Discovering work prioritisation patterns from event logs

Suriadi, S.; Wynn, M.T.; Xu, J.; van der Aalst, W.M.P.; ter Hofstede, A.H.M.

Published in:
Decision Support Systems

DOI:
10.1016/j.dss.2017.02.002

Published: 01/08/2017

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Decision Support Systems 100 (2017) 77–92

Contents lists available at ScienceDirect

Decision Support Systems

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

Discovering work prioritisation patterns from event logs

Suriadi Suriadi[a,*], Moe T. Wynn[a], Jingxin Xu[a], Wil M.P. van der Aalst[b,a], Arthur H.M. ter Hofstede[a,b]

[a] Queensland University of Technology, Australia
[b] Eindhoven University of Technology, The Netherlands

ARTICLE INFO

Available online 8 February 2017

Keywords:
Resource behaviour mining
Queuing
Process mining

ABSTRACT

Business process improvement initiatives typically employ various process analysis techniques, including evidence-based analysis techniques such as process mining, to identify new ways to streamline current business processes. While plenty of process mining techniques have been proposed to extract insights about the way in which activities within processes are conducted, techniques to understand resource behaviour are limited. At the same time, an understanding of resources behaviour is critical to enable intelligent and effective resource management - an important factor which can significantly impact overall process performance. The presence of detailed records kept by today's organisations, including data about who, how, what, and when various activities were carried out by resources, open up the possibility for real behaviours of resources to be studied. This paper proposes an approach to analyse one aspect of resource behaviour: the manner in which a resource prioritises his/her work. The proposed approach has been formalised, implemented, and evaluated using a number of synthetic and real datasets.

1. Introduction

Business process management (BPM) enables organisations to improve the effectiveness and efficiency of their business operations by systematically documenting, managing, automating and optimising their business processes [1]. To achieve more with less, organisations need to focus on process efficiency, i.e., how their business operations could be improved. A plethora of literature and methodology exists on how one can improve process efficiency, e.g. Six Sigma [2]. However, as most business operations rely on human resources, e.g. employees, it is equally important to investigate whether these resources can be used in a more efficient manner; for example, how do employees spend their time between productive (e.g., waiting time) and non-productive (e.g., idle time) tasks? Are there any opportunities for increased resource utilisation?

Today's information systems record a wide variety of "events". Events may be generated by human behaviour (e.g., customers and employees), machines, and software. By leveraging state-of-the-art data analytics (including data mining, machine learning, and statistical techniques), valuable insights about resource behaviour can be extracted from this data to not only address the questions just presented, but also facilitate smarter resource management.

Within business processes, while resources are normally guided by business rules from the organisation and are constrained by the associated IT systems in terms of how they perform their work, resources typically have some freedom in prioritising their work, including the selection of activities (also known as *work items*) to perform and the order in which these activities are carried out. The way in which resources select the tasks to perform essentially forms the type of *queuing discipline* he/she applies. A *queuing discipline* refers to "the manner in which customers are selected for service when a queue is formed" [3]. The most common queuing discipline used in day-to-day life is the first-in-first-out style (FIFO) where work items that arrive first receive the highest priority, last-in-first-out (LIFO) where work items that arrive last receive the highest priority, and priority-based where priority is determined by some pre-determined rules.

The versatility of the concept of a queue has seen its application in many domains, from computer networks to business processes. Studies in the use of queues show that knowledge of queuing disciplines employed is important to design effective resource management strategy for ensuring appropriate staffing level [4,5] or performance stabilization [6,7]. Furthermore, studies show that queuing discipline may have a significant impact on the overall performance [8–13]. For example, the use of Shortest Process Time first discipline has

* Corresponding author.
E-mail addresses: s.suriadi@qut.edu.au (S. Suriadi), m.wynn@qut.edu.au (M.T. Wynn), j15.xu@qut.edu.au (J. Xu), w.m.p.v.d.aalst@tue.nl (W.M. vander Aalst), a.terhofstede@qut.edu.au (A.H. ter Hofstede).

http://dx.doi.org/10.1016/j.dss.2017.02.002
0167-9236/

been shown to reduce cycle time as compared to FIFO in certain settings [11]. Within business processes, one can draw a parallel in how queuing discipline employed by resources can substantially impact overall process performance. For example, a predominantly LIFO work prioritisation behaviour of resources may very likely lead to a LIFO case completion trend - a phenomenon that is not desirable from a customer satisfaction perspective. The interplay between the assignment of work items to resources and their queuing discipline also impacts overall process performance. For example, assigning a work item involving *calling customers* to a resource who always prioritises the execution of *e-mailing customers* will easily lead to the building up of longer (and rather unfair) waiting times for the former task. This highlights an undesirable situation where the assignment mechanisms of work items to resources, and the choice of queuing discipline of the resources in the process are *out of sync*.

A clear understanding of resource behaviour can assist organisations in identifying undesirable (and perhaps unexpected) working patterns which will guide them in investigating contextual factors (e.g. the way in which a list of tasks is presented to users on their screens) that may inadvertently encourage the expression of such behaviours by employees, leading to a clear direction for process improvement (e.g. changing the default ordering of work items). As reported in this article, this is precisely one of the insights we extracted.

The scenarios above clearly demonstrate the importance of understanding resource behaviour: it allows one to identify individual resource behaviour (which may be problematic) and to understand their compound effects on overall process performance. Most importantly, insights about resource behaviour will nicely complement existing process improvement strategy, enabling intelligent adaptation of the way in which processes are designed (to achieve the best process outcomes) to the way in which resources tackle their tasks in the processes.

In this article, we present a new data-driven approach to learning the prioritisation order used by a resource to carry out the work items (in relation to a particular business process). As shown in Fig. 1 (left-hand-side figure), a business process is typically guided by a process model. A process model captures those activities that need to be performed, the temporal order in which they are to be executed (e.g. sequentially or in parallel), and the resource(s) who can execute the various activities in the process. The execution of various instances of a process is often recorded in transactional records (also known as *event logs*).

Event logs typically contain information about the activities (or work items) that have been executed, the time they occurred, and the identifiers of employees who carried out the activities. By combining process analysis and data mining techniques, the emerging discipline of *process mining* provides a collection of novel techniques to exploit and extract process-related insights from raw event data [14]. Research in the domain of process mining has traditionally been focused on *process discovery* (i.e., automated discovery of the control flow of a process from data attributes recorded in an event log), *conformance checking* (i.e., detection of where and how deviances in processes occurred by comparing observation seen in a log with normative process models or business rules), and *performance analysis* (i.e., identifying bottlenecks and extracting process performance metrics). Relatively few research studies have been conducted that focus on the resource perspective [15–20], and to our knowledge, none of these works focus on discovering resources work prioritisation order.

Our approach makes use of detailed transactional records of executed processes (i.e. event logs) to determine the queuing discipline employed by the resources (Fig. 1 - right-hand-side). Such a data-driven approach has the advantage of objectively exposing the actual way in which resources work, which may, and often do, contradict anecdotal wisdom or recommended business practices.

It is not our goal to monitor and control the way in which resources work. This paper is about discovering the work prioritisation patterns of resources and their effects on the overall process which can be performed in a privacy-preserving manner (see Section 5).

Our approach has been implemented as a plug-in for the open-source process mining tool ProM[1]. We evaluate the correctness of our approach and implementation by testing the tool using synthetic logs. We demonstrate the usefulness of our approach in a case study with an Australian-based insurance organisation. In particular, our case study manages to extract useful insights about behaviours of resources that may be useful for the stakeholders to design a more targeted actions.

The rest of the paper is organised as follows. Section 2 presents the proposed approach for learning work prioritisation patterns. Section 3 discusses a prototype implementation of the approach within the open-source process mining tool, ProM. Sections 4 and 5 present findings from the evaluation of the proposed approach using synthetic and real-life datasets. Section 6 summarises related work in the areas of organisational mining and queuing theory. Section 7 concludes the paper.

2. Learning work prioritisation patterns

A descriptive overview of our approach is provided in Section 2.1, and formalised in Section 2.2.

2.1. Approach

The proposed approach is illustrated in Fig. 2. The log shown at the top of Fig. 2 depicts a snippet of the events performed by two resources: Carol and Eliza. Each row in the log represents an event. For example, the first row of the log records an event capturing the assignment of a work item to the resource named Carol. The work item in this event is defined by the activity 'create PO' that needs to be executed for a particular process instance of which the identifier is '330'. As a short form, we give an identifier for the work item captured by every event in the log (e.g. C1 for the work item represented by the first event in the log).

By observing such an event log, we can build the worklist of a resource, ordered according to the times the work items are assigned to the resource (i.e., the in-list) and the corresponding (partial) list of work items completed by the resource, ordered according to the time the work items are completed (i.e., the out-list). For example, the bottom-left part of Fig. 2 depicts an in-list for resource Carol at a particular timestamp t'_3 (which happened *just immediately before* t_3) whereby three work items (C1, C2, and C3) have been assigned to her.

From this in-list, we build the *expected ordering of work items output* at time t_3 by assuming a certain queuing discipline. For example, if we hypothesise that Carol works on a FIFO basis, then we should expect the order in which the work items are completed to be the same as the order in which the work items were assigned.

The bottom-right side of Fig. 2 shows the out-list of Carol at time t_3, just after the completion of work item C3. Whenever we see a work item being completed, we first determine the expected work item that should be seen at the out-list based on the assumed queuing discipline and extract the in-list position of that work item. Next, we calculate the *distance* between the in-list *position* of the *expected* work item and the *in-list position* of the work item that *actually appears* in the out-list. For example, in Fig. 1, if Carol adopts the FIFO queuing discipline, the expected work item to be seen at time t_3 is C1 (which assumes the first position in Carol's in-list). However, if Carol adopts the LIFO discipline, the expected work item to appear in the out-list at time t_3 is C3 (which assumes the third position in

[1] www.promtools.org - Resource Queue Behaviour package.

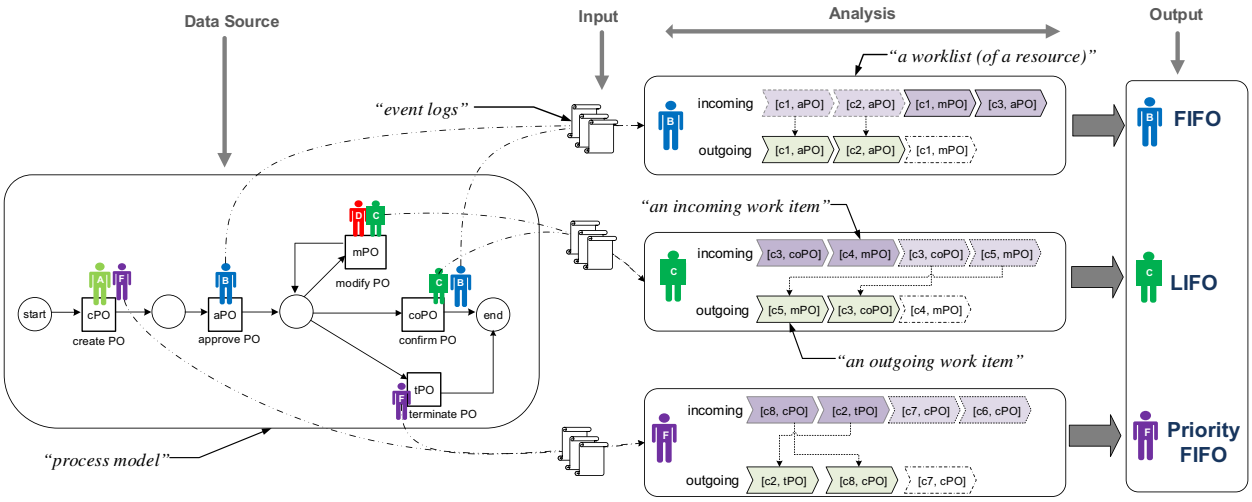

Fig. 1. The left-hand side figure shows an example of a 'Purchase Order' (PO) process which is made up of five activities: create PO, approve PO, modify PO, confirm PO, and terminate PO. The execution of a process leaves traces that are recorded in event logs. Our approach uses information in event logs to determine the prioritisation order employed by resources in tackling their work items. A work item is represented by the activity name and the case to which the activity belongs (e.g. [c1, aPO] represents an 'approve PO' task that needs to be performed for a case identified as 'c1'. Resource *B* employs a FIFO discipline. Resource *C* employs a LIFO discipline. Resource *F* employs a priority FIFO queue because work items with the highest priority are executed first. In this case, activity 'terminate PO' (tPO) has a higher priority than other activities, thus are executed first, while work items that share the same priority (all three work items containing activity 'create PO' - cPO) are executed in FIFO manner.

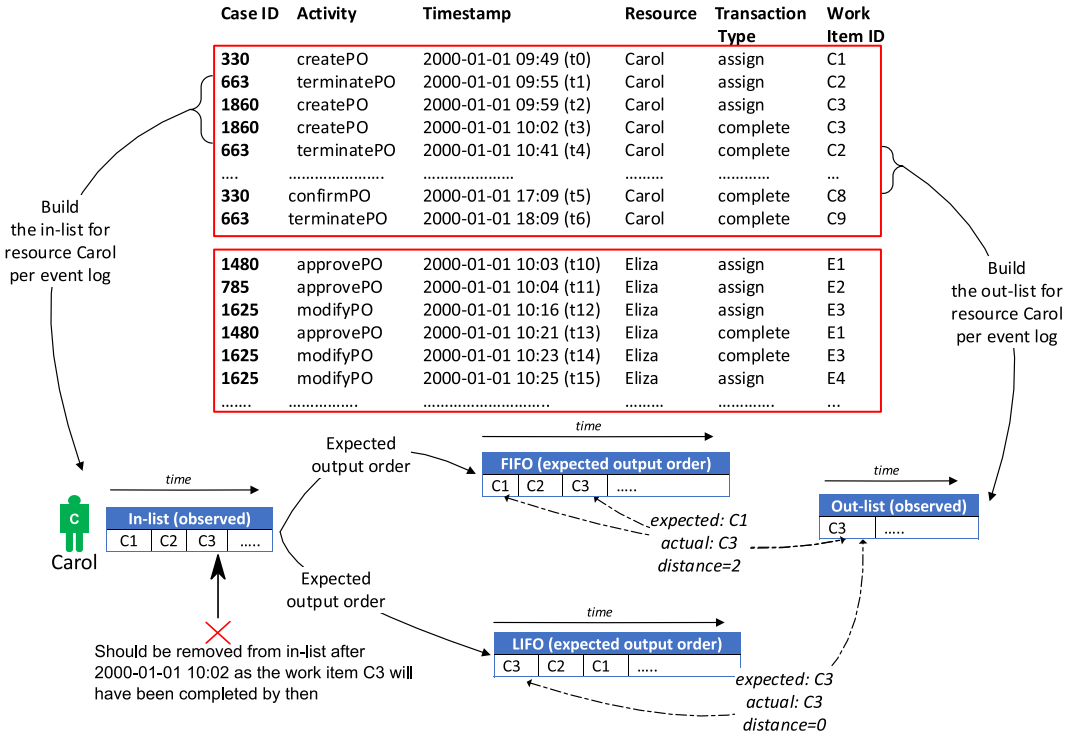

Fig. 2. Overview of the approach.

Carol's in-list). The actual work item that appears on Carol's out-list is C3. Therefore, the actual work-item is off by *two* positions for a FIFO queuing discipline, while for a LIFO queue, the distance is *zero*.[2]

It is difficult to learn queuing discipline by just looking at one distance value. What we need to obtain instead is the *trend* of the distance values over time. By calculating the distance value every time a work item is added to a resource's out-list and by aggregating these values, the trend in terms of the fluctuation, or its lack thereof, of the distance value can be observed. By observing this trend, we can then predict the queuing discipline employed by resources.

In the remainder of this paper, we use the term *resource perspective* to refer to the learning of queuing discipline from the perspective

[2] Note that the in-list, and the corresponding expected output order list, changes as work items are added and removed from these lists. As a work item is added to the in-list, the expected output order is likely to change (depending on the queuing discipline being analysed). Similarly, when a work item appears on the out-list, that work item should be removed from the in-list.

of resources, similar to the examples given earlier in this section: a queue is made up of work items related to *one particular resource*.

We can also learn the prioritisation order from the perspectives of activity types (referred to as *activity perspective*). Here, a queue is made up of a collection of work items of *one particular activity*. Using the same log shown in Fig. 2, the in-list for activity `createPO` include C1 (at time t_0 only) and C3 (at time t_2) (C2 relates to a *different* activity, thus cannot be a member in this case). Similarly, for activity `modifyPO`, the in-list consists of work items E3 (at time t_{12}), while the out-list will consist of the same work item E3 but related to the event at time t_{14}.

Finally, we can also learn work prioritisation order from the perspective of a case (referred to as *case perspective*). In this case, an in-list queue is made up of the earliest activity per case, and the out-list queue is made up of the latest activity per case. For example, using the log shown in Fig. 2, an in-list built from a case perspective will consist of work items C1 (for case 330 at time t_0) and C3 (for case 663 at time t_2) because `createPO` is the activity that signifies the start of a case as per process model in Fig. 1. The out-list corresponding to this in-list will consist of work items C8 and C9 (for cases 330 and 664 at time t_5 and t_6 respectively) because they represented possible end activities of a case. Note that for analysis from the *case perspective*, the pair of in-list and out-list members are of different work items though they relate to the same case.

2.2. Formalisations

An event log consists of a set of events. Each event has a timestamp. The timestamp of an event is one of a range of attributes of an event. These attributes can be mandatory or optional.

Definition 1 (*Event, attribute*)**.** Let $\mathcal{E}$ be the *event universe*, i.e., the set of all possible event identifiers. An event may be characterised by various *attributes*, e.g., an event has a timestamp, corresponds to an activity, and belongs to a particular case. Let AN be a set of all possible attribute names. For any event $e \in \mathcal{E}$ and an attribute name $a \in AN$: $\#_a(e)$ is the value of attribute named a for event e. If an event e does not have an attribute a, then we write $\#_a(e) = \perp$ (null value).

Let $\mathcal{D}_{case}$ be the set of case identifiers (case ID), $\mathcal{D}_{act}$ be the set of activities, $\mathcal{D}_{time}$ be the set of timestamps, $\mathcal{D}_{res}$ be the set of resources, $\mathcal{D}_{type} = \{schedule, assign, start, resume, suspend, manual-skip, auto-skip, complete\}$ be the set of event transaction types, and $\mathcal{D}_{data}$ be the set of data values (these may have a complex structure).

For each event $e \in \mathcal{E}$, we define a number of standard attributes: $\#_{case}(e) \in \mathcal{D}_{case}$ (the case ID of e), $\#_{act}(e) \in \mathcal{D}_{act}$ (the activity of e), $\#_{time}(e) \in \mathcal{D}_{time}$ (the timestamp of e), $\#_{res}(e) \in \mathcal{D}_{res}$ (the resource who triggered the occurrence of e), and $\#_{type}(e) \in \mathcal{D}_{type}$ (the transaction type of e).

Definition 2 (*Event log*)**.** An event log $\mathcal{L} \subseteq \mathcal{E}$ is a set of events.

The timestamps associated with events in $\mathcal{L}$ naturally provide a partial order of events (it is partial because more than one event can occur at the same time). To establish a total order of events, we define an event order identifier.

Definition 3 (*Event order identifier*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log. $id_{\mathcal{L}} : \mathcal{L} \rightarrow \{1, \ldots, |\mathcal{L}|\}$ is a bijective function that maps each event $e \in \mathcal{L}$ to a unique natural number whereby for all $e_1, e_2 \in \mathcal{L}$: if $\#_{time}(e_1) < \#_{time}(e_2)$, then $id_{\mathcal{L}}(e_1) < id_{\mathcal{L}}(e_2)$, i.e. function $id_{\mathcal{L}}$ provides a total order of events.

Definition 4 (*Case*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log. A finite sequence of events over $\mathcal{L}$ of length $n \in \mathbb{Z}_{>0}$ is a mapping $\alpha \in \{1, 2, \ldots, n\} \rightarrow \mathcal{L}$. We represent such a sequence as a string $\alpha = \langle e_1, e_2, \ldots, e_n \rangle$ where $\alpha(i) = e_i$ for $1 \leq i \leq n$.

Let $cid \in \mathcal{D}_{case}$ be a case identifier. A *case* with cid as its case identifier (denoted as α_{cid}) is a finite sequence of events over $\mathcal{L}$ of length $n \in \mathbb{Z}_{>0}$ such that for any $i, j \in \{1, 2, \ldots, n\}$ (where $i < j$), $id_{\mathcal{L}}(\alpha(i)) < id_{\mathcal{L}}(\alpha(j))$ and $\#_{case}(\alpha(i)) = \#_{case}(\alpha(j)) = cid$.

As mentioned in Section 1, an activity that is executed within a case is referred to as a *work item*. A process may allow the same activity to be repeated within a case. For example, in the process model shown in Fig. 1, the activity modifyPO is allowed to be repeated. In Fig. 2, we can see that this same activity was assigned twice to Eliza in the case number 1625. Therefore, it is possible that two or more work items within the same case may refer to the same activity. To uniquely identify multiple instantiations of the same activity within the same case, we introduce the notion of *work item identifier*. Two work items of the same activity executed within the same case have *different* work item identifiers. For example, in Fig. 2, the two work items related to the activity modifyPO for case 1625 have two different work item identifiers (E3 and E4).

Definition 5 (*Work item*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $\mathcal{D}_{wid}$ be a set of possible work item identifiers, and $\tau : \mathcal{L} \rightarrow \mathcal{D}_{wid}$ be a function that assigns a work item identifier to an event such that for any two events $e_1, e_2 \in \mathcal{L}$ where $id_{\mathcal{L}}(e_1) \neq id_{\mathcal{L}}(e_2)$, e_1 and e_2 refer to the *same work item if and only if*: $\#_{case}(e_1) = \#_{case}(e_2)$ and $\#_{act}(e_1) = \#_{act}(e_2)$ and $\tau(e_1) = \tau(e_2)$. A *work item* can therefore be uniquely identified as a tuple of $\mathcal{D}_{case} \times \mathcal{D}_{act} \times \mathcal{D}_{wid}$.

Definition 5 implies that two or more events may refer to the same work item. This is possible because a work item goes through a number of states. The states and the corresponding transitions that a work item traverses are described using a Deterministic Finite Automata [21] diagram as shown in Fig. 3. The edges of the figure capture the possible transaction types (i.e. $\mathcal{D}_{type}$), while the nodes capture the possible states of work items. The transition of a work item from one state to another is captured by an *event*, and the exact type of the transition is codified by the value of the transaction type of the event.

We use the notation $tp_x \Rightarrow tp_{x'}$ to say that $tp_{x'} \in \mathcal{D}_{type}$ is a transaction type that can be reached from an earlier transaction type $tp_x \in \mathcal{D}_{type}$ as per Fig. 3.

As explained in Section 2.1, to learn resource work prioritisation from various perspectives, corresponding in-lists and out-lists need

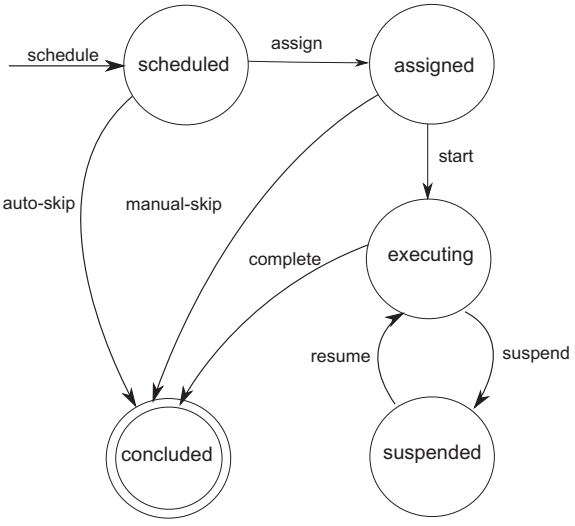

Fig. 3. Possible transaction types and states of work items (based on XES standard definition [22]).

to be built. For the resource perspective and activity perspective, the in-lists and out-lists are built based on the transaction types of the events in the list. For the case perspective, the corresponding in-lists and out-lists need to be built based on the first and the last event of a case. For these purposes, we define event-typed log and case log.

Definition 6 (*Event-typed log*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log. Given $tp \in \mathcal{D}_{Type}$, $\mathcal{L}_{tp} = \{e \in \mathcal{L} \mid \#_{type}(e) = tp\}$ is an *event-typed log* whose events in $\mathcal{L}$ have the same transaction type.

Definition 7 (*Case start log, case end log*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $\mathcal{D}_{startAct} \subseteq \mathcal{D}_{act}$ be a set of all possible activity names signifying the start of a case, and $\mathcal{D}_{finalAct} \subseteq \mathcal{D}_{act}$ be a set of all possible activity names signifying the end of a case.

$\mathcal{L}_{first} = \{e \in \mathcal{L} \mid \nexists_{e' \in \mathcal{L}} : (id_{\mathcal{L}}(e') < id_{\mathcal{L}}(e) \wedge \#_{case}(e) = \#_{case}(e') \wedge \#_{act}(e') \in \mathcal{D}_{startAct})\}$ is the *case start log* of $\mathcal{L}$ whereby each event in $\mathcal{L}_{first}$ is the earliest event representing the start of the case to which the event e belongs. Similarly, $\mathcal{L}_{last} = \{e \in \mathcal{L} \mid \nexists_{e'' \in \mathcal{L}} : (id_{\mathcal{L}}(e'') > id_{\mathcal{L}}(e) \wedge \#_{case}(e) = \#_{case}(e'') \wedge \#_{act}(e'') \in \mathcal{D}_{finalAct})\}$ is the *case end log* of $\mathcal{L}$ whereby each event in $\mathcal{L}_{last}$ is the latest event representing the end of the case to which the event e belongs.

To capture the pairing of two events representing the entry of a particular work item into a queue and its corresponding exit, we define the concept of *segment*.

Definition 8 (*Work item segment*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $tp_{in}, tp_{out} \in \mathcal{D}_{type}$ be event transaction types, and $\mathcal{L}_{tp_{in}}, \mathcal{L}_{tp_{out}} \subseteq \mathcal{L}$ be event-typed logs. $\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}} = \{(e_i, e_o) \in \mathcal{L}_{tp_{in}} \times \mathcal{L}_{tp_{out}} \mid id_{\mathcal{L}}(e_i) < id_{\mathcal{L}}(e_o) \wedge (\#_{case}(e_i), \#_{act}(e_i), \tau(e_i)) = (\#_{case}(e_o), \#_{act}(e_o), \tau(e_o)) \wedge tp_{in} \Rightarrow tp_{out}\}$ is a set of *work item segments* which are defined by pairs of two events (e_i, e_o) with e_i marking the beginning of a particular work item segment and e_o marking the end of the corresponding segment.

An event log may not contain information for all transaction types of a work item. Definition 8 only requires that an event log records two particular transaction types for each work item (e.g. tp_{in}= start and tp_{out} = complete). A limitation of this requirement is that our approach may not work when the event log used only records *one* particular transaction type per work item.[3]

Definition 9 (*Cluster of work item segments*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $tp_{in}, tp_{out} \in \mathcal{D}_{type}$ be event transaction types, $\mathcal{L}_{tp_{in}}, \mathcal{L}_{tp_{out}} \subseteq \mathcal{L}$ be event-typed logs, $\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}$ be a set of work item segments, $a \in AN$ be an attribute name, $\mathcal{D}_a$ be the possible values for attribute a, and $v_a \in \mathcal{D}_a$ be a particular value of the attribute a.

We define $\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}{\restriction}^{a}_{v_a} = \{(e_i, e_o) \in \mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}} \mid \#_a(e_i) = \#_a(e_o) = v_a\}$ as a *cluster of work item segments* where all segments within the cluster share the same attribute value for the given attribute a.

For example, let $res \in AN$ be an attribute referring to resource identifier, $r_1 \in \mathcal{D}_{res}$ be a particular resource identifier value, and assign, start $\in \mathcal{D}_{type}$ be two specific transaction types. $\mathcal{S}_{\mathcal{L}_{assign},\mathcal{L}_{start}}{\restriction}^{res}_{r_1} = \{(e_i, e_o) \in \mathcal{S}_{\mathcal{L}_{assign},\mathcal{L}_{start}} \mid \#_{res}(e_i) = \#_{res}(e_o) = r_1\}$ refers to a cluster of work item segments whereby all segments within the cluster share the same resource identifier value and the transaction types that signify the entry and exit of a work item to/from a segment are assign and start respectively.

Using Definition 9, we can see that the concept of *resource perspective* (mentioned towards the end of Section 2.1) is in fact captured by $\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}{\restriction}^{res}_{r_i}$ for any $r_i \in \mathcal{D}_{res}$. Similarly, the concept of *activity perspective* applies when segments are defined as: $\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}{\restriction}^{act}_{a_i}$ for any $a_i \in \mathcal{D}_{act}$ where $act \in AN$ is an attribute representing activity name, and $a_i \in \mathcal{D}_{act}$ is a possible activity name.

Definition 10 (*Case segment*)**.** Let $\mathcal{L}$ be an event log, $\mathcal{L}_{first}$ be the case start log for $\mathcal{L}$, and $\mathcal{L}_{last}$ be the case end log of $\mathcal{L}$. $\mathcal{S}_{\mathcal{L}_{first},\mathcal{L}_{last}} = \{(e_{ci}, e_{co}) \in \mathcal{L}_{first} \times \mathcal{L}_{last} \mid \#_{case}(e_{ci}) = \#_{case}(e_{co})\}$ is a set of *case segments* which are defined by pairs of (e_{ci}, e_{co}) with e_{ci} marking the beginning of a particular case segment and e_{co} an event referring to the same case marking the end of the corresponding segment.

The concept of *case perspective*, mentioned towards the end of Section 2.1, thus applies when segments are defined as $\mathcal{S}_{\mathcal{L}_{first},\mathcal{L}_{last}}$ as per Definition 10 above.

Definition 11 (*Collection of segments*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $tp_{in}, tp_{out} \in \mathcal{D}_{type}$ be event transaction types, $\mathcal{L}_{tp_{in}}, \mathcal{L}_{tp_{out}} \subseteq \mathcal{L}$ be event-typed logs, $\mathcal{L}_{first}$ be a case start log, $\mathcal{L}_{last}$ be a case end log, $\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}$ be a set of work item segments, $\mathcal{S}_{\mathcal{L}_{first},\mathcal{L}_{last}}$ be a set of case segments, $a \in AN$ be a particular attribute name, $\mathcal{D}_a$ be the set of all possible values for the attribute a, and $v_a \in \mathcal{D}_a$ be a value of the attribute a.

$\mathbb{S}_{\mathcal{L}} = \{\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}\} \cup \{\mathcal{S}_{\mathcal{L}_{first},\mathcal{L}_{last}}\} \cup \mathbb{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}{\restriction}_{AN}$ is the set of all possible *sets of segments* that may exist in $\mathcal{L}$ where

- $\{\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}\}$ is a set of the set of all work item segments,
- $\{\mathcal{S}_{\mathcal{L}_{first},\mathcal{L}_{last}}\}$ is a set of the set of all case segments, and
- $\mathbb{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}{\restriction}_{AN} = \bigcup_{a \in AN, v_a \in \mathcal{D}_a} \{\mathcal{S}_{\mathcal{L}_{tp_{in}},\mathcal{L}_{tp_{out}}}{\restriction}^{a}_{v_a}\}$ is a set of the sets of all possible clusters of work item segments.

Having defined the concept of *segment*, we can now define the concept of in-list and out-list.

Definition 12 (*In-list, out-list*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log and $\mathbb{S}_{\mathcal{L}}$ be the set of all possible sets of segments that may exist in $\mathcal{L}$. For a given $\mathcal{S}' \in \mathbb{S}_{\mathcal{L}}$, $\mathcal{L}^{\mathcal{S}'}_{in} = \{e_i \in \mathcal{L} \mid \exists_{e_o \in \mathcal{L}} : (e_i, e_o) \in \mathcal{S}'\}$ is an *in-list* whose members represent the starting events of segments seen in $\mathcal{S}'$ (henceforth, referred to as the *in-list of* $\mathcal{S}'$).

Conversely, $\mathcal{L}^{\mathcal{S}'}_{out} = \{e_o \in \mathcal{L} \mid \exists_{e_i \in \mathcal{L}} : (e_i, e_o) \in \mathcal{S}'\}$ is an *out-list* whose members represent the ending events of segments seen in $\mathcal{S}'$ (henceforth, referred to as the *out-list of* $\mathcal{S}'$).

Definition 13 (*Timed in-list, timed out-list*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $\mathcal{S}' \in \mathbb{S}$ be a set of a particular type of segment, $\mathcal{L}^{\mathcal{S}'}_{in}$ be the in-list of $\mathcal{S}'$, and $\mathcal{L}^{\mathcal{S}'}_{out}$ be the out-list of $\mathcal{S}'$.

Given a timestamp $t \in \mathcal{D}_{time}$,

- $\mathcal{L}^{t,\mathcal{S}'}_{out} = \{e_o \in \mathcal{L}^{\mathcal{S}'}_{out} \mid \#_{time}(e_o) \leq t\}$ is the *timed out-list* of $\mathcal{S}'$ at time t whereby all activity instances or cases captured by the events within the list have completed their segments at or before time t.
- $\mathcal{L}^{t,\mathcal{S}'}_{in} = \{e_i \in \mathcal{L}^{\mathcal{S}'}_{in} \mid \exists_{(e_i,e'_o) \in \mathcal{S}'} : \#_{time}(e_i) \leq t \wedge \#_{time}(e'_o) > t\}$ is the *timed in-list* of $\mathcal{S}'$ at time t whereby all activity instances or cases captured by the events within the list have not yet completed their segments at, or before, time t.

Definition 14 (*Ranking function*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log. $\mathcal{R}$ is a set of bijective functions that rank every event in an event log $\mathcal{L}$ based on some criteria, i.e. for any $\rho^{\mathcal{L}} \in \mathcal{R}$, $\rho : \mathcal{L} \rightarrow \{1, \ldots, |\mathcal{L}|\}$.

For example, a FIFO ranking function $\rho^{\mathcal{L}}_{FIFO} \in \mathcal{R}$ will rank all events in an event log $\mathcal{L}$ such that for any two events $e_j, e_k \in \mathcal{L}$, $\rho^{\mathcal{L}}_{FIFO}(e_j) < \rho^{\mathcal{L}}_{FIFO}(e_k)$ if and only if $id_{\mathcal{L}}(e_j) < id_{\mathcal{L}}(e_k)$. A LIFO ranking function $\rho^{\mathcal{L}}_{LIFO} \in \mathcal{R}$ is the reverse: $\rho^{\mathcal{L}}_{LIFO}(e_j) < \rho^{\mathcal{L}}_{LIFO}(e_k)$ if and only if $id_{\mathcal{L}}(e_j) > id_{\mathcal{L}}(e_k)$.

More concretely, given an event log $\mathcal{L}' \subseteq \mathcal{L}$ that consists of three events $\{e_1, e_2, e_3\}$ where $id_{\mathcal{L}}(e_1) < id_{\mathcal{L}}(e_2) < id_{\mathcal{L}}(e_3)$, $\rho^{\mathcal{L}'}_{FIFO}(e_1) =$

[3] Nevertheless, through other types of process mining analysis, e.g., [23], it is possible to derive a new transaction type for a work item form a row event log that only contains one transaction type.

$1, \rho^{\mathcal{L}'}_{FIFO}(e_2) = 2$, and $\rho^{\mathcal{L}'}_{FIFO}(e_3) = 3$. The converse is true for a LIFO: $\rho^{\mathcal{L}'}_{LIFO}(e_1) = 3, \rho^{\mathcal{L}'}_{LIFO}(e_2) = 2$, and $\rho^{\mathcal{L}'}_{LIFO}(e_3) = 1$.

Definition 15 (*Priority attribute, priority value, priority class*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, *priority* $\in AN$ be a *priority attribute name*, $\mathcal{D}_{priority}$ be the set of possible priority values, $PriorityClass = \{1, 2, \ldots, |\mathcal{D}_{priority}|\}$ be a set of possible priority classes with '1' referring to the class with the highest priority, and $\psi : \mathcal{D}_{priority} \rightarrow PriorityClass$ be a function that maps each value in $\mathcal{D}_{priority}$ into a particular priority class. For a given $e \in \mathcal{L}$, $\#_{priority}(e) \in \mathcal{D}_{priority}$ is the *priority value* of event e, and $\psi(\#_{priority}(e))$ is the corresponding *priority class*.

Given an event log $\mathcal{L} \subseteq \mathcal{E}$, a priority ranking function $\rho^{\mathcal{L}}_{priority} \in \mathcal{R}$ will rank all events in $\mathcal{L}$ such that for any two events $e_j, e_k \in \mathcal{L}$, $\rho^{\mathcal{L}}_{priority}(e_j) < \rho^{\mathcal{L}}_{priority}(e_k)$ if and only if $\psi(\#_{priority}(e_j)) > \psi(\#_{priority}(e_k))$ or $\psi(\#_{priority}(e_j)) = \psi(\#_{priority}(e_k))$ and $id_{\mathcal{L}}(e_j) < id_{\mathcal{L}}(e_k)$.

For example, assume an event log $\mathcal{L}' \subseteq \mathcal{L}$ that consists of three events $\{e_1, e_2, e_3\}$ where $id_{\mathcal{L}}(e_1) < id_{\mathcal{L}}(e_2) < id_{\mathcal{L}}(e_3)$, $\psi(\#_{priority}(e_1)) = \psi(\#_{priority}(e_3)) = 2$, and $\psi(\#_{priority}(e_2)) = 1$. The priority ranking function $\rho^{\mathcal{L}'}_{priority}$ will yield the following: $\rho^{\mathcal{L}'}_{priority}(e_1) = 2, \rho^{\mathcal{L}'}_{priority}(e_2) = 1$, and $\rho^{\mathcal{L}'}_{priority}(e_3) = 3$. This is because e_2 has the highest priority class amongst all events. Furthermore, while e_1 and e_2 have the same priority class, the event order identifier for e_1 is lower than e_3, thus giving e_1 a higher ranking than e_3.

Definition 16 (*Distance, queue score*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log, $\mathcal{S}' \in \mathbb{S}_{\mathcal{L}}$ be a set of a particular type of segments that may exist in $\mathcal{L}$, and $\mathcal{L}^{\mathcal{S}'}_{out}$ be the out-list of $\mathcal{S}'$. Given an output event $e_o \in \mathcal{L}^{\mathcal{S}'}_{out}$, we can determine the following: t_o (the timestamp of e_o which is $\#_{time}(e_o)$); $\mathcal{L}^{t_o,\mathcal{S}'}_{in}$ (the timed in-list at time t_o); $e_i \in \mathcal{L}^{t_o,\mathcal{S}'}_{in}$ (the corresponding input event of e_o such that $(e_i, e_o) \in \mathcal{S}'$); $\rho^{\mathcal{L}^{t_o,\mathcal{S}'}_{in}} \in \mathcal{R}$ (a ranking function over $\mathcal{L}^{t_o,\mathcal{S}'}_{in}$); and $e_e \in \mathcal{L}^{t_o,\mathcal{S}'}_{in}$ such that $\forall_{e_k \in \mathcal{L}^{t_o,\mathcal{S}'}_{in}} : \rho^{\mathcal{L}^{t_o,\mathcal{S}'}_{in}}(e_e) < \rho^{\mathcal{L}^{t_o,\mathcal{S}'}_{in}}(e_k)$ (i.e. e_e is the event representing the work item that is expected to exit its segment at time t_o).

We define *distance*: $\mathcal{L}_{out} \times \mathbb{P}(\mathcal{L}) \rightarrow \mathbb{Z}_{\geq 0}$ as a function that returns a non-negative integer representing the distance between e_i, $e_e \in \mathcal{L}^{t_o,\mathcal{S}'}_{in}$. The formula of the *distance* function is as follows: $distance(e_o, \mathcal{L}) = |id_{\mathcal{L}}(e_e) - id_{\mathcal{L}}(e_i)|$.

We also define *max_distance*: $\mathcal{L}_{out} \times \mathbb{P}(\mathcal{L}) \rightarrow \mathbb{Z}_{\geq 0}$ as a function that returns the maximum distance between any two events in $\mathcal{L}^{t_o,\mathcal{S}'}_{in}$. The formula for *max_distance* function is: $max\_distance(e_o, \mathcal{L}) = id_{\mathcal{L}}(e_{max}) - id_{\mathcal{L}}(e_{min})$ where $e_{min}, e_{max} \in \mathcal{L}^{t_o,\mathcal{S}'}_{in}$ such that $\forall_{e_l \in \mathcal{L}^{t_o,\mathcal{S}'}_{in} \setminus \{e_{min}\}}$: $id_{\mathcal{L}}(e_{min}) < id_{\mathcal{L}}(e_l)$, and $\forall_{e_l \in \mathcal{L}^{t_o,\mathcal{S}'}_{in} \setminus \{e_{max}\}}$: $id_{\mathcal{L}}(e_{max}) > id_{\mathcal{L}}(e_l)$.

Finally *queue_score*: $\mathcal{L}_{out} \times \mathbb{P}(\mathcal{L}) \rightarrow \mathbb{R}$ is a function that estimates the extent to which work items deviate from the expected input and output ordering. The formula for the *queue_score* function is as follows: $queue\_score(e_o, \mathcal{L}) = 1 - (distance(e_o, \mathcal{L}) / max\_distance(e_o, \mathcal{L}))$.

The *average queue score* for $\mathcal{L}^{\mathcal{S}'}_{out}$ is the simple *mean* of all the queue scores calculated for events in $\mathcal{L}^{\mathcal{S}'}_{out}$: $\sum_{e_o \in \mathcal{L}^{\mathcal{S}'}_{out}} (queue\_score(e_o, \mathcal{L})) / |\mathcal{L}^{\mathcal{S}'}_{out}|$.

To account for the changes in resource behaviour over time, one could adjust *distance* values using a forget function. Such a function deliberately gives more weight to events that occurred recently and lower weight to events that occurred further in the past.

Definition 17 (*Forget function*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log. We define $\mathcal{S}' \in \mathbb{S}_{\mathcal{L}}$ be a set of a particular type of segments that may exist in $\mathcal{L}$, and $\mathcal{L}_{out}$ be the out-list of $\mathcal{S}'$. Furthermore, let $T_{\mathcal{L}_{out}} = \{\#_{time}(e) \in \mathcal{D}_{time} | e \in \mathcal{L}_{out}\}$, $t_{min} \in T_{\mathcal{L}_{out}}$ where $\nexists_{t'_{min} \in T_{\mathcal{L}_{out}}} : t'_{min} < t_{min}$, and $t_{max} \in T_{\mathcal{L}_{out}}$ where $\nexists_{t'_{max} \in T_{\mathcal{L}_{out}}} : t'_{max} > t_{max}$.

We define a *forget function* $\mathcal{F} : T_{\mathcal{L}_{out}} \rightarrow [0,1]$ that maps the timestamp of all events in $\mathcal{L}_{out}$ to a real number such that $\mathcal{F}(t_1) < \mathcal{F}(t_2)$ iff $t_1 < t_2, \mathcal{F}(t_{min}) = 0$, and $\mathcal{F}(t_{max}) = 1$.

In practice, many cumulative probabilities distribution functions can be used to represent the function $\mathcal{F}$ defined above.

Definition 18 (*Adjusted queue score*)**.** Let $\mathcal{L} \subseteq \mathcal{E}$ be an event log. We define *adjusted_queue_score*: $\mathcal{L}_{out} \times \mathbb{P}(\mathcal{L}) \rightarrow \mathbb{R}$ as a function that estimates the extent to which activity instances deviate from the expected input and output ordering, *adjusted* with the forget function. The formula for the *adjusted_queue_score* function is as follows: $adjusted\_queue\_score(e_o, \mathcal{L}) = \mathcal{F}(\#_{time}(e_o))^* [1 - (distance(e_o, \mathcal{L}) / max\_distance(e_o, \mathcal{L}))]$. The *average adjusted queue score* for $\mathcal{L}^{\mathcal{S}'}_{out}$ is the simple *mean* of all the adjusted queue scores calculated for events in $\mathcal{L}^{\mathcal{S}'}_{out}$.

3. Implementation

The proposed approach has been implemented as a plug-in to the ProM framework.[4] The input to this plug-in is an event log in the standard XES format [22], and the output is a panel that consists of an option panel and a chart panel. The option panel (Fig. 4 - left) provides an interface for users to configure a number of options, such as the queue type (i.e., the queuing discipline to test: FIFO, LIFO, and Priority), the analysis perspectives (i.e., resource, activity, or case perspectives), the transaction types signifying the entry and exit of an item into a queue, the attribute name that is to be used for a priority attribute (if the priority queue type is chosen), and the type of cumulative probability distribution function that one would like to use to represent the forget function. Our implementation supports the following probability cumulative distribution functions: normal distribution, exponential distribution, poisson distribution, and logistic distribution.

Based on the chosen perspective, the "Select items to view" listbox will be populated with the values corresponding to the chosen "perspective". For example, if a user chooses "activity" as the desired perspective, the listbox displays all possible activity names in the log. Users can then refine the results by choosing only those values that they want to see. By default, the results for all of the values listed in the listbox will be shown.

Our implementation also allows user to customise how they want the queue score to be calculated when the number of items in a queue (i.e. the queue length) being analysed is *one*. As discussed in further details in Section 4.1, when the length of a queue is one, it is difficult to distinguish if a resource or if a case works in either FIFO, LIFO, or Priority queue fashion. We therefore allow users to customise the calculation in this situation to either (1) ignore the inclusion of that particular queue score, (2) include it in the calculation as normal, or (3) give a value of 0.5 (indicating no strong preference for the queue to follow either queuing discipline).

The chart panel (Fig. 4 - right) can be divided into two parts: the chart area at the top and the chart legend below the chart area. The chart area displays a time series that shows the trend of the *queue score* (y-axis - left) and the queue length (y-axis - right) over a period of time (x-axis). The queue score used at the y-axis coordinate (left) is the *queue score* as defined in Section 2. This queue score (x), normalised to the range of $[0, 1]$, measures the distance between the actual queue behaviour (as seen in the log) and the expected queue behaviour (based on a particular queuing discipline). A higher

[4] The plug-in is available in the nightly build version of the ProM Tool http://www.promtools.org/prom6/nightly/. Installation instruction is available from https://www.dropbox.com/s/j4boic9tmmm8clb/ResourceQueue_installationInstruction.pdf?dl=0.

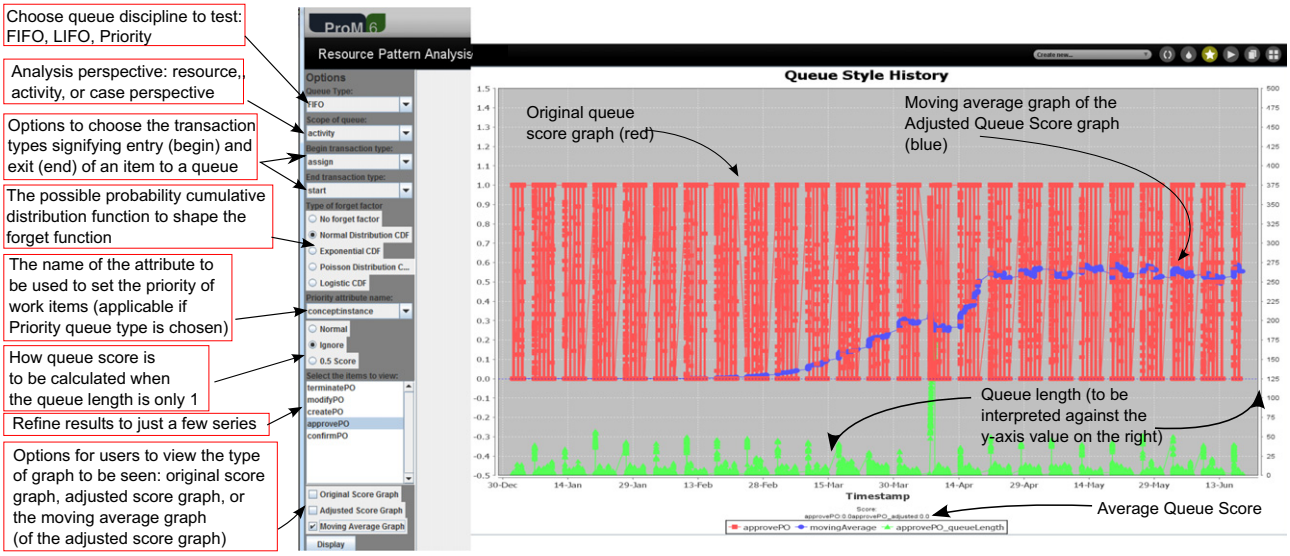

Fig. 4. The option panel.

x score indicates a greater match between the expected and the actual queuing behaviours.

Graphs for queue scores can be displayed using the original queue score values. However, experience shows that original queue scores are often volatile, resulting in a rather 'jagged' graph. We therefore allow users to 'smooth' the graph out using the *moving average* function [24]. The numbers shown on the legend area are the *average queue scores* calculated over the whole period captured in the event log. A higher average queue score indicates a closer match between the expected and actual queue behaviours.

4. Evaluation using synthetic data

This section presents the evaluation of the proposed approach using synthetic data sets with known "ground truths" (i.e. expected queue behaviours). Evaluation of our approach using a real-life data set with an Australian insurance company is provided in Section 5.

Fifty-five synthetic data sets were used, each with their specific known 'ground truths' and queuing disciplines (see Table 1 for details). These data sets were generated based on the same business process model shown in Fig. 5. Four life-cycle transitions are present in all the dataset: "schedule", "assign", "start", and "complete". The event logs contain finalised cases only. Work items in the synthetic data sets are performed by six resources.[5] All cases in these logs are started within a 24-week period. In the remainder of this article, each log is identified by its unique identifier per Table 1.

We have generated event logs for all the three perspectives ("activity", "resource" and "case"), each with FIFO and LIFO queue types. For the "resource perspective" logs, we also generated event logs with "Priority" queue type as the ground truth (Log ID "37" to "42").

In the following discussion, we denote a queue as a "sample". A queue with a ground truth equal to the tested queue type is referred to as a "positive sample", and a queue with a ground truth that is different from the queue type being tested is referred to as a "negative sample". For example, if we test for a FIFO-type queue in the activity perspective, the event logs "13" to "18" (FIFO ground truth) contain positive samples, while event logs "19" to "24" (LIFO ground truth) contain negative samples.

The *average queue score* (see Definition 16) is used as the primary metric for estimating the strength of the type of queue being detected: a higher score indicates a better agreement between the queue behaviour exhibited by resources seen in an event log and the tested queue type. Unless specified otherwise, the "assign" and "start" transitions are used as the "begin" and "end" transactional types for the activity and resource perspectives tests, and the "start" and "complete" transitions are used for the case perspective tests.

Due to space limitation, we summarise the results of our evaluations using synthetic data sets and elaborate key findings from these exercises.[6]

4.1. Logs with varying workload rates

Logs L1 to L42 are used to evaluate the detection of queue styles under different *workload rate* (i.e., the number of new cases started per week). Our experiments not only confirm the fact that our approach and its implementation *manage to detect the correct queuing disciplines* as per the actual ground truths, but also highlight a number of interesting phenomena. Uncovering such phenomena often provides valuable information that can be used to improve process management.

Without refinement in our analysis approach, the average queue scores will be quite high for any queuing discipline if we frequently see queues of length 1. For example, the average queue scores for positive samples (i.e. the tested queuing discipline matches the real one) in logs L13–L42 are all close to 1; however, the average queue scores for the negatives samples are also high, ranging from 0.2 to

[5] The synthetic event logs are generated by a Java program developed by the authors. These logs can be downloaded from https://www.dropbox.com/s/y30fy6e9t2px9ns/SyntheticEventLogs_FirstRevision.zip?dl=0.

[6] Details about the evaluations and the results are available in the other version of this paper https://www.dropbox.com/s/9na8e0ntt7u28ru/Suriadi_et_al_DSS_SpecialIssue_SmartBPM_FullVersion.pdf?dl=0.

Table 1
Overview of the synthetic data sets. For the "Number of cases" column, we present the total number of cases in the format of $X \times Y$, where X represents the number of started cases per week (ranges from 50, 100, 150, 200, 300, and 500), and Y is the number of weeks with started cases.

Log ID	Perspective	Ground truth	Number of cases	Noise
L1 – L6	Case	FIFO	50 × 24, 100 × 24..., 500 × 24	0%
L7 – L12	Case	LIFO	50 × 24, 100 × 24..., 500 × 24	0%
L13 – L18	Activity	FIFO	50 × 24, 100 × 24..., 500 × 24	0%
L19 – L24	Activity	LIFO	50 × 24, 100 × 24..., 500 × 24	0%
L25 – L30	Resource	FIFO	50 × 24, 100 × 24..., 500 × 24	0%
L31 – L36	Resource	LIFO	50 × 24, 100 × 24..., 500 × 24	0%
L37 – L42	Resource	Priority	50 × 24, 100 × 24..., 500 × 24	0%
L43	Resource	FIFO	250 × 24	10%
L44	Resource	FIFO	250 × 24	30%
L45	Resource	FIFO	250 × 24	50%
L46	Resource	FIFO	50 × 24	30%
L47 – L53	Resource	First FIFO, then LIFO	50 × 24, 100 × 24..., 500 × 24	0%
L54	Resource	Purely random	500 × 24	100%
L55	Resource	Alternate between FIFO and LIFO	500 × 24	100%

over 0.8. This phenomenon can be explained by the fact that the states of the queues in these logs contain, often, *only one element*. Thus, when this one element exits its queue, its behaviour is reflected as both a LIFO style and a FIFO style. In other words, when there is only one element in the queue, FIFO behaviour is manifested exactly as LIFO, and vice versa.

As explained in Section 3, the implementation of our approach allows users to handle the situation of a single item queue, such as providing options for users to *ignore* the inclusion of those queue scores calculated from a single item queue, or to assign a score of 0.5. Our experiments show that our approach can better decide queue styles when we ignore those queue scores obtained when the queue length is 1. In fact, when we ignored queue scores calculated when the queue length is 1, the average queue scores for negative samples in our experiments go down and approach 0 (as expected). There are a few exceptions, however. After ignoring the queue scores obtained when queue length is 1, we do expect queue scores for negative samples to be close to 0. However, in some of our experiments, the average queue scores for some negative samples were still rather high (sometimes as high as 0.35–0.4). We found that this unexpected phenomenon is caused by the queue containing items with exactly the same input or output timestamps. For example, we find that the unexpected high negative sample queue scores for log L13 exist when many instances of a particular activity are being started at the same timestamp. This is a limitation of the proposed approach as there is no mechanism to effectively handle the situation where there exist multiple items with the same timestamps in the queue.

Finally, our experiments also demonstrate that a priority queue of any length can also be interpreted as a FIFO queue or a LIFO queue. For example, suppose a queue contains two items A and B. Suppose A arrives earlier than B, and A has a higher priority attribute. The expected FIFO out-list is "A–B", which is also the expected order for Priority queue out-list. Our experiments using logs L37–L42 (logs with priority ground truth) show precisely this: the average queue scores for the negative samples are all over 0.3, with some as high as 0.7.

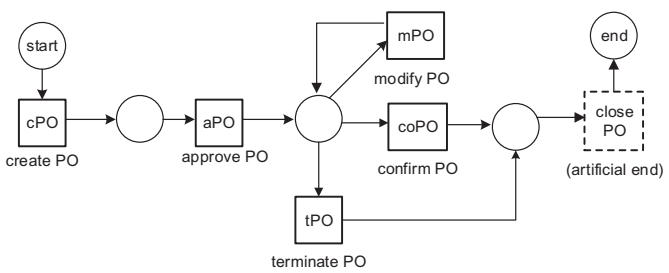

Fig. 5. The process model for the synthetic log.

4.2. Logs with noise

Our experiments using logs containing noise at varying intensity (L43 to L46) show that the average queue scores for the positive samples, while still high (above 0.7 in most cases), are generally lower than our previous experiments (with logs without noise); the scores for the negative samples (LIFO tests) are still low (all below 0.35). Furthermore, at a fixed workload rate (250 cases per week - logs L43, L44, and L45), as the noise level increases, the average queue scores for positive samples (FIFO tests) generally decrease while the scores for the negative samples (LIFO tests) generally increases. For a fixed noise level (30% - logs L44 and L46), we observe that the log with lower workload rate (L46) produces lower positive samples scores and higher negative sample scores, as compared with the log with higher workload rate (L44).

Based on these evaluation results, we can see that our approach performs as expected in the presence of noise: lower noise level leads to more accurate results, and vice versa. Furthermore, we can also see a positive correlation between the workload rate and the queue detection accuracy level. Again, this is expected: the higher the workload is, the longer the queue formed, thus allowing one to better see the differences between various queuing disciplines.

When resources randomly pick the next work item to perform (log L54), our approach shows that resources are more likely to exhibit LIFO behaviour than FIFO (Fig. 6). This may seem unexpected but it is actually unsurprising. Assume a worklist of 5 work items $\{w_1, w_2, w_3, w_4, w_5\}$ with their respective *event order identifier* of {1,2,3,4,5} (Definition 3 in Section 2.2). At time t_1, let's say the resource picks the work item located in the second position *of the worklist*, i.e. w_2, to execute. The resource is seen as choosing the *second-in-line* work item to execute, out of 5 that has been assigned to the resource thus far. This behaviour seems to align with FIFO better than LIFO (because the resource chooses a work item from the first-half of the queue).

Upon completion of w_2, at time $t_2 (> t_1)$, the worklist of the resource is now $\{w_1, w_3, w_4, w_5\}$. Assume the resource, again, picks the work item located in the second position *of the worklist*, i.e. w_3. Historically seen, the resource now picks the *third-in-line* work item to execute (from FIFO perspective) . That is, the resource starts to show preference of executing work items that were assigned

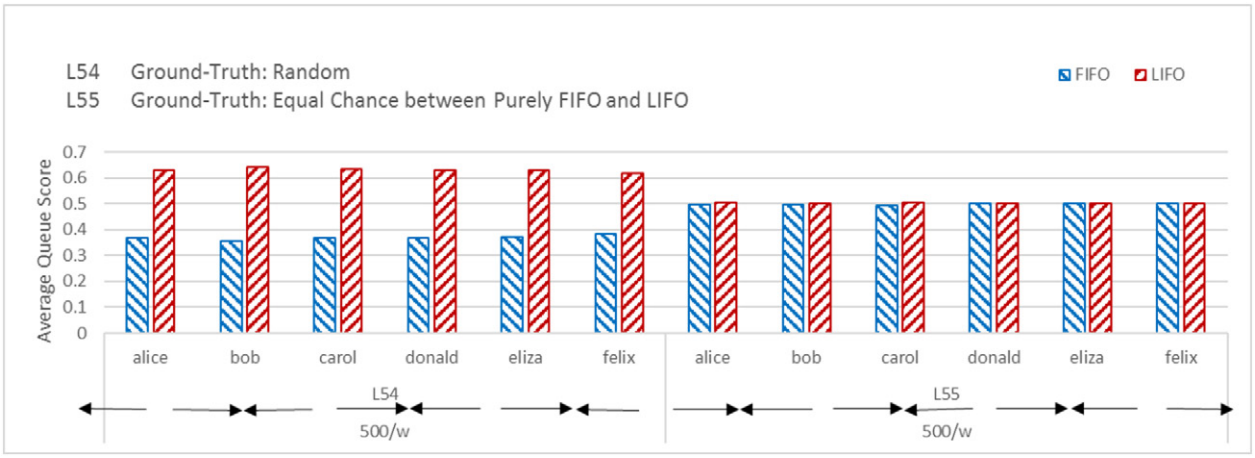

Fig. 6. Experimental results for event logs L54 (randomly picking the next task to perform from a set of ordered worklist) and L55 (alternating between pure FIFO and pure LIFO behaviour).

later, rather than earlier. At time t_3 (upon completion of w_3), if the resource also picks the work item located in the second position of the worklist (which will be w_4), the resource is now executing the *fourth-in-line* work item to execute. Here, a drift towards LIFO behaviour becomes more evident even though the resource still executes work item located in the first-half of the worklist. Obviously, such a drift is even more pronounced if the resource were to pick work items located on the third or higher positions on each iteration. Such a drift will only be neutralised if the resource were to pick the work item located on the first position of the worklist.

When a resource randomly picks the next work item to execute, each work item at each position in the worklist has an equal probability to be executed. However, as demonstrated above, it is sufficient for the resource to pick the second *or higher* position to execute for the drift towards LIFO to become visible. Hence, given a worklist of x-number of work items, there is a $\frac{1}{x}$ probability that the LIFO drift is neutralised (that is, picking the work item at position 1), while there is $1 - \frac{1}{x}$ probability that the LIFO drift is exacerbated. Thus, it is not difficult to see why a drift towards LIFO may be visible even when resources choose randomly the next work item to execute.

Nevertheless, in a situation where a resource is as likely to behave in a strictly FIFO manner as in LIFO manner, our approach behaves as expected: the average queue scores are close to 0.5 for both LIFO and FIFO assessments for all resources (as shown in Fig. 6 for log L55).

4.3. Logs with concept drift

Event logs L47 to L53 are resource perspective logs generated with a queue style ground truth that switches from FIFO to LIFO, thus exhibiting a form of "concept drift" [25]. Here, we use the scenario where the resources start working in the FIFO style, but switch to the LIFO style after a certain period of time. With the introduction of concept drift, one cannot rely on average queue scores to decide the queue style to which a queue belongs as the value takes into account the behaviour seen *for the whole period of the log*, not the behaviour seen at various points in time over which change is normally detected. Therefore, to detect change, we need to plot the queue scores over time and observe any change in the trend over time. Our experiments with logs L47 to L53 confirm that a change in the queuing disciplines can indeed be detected.[7]

[7] For details of our experiment results, please refer to the other version of this article at https://www.dropbox.com/s/9na8e0ntt7u28ru/Suriadi_et_al_DSS_SpecialIssue_SmartBPM_FullVersion.pdf?dl=0.

5. Evaluation using a real-life dataset

Having established the correctness of our approach and its implementation in Section 4, this section presents insights gained from the evaluation of the proposed techniques with a real-life dataset depicting the claims handling process from an Australian insurance company, NTI (National Transport Insurance). A brief overview of the claims handling process is as follows: when a claim notification of loss is received by the company (e.g., after a truck rolls over or is involved in an accident), an assessment of the claim is started. Notifications are also sent to the underwriting, recovery and settlement teams for processing. After the assessment is finalised, the claim is settled and a number of payments are made (including business interruption payments).

This data set was used to detect the three different queuing styles at different perspectives. The results were presented to *three* stakeholders (national claims manager, business reporting manager and national business operations manager) to evaluate the applicability and usefulness of the proposed approach in an organisational setting.

The stakeholders from NTI are also interested in comparing insurance claims behaviours across different states. Thus, we present not only analysis results obtained using Australia-wide data, but also comparative analysis results obtained using data from the states of QLD and VIC only. The comparative analysis for claims from QLD and VIC was carried out for the following reasons: (1) these two states have the largest number of cases, (2) the number of cases from these two states is comparable, but (3) the total number of events in QLD is higher than that for the VIC data with slightly longer mean and median case throughput times - approx. 10% (see Table 2). Through comparative analysis, we are interested in determining whether different queuing disciplines could provide insights into these performance differences.

5.1. Dataset pre-processing

Basic characteristics of the dataset used are provided in Table 2. There are 129 employees (anonymised) seen in the log, holding a total of 30 different organisational roles.

To evaluate the likelihood of the priority queuing discipline, each work item is labelled with a priority attribute whose values include "Highest", "High", "Normal" and "Low". This labelling is based on the stakeholders' input. It is important to know that the user interface of the claims processing software (that an NTI employee uses) displays tasks in a worklist that is ordered based on *task due dates* by default which could be in conflict with the priority order of tasks.

Table 2
Overview of the datasets (Australia vs. QLD vs. VIC).

	Australia	QLD	VIC
Num of events	154,885	43,645	37,008
Num of cases	11,995	3110	3043
Number of activities	45	45	44
Median case duration	71.8 days	76.9 days	69.9 days
Mean case duration	14 weeks	14.9 weeks	13.3 weeks
Time frame	01/07/2012–02/08/2014	03/07/2012–23/07/2014	02/07/2012–11/07/2014

The transaction types "assign" and "complete" are used as the "begin" and "end" markers in our analysis. Queues with lengths of one are ignored when calculating the average queue scores.

This remainder of this section presents the results of our analysis along with the feedback from stakeholders as appropriate and a few caveats w.r.t. our approach.

5.2. Analyses

We performed three types of analysis according to the three analysis perspectives that our approach supports: case, resource, and activity perspectives.

5.2.1. Case perspective analysis

Fig. 7 shows the average queue scores for the FIFO and LIFO styles. The average queue score for QLD, VIC and Australia are similar: the average queue scores for FIFO are between 0.2614 and 0.2760 and the average queue scores for LIFO are between 0.7240 and 0.7433 (out of 1). These results indicate that most cases did not follow the FIFO queue style and that cases seemed to be completed predominantly in the LIFO style. This can perhaps be explained by our earlier observation (Section 4.2) whereby resources who simply pick work items at random are likely to exhibit LIFO behaviours.

Stakeholders' feedback. This analysis has extracted an interesting insight for the stakeholders: it reveals *undesirable* work prioritisation styles (from the stakeholders' perspective), as they would prefer to see cases that started earlier to be finished first (i.e. a FIFO queue style), whereas our analysis results (Fig. 7) demonstrate that this is not the case; instead, we found the LIFO queue style to be more dominant. Further analysis of resource prioritisation behaviour in the subsequent sections sheds some light on this phenomenon.

5.2.2. Role/resource perspective analysis

Our analysis in the resource perspective is based on the *role* attribute as per the request of the stakeholders. A role is occupied by one or more employees and employees occupying the same role in different states is expected to carry out the same type of activities. Fig. 8 shows the average queue scores for Australia-wide analysis which reveals that 17 out of the total 21 roles were likely to follow LIFO queue. A few roles, however, were detected to execute work items mainly in the FIFO manner (e.g. roles identified by numbers 13, 6, and 12). However, even the highest FIFO score is only 0.729 (Role "6"), which is not as high as those scores obtained from running LIFO queue test. Interestingly, Fig. 8 shows that (1) roles with a maximum queue length of between 4 to 9 tend to exhibit FIFO-queue behaviour and (2) LIFO-queue behaviour becomes more dominant as the maximum queue length increases.

Fig. 9 (top and middle) present our role analysis results for QLD and VIC respectively. In these two states, the LIFO style is more dominant for roles with higher frequencies. Fig. 9 (bottom) shows several roles that behaved very differently between the two states: role "5" has different priority queue scores between the two states (QLD: 0.6022; VIC: 0.2286). Both roles "7" and "16" are detected to use the LIFO style in QLD, but not in VIC. The FIFO scores for role "7" are 0.1644 and 0.6618 in QLD and VIC respectively. Role "16" has a priority queue score as high as 0.6667 in VIC, while the priority score for this role is only 0.2296 in QLD. Role "13" is found to be exclusively using the FIFO/Priority-style queue in QLD (i.e., a score of 1 for both), whereas in VIC, the queue style adopted by this role is predominantly FIFO and Priority (0.8 for both).

Interestingly, for roles with high FIFO queue score, their priority queue scores are also high. For all the three datasets (i.e. Australia, QLD, and VIC), we can observe that, for a role with a FIFO score higher than 0.6, the priority queue score for the same role will also be higher than 0.6. In addition, role "13" in the QLD dataset has both the FIFO and priority queue scores equal to 1 (Fig. 9 - top). This indicates that the FIFO order and the priority order of the work items for this role are identical to each other. By further examining the dataset, we found that, role "13" in QLD has 77.78% work-items with a priority attribute of "High", and the remaining work-items with a priority attribute of "Normal".

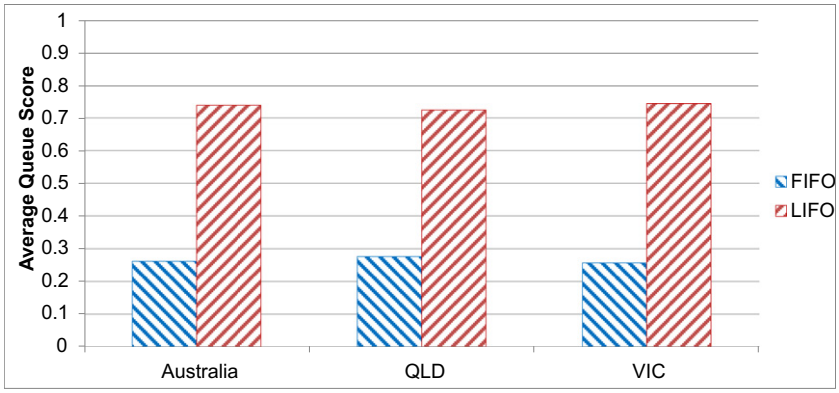

Fig. 7. Average queue scores (case perspective).

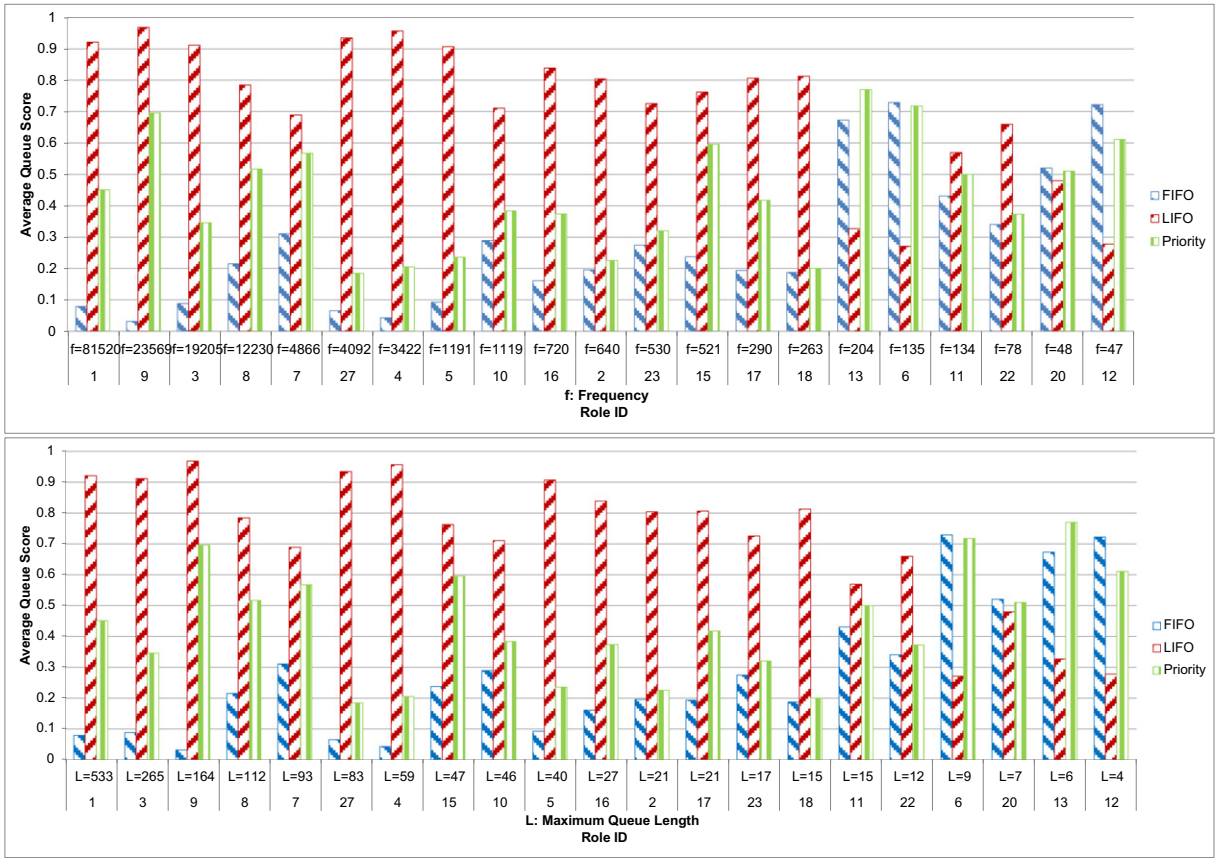

Fig. 8. Role perspective results for Australia, sorted by frequency (top) and maximum queue length (bottom).

For completeness, we also conducted resource-perspective analysis using the anonymised resource attribute. In this analysis, no particular queuing discipline stands out across resources in QLD and VIC. An interesting observation, however, is the fact that the maximum queue lengths for resources in QLD tend to be longer than those in VIC. There are no resources with a maximum queue length of over 50 in VIC, while in QLD there are four resources (identified as "FDLN", "DIQX", "PNCS" and "ZGAP") with maximum queue lengths greater or equal to 100.

Stakeholders' feedback. The stakeholders did expect roles "6", "13", and "12" (see Fig. 9 - top and middle) to exhibit different queuing behaviours compared to the rest. Roles "6" and "13" refer to managerial/senior roles and the tasks to be completed by role "12" are not driven by specific deadlines unlike other roles in the organisation.

The differences in the queue styles in QLD and VIC for roles "7" and "16" (see Fig. 9 - bottom) are interesting to note though the stakeholders are unable to explain the possible reasons behind the differences. The stakeholders indicate that role "7" is responsible for customer service, whereas the tasks for role "16" require more critical thinking and internal discussion. Further investigation into this matter is needed in order to explain these differences.

In relation to the longer queue lengths for resources in QLD as compared to VIC, the stakeholders indicated that the organisation has a high-functioning, long-time, and task-focused team in VIC, whereas the turnover rate for the employees in the QLD team is quite high. Thus, it is not surprising to see a higher number of items in the worklists of resources in QLD.

5.2.3. Activity perspective analysis

Our analysis in the activity perspective, with a few exceptions, also shows LIFO to be the dominant queuing discipline for all activity types Australia-wide, in QLD, and in VIC.

Fig. 10 compares queuing disciplines across two states in the activity perspective. There are only three activities that were dominantly executed in the FIFO style in both states (Fig. 10 - top): "Claim Auto Notification of Loss" (FIFO score: 0.6638), "Claim Settlement Consultant Notification" (FIFO score: 0.7672), and "Follow-up Unaccepted Assessment" (FIFO score: 0.6344). There are 22 activities executed mainly in the LIFO style with the LIFO scores ranging from 0.54 to 1. Fig. 10 (middle) shows the results of those activities. Interestingly, activities "Business Interruption Settlement Payment Due" and "Fleet Claim Notification Estimate Movement" were exclusively handled in the LIFO manner in VIC, whereas in QLD, they were sometimes processed in the FIFO style. Most of the activities shown in Fig. 10 (bottom) were executed in a random manner (as suggested by average queue scores of around 0.5).

Stakeholders' feedback. The results of our activity perspective analysis are well-received by the stakeholders. They found that these results confirmed their intuition that the way in which work items are presented to the employees on their computer screens influences their working style. Work items of the same activity type are sorted (from the top to the bottom of the screen) from the most recent

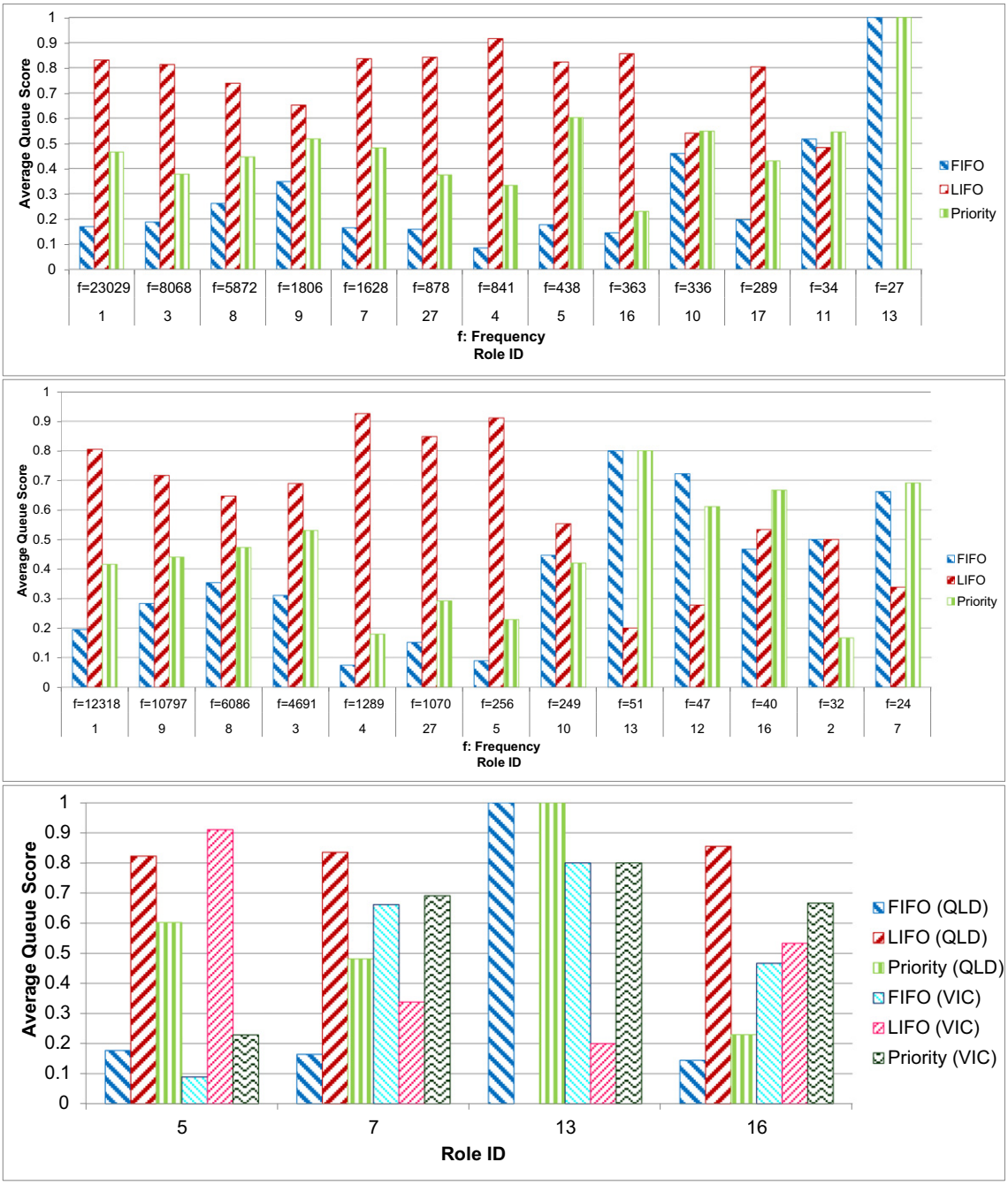

Fig. 9. Results at the role perspective for QLD (top) and VIC (middle), sorted by frequency. The bottom figure compares the queue scores for the two states.

ones to the oldest ones. Our analysis results indicate that people in the organisation mostly execute the tasks that come into their vision first (which are the most recent ones as they appear at the top of the screen), rather than scrolling down to the bottom of the screen to check for all tasks that may have arrived earlier before deciding on the tasks to be executed first (i.e., FIFO). Furthermore, the bottom bar chart of Fig. 10 shows the queue behaviours for some activities to be random - no predominant queuing discipline - as both the FIFO and LIFO scores are around 0.5. The stakeholders confirm that this is expected, and is indeed desirable, for some activities such as the "Assessment Type Changed" activity to be processed not according to a strict queuing discipline. Therefore, the insight gained here aligns with the stakeholders' expectations of those activity types.

Finally, with regards to the exclusively-LIFO queue style for the activities "Business Interruption Settlement Payment Due" (see the middle chart in Fig. 10) and "Business Interruption Progress Payment (Weekly)" (see the bottom chart in Fig. 10) seen in the state of VIC, the stakeholders told us that these activities were about providing payment to clients. Detecting an exclusively LIFO pattern may indicate poor customer experience as those who should receive payments earlier could be made to wait longer. From the stakeholders' perspective, it is acceptable to conduct these activities based on the needs of the customers rather than in a strict time-based order of either FIFO or LIFO. Therefore, the somewhat random queue style detected for these activities in the state of QLD (where neither LIFO queue score nor FIFO queue score dominates the analysis results) is expected and desirable.

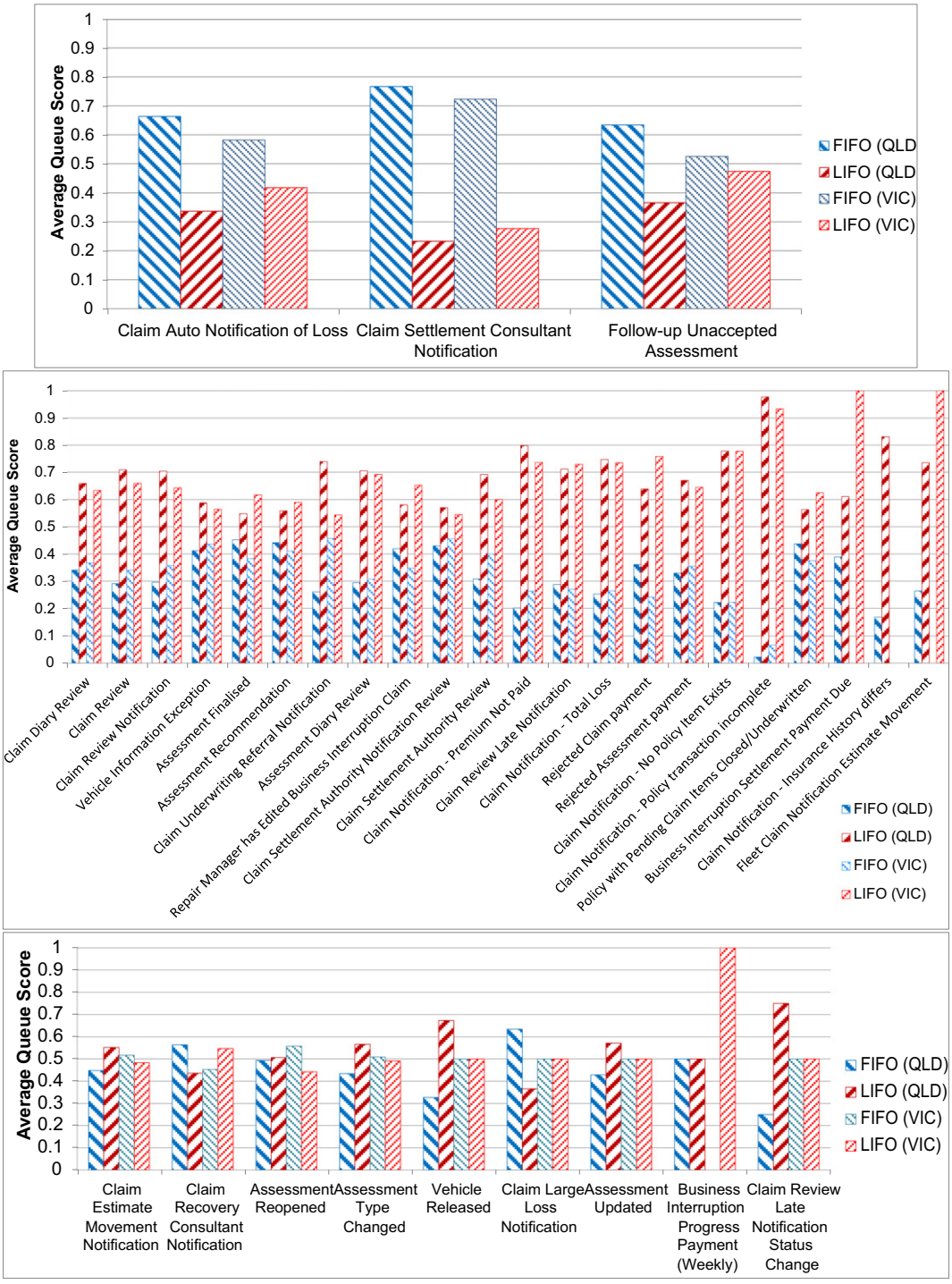

Fig. 10. Results in the activity perspective (QLD vs. VIC). Top: Activities where FIFO values larger than LIFO in both states; Middle: Activities with LIFO values larger than FIFO in both states; Bottom: The rest. Sorted by frequency.

5.3. Discussions and limitations

All three stakeholders found the insights gained from applying the resource work prioritisation approach to their datasets to be useful and highly detailed. From the discussions so far, a key revelation for the stakeholders is the fact that the LIFO styles are popular, regardless of the perspective of the analysis (i.e., case, role/resource, and activity perspectives). Analysis from the resource perspective shows that there are quite a number of employees who prefer the LIFO style. This is a useful insight to the stakeholders as, from their perspective, a more appropriate prioritisation style is to give a higher priority to the oldest work-items/cases, while taking into consideration the priority attributes and the due-date time where applicable.

We also found out that the organisation actively monitors the workload of employees on a daily basis in order to prioritise certain

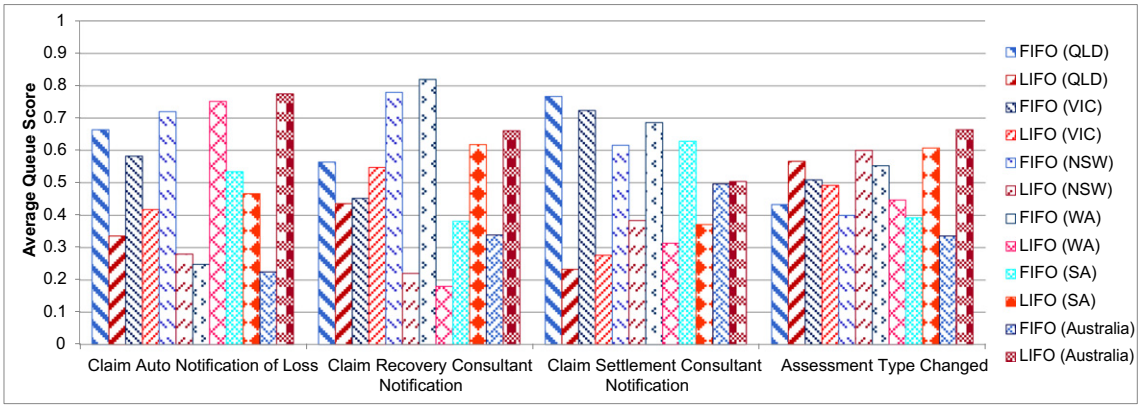

Fig. 11. Comparison of some activities across Australia.

activities/cases and to balance the workloads.[8] An employee records his/her worklist and itemises all work items that are due in 2 days or are overdue by 2 days. The worklists are then reviewed by a manager and the tasks are prioritised according to when they are started (the oldest first). The work is then re-distributed around the team where necessary. This monitoring and prioritisation activity is not being performed using any data analysis tool. The stakeholders mentioned that the monitoring of work prioritisation is a labour-intensive process and that their current workload/prioritisation analysis is not at the same level of sophistication as what we have presented to them. We then asked the stakeholders whether they would find this type of analysis useful. They all believed this type of analysis to be useful and expressed their interest to use our proposed technique on a regular basis (i.e., quarterly/annually) in order to monitor the changes in the work prioritisation styles over time. These insights point to the practical usefulness of the proposed work prioritisation detection approach.

This case study reveals a caveat in our approach: an activity that is detected to be FIFO when seen separately in each state may be detected to be handled mainly in LIFO when cases from different states are combined (i.e. across the whole of Australia) - see Fig. 11. For example, activity "Claim Settlement Consultant Notification" is detected to be handled mainly in the FIFO style when a separate analysis for each state is conducted; however, the combined analysis result (Australia) shows that both the FIFO and LIFO queue scores are around 0.5 with the LIFO score slightly higher than the FIFO score. The above phenomenon is entirely possible using our approach. Assume an input list consisting of work items from three different groups/cohorts A, B, and C and each cohort has two work items. That is, the combined input list is as follows: $(A_1, A_2, B_1, B_2, C_1, C_2)$. Let the output list be $(C_1, B_1, A_1, C_2, B_2, A_2)$. Then for each cohort, the output order is FIFO, but the output order for the combination of the three cohorts is not FIFO. Therefore, how one filters event logs greatly affects the type of queue detected. In this case study, analysing resource behaviour separately within each state makes more sense as work items are not assigned to resources at the national level, but locally within each state.

Strictly speaking, our approach is applicable when the event logs used contain two life-cycle transitions recorded for each work item. In practice, it is not uncommon to see an event log where each work item only has timestamp information related to just one life-cycle transition recorded (e.g., the "complete" time). While our approach cannot handle such a situation, through log pre-processing, we should be able to overcome this issue (we can derive a new transaction timestamp using, for example, the event interval analysis approach [23]). Nevertheless, in evaluating our approach using the real-life dataset from the NTI organisation, we have excluded those work items with only one life-cycle transition timestamp. The exclusion of these work items may lead to a mismatch between the analysis results and an employee's actual work prioritisation behaviour.

Finally, our approach is yet to improve the way it handles the situation whereby multiple work items enter and/or exit a queue at exactly the same time as detailed towards the end of Section 4.1.

6. Related work

In the last decade, a variety of process mining algorithms have been proposed [14,26]. Early process mining techniques were developed to discover *process models* from event logs [27]. A process model captures the *control-flow perspective* of a process, i.e. the temporal dependencies between various activities in business processes.

Recently, process mining algorithms have been expanded to include techniques to discover models capturing other process perspectives, such as the data-flow perspective [28] (which captures the way in which data was consumed and transformed) and the organisational perspective [14,17-19,26,29] (which discovers knowledge about the involvement of resources within process executions). The latter is also known as *organisational mining*.

Research in the domain of organisational mining mainly focused on addressing issues related to resource performance. For example, Nakatumba and van der Aalst investigate the effects of employee workloads on service times using regression analysis [18]. Huang et al. propose measures for resource preference, availability, competence, and cooperation, and showed how they can be discovered from logs [19]. Kim et al. [30] propose a method to construct a decision tree (constructed from past performances of resources) that can be used at run-time to decide the best resource to be assigned a particular work-item given an objective (lowest cost or fastest completion time). In [20], Pika et al. present a general framework to detect changes in resource behaviour over time by making use of time series analysis techniques. Senderovich et al. present two different approaches, namely, data mining with decision trees and queuing heuristics to mine resource scheduling protocols for service systems [31]. Finally, the work by Suriadi et al. [23] manipulates time intervals between various events to build various pictures of resource performance, including their throughput, workload, and idle times. By contrast, in this paper, we focus less on understanding

[8] The dataset that was used for the analysis is from 2012 to 2014. The discussions with the stakeholders were held in late 2015 and the organisation has implemented a number of improvements to their work prioritisation style since.

resource performance, but more on learning the way resources select and prioritise their work. In particular, we propose techniques to learn the order in which work items are being executed by a resource (i.e., the queuing discipline adopted by resources). We contend that such a queue discovery approach has not been properly addressed within the process mining discipline.

Recent work within the process mining community that is quite closely related to our approach is the work by Senderovich et al. [32–34] which seeks to learn the queue lengths for the purpose of online delay prediction, assuming a particular type of queuing discipline employed. Our work presented in this article is distinct in that we want to discover the *queuing discipline employed* by resources, instead of predicting queue length or waiting time.

Within business process simulation community, process mining is often applied to learn historical resource behaviour such that realistic simulation models can be realised (e.g. Rozinat et al. [35] and Wynn et al. [36]). Appropriately configuring the queuing discipline for each pool of resources is key for business process simulation [37,38]. Within this context, our work can be positioned as complementing process simulation research as it allows creators of business process simulation models to learn from historical data the most appropriate work prioritisation behaviour.

The work by Akhavian and Behzadan [39] is closely related to ours. In this work, a technique to learn the queuing discipline employed by resources is proposed. Their approach, however, is different in that their starting point of analysis is a collection of sensor data, *instead of* an event log like in our approach. Consequently, their approach does not support the learning of queuing disciplines from various perspectives (such as resource, activity, and case perspectives), which our approach supports. Finally, the approach proposed by Akhavian and Behzada [39] learns queuing discipline in a *batch* fashion whereby assessment of queuing discipline employed happens *once* after entries in the log are processed. Our approach, on the other hand, operates in a *streaming* fashion whereby assessment of the queuing discipline employed by a resource is performed every time a work item exits a queue. As such, drifts in the queuing discipline employed by resources over time can be easily captured. Furthermore, the *streaming* nature of our approach also means that it can be easily adapted to learn the queuing discipline of a resource in real-time.

While the concept of queue is used heavily in this article, it is different from other work concerned with queuing theory as flowing from the work by Agner Krarup Erlang [3] and its variants [40–45]. By looking at these literature, one can see that the type of problems this work addresses is around the issues of capacity planning and resource optimisation. For example, the work by Erlang [3] looks at the problem of telephone traffic congestion and proposed models to predict the behaviour of systems with randomly arising demands. The work by Maghsoudlou et al. [43] also apply queuing theory to assess the performance of a supply chain network. Queuing theory has also been widely applied in hospital settings to better plan bed capacity (e.g. Shen and Wang [44]) or to assess the capacity of an emergency department load (e.g. Rosen [45]).

Examples mentioned above mostly involve the creation of a model of queues with certain behavioural assumptions, including the arrival rate of new work items and the mean working time to serve a customer. This model is then studied to predict how a system behaves under various loads. The work prioritisation approach proposed in this article is therefore different in that we do not attempt to create a top-down model of a queue with certain characteristics; instead, we attempt to detect, from data, *how* a resource prioritises the tasks that he/she has been assigned to, whether in a FIFO fashion, LIFO style, or through some other prioritisation mechanism.

Another stream of research focuses on the impact of the choice of queuing discipline on the performances of the systems/processes being studied. Research in this area (for example [46–48]), however, simply assumes a particular type of queuing discipline as a starting point. Our work serves as input for such approaches. Our work starts with the assumption that we *do not know* the queuing discipline that exists in the process being investigated; instead, it is the *goal* of our approach to detect the queuing discipline being used. Furthermore, resource behaviour may change over time: a resource could at some point follow a FIFO queuing discipline and, at a later point in time, shift to a priority-based queue. Our approach allows one to observe such changes.

Finally, the implementation of our approach currently only supports three queuing disciplines (FIFO, LIFO, and Priority), while there exist many other queuing disciplines (for example, Kumar and Sharma [49]). As detailed in Section 2, our approach attempts to determine the queuing discipline of resources by looking at the alignment between 'expected' vs. 'actual' work items that exit a queue (given a particular queuing discipline). Therefore, as long as it is possible to determine the expected work item to exit a queue from the current state of a queue, our approach can support the assessment of that particular queuing discipline. Note that this holds for almost any queuing discipline provided the required contextual data is available.

7. Conclusion and future work

This paper described a novel approach to determining the degree of conformance of the behaviour of a resource to some prescribed queuing discipline related to the prioritisation of work items. The approach makes use of transactional data recorded in event logs to learn the prioritisation orders of resources when undertaking work. The approach has been generalised to also learn the prioritisation order of cases and activities. The approach supports well-known queuing disciplines including FIFO, LIFO, and Priority. Through the evaluation of our approach, it is also interesting to note that LIFO behaviours may simply be a consequence of resources choosing work items at random. The proposed approach has been implemented as a plug-in of the open-source process mining framework, ProM. The approach was evaluated using a range of synthetic and real life datasets. The paper also discussed the findings from a case study conducted at an Australian insurance company which shows the usefulness of our approach in practice. The main future work in this regard involves improving the way in which score for a particular queuing discipline is calculated to take into account the situation where two or more work items enter or exit a queue at the same time and where there is only one transaction lifecycle recorded for each work item (the limitations of our current approach as mentioned in Section 4).

Acknowledgements

This work is partly supported by the ARC Discovery Cost-Aware Business Process Management grant (DP120101624).

References

[1] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag, Berlin, 2007.
[2] T. Pyzdek, P.A. Keller, The Six Sigma Handbook, McGraw-Hill Education. 2014.
[3] D. Gross, Fundamentals of Queueing Theory, John Wiley & Sons. 2008.
[4] Z. Feldman, A. Mandelbaum, W.A. Massey, W. Whitt, Staffing of time-varying queues to achieve time-stable performance, Management 54 (2) (2008) 324–338.
[5] A. Li, W. Whitt, J. Zhao, Staffing to stabilize blocking in loss models with time-varying arrival rates, Probab. Eng. Inf. Sci. 30 (2) (2016) 185–211.
[6] Y. Liu, W. Whitt, Stabilizing customer abandonment in many-server queues with time-varying arrivals, Oper. Res. 60 (6) (2012) 1551–1564.
[7] W. Whitt, Stabilizing performance in a single-server queue with time-varying arrival rate, Queueing Syst. 81 (4) (2015) 341–378.
[8] W. Mélangea, J. Walraevensa, D. Claeysa, B. Steyaerta, H. Bruneela, The impact of a global FCFS service discipline in a two-class queue with dedicated servers, Comp. Oper. Res. 71 (2016) 23–33.

[9] J. Walraevensa, T. Maertens, H. Bruneel, A semi-preemptive priority scheduling discipline: performance analysis, Eur. J. Oper. Res. 224 (2013) 324–332.
[10] H.M. Asif, E.-S.M. El-Alfy, Performance Evaluation of Queuing Disciplines for Multi-class Traffic using OPNET Simulator, MMACTE'05, WSEAS. 2005, pp. 1–6.
[11] O. Rose, The Shortest Processing Time First (SPTF) Dispatch Rule and Some Variants in Semiconductor Manufacturing, Winter Simulation Conference, vol. 2, 2001. pp. 1220–1224.
[12] J. Nzouonta, T. Ott, C. Borcea, Impact of queuing discipline on packet delivery latency in ad hoc networks, Perform. Eval. 66 (12) (2009) 667–684.
[13] R. Rönngren, R. Ayani, A comparative study of parallel and sequential priority queue algorithms, ACM Trans. Model. Comput. Simul. 7 (2) (1997) 157–209.
[14] W.van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer-Verlag, Berlin, 2011.
[15] J. Bergs, D. Vandijck, O. Hoogmartens, P. Heerinckx, D.V. Sassenbroeck, B. Depaire, W. Marneffe, S. Verelst, Emergency department crowding: time to shift the paradigm from predicting and controlling to analysing and managing, Int. Emerg. Nurs. 24 (2016) 74–77.
[16] C. Fernandez-Llatas, J.-M. Benedi, J.M. Garcia-Gomez, V. Traver, Process mining for individualized behavior modeling using wireless tracking in nursing homes, Sensors 13 (11) (2013) 15434–15451.
[17] M. Song, W. van der Aalst, Towards comprehensive support for organizational mining, Decis. Support. Syst. 46 (1) (2008) 300–317.
[18] J. Nakatumba, W. van der Aalst, Analyzing Resource Behavior using Process Mining, Proceedings of BPI'2009, Vol. 43 of LNBIP, Springer. 2010, pp. 69–80.
[19] Z. Huang, X. Lu, H. Duan, Resource behavior measure and application in business process management, Expert Systems with Applications 39 (7) (2012) 6458–6468.
[20] A. Pika, M. Wynn, C. Fidge, A. ter Hofstede, M. Leyer, W. van der Aalst, An Extensible Framework for Analysing Resource Behaviour Using Event Logs, CAiSE, Vol. 8484 of LNCS, Springer. 2014, pp. 564–579.
[21] J.E. Hopcroft, Introduction to Automata Theory, Language, and Computation, Addison Wesley. 2001.
[22] C.W. Günther, E. Verbeek, XES Standard Definition, 2nd ed., Eindhoven University of Technology, The Netherlands, March 2014.
[23] S. Suriadi, C. Ouyang, W. van der Aalst, A.H.M. ter Hofstede, Event interval analysis: why do processes take time? Decis. Support. Syst. 79 (2015) 77–98.
[24] C. Droke, Moving Averages Simplified, Marketplace Books. 2001.
[25] J. Bose, W. van der Aalst, I. Zliobaite, M. Pechenizkiy, Dealing with concept drifts in process mining, IEEE Trans. Neural Netw. Learn. Sys. 25 (1) (2014) 154–171.
[26] W. van der Aalst, M. Schonenberg, M. Song, Time prediction based on process mining, Inf. Syst. 36 (2) (2011) 450–475.
[27] W. Gaaloul, S. Alaoui, K. Baina, C. Godart, Mining Workflow Patterns through Event-data Analysis, The 2005 Symposium on Applications and the Internet Workshops, IEEE. 2005, pp. 226–229.
[28] M. de Leoni, W. van der Aalst, Data-aware Process Mining: Discovering Decisions in Processes Using Alignments, SAC'13, ACM, New York, NY, USA, 2013, pp. 1454–1461.
[29] W. van der Aalst, H. Reijers, M. Song, Discovering social networks from event logs, Comput. Supported Coop. Work (CSCW) 14 (6) (2005) 549–593.
[30] A. Kim, J. Obregon, J.-Y. Jung, Constructing decision trees from process logs for performer recommendation, BPM Workshops LNBIP (171) (2014) 224–236.
[31] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Mining Resource Scheduling Protocols, BPM Vol. 8659 of LNCS, Springer. 2014, pp. 200–216.
[32] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue Mining - Predicting Delays in Service Processes, CAiSE 2014, Vol. 8484 of LNCS, Springer. 2014, pp. 42–57.
[33] A. Senderovich, S.J.J. Leemans, S. Harel, A. Gal, A. Mandelbaum, W. van der Aalst, Discovering Queues from Event Logs with Varying Levels of Information, in: Reichert, M. Reijers, H. (Eds.) Business Process Management Workshops. BPM 2015. Lecture Notes in Business Information Processing, vol. 256, Springer 2016,
[34] A. Senderovich, Matthias Weidlich, Avigdor Gal, A. Mandelbaum, Queue mining for delay prediction in multi-class service processes, Inf. Syst. 53 (2015) 278–295.
[35] A. Rozinat, R.S. Mans, M. Song, W. van der Aalst, Discovering simulation models, Inf. Syst. 34 (3) (2009) 305–327.
[36] M. Wynn, A. Rozinat, W. der Aalst, A.H. ter Hofstede, C. Fidge, Process Mining and Simulation, Modern Business Process Automation, Springer. 2010,
[37] W. van der Aalst, Business Process Simulation Survival Guide, Handbook on Business Process Management 1, Springer. 2014, pp. 337–370.
[38] N. Martin, B. Depaire, A. Caris, The use of process mining in business process simulation model construction - structuring the field., Bus. Inf. Sys. Eng. 58 (1) (2016) 73–87.
[39] R. Akhavian, A.H. Behzadan, Evaluation of queueing systems for knowledge-based simulation of construction processes, Automation in Construction 47 (2014) 37–49.
[40] I. Adan, J. Resing, Queueing Systems, Eindhoven. 2002, http://www.win.tue.nl/iadan/queueing.pdf.
[41] B. Avi-Itzhak, H. Levy, On measuring fairness in queues, Advances in Applied Probability 36 (3) (2004) 919–936.
[42] B. Avi-Itzhak, H. Levy, D. Raz, A resource allocation queueing fairness measure: properties and bounds, Queueing Syst 56 (2) (2007) 65–71.
[43] H. Maghsoudlou, M.R. Kahag, S.T.A. Niaki, H. Pourvaziri, Bi-objective optimization of a three-echelon multi-server supply-chain problem in congested systems: modeling and solution, Comput. Ind. Eng. 99 (2016) 41–62.
[44] X. Shen, X. Wang, Improving the health-care delivery process at hospital emergency services by a better use of inpatient bed information, Electron. Commer. Res. Appl. 14 (2015) 14–22.
[45] B.L.P. Rosén, Measuring effective capacity in an emergency department, J. Health Organ. Manag. 30 (1) (2016) 73–84.
[46] S. Creemers, M. Lambrecht, Modeling a Hospital Queueing Network, Queueing Networks: A Fundamental Approach, International Series in Operations Research and Management Sciences Springer. 2011, Chap. 18.
[47] S. Saghafian, G. Austin, S.J. Traub, Operations research/management contributions to emergency department patient flow optimization: review and research prospects, IIE Trans. Healthcare Syst. Eng. 5 (2) (2015) 102–123.
[48] L. Green, Queueing Analysis in Healthcare, Patient Flow: Reducing Delay in Healthcare Delivery, Springer. 2006. Chap. 10
[49] M. Kumar, S.C. Sharma, Priority Aware Longest Job First (PA-LJF) Algorithm for Utilization of the Resource in Cloud Environment, INDIACom, 2016. pp. 415–420.

Dr. Suriadi Suriadi is a Senior Research Fellow at Queensland University of Technology. Before this, from 2014 to 2016, he was a Lecturer within the College of Sciences of Massey University, New Zealand. He obtained his PhD degree in the discipline of Information Security in late 2010 from the Queensland University of Technology (QUT). Since 2007, he has been involved in a number of research projects in the area of information security. From 2011 to late 2014, he was a Research Fellow within the Business Process Management discipline at Queensland University of Technology, Brisbane, Australia. He enjoys working in collaborative, cross-domain research projects that allow the application of research outcomes to address real-world problems. His main research interests are in the area of process mining and information security.

Dr. Moe T. Wynn is a Senior Lecturer in the field of Business Process Management (BPM) within the school of Information Systems at Queensland University of Technology. Her research interests include process automation, process mining, comparative process analytics and cost-aware business process management. She has published over 60 refereed research papers. Her work appeared in well-known journals in the field including Information Systems, Information Sciences, Data and Knowledge Engineering, Information and Software technology, Formal Aspects of Computing, Journal of Computer and System Sciences, and Transactions on Petri Nets and Other Models of Concurrency. Her work is supported by over AUD $3.3 million in grant funding in the past five years.

Dr. Jingxin Xu received his bachelors degree in telecommunications engineering from Xidian University, Xian, China, in 2008, masters degree in information technology from the Queensland University of Technology (QUT), Brisbane, QLD, Australia, in 2010, and PhD degree in signal processing from QUT, in 2014. He is currently a Postdoctoral Research Fellow with QUT, investigating process mining techniques for business process management.

Prof.dr.ir. Wil van der Aalst is a Full Professor of Information Systems at the Technische Universiteit Eindhoven (TU/e), The Netherlands. He is also the Academic Supervisor of the International Laboratory of Process-Aware Information Systems of the National Research University, Higher School of Economics in Moscow. Moreover, since 2003 he has a part-time appointment at Queensland University of Technology (QUT). At TU/e he is the scientific director of the Data Science Centre Eindhoven (DSC/e). Wil van der Aalst has published more than 200 journal papers, 20 books (as author or editor), 450 refereed conference/workshop publications, and 60 book chapters. Many of his papers are highly cited (he is one of the most cited computer scientists in the world and has an H-index of 131 according to Google Scholar) and his ideas have influenced researchers, software developers, and standardization committees working on process support. In 2012, he received the degree of doctor honoris causa from Hasselt University. In 2013, he was appointed as Distinguished University Professor of TU/e and was awarded an honorary guest professorship at Tsinghua University. He is also a member of the Royal Holland Society of Sciences and Humanities (Koninklijke Hollandsche Maatschappij der Wetenschappen) and the Academy of Europe (Academia Europaea).

Prof.dr. Arthur ter Hofstede is a Professor in the Information Systems School in the Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia, where he is Head of the Business Process Management Discipline. He is also a Professor in the Information Systems Group of the Department of Industrial Engineering of the Technische Universiteit Eindhoven (TU/e). His research interests are in the areas of business process automation and process mining.