can be either distribution-driven or trace-driven, as we discussed above.

Lastly, events can be viewed as a continuous flow, rather than occurring in discrete times (e.g. when a fluid network is considered). However, the simulation model must then be distribution-based and not trace-based (real events occur at discrete points in time).

Simulation mining of queueing models provides a framework for performance analysis and online operational support. However, describing the control flow perspective via queueing networks is not practical in processes with large activity spaces, since one must define the number of customer classes according to the size of the activity space. Therefore, we propose a unified framework for modeling business processes that would accommodate both control flow and queueing perspectives.

9.3 A Unified Framework for Business Process Analysis and Simulation

Business processes can be viewed from three perspectives: the *control flow perspective* (activity-centric), the *queueing perspective* (resource-centric) and the combined *control flow and queueing perspective* [1]. We claim that other perspectives, e.g. case, resource and organizational, can be embedded within the latter view. For example, the case perspective that corresponds to service paths can be viewed from the control flow perspective (which activities will comprise the path), the queueing perspective (which resources will be involved in the path) and the combined perspective (which resources and activities were related to the path).

Models that correspond to the control flow perspective include *stochastic PERT* [41], *stochastic Petri nets* [10], UML activity diagrams [21] and BPMN [54]. We hope that we convinced the reader that queueing networks are first-choice models for the queueing perspective (for both customers and resources). We claim that complex data-driven analysis, such as simulation mining, requires a comprehensive view that would unite these two perspectives under a common roof.

We conclude this proposal by naming two frameworks as candidates for the combined approach. The first framework is a combination of stochastic Petri nets (for the control flow perspective) and queueing networks (for the queueing perspective). The second framework, namely *stochastic*

processing networks, provides a single general formalism that includes many of the models that we mentioned throughout the proposal (e.g. stochastic Petri nets, stochastic PERT, queueing networks).

9.3.1 Combining Stochastic Petri Nets and Queueing Networks

Stochastic Petri nets (SPNs) have been proven to provide a strong basis for both control flow analysis (e.g. conformance checking) and operational support (e.g. predicting remaining times) [56]. The formalism is highly expressive and is able to model complex system behavior (e.g. synchronizations, time-outs) [85]. Moreover, SPN is a formal method that enables model verification to ensure its correctness; this feature is useful when considering simulation models [31]. A major drawback of the SPN formalism is its computational inefficiency for operational analysis, especially when it concerns the modeling of large systems [85]. To solve this inefficiency, fluid and product-form stochastic Petri nets were developed by [70] and [7], respectively.

However, these formalisms do not explicitly consider the queueing perspective and are not as well-developed theoretically as the corresponding product-form and fluid queueing networks. Analytically tractable (product-form) queueing networks (QNs) are efficient for operational analysis, even when complex systems are considered [13, 85]. Moreover, fluid and diffusion approximations to queueing networks result the so-called 'state-space collapse' [35], which makes these methods efficient and useful in practical use (e.g. our snapshot predictors from Section 7).

This research direction is related to previous works that considered the combination of stochastic Petri nets and queueing networks for modeling complex systems [6, 9, 11]. Moreover, a recent work by Menasce [48] presented an automated methodology for combining the two formalisms into a single model. We hope that combining these two formalisms into a single model would capture both the control flow perspective and queueing perspective. As we already discussed in the context of simulation formalisms, queueing petri nets (QPN) combine the benefits of SPN and QN [9]. Therefore, we may choose between a model that preserves notation from both formalisms or use QPN as a single framework.

9.3.2 Stochastic Processing Networks

Stochastic processing networks is a general family of models that were considered in the context of process management in [1] and formalized in [33]. These networks include both the control flow (activity) and queueing perspectives (resources) in a single model. A processing network takes inputs of materials (e.g. customers, jobs, packets, servers) and produces other materials as outputs. For example, in stochastic processing networks, customers may be consumed by resources or resources can be consumed by customers, which corresponds to our approach in Section 8. Since the definition of stochastic processing networks is lose, these models can be viewed as a generalization to the

stochastic PERT and to any of the Petri net formalisms that are suitable for modeling business processes (e.g. Workflow net, stochastic Petri net). Moreover, multi-class queueing networks, which is the most complex type of queueing models are a special case of a stochastic processing network [33].

On one hand, this generality of stochastic processing networks makes the formalism attractive for its discovery from event logs, since it provides a single model for all perspectives. On the other hand, the abstractness of stochastic processing networks requires further assumptions and thus, may drive us back to the previous formalisms (e.g. Petri nets, queueing networks) for practical analysis. We leave the selection of a unified framework, as an open question for further research.

References

[1] Paul S Adler, Avi Mandelbaum, Viên Nguyen, and Elizabeth Schwerer. From project to process management: an empirically-based framework for analyzing product development time. *Management Science*, 41(3):458–484, 1995.

[2] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Conformance Checking Using Cost-Based Fitness Analysis. In *EDOC*, pages 55–64. IEEE Computer Society, 2011. ISBN 978-1-4577-0362-1.

[3] Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Alignment Based Precision Checking. In Rosa and Soffer [59], pages 137–149. ISBN 978-3-642-36284-2.

[4] Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors. *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, volume 4714 of *Lecture Notes in Computer Science*, 2007. Springer. ISBN 978-3-540-75182-3.

[5] Rami Atar, Avi Mandelbaum, and Asaf Zviran. Control of Fork-Join Networks in heavy traffic. In *Allerton Conference*, pages 823–830. IEEE, 2012. ISBN 978-1-4673-4537-8.

[6] Gianfranco Balbo, Steven C. Bruell, and Subbarao Ghanta. Combining queueing networks and generalized stochastic Petri nets for the solution of complex models of system behavior. *Computers, IEEE Transactions on*, 37(10):1251–1268, 1988.

[7] Simonetta Balsamo, Peter G Harrison, and Andrea Marin. Methodological construction of product-form stochastic Petri nets for performance evaluation. *Journal of Systems and Software*, 85(7):1520–1539, 2012.

[8] Falko Bause. ”QN+PN=QPN” - Combining Queueing Networks and Petri Nets. Technical report, 1993.

[9] Falko Bause and Peter Kemper. QPN - Tool for Qualitative and Quantitative Analysis of Queueing Petri Nets. In Günter Haring and Gabriele Kotsis, editors, *Computer Performance Evaluation*, volume 794 of *Lecture Notes in Computer Science*, pages 321–334. Springer, 1994. ISBN 3-540-58021-2.

[10] Falko Bause and Pieter S Kritzinger. *Stochastic Petri Nets*. Springer, 2002.

[11] Matthias Becker and Helena Szczerbicka. PNiQ: Intergration of queuing networks in generalised stochastic Petri nets. *IEE Proceedings - Software*, 146(1):27–32, 1999.

[12] Piergiorgio Bertoli, Chiara Di Francescomarino, Mauro Dragoni, and Chiara Ghidini. Reasoning-Based Techniques for Dealing with Incomplete Business Process Execution Traces. In Matteo Baldoni, Cristina Baroglio, Guido Boella, and Roberto Micalizio, editors, *AI*IA*, volume 8249 of *Lecture Notes in Computer Science*, pages 469–480. Springer, 2013. ISBN 978-3-319-03523-9.

[13] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications; 2nd Edition*. Wiley, 2006. ISBN 978-0-471-56525-3.

[14] R. P. Jagadeesh Chandra Bose and Wil M. P. van der Aalst. Process diagnostics using trace alignment: Opportunities, issues, and challenges. *Inf. Syst.*, 37(2):117–141, 2012.

[15] R. P. Jagadeesh Chandra Bose, Wil M. P. van der Aalst, Indre Zliobaite, and Mykola Pechenizkiy. Handling Concept Drift in Process Mining. In Haralambos Mouratidis and Colette Rolland, editors, *CAiSE*, volume 6741 of *Lecture Notes in Computer Science*, pages 391–405. Springer, 2011. ISBN 978-3-642-21639-8.

[16] Lawrence Brown, Noah Gans, Avi Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. Statistical Analysis of a Telephone Call Center. *Journal of the American Statistical Association*, 100(469):36–50, 2005. doi: 10.1198/016214504000001808. URL http://www.tandfonline.com/doi/abs/10.1198/016214504000001808.

[17] Ziv Carmon and Daniel Kahneman. The experienced utility of queuing: real time affect and retrospective evaluations of simulated queues. Technical report, Working paper, Duke University, 1996.

[18] Mark S Daskin. *Service Science*. Wiley. com, 2011.

[19] Massimiliano de Leoni, Wil M. P. van der Aalst, and Boudewijn F van Dongen. Data-and Resource-Aware Conformance Checking of Business Processes. In *Business Information Systems*, pages 48–59. Springer, 2012.

[20] Ana Karla A. de Medeiros, A. J. M. M. Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, 14(2):245–304, 2007.

[21] Marlon Dumas and Arthur H. M. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In Martin Gogolla and Cris Kobryn, editors, *UML*, volume 2185 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2001. ISBN 3-540-42667-1.

[22] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 2013. ISBN 978-3-642-33142-8.

[23] Dirk Fahland and Wil M. P. van der Aalst. Model RepairAligning Process Models to Reality. *Information Systems*, 2013.

[24] Diogo R. Ferreira, Marielba Zacarias, Miguel Malheiros, and Pedro Ferreira. Approaching Process Mining with Sequence Clustering: Experiments and Findings. In Alonso et al. [4], pages 360–374. ISBN 978-3-540-75182-3.

[25] James A Fitzsimmons and Mona J Fitzsimmons. *Service Management: Operations, Strategy, Information technology*. McGraw-Hill/Irwin Boston, 2004.

[26] Noah Gans, Ger Koole, and Avi Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):79–141, 2003.

[27] Ofer Garnett and Avi Mandelbaum. An introduction to skills-based routing and its operational complexities. *Teaching notes*, 2000.

[28] Peter W. Glynn and Donald L. Iglehart. Simulation Methods for Queues: An Overview. *Queueing Syst.*, 3(3):221–255, 1988.

[29] Christian W. Günther and Wil M. P. van der Aalst. Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics. In Alonso et al. [4], pages 328–343. ISBN 978-3-540-75182-3.

[30] HW Gustafson. Force-loss analysis. *Employee turnover: Causes, consequences, and control*, pages 139–185, 1982.

[31] Peter J Haas. *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer, 2002.

[32] Randolph W Hall. *Queueing Methods: For Services and Manufacturing*. Prentice Hall, Englewood Cliffs NJ, 1991.

[33] J Michael Harrison. Stochastic networks and activity analysis. *Translations of the American Mathematical Society-Series 2*, 207:53–76, 2002.

[34] J. Michael Harrison and Austin J. Lemoine. A Note on Networks of Infinite-Server Queues. *Journal of Applied Probability*, 18(2):pp. 561–567, 1981. ISSN 00219002. URL http://www.jstor.org/stable/3213306.

[35] J Michael Harrison and Jan A Van Mieghem. Dynamic control of Brownian networks: state space collapse and equivalent workload formulations. *The Annals of Applied Probability*, pages 747–771, 1997.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[37] Mark B. Houston, Lance A. Bettencourt, and Sutha Wenger. The relationship between waiting in a service queue and evaluations of service quality: A field theory perspective. *Psychology and Marketing*, 15(8):735–753, 1998. ISSN 1520-6793. doi: 10.1002/(SICI)1520-6793(199812)15:8⟨735::AID-MAR2⟩3.0.CO;2-9. URL http://dx.doi.org/10.1002/(SICI)1520-6793(199812)15:8<735::AID-MAR2>3.0.CO;2-9.

[38] Rouba Ibrahim and Ward Whitt. Real-Time Delay Estimation Based on Delay History. *Manufacturing and Service Operations Management*, 11(3):397–415, 2009. doi: 10.1287/msom.1080.0223. URL http://msom.journal.informs.org/content/11/3/397.abstract.

[39] John D Kalbfleisch and Ross L Prentice. *The Statistical Analysis of Failure Time Data*, volume 360. John Wiley & Sons, 2011.

[40] David G. Kendall. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics*, 24(3): pp. 338–354, 1953. URL http://www.jstor.org/stable/2236285.

[41] V. G. Kulkarni and V. G. Adlakha. Markov and Markov-Regenerative PERT Networks. *Operations Research*, 34(5):pp. 769–781, 1986. ISSN 0030364X. URL http://www.jstor.org/stable/170733.

[42] Richard C. Larson. Perspectives on Queues: Social Justice and the Psychology of Queueing. *Operations Research*, 35(6):895–905, 1987. doi: 10.1287/opre.35.6.895. URL http://or.journal.informs.org/content/35/6/895.abstract.

[43] Richard C Larson. The queue inference engine: Deducing queue statistics from transactional data. *Management Science*, 36(5):586–601, 1990.

[44] Fabrizio Maria Maggi, Marco Montali, Michael Westergaard, and Wil M. P. van der Aalst. Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In Stefanie Rinderle-Ma, Farouk Toumani, and Karsten Wolf, editors, *BPM*, volume 6896 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2011. ISBN 978-3-642-23058-5.

[45] Avi Mandelbaum. Service engineering (science, management): A subjective view. Technical report, Technical report, Technion-Israel Institute of Technology, 2007.

[46] Avi Mandelbaum and Sergey Zeltyn. Estimating characteristics of queueing networks using transactional data. *Queueing systems*, 29(1):75–127, 1998.

[47] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University, Press Cambridge, 2008.

[48] D Menascé. A Methodology for Combinining GSPNs and QNs. In *Computer Measurement Group Conference*, 2011.

[49] IEEE TASK FORCE ON PROCESS MINING. Process Mining Manifesto. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, *Business Process Management Workshops (1)*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer, 2011. ISBN 978-3-642-28107-5.

[50] Joyce Nakatumba. *Resource-Aware Business Process Management: Analysis and Support*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, 12 2013.

[51] Joyce Nakatumba and Wil M. P. van der Aalst. Analyzing Resource Behavior Using Process Mining. In Stefanie Rinderle-Ma, Shazia Wasim Sadiq, and Frank Leymann, editors, *Business Process Management Workshops*, volume 43 of *Lecture Notes in Business Information Processing*, pages 69–80. Springer, 2009. ISBN 978-3-642-12185-2.

[52] Efrat Nakibly. Predicting waiting times in telephone service systems. Master's thesis, Technion–Israel Institute of Technology, 2002.

[53] Viên Nguyen. The trouble with diversity: Fork-join networks with heterogeneous customer population. *The Annals of Applied Probability*, pages 1–25, 1994.

[54] Business Process Model OMG. Notation (BPMN) 2.0. *Object Management Group: Needham, MA*, 2494:34, 2011.

[55] Elham Ramezani, Dirk Fahland, and Wil M. P. van der Aalst. Where Did I Misbehave? Diagnostic Information in Compliance Checking. In Alistair P. Barros, Avigdor Gal, and Ekkart Kindler, editors, *BPM*, volume 7481 of *Lecture Notes in Computer Science*, pages 262–278. Springer, 2012. ISBN 978-3-642-32884-8.

[56] Andreas Rogge-Solti and Mathias Weske. Prediction of Remaining Service Execution Time Using Stochastic Petri Nets with Arbitrary Firing Delays. In Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu, editors, *ICSOC*, volume 8274 of *Lecture Notes in Computer Science*, pages 389–403. Springer, 2013. ISBN 978-3-642-45004-4.

[57] Andreas Rogge-Solti, Ronny Mans, Wil M. P. van der Aalst, and Mathias Weske. Repairing Event Logs Using Timed Process Models. In Yan Tang Demey and Hervé Panetto, editors, *OTM Workshops*, volume 8186 of *Lecture Notes in Computer Science*, pages 705–708. Springer, 2013. ISBN 978-3-642-41032-1.

[58] Andreas Rogge-Solti, Wil M. P. van der Aalst, and Mathias Weske. Discovering stochastic Petri nets with arbitrary delay distributions from event logs. In *BPM Workshops*, 2013.

[59] Marcello La Rosa and Pnina Soffer, editors. *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, volume 132 of *Lecture Notes in Business Information Processing*, 2013. Springer. ISBN 978-3-642-36284-2.

[60] Anne Rozinat and Wil M. P. van der Aalst. Decision Mining in ProM. In Schahram Dustdar, José Luiz Fiadeiro, and Amit P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer, 2006. ISBN 3-540-38901-6.

[61] Anne Rozinat and Wil M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008.

[62] Anne Rozinat, RS Mans, Minseok Song, and Wil M. P. van der Aalst. Discovering simulation models. *Information Systems*, 34(3):305–327, 2009.

[63] Anne Rozinat, Moe Thandar Wynn, Wil M. P. van der Aalst, Arthur HM ter Hofstede, and Colin J Fidge. Workflow simulation for operational decision support. *Data & Knowledge Engineering*, 68(9):834–850, 2009.

[64] Robert G. Sargent. Verification and validation of simulation models. In S. Jain, Roy R. Creasey Jr., Jan Himmelspach, K. Preston White, and Michael C. Fu, editors, *Winter Simulation Conference*, pages 183–198. WSC, 2011.

[65] Joseph L Schafer and John W Graham. Missing data: our view of the state of the art. *Psychological methods*, 7(2):147, 2002.

[66] Helen Schonenberg, Barbara Weber, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Supporting Flexible Processes through Recommendations Based on History. In Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors, *BPM*, volume 5240 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2008. ISBN 978-3-540-85757-0.

[67] Lee Schruben and Radhika Kulkarni. Some consequences of estimating parameters for the M/M/1 queue . *Operations Research Letters*, 1(2):75 – 78, 1982. ISSN 0167-6377. doi: http://dx.doi.org/10.1016/0167-6377(82)90051-7. URL http://www.sciencedirect.com/science/article/pii/0167637782900517.

[68] Arik Senderovich. Multi-Level Workforce Planning in Call Centers. Master's thesis, 2012.

[69] Arik Senderovich, Matthias Weidlich, Avigdor Gal, and Avi Mandelbaum. Queue Mining–Predicting Delays in Service Processes. Technical report, 2013.

[70] Manuel Silva and Laura Recalde. On fluidification of Petri Nets: from discrete to hybrid and continuous models. *Annual Reviews in Control*, 28(2):253–266, 2004.

[71] Minseok Song, Christian W. Günther, and Wil M. P. van der Aalst. Trace Clustering in Process Mining. In Danilo Ardagna, Massimo Mecella, and Jian Yang, editors, *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 109–120. Springer, 2008. ISBN 978-3-642-00327-1.

[72] Suriadi Suriadi, Chun Ouyang, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Root Cause Analysis with Enriched Process Logs. In Rosa and Soffer [59], pages 174–186. ISBN 978-3-642-36284-2.

[73] Wil M. P. van der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets 1997*, pages 407–426. Springer, 1997.

[74] Wil M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.

[75] Wil M. P. van der Aalst. Business process simulation revisited. In *Enterprise and Organizational Modeling and Simulation*, pages 1–14. Springer, 2010.

[76] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011. ISBN 978-3-642-19344-6.

[77] Wil M. P. van der Aalst. Process Mining: Overview and Opportunities. *ACM Trans. Management Inf. Syst.*, 3(2):7, 2012.

[78] Wil M. P. van der Aalst. Challenges in Service Mining: Record, Check, Discover. In Florian Daniel, Peter Dolog, and Qing Li, editors, *ICWE*, volume 7977 of *Lecture Notes in Computer Science*, pages 1–4. Springer, 2013. ISBN 978-3-642-39199-6.

[79] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[80] Wil M. P. van der Aalst, H. T. de Beer, and Boudewijn F. van Dongen. Process Mining and Verification of Properties: An Approach Based on Temporal Logic. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *OTM Conferences (1)*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2005. ISBN 3-540-29736-7.

[81] Wil M. P. van der Aalst, Hajo A. Reijers, and Minseok Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative Work*, 14(6):549–593, 2005.

[82] Wil M. P. van der Aalst, J Nakatumba, A Rozinat, and N Russell. Business Process Simulation: How to get it right. *BPM Center Report BPM-08-07, BPMcenter. org*, 2008.

[83] Wil M. P. van der Aalst, MH Schonenberg, and Minseok Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011.

[84] Boudewijn F van Dongen, RA Crooy, and Wil M. P. van der Aalst. Cycle Time Prediction: When Will This Case Finally Be Finished? In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 319–336. Springer, 2008.

[85] Mary Vemon, John Zahorjan, and Edward D Lazowska. A comparison of performance Petri nets and queueing network models. Technical report, Technical Report 86-09-09, University of Washington, Dept. of Computer Science, 1986.

[86] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling, and Mathias Weske. Process compliance analysis based on behavioural profiles. *Information Systems*, 36(7):1009–1025, 2011.

[87] A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible Heuristics Miner (FHM). In *CIDM*, pages 310–317. IEEE, 2011. ISBN 978-1-4244-9925-0.

[88] Anton JMM Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

[89] Ward Whitt. *Stochastic-process limits: an introduction to stochastic-process limits and their application to queues*. Springer, 2002.

[90] Lars Wohlrab and Johannes Fürnkranz. A review and comparison of strategies for handling missing values in separate-and-conquer rule learning. *Journal of Intelligent Information Systems*, 36(1):73–98, 2011.

[91] G Yom-Tov and A Mandelbaum. The Erlang-R queue: time-varying QED queues with re-entrant customers in support of healthcare staffing. *preprint*, 2010.

[92] Itamar Zaied. The Offered Load in Fork Join Networks: Calculations and Applications to Service Engineering of Emergency Department. Master's thesis, 2012.