BPI Challenge 2019 Report: a Purchase-to-Pay Process Analysis

Adriano Augusto, Volodymyr Leno, and Daniel Reissner

University of Melbourne
{aaugusto, vleno, dreissner}@student.unimelb.edu.au

Abstract. Every year, since 2011, the Business Process Intelligence Challenge (BPI Challenge) allow students and practitioners to test their process mining skills: analysing process data sourced from real operating companies, addressing specific stakeholders questions, and trying to derive valuable insights. This year (2019), the process data captures the purchase-to-pay process of a company operating in the Netherlands in the area of coatings and paints. This paper reports the results of our process mining analysis, which revolved around the three main questions set by the stakeholders. Precisely, we (i) discovered a set of as-is process models, and a set of to-be process models representing the purchase-to-pay process; (ii) developed a Java application to run a throughput analysis of the process execution; (iii) identified the process deviant behavior and highlighted the most interesting and valuable insights.

1 Introduction

Business processes shape the way organizations operate in order to provide services and products to their customers. Given their importance, business processes are most of the times supported by information systems, which record data about individual executions of the processes. This data is stored in the form of event logs, where each event represents the execution of an activity in the context of a case. However, this raw process data would be useless if not analysed. In the last two decades, numerous research studies considered the problem of process data analysis, proposing a range of techniques that help organizations to gain useful information and knowledge about their business processes, with the ultimate goal of assessing and (where possible) improving them. These techniques belong to the discipline of *Process Mining* and they can be categorised into three macro groups: (i) automated process discovery (generating a process model from an event log); (ii) conformance checking (identifying and diagnosing mismatches between reference behavior and recorded behavior); and (iii) process enhancement (enriching a process model using the information recorded on the event log) [3].

Given the rising interest in Process Mining both in research and industry, in 2011, the Business Process Intelligence Challenge (BPI Challenge) was established, with the goal of giving an opportunity to students, academics, and practitioners to test their process mining knowledge, techniques, and skills: analysing process data sourced from real operating companies, addressing specific stakeholders questions, and trying to derive valuable insights.

This year, for the BPI Challenge 2019, we analysed the process data recorded during the execution of the Purchase-to-Pay (P2P) process of a company operating in the Netherlands in the area of coatings and paints[4]. This report summarises the results of our analysis, which was driven by three main questions proposed by the company stakeholders. Precisely, the company is interested in:

1. obtaining a collection of process models that clearly explain their P2P process.
2. analysing the throughput of the P2P process, with focus on its time performance.
3. identifying compliance issues, comparing the process behavior recorded in the process execution data with the expected behavior captured in the process models and its high-level textual description.

The structure of this report follows the three main questions of the company. In Section 2, we focus on process identification and discovery, describing the steps we applied (and tools we used) to analyse and understand the P2P process execution in order to produce a set of process models capturing the recorded process behavior (as-is processes) and a set of process models representing the expected process behavior (to-be processes). In Section 3, we focus on the throughput of the P2P process, with a great focus on its time performance, we describe the application we implemented for analysing this latter, and we report a wide range of measurements. In Section 4, we discuss our conformance checking analysis, focusing on the compliance issues we identified and highlighting the most interesting mismatches between desired process behavior and recorded process behavior. Finally, the conclusion summarises the output of our process (mining) analysis.

2 Process Discovery

In this section, we first provide a textual description of the P2P process, as a reference to understand in depth this report and our analysis. [1] Then, we analyse the process behavior recorded in the event log. Finally, we show how we generated the *as-is process models*, combining automated process discovery with ad-hoc refinements driven by our process map analysis. We conclude the section proposing a set of *to-be process models*, which capture in a more precise way the process description and leave out infrequent (and senseless) behavior that we identified in the event log during this first step of our analysis.

2.1 Process Description

The P2P process handles purchase orders (POs), from their creation to their clearance. Each PO is recorded into a *PO document* containing several lines. Each line of a PO document refers to a *PO item*, specifying its details (e.g.

[1] The process description we provide can be found also in the BPI Challenge Manifest, available at: https://icpmconference.org/icpm-2019/contests-challenges/bpi-challenge-2019/

item type, category, etc.) and its value (i.e. its cost). Each PO item is processed separately and according to a specific data flows. The data flows are characterized by three main steps and their execution order, these are: (i) *Record Goods Receipt*, (ii) *Record Invoice Receipt*, (iii) *Clear Invoice*. Here, we summarise the four possible data flows.

DF1. *3-way matching, invoice after goods receipt.* In this type of data flow, *Record Goods Receipt* must occur before *Record Invoice Receipt*, and this latter before *Clear Invoice*. The value of the Goods Receipt message must match the value of the Invoice Receipt message and the value at the creation of the PO item.

DF2. *3-way matching, invoice before or after goods receipt.* In this type of data flow, *Record Invoice Receipt* can occur either before or after *Record Goods Receipt*. However, if it occurs before *Record Goods Receipt*, the data flow is blocked until the Goods Receipt message is received. Also in this case, as the previous, *Clear Invoice* can occur only if the value of the Goods Receipt message matches the value of the Invoice Receipt message and the value at the creation of the PO item.

DF3. *2-way matching, no goods receipt needed.* In this type of data flow, no Goods Receipts message is received. Instead, the Invoice Receipt message is received and its value must match the value at the creation of the PO item. However, the entire value of the PO item could be consumed partially, such that multiple Invoice Receipt messages are received, each of them covering a sub-value of the total PO item value (until the total value is covered).

DF4. *Consignment.* This data flow is the simplest one, it requires only to *Record Goods Receipt*, whilst it does not need to *Record Invoice Receipt*, nor to *Clear Invoice*.

The data flows give an overview of what are the main steps of a PO item processing and their execution order, however, a PO item could follow a certain data flow multiple times. For example, a PO item processed with the data flow DF1 could have multiple Goods Receipt messages, each of them followed by an Invoice Receipt message and its clearance (i.e. *Clear Invoice* step).

In the following, we refer to this process description to interpret the process execution data recorded in the event log, as well as to provide the final to-be process models that capture both the behavior in the event log and the behavior description.

2.2 Process Analysis

We started our process analysis by visualising the data recorded in the original event log [4]. To do so, we uploaded the event log into Disco[2], which allows us to visualise it as a process map, as well as to apply filters on events, activities, paths, timeframe etc. Figure 1 shows the unfiltered process map. We can easily understand that the data recorded in the event log is noisy, and filters are necessary to interpret it, this is common when dealing with real-life data.

[2] A commercial process mining platform, available at: https://fluxicon.com/disco/

Given that each case in the event log records the data flow of a PO item within a PO document, and that each data flow belongs to one of the four types described above (i.e. DF1-DF4), we decided to split the original log into four sublogs, each capturing one of the four data flows. Then, we took into account the timeframe of the events recorded for each data flow, and we observed that several events where recorded in doubtful dates and times (see Table 1). Consequently, we applied a filter on the timeframe, retaining only the data flows recorded between the *31 Dec 2017* and the *18 Jan 2019*. We believe that the events falling outside this timeframe are due to either recordings errors or data corruption, the company may be interested in revising the robustness of its process recording software and/or the security of its databases.

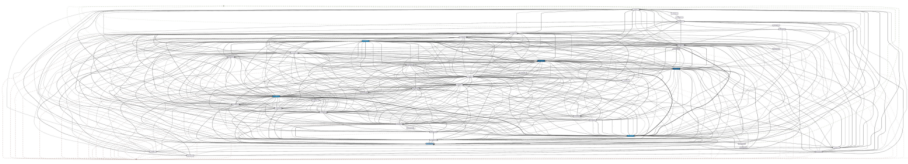

Fig. 1. Process map of the original log, unfiltered.

Data Flow	**Recordings Timeframe**	
	Start	**End**
DF1	*24 Jan 2001*	*10 Apr 2020*
DF2	*27 Jan 1948*	*06 Dec 2019*
DF3	*26 Jan 2017*	*02 Feb 2019*
DF4	*31 Dec 2017*	*18 Jan 2019*

Table 1. Data flows timeframes observed in the event log.

At this stage, we obtained four different event logs (one per data flow type) having a restrained timeframe. We proceed with analysing each of them separately.

Data Flow 1. The DF1 recordings show that up to 38 unique activities can be executed during the processing of the PO items. However, having a closer look at the activities, we notice that some of them are likely to be *external*. We consider external those activities that do not relate to the PO item processing because either belonging to another context or executed by external entities.
Assumption-1. All the *SRM*-tagged activities (e.g. *SRM: Created*, *SRM: Ordered*) are external, since they clearly relate to the *Supplier Relationship Management* (SRM) process.

It is not easy to understand why the SRM activities are recorded in the event log of the P2P process, and unfortunately, there is no mention of the SRM

process in the details provided by the company. For the remaining, we resort to Assumption-1 when analysing the SRM activities. In fact, it is likely that they follow a separate workflow, being performed most of the times by *Batch Users*, and often in between different (random) activities not relating to the SRM process.

Assumption-2. The activities performed by the *Vendor*[3] are external, which are the following two: *Vendor Creates Debit Memo* and *Vendor Creates Invoice*. Furthermore, these activities are always recorded by the user *NONE*, in the event log.

Following assumption-2, we analysed the user *NONE*. This latter performs only four activities in the event log: (i) *Vendor Creates Debit Memo*; (ii) *Vendor Creates Invoice*; (iii) *Record Service Entry Sheet*; (iv) *Clear Invoice*. This finding led us to formulate the next assumption.

Assumption-3. Tthe user *NONE* represents the *Vendor* within our event log.

Assumption-3 associates the activity *Record Service Entry Sheet* to the Vendor as well, since this activity is always and only performed by the user *NONE*. More complicated is the case of the activity *Clear Invoice*, being this latter performed 4.1% of the times by *NONE*, but the remaining by company users. This fact was remarkable, indeed, according to the process description the *Clear Invoice* activity is one of the most important of the P2P process, how come such activity could be recorded by a user without leaving traces?[4] Consequently, we did not associate *Clear Invoice* to the Vendor, but we considered outliers those instances of *Clear Invoice* which are recorded in the event log by the user *NONE*.

After identifying the external activities, i.e. SRM and Vendor activities, we filtered them out from the event log, being left with 23 activities. Among these 23 activities, we identified 5 which are rework activities (see Table 2), yet the total number of activities is very high to have an overview of the DF1 processing procedure. At last, we decided to filter the activities with frequency less than 1%. This last filter highly reduced the total number of activities to five: *Create Purchase Order Item*; *Record Goods Receipt*; *Record Invoice Receipt*; *Clear Invoice*; *Remove Payment Block*. Except for the latter, we can immediately identify the DF1 provided in the textual description. Therefore, no more filters were applied, Table 3 summarises the number of activities identified in each data flow.

Rework Activities
Change Quantity
Change Price
Change Delivery Indicator
Change Storage Location
Change Currency
Change Payment Term

Table 2. Rework activities.

Later in this section, we show how we discovered the as-is process model of the DF1 starting from this highly filtered log.

Data Flow 2. The recordings for the DF2 are very similar to the ones of DF1. The DF2 counts up to 39 distinct activities, 13 of which are external (11 SRM and 2 Vendor activities). Of the remaining 26 activities, 6 are rework activities, and 9 occur with a frequency rate over 1%. The latter ones are the

[3] According to their activity name.

[4] A *NONE* user cannot be tracked.

Data Flow	Activities					
	Unique	**External**	**SRM**	**Vendor**	**Rework**	**Freq. > 1.0%**
DF1	38	15	12	3	5	5
DF2	39	13	11	2	6	9
DF3	11	2	0	2	1	4
DF4	15	0	0	0	6	7

Table 3. Actvities per data flow.

following: *Record Goods Receipt*; *Create Purchase Order Item*; *Record Invoice Receipt*; *Clear Invoice*; *Remove Payment Block*; *Create Purchase Requisition Item*; *Receive Order Confirmation*; *Change Quantity*; *Change Price*. Compared to the most frequent activities identified for the DF1, in DF2 we have almost double, some of which are not mentioned in the process description, e.g. *Create Purchase Requisition Item*.

Data Flow 3. The DF3 is the simplest among the four data flows. Its total number of distinct activities is only 11. There is no trace of SRM activities, but we find again the 2 Vendor activities. Removing the latter ones and focusing on the activities with frequency rate over 1%, we are left with the following: *Change Approval for Purchase Order*; *Create Purchase Order Item*; *Record Invoice Receipt*; *Clear Invoice*. Compared to the most frequent activities identified for the DF1 and DF2, in DF3 it is already very clear the full process behavior, which matches (almost) straightforward with the process description for the DF3.

Data Flow 4. The DF4 recordings contain up to 15 activities, 6 of which are rework, and none of them external (this was expectable, given that no invoice clearance is necessary in this data flow). The most frequent activities (freq. > 1%) are: *Record Goods Receipt*; *Create Purchase Order Item*; *Create Purchase Requisition Item*; *Change Quantity*; *Receive Order Confirmation*; *Change Delivery Indicator*; *Delete Purchase Order Item*.

Having identified external activities, rework activities, and most frequent activities, we further filtered the four event logs. First, we removed all the external activities: (i) we discarded the activities relating to the Vendor tasks, and (ii) we generated a fifth event log comprising only and all the SRM activities (we will use this latter to discover the SRM process, keeping it disjointed from the P2P process). Then, we filtered out from each of the four event logs the infrequent activities (absolute frequency less than 1%), and the rework activities (we postpone their analysis). These five event logs are the input for the next phase of our analysis, which is the discovery of the as-is process models. Table 4 briefly summarises the five (filtered) event logs key statistics.

Filtered	Cases			Duration (days)		
Event Logs	Total	Unique	Events	AVG	MAX	MIN
DF1	15,129	881	122,427	66.6	379.6	<1
DF2	220,810	1,539	960,838	70.8	365.0	<1
DF3	1,027	104	5,038	46.5	348.3	<1
DF4	14,498	37	33,923	23.4	209.7	<1
SRM	1,426	58	11,415	23.7	379.6	<1

Table 4. Filtered event logs statistics.

2.3 As-Is Process Models

To discover the as-is process models, we uploaded the five event logs to Apromore,[5] which allows us to visualise the event logs as process maps, to easily turn them into BPMN models applying Split Miner [2], and (if needed) to manually edit the discovered models for enhancing them.

Figures 2 to 6 show the BPMN models discovered on Apromore (using the Process Map Discovery Plugin and its embedded Split Miner [6]) from the four (filtered) event logs capturing the four data flows and the SRM event log containing only the SRM activities. We note that these process models highly match the four data flow descriptions, however, we need to mention a few remarks.

Remark-1. In the DF1 process model, the *Remove Payment Block* activity is unexpected, given that for the DF1 the *Record Invoice Receipt* activity is always following the *Record Goods Receipt* activity, the payment block should never be required. Also, if a payment block is applied, it is done in a transparent way, since the payment block removal is frequently observed in the event log, but not its setting (i.e. the activity *Set Payment Block* is rare).

Remark-2. In the DF2 process model, the *Record Invoice Receipt* activity is rarely observed before the *Record Goods Receipt* activity, despite it is allowed according to the process description. Often, an order confirmation is required for the PO items processed with the DF2, though we were not able to determine when or why. Finally, another highly frequent activity not mentioned in the DF2 description is: *Create Purchase Requisition Item*. The name of this latter, unfortunately, does not give us any hint in determining the purpose of the activity, but we believe it is worth mention its frequency (42564 occurrences, 4.17%), the company stakeholders may find it interesting or unusual.

Remark-3. The DF3 process model is the simplest out of the four, as well as the rarest type of data flow, with only 1027 cases (see also Table 4). The model clearly captures its description except for the activity *Change Approval for Purchase Order*, which occurs at least once in all the cases. This activity seems to be necessary (or even mandatory) for this type of data flow.

Remark-4. The DF4 process model presents some characteristics similar to the DF2 one, for example, the execution of the activities *Create Purchase Requisition Item* and *Receive Order Confirmation*. However, also in this case, we do not

[5] Apromore is the web-based process analytics platform maintained by the BPM research group of the University of Melbourne, more info at: apromore.org. Apromore is free and publicly available at: apromore.cis.unimelb.edu.au

[6] With parameters: 100(*activities*), 20(*arcs*), and 40(*parallelism*)

have enough information to explain their recurrent appearance. It is interesting to note that the DF4 is the only type of data flow where the deletion of the PO item occurs frequently (422 times out of 14,498).

Remark-5. Regarding the SRM process model, we have no information nor reference behavior to claim or reject its correctness. However, its activities names are self-explicative and can be easily understood. Accordingly, we believe the behavior captured in the SRM process model is either correct or very close to the correct one, being its control flow reasonable and having applied no filters on the SRM activities.

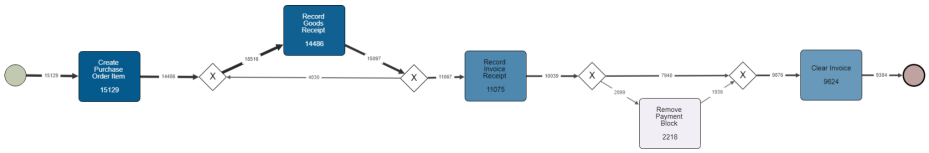

Fig. 2. As-is BPMN process model of the DF1, filtered infrequent behavior.

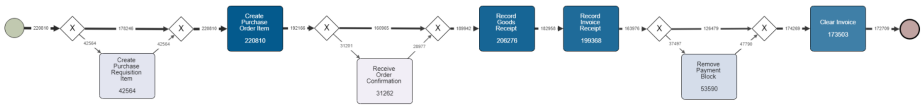

Fig. 3. As-is BPMN process model of the DF2, filtered infrequent behavior.

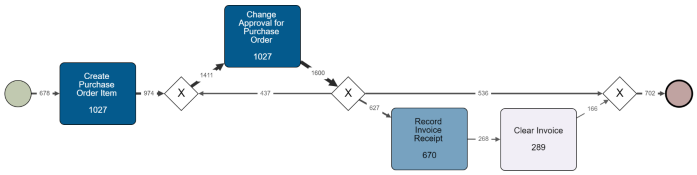

Fig. 4. As-is BPMN process model of the DF3, filtered infrequent behavior.

The filtered as-is process models are a good representation of the most frequent behavior observed in the four data flows. Exception made for the above remarks, the company should be relieved knowing that the majority of the times the workflow adheres to the prescribed one. Nevertheless, as we highlighted in the process analysis, several activities are observed in the original event log that generate noise (infrequent or deviant behavior). Given that the noise recorded in the event log may conceal interesting insights, which the company may find valuable, we tried to incorporate that information into the models showed in

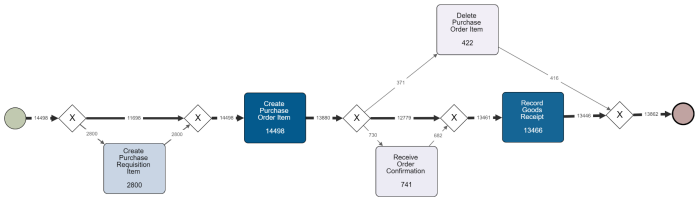

Fig. 5. As-is BPMN process model of the DF4, filtered infrequent behavior.

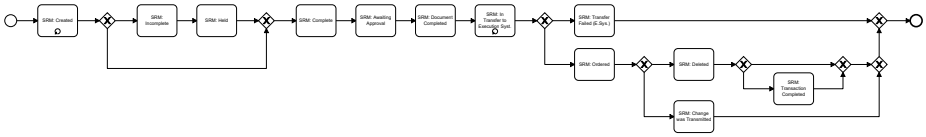

Fig. 6. The SRM process model.

Figures 2 to 5. To do so, we did the following. For each of the four data flows, we considered all the infrequent activities that were filtered out (i.e. activities with frequency $< 1\%$). For each infrequent activity, we analysed only the event log cases containing it, we detected its most frequent preceding activity and its most frequent succeeding activity, and we placed the infrequent activity between the former and the latter.

The process models we obtained after this procedure were then enhanced using the complete BPMN elements (to reduce complexity and increase understandability), as an example we reported in Figure 7 the DF2 complete and enhanced model (we suggest view on screen, for zooming in). It is easy to note that when trying to capture all the infrequent behavior recorded in the event log, the complexity of the process model explodes. This should not discourage the company stakeholders, but rather become the starting point for a deeper analysis of the P2P process behavior. Indeed, reaching this stage in our analysis allowed us to interpret further the process behavior and generate the to-be process models.

2.4 To-Be Process Models

Starting from the as-is process models (the complete and unfiltered versions), we decided to *redesign* them in what we think could/should be the actual process models. [7] The redesign did not follow a structured approach, instead, we integrated our knowledge and understanding of the process behavior into the models.

Figure 8 shows the root process, we assumed that a PO document is processed as a whole, whilst its line items are processed by subprocesses, each referring to a specific data flow (DF1-DF4). Furthermore, we assumed that any rework

[7] All the process models discovered in our analysis are available at: https://www.dropbox.com/s/j83l5m5y9gkiluo/bpic2019models.zip?dl=0

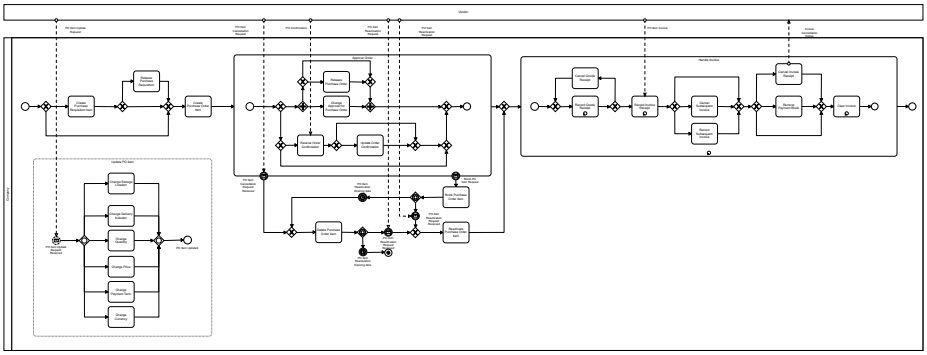

Fig. 7. As-is BPMN process model of the DF2, unfiltered and enhanced with complete BPMN elements.

activity can be required at any time and it must be triggered by an external message/notification, this is captured by the event subprocess in the root process.

The subprocesses modeling the behavior of the four data flows are shown in Figures 9 to 12. The behavior represented matches the reference behavior, but at the same time allows for the extra (infrequent) behavior identified in the original event log. Describing in details the behavior of the process models is out of the scope of this report, since the BPMN representation is straightforward and self-explicative. We recommend to visualise the models on screen, or to download the original PDF files and print them full scale on paper.

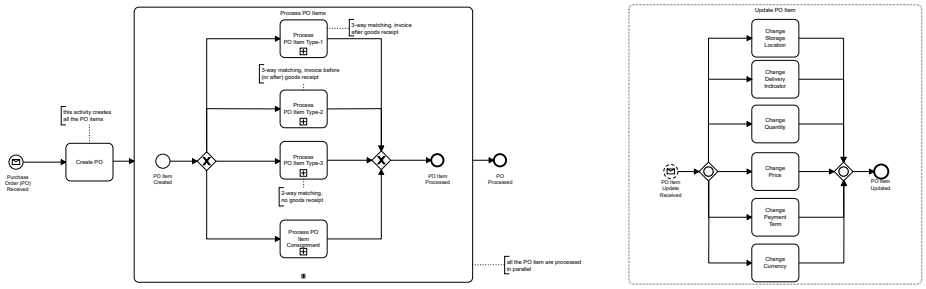

Fig. 8. Root process model.

3 Process Performance

In this section, we compute and analyse the throughput time and the throughput of the P2P process. First, we calculate the throughput time of the process, which we defined as the time between the main steps of the data flows (i.e. *Record Goods Receipt*, *Record Invoice Receipt* and *Clear Invoice*), and we propose a control-flow based technique to match these steps when recursively executed during a

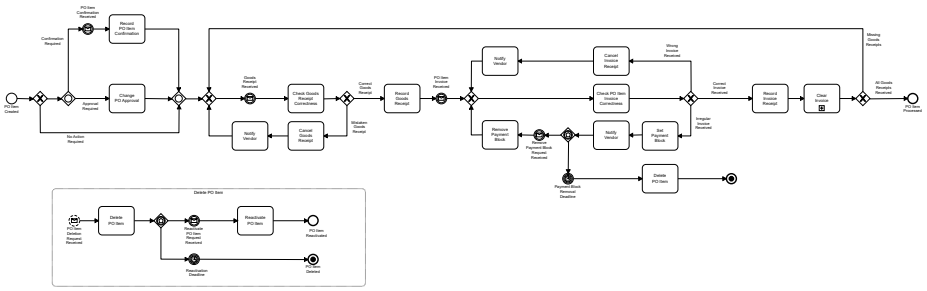

Fig. 9. To-be BPMN process model of the DF1.

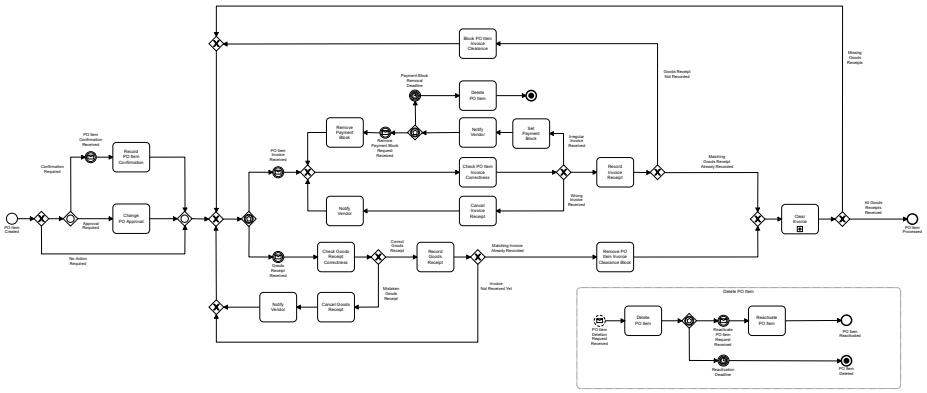

Fig. 10. To-be BPMN process model of the DF2.

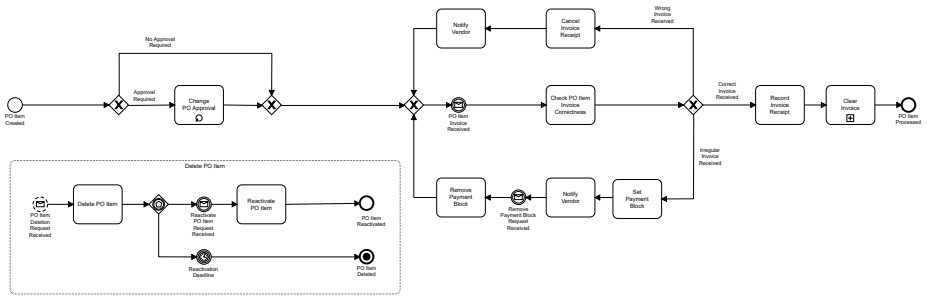

Fig. 11. To-be BPMN process model of the DF3.

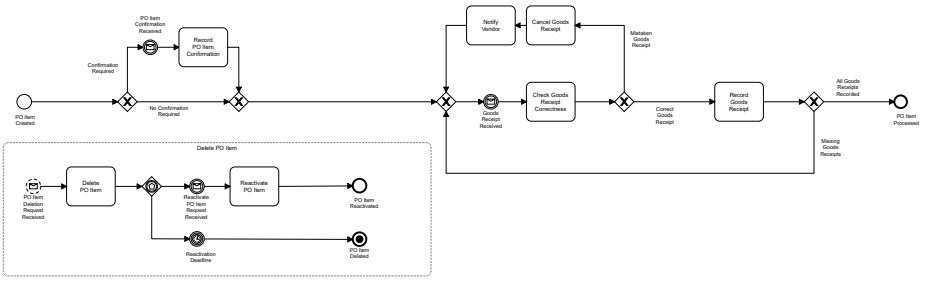

Fig. 12. To-be BPMN process model of the DF4.

PO item processing. Then, we calculate the throughput of the process in terms of net worth flow and payments rate (average number of payments per day).

3.1 Throughput time

In the context of our P2P process analysis, we define the *Throughput Time* as the amount of time required to handle single invoice, with reference to the times between the three main steps: *Record Goods Receipt*, *Record Invoice Receipt* and *Clear Invoice*.

We performed our throughput analysis using the four event logs described in the previous section (see Table 4), one per data flow, except for the one capturing case DF4. We held out this latter because it does not contains any invoicing information. The main challenge of computing the throughput times between the three main steps of the data flows, is the presence of recurring events representing multiple times the same steps for a given PO item. For example, given a PO item, we can observe in the event log multiple *Record Goods Receipt* activities, followed by multiple *Record Invoice Receipt* activities, followed by multiple *Clear Invoice* activities. Therefore, we asked the following question: can we match each of the *Record Goods Receipt* activity to the corresponding *Record Invoice Receipt*, and each of the latter ones to the matching *Clear Invoice* activity? Unfortunately, the information recorded in the attributes of the events recorded in the logs is not sufficient to efficiently cluster the three main steps into triplets. Therefore, we designed an ad-hoc matching technique to compute the throughput time.[8] Our technique is based on the First In First Out (FIFO) principle, meaning that the earliest invoice will be closed first, e.g. the earliest *Record Goods Receipt* activity will be matched with the earliest *Record Invoice Receipt* activity, which will be then matched with the earliest *Clear Invoice* activity.

Given a data flow event log, e.g. DF1, first, we create three vectors: idx_1, idx_2, and idx_3, representing (respectively) the observations of *Record Goods Receipt*, *Record Invoice Receipt* and *Clear Invoice*. Then, we scan all the cases in the event log searching for the aforementioned events and we save their positions in the corresponding vectors. For example for the case presented in Table 5 we

[8] The application and its sources are available at: https://github.com/volodymyrLeno/BPIC2019

will have $idx_1 = (1, 2, 7, 8, 9, 10, 16, 17, 18)$, $idx_2 = (3, 4, 6, 11, 12, 13, 14, 19)$ and $idx_3 = (5, 15, 20)$.

	Case ID	*Activity*	*Timestamp*
1	2000000015_00001	Record Goods Receipt	2018-01-25T11:16:00.000+10:00
2	2000000015_00001	Record Goods Receipt	2018-02-05T21:46:00.000+10:00
3	2000000015_00001	Record Invoice Receipt	2018-02-07T02:46:00.000+10:00
4	2000000015_00001	Record Invoice Receipt	2018-03-14T03:03:00.000+10:00
5	2000000015_00001	Clear Invoice	2018-03-23T17:24:00.000+10:00
6	2000000015_00001	Record Invoice Receipt	2018-03-23T17:26:00.000+10:00
7	2000000015_00001	Record Goods Receipt	2018-03-26T18:32:00.000+10:00
8	2000000015_00001	Record Goods Receipt	2018-03-26T18:34:00.000+10:00
9	2000000015_00001	Record Goods Receipt	2018-03-26T18:36:00.000+10:00
10	2000000015_00001	Record Goods Receipt	2018-03-26T18:39:00.000+10:00
11	2000000015_00001	Record Invoice Receipt	2018-03-26T19:01:00.000+10:00
12	2000000015_00001	Record Invoice Receipt	2018-03-27T01:05:00.000+10:00
13	2000000015_00001	Record Invoice Receipt	2018-03-27T01:06:00.000+10:00
14	2000000015_00001	Record Invoice Receipt	2018-03-27T02:28:00.000+10:00
15	2000000015_00001	Clear Invoice	2018-04-05T23:51:00.000+10:00
16	2000000015_00001	Record Goods Receipt	2018-04-30T02:26:00.000+10:00
17	2000000015_00001	Record Goods Receipt	2018-04-30T02:28:00.000+10:00
18	2000000015_00001	Record Goods Receipt	2018-04-30T02:30:00.000+10:00
19	2000000015_00001	Record Invoice Receipt	2018-04-30T23:19:00.000+10:00
20	2000000015_00001	Clear Invoice	2018-05-09T22:12:00.000+10:00
..	..	..	..

Table 5. First 20 events of case 2000000015_00001

Afterwards, we can match the events into corresponding triplets. This procedure differs for each data flow as we have to consider their unique requirements. For example, DF1 and DF2 require the execution fo the *Record Goods Receipt* activity, whilst DF3 does not; in DF2 the *Record Invoice Receipt* activity can occur before the *Record Goods Receipt* activity, whilst in DF1 the former always follows the latter.

In DF1, for each element $idx_{1i} \in idx_1$, we take the first element $idx_{2j} \in idx_2$ such that $idx_{2j} > idx_{1i}$, and the first element $idx_{3k} \in idx_3$ such that $idx_{3k} > idx_{2j}$. Together, idx_{1i}, idx_{2j}, and idx_{3k} create a triplet of activities representing the main steps of the P2P process. One *Record Goods Receipt* can be matched only with one *Record Invoice Receipt*, and one *Clear Invoice* activities. Thus, after matching the activities into triplet, their corresponding indexes are removed from the vectors. Continuing the example given in Table 5, we can identify three triplets: (1, 3, 5), (2, 4, 15) and (7, 11, 20). Note that, the other events representing the *Record Goods Receipt* and *Record Invoice Receipt* activities (e.g. events 6, 8, 9) cannot be associated with any *Clear Invoice* activity, meaning that they belong to *incomplete invoices*. In DF1, we consider an invoice incomplete when it is missing one of the three main steps. For the computation of the throughput time we consider only completed invoices.

In DF2, we identify the first activity in the case and then select the corresponding matching order: *Record Goods Receipt* → *Record Invoice Receipt* → *Clear Invoice* or *Record Invoice Receipt* → *Record Goods Receipt* → *Clear Invoice*. In the former case, the procedure is the same as for DF1. In the latter case, we match each element of idx_{2i} with the first element $idx_{1j} \in idx_1$ such that $idx_{1j} > idx_{2i}$, and then select the first element $idx_{3k} \in idx_3$ such that $idx_{3k} > idx_{1j}$.

In DF3, we use only the vectors idx_2 and idx_3, given that the *Record Goods Receipt* activity is not observed in DF3. For each element $idx_{2i} \in idx_2$ we select the first element $idx_{3k} \in idx_3$ such that $idx_{3k} > idx_{2i}$, and we match them as couples.

After obtaining the triplets (couples in the case of DF3), we can compute the time between *Record Goods Receipt* (GR), *Record Invoice Receipt* (IR) and *Clear Invoice* (CI) activities simply subtracting their timestamps:

$$time(GR, IR) = IR.timestamp - GR.timestamp \tag{1}$$

$$time(IR, CI) = CI.timestamp - IR.timestamp \tag{2}$$

$$time(GR, CI) = CI.timestamp - GR.timestamp \tag{3}$$

Figure 13 shows the throughput times between the GR and IR activities. Since DF3 does not contain GR activities, we held it out, and we report only the results for DF1 and DF2. As we can observe, in more than 50% of the cases, the throughput time between GR and IR does not exceed 10 days. Whilst, for 80% of the cases the throughput time is in the range 0 to 40 days. Table 6 reports the minimum, maximum, average, and median throughput times, which complement the histograms chart in Figure 13. Although, at a first glance, the throughput time distributions look very similar for DF1 and DF2, we note that (on average) the DF1 case is slower than the DF2 case to *Record Invoice Receipt* after a Goods Receipt is recorded (29.37 days vs. 17.47 days). On the other hand, the maximum throughput time between GR and IR is much greater in the case of the DF2, 325.06 days against the 226.85 days for DF1. The minimum throughput time seems less reliable, especially in the case of DF2 which is equal to 0. This was observed because in some cases (of the DF2) GR and IR are recorded in the event log with identical timestamps. We were not able, though, to understand whether this was a recording error, a compliance issue, or just a normal execution.

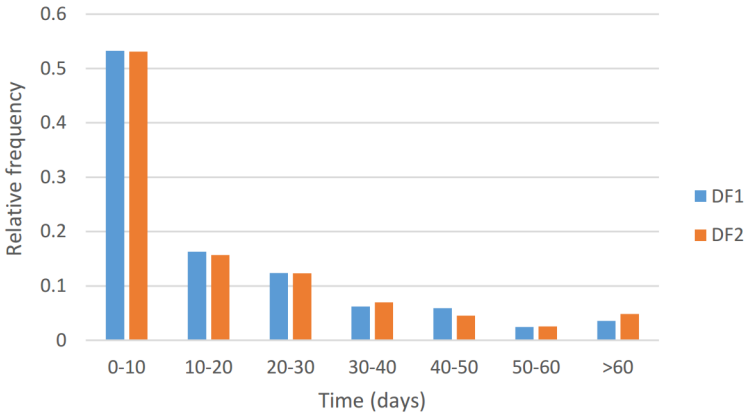

Fig. 13. Throughput times between *Record Goods Receipt* and *Record Invoice Receipt* activities, histograms chart.

	DF1	DF2
MIN	1 minute	0 minutes
MAX	226.85 days	325.06 days
AVG	29.37 days	17.47 days
MEDIAN	20.82 days	8.77 days

Table 6. Throughput times between *Record Goods Receipt* and *Record Invoice Receipt* activities, statistics.

Figure 14 and Table 7 summarises the throughput times between IR and CI. We notice that these throughput times increase substantially with respect to (w.r.t.) the throughput times between GR and IR. Indeed, more than 50% of the cases takes more than 40 days to *Clear Invoice* after the Invoice is recorded. If we consider the average throughput time (in the case of DF2), we can see that it more than doubles w.r.t. the average throughput time between GR and IR (17.47 days against 49.06 days, see Table 7). Even worse it is the case of the median throughput time which in the DF2 increases of 400%, and in the DF1 increases of 40%. On the other hand, the throughput times between IR and CI remain low for the DF3 case.

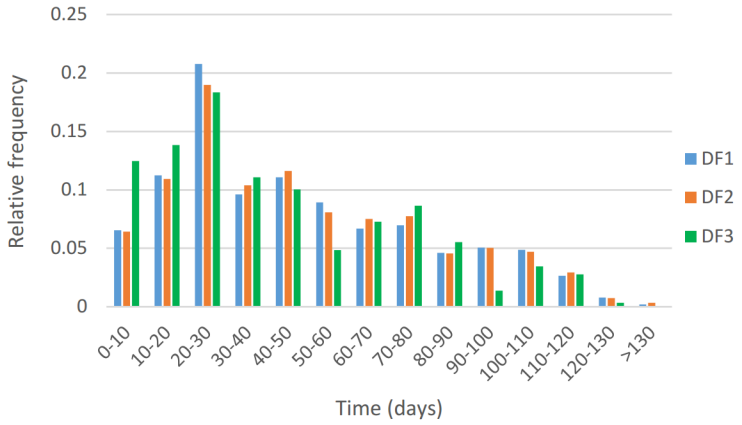

Fig. 14. Throughput time between *Record Invoice Receipt* and *Clear Invoice*, histograms chart.

	DF1	DF2	DF3
MIN	1 minute	0 minutes	2 minutes
MAX	317.46 days	341.94 days	202.1 days
AVG	37.61 days	49.06 days	10.04 days
MEDIAN	27.81 days	42.93 days	5.25 days

Table 7. Throughput time between *Record Invoice Receipt* and *Clear Invoice*, statistics.

Finally, Figure 15 and Table 8 report the throughput times between GR and CI, for the DF1 and the DF2 (not applicable to DF3). In contrast with the throughput times between GR and IR, and IR and CI, in this case both the DF1 and the DF2 throughput times are in line, with average close to 66 days.

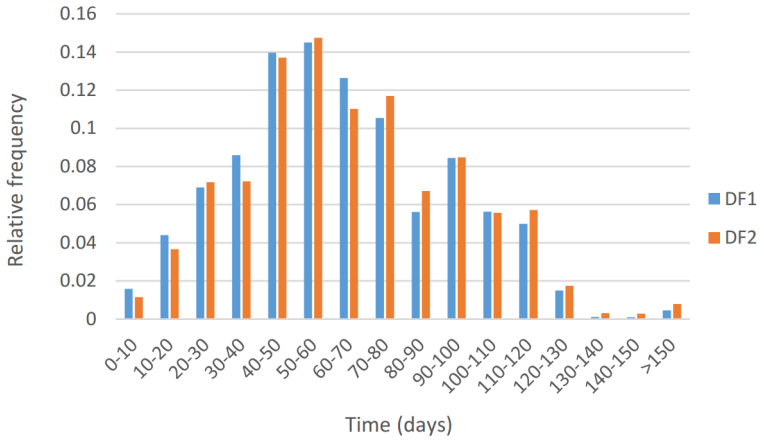

Fig. 15. Throughput time between *Record Goods Receipt* and *Clear Invoice*, histograms chart.

	DF1	DF2
MIN	2.13 hours	2.2 hours
MAX	330.51 days	345.22 days
AVG	66.99 days	65.98 days
MEDIAN	63.16 days	63.01 days

Table 8. Throughput time between *Record Goods Receipt* and *Clear Invoice*, statistics.

Data Flow	Cleared Invoices	Payments per day (on average)	Total net worth (in millions)	Avg. net worth per day (in millions)
DF1	13150	34.55	1243.623	3.267
DF2	180156	473.74	624.145	1.641
DF3	291	0.78	2.022	0.005
Total	193597	508.575	1869.79	4.912

Table 9. Throughput statistics

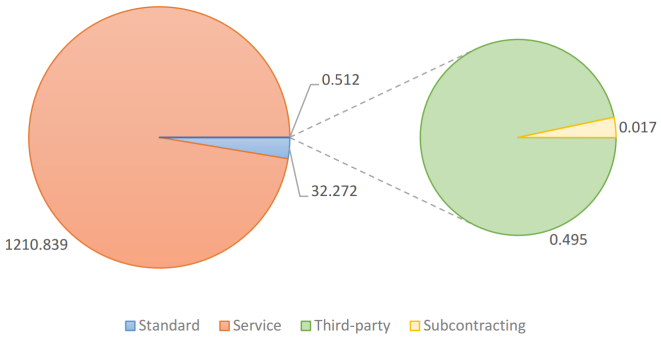

Fig. 16. Net worth per item type for DF1 (in millions).

3.2 Throughput

We define the *throughput* of the P2P process as the average amount of payments (i.e. invoices cleared) performed per day. Consequently, to compute the throughput, we consider only the cases where the *Clear Invoice* activity is observed. Also, we report on the total net worth, and average net worth per day, to compute the net worth we referred to the attribute *Cumulative Net Worth* of the *Clear Invoice* events recorded in the event log. The results are shown in Table 9. Although the number of payments performed in DF2 is the highest, their net worth is less than the net worth of the payments performed in DF1 (being this latter almost double the net worth of DF2).

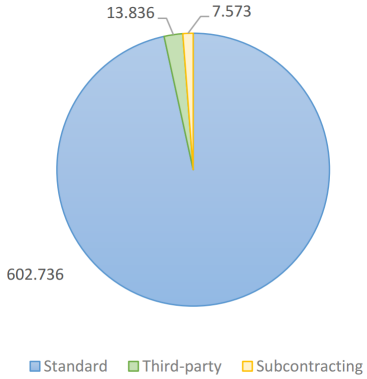

Fig. 17. Net worth per item type for DF2 (in millions).

On average there are around 508 payments performed every day across all data flows. The total net worth is equal to 1.8 billions euros, 66.5% of which is coming from DF1 and 33.4% from DF2 correspondingly. DF3 accounts only for the 0.1% of the total net worth. Furthermore, we broke down the net worth distribution into the different items types. Figure 16 and 17 show the net worth distributions for Df1 and DF2, resp. In DF1 most of the net worth is allocated to the *Service* item type, accounting for 97% of the total. By contrast, DF2 does not handle any payment for *Service* item type, and the most dominant item type is *Standard* with net worth over 602 millions (around 96% of total net worth of DF2). In DF1, the *Standard* item type covers only about 32 millions of net worth, around 2% of the total net worth. In both DF1 and DF2, the *Third Party* item type is characterized by higher net worth than *Subcontracting* (almost 30 times higher in DF1 and 2 times higher in DF2). In DF3, all the payments involve only one item type which is *Limit*.

4 Process Compliance

In this section, we analyse the compliance of the processes w.r.t different objectives, i.e. completeness of cases, exceptional cases, cases with compliance issues and cases with re-work. We exclude data flow DF4 from our compliance analysis, i.e. items of the consignment type, since this process is handled externally and activities from the external company are not recorded in the event log. Thus, an analysis will not provide inside into company internal compliance issues. For the analysis we then consider the BPI challenge log 2019 with a time-frame filter from 31st Dec 2017 - 18 January 2019, as discussed in section 2.2, and an attribute filter removing cases with item type 'consignment. This filtered log maintains 94% of the cases and 97% of all events. For finding the compliance issues, we applied the tool Disco provided by the company Fluxicon.

As a first step, we extract a set of incomplete cases from the event log and categorize them according to different characteristics. For the following analysis, we then remove this set of incomplete cases from the event log. As a next step, we define criteria for classifying cases as exceptional and investigate some example cases. We divided the analysis of compliance issues into two categories: compliance issues resulting from (1) high level control flow issues and from (2) mismatching values of invoice receipts. Finally, we will identify the customers of the process that cause the most rework.

4.1 Incomplete cases

In this section, we consider a case to be incomplete, if it does not start or end with specified start or end activities. For the BPIC19 log, we consider 'Create Purchase Order Item' or 'Create Purchase Order requisition item' and 'Vendor creates invoice' to be relevant starting activities and 'Clear invoice' to be the only relevant end activity of a given case. Both activities 'Create Purchase Order requisition item' and 'Clear invoice' are mentioned in the high-level description for all data flows one to three as necessary activities, i.e. the goods receipts and invoice values need to be matched against the value at the creation of an item and only if these values match, the payment is issued. During the analysis of the data, we identified that sometimes activity 'Create Purchase Order Item' is preceded by either activity 'Create Purchase Order requisition item' or 'Vendor creates invoice' and thus we included this activity as starting activities as well. To filter the BPIC19 log to only maintain complete traces, we apply an endpoint filter with the option to discard traces that have not one of the aforementioned starting or end activities. As a result, we retrieve an Event log that maintains 71% of the cases and 72% of the events. In the following, we will investigate the 23% of the cases that were filtered in this step and try to categorize them according to different criteria.

To maintain only the set of incomplete cases, we apply the endpoint filter with the option to discard cases that have 'Clear Invoice' as an end event. As a result we retrieve an event log with 22% of the cases. We noticed several cases with deletions and cancellations that should be separated from the set of open cases. For that purpose, we used the Attribute filter that requires the all activities with 'Cancel' or 'Delete' in their activity names as mandatory. 4% of

all cases were deleted or cancelled after their creation. When we change this filter from the option mandatory to forbidden, we can receive an event log of currently open cases (18%) that are not cancelled or deleted. As there are strict payment deadlines for invoices, usually within one moth, we want to apply a Timeframe filter to divide the set of open cases into cases younger (8%) and cases older than one month (9%). For open cases younger than one month, no action is usually required and the cases can just proceed naturally. For cases that are older than one month usually a follow up or an action is required. For that purpose, we distinguish the set of older cases into cases that never received an invoice (4%) and cases that received and invoice but no payment (5%). Since a relatively high amount of cases is open longer than one month, the data provider could consider to implement payment reminders for cases that received invoices or send follow-up requests for invoices to the vendors for cases without invoices.

Table 10 summarizes the filter options applied and table 11 summarizes the resulting event log sizes. Moving forward, we will focus the analysis of compliance issues on the event log of complete cases, i.e. after applying filter (1).

4.2 Exceptional Purchase Order Documents

We deem a PO document to stand out from the event log, if it has a high number of events, i.e. is in the top 1% of cases when sorting the cases according to their number of events. Alternatively, it is possible to use case durations for the identification of exceptional cases. In this section, however, we focus more on the analysis of compliance issues and a high number of events is usually an indicator for compliance issues. To identify exceptional cases, we apply a performance filter by the number of events. We identify the cut-off for the 1% of cases with the highest number of events for cases that have a minimum number of 20 events. The 1% of cases cover 5% of the events of the BPIC19 log. Fig. 19 shows the top 5 exceptional cases. These cases have more than 300 events and some cases have a duration of over one year. We also discovered a high-level process map with the tool Apromore for 50% of the activities and 5% of the arcs. It is noticeable that these cases contain a lot of rework, i.e. several 'Change' activities and also cancelled goods receipts or invoices. These cases can be used for further analysis with domain knowledge to identify causes for these rework activities.

Case ID	▲ Events	Variant	Started	Finished	Duration
4507000436_00010	513	Variant 69	03.01.2018 01:33:00	11.01.2019 01:45:00	1 year, 8 days
4507000449_00060	500	Variant 73	03.01.2018 02:20:00	18.01.2019 00:41:00	1 year, 14 days
4507013395_00001	458	Variant 348	02.03.2018 08:45:00	07.06.2018 21:13:00	97 days, 13 hours
4507000684_00010	399	Variant 78	04.01.2018 00:23:00	18.01.2019 00:41:00	1 year, 14 days
4507036789_00001	345	Variant 676	15.06.2018 06:44:00	06.09.2018 22:09:00	83 days, 15 hours

Fig. 18. Top 5 exceptional cases.

Filter ID	Description	Filter type	Options	Filter parameters
(1)	Filter for complete cases	Endpoints Filter	Discard cases	**Start event values:** 'Create Purchase Requisition Item', 'Create Purchase Order Item', 'Vendor creates invoice' **End event values:** 'Clear Invoice'
(2)	Filter for incomplete cases	Endpoints Filter	Discard cases	**Start event values:** any **End event values:** any besides 'Clear Invoice'
(3)	Filter for cancelled or deleted cases	Attribute Filter	Mandatory	**Event values:** 'Cancel Goods Receipt' 'Cancel Invoice Receipt' 'Cancel Subsequent Invoice' 'Delete Purchase Order Item'
(4)	Filter for open cases	Attribute Filter	Forbidden	**Event values:** 'Cancel Goods Receipt' 'Cancel Invoice Receipt' 'Cancel Subsequent Invoice' 'Delete Purchase Order Item'
(5)	Filter for cases with invoices	Attribute Filter	Mandatory	**Event values:** 'Record Invoice Receipt'
(6)	Filter for cases without invoices	Attribute Filter	Forbidden	**Event values:** 'Record Invoice Receipt'
(7)	Filter for cases before 18th of Dec.18	Timeframe Filter	Keep cases: Completed in Timeframe	**Start value:** 31/12/17 **End value:** 18/12/18
(8)	Filter for cases after 18th of Dec.18	Timeframe Filter	Keep cases: Completed in Timeframe	**Start value:** 18/12/18 **End value:** 12/01/1

Table 10. Applied Filters for identifying and classifying incomplete cases

Category	Filters	%Cases	%Events
Complete cases	(1)	71%	72%
Incomplete cases	(2)	22%	24%
Cancelled or deleted cases	(2) & (3)	4%	3%
Open cases younger than one month	(2) & (4) & (8)	8%	6%
Open cases without invoice older than one month	(2) & (4) & (6) & (7)	4%	9%
Open cases with invoice older than one month	(2) & (4) & (5) & (7)	5%	4%

Table 11. Event log sizes after filtering for incomplete cases

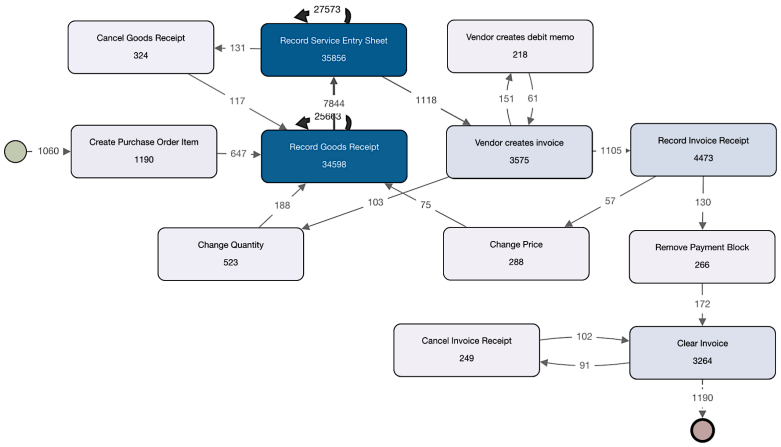

Fig. 19. Exceptional cases process map discovered with the tool Apromore.

4.3 Compliance issues from a control flow perspective

Typically, compliance issues from a control flow perspective when comparing event logs with process models can be assessed with techniques such as trace alignments [1]. The process models from section 2, however, support a wide range of BPMN elements that are not currently supported by these trace comparison techniques, i.e. events, sub-processes or message flows. Since removing these elements of the process models would not properly represent the processes developed in section 2 any more, we decided to not apply common conformance checking techniques and rather fall back to high level analysis of compliance issues with filters.

At first, we checked high level constraints of data flows one and two given in the description of the challenge. In particular, in DF1 an invoice can only be cleared, if an invoice is received after the goods are received. Hence, if an invoice would be received for this item category before the goods are received, this case would cause a compliance issue. We tested for these cases by applying an attribute filter for the corresponding item category and a follower filter, i.e. filtering for cases where activity 'Record Invoice Receipt' is eventually followed by activity 'Record Goods Receipt'. When investigating the resulting 306 cases, however, we realized that most cases just resolve around several goods and invoice receipts. When filtering these cases with another follower filter, i.e. 'Record Goods Receipt' is never eventually followed by 'Record Goods Receipt', no cases remained, i.e. this compliance rule is never violated.

Activity	Resource
Create Purchase Order Item	user_038
Vendor creates invoice	NONE
Record Invoice Receipt	batch_00
Remove Payment Block	user_006
Record Goods Receipt	user_029
Clear Invoice	user_005

Fig. 20. Example Case for preemptive removal of payment block (ID 4507000254_00010).

Another possible compliance issue is described in DF2: If an invoice is received before the goods receipt, then an automatic payment block needs to be removed before the invoice can be cleared. It would be a compliance issue,

Activity	Resource
Create Purchase Requisition Item	user_292
Create Purchase Order Item	user_136
Record Invoice Receipt	user_012
Vendor creates invoice	NONE
Record Goods Receipt	user_136
Clear Invoice	user_015

Fig. 21. Example Case of missing activity 'Remove Payment Block' (ID 4508069895_00010).

if the activity 'Remove Payment Block' is missing. We check for this compliance issue by applying an attribute filter for item category of DF2, a follower filter, where activity 'Record Invoice Receipt' is eventually followed by activity 'Record Goods Receipt', and another follower filter, where activity 'Record Goods Receipt' is never eventually followed by 'Remove Payment Block'. In addition, we apply an additional follower filter to remove cases with duplicate invoices as those cases interfered with the analysis.

As a result, we found 291 cases with compliance issues. When investigating these cases, we found that activity 'Remove Payment Block' was sometimes executed before the goods were received, which can also be observed in Fig. 20. We can filter for these cases with another follower filter, i.e. where 'Remove Payment Block' was eventually followed by activity 'Record Goods Receipt'. Cases with pre-emptive removal of the payment block make up 263 out of 291 cases. The remaining 28 cases, where activity 'Remove Payment Block' was missing, can be identified by choosing the option 'never eventually followed' for the last mentioned filter. Fig. 21 shows a sample case, where activity 'Remove Payment Block' is missing.

Activity	Resource
Create Purchase Order Item	user_142
Vendor creates invoice	NONE
Record Goods Receipt	user_080
Vendor creates invoice	NONE
Record Invoice Receipt	user_007
Clear Invoice	user_005
Clear Invoice	user_005

Fig. 22. Example Case of duplicate activity 'Clear Invoice' (ID 4507002162_00010).

We also found other compliance issues: When applying a follower filter, when 'Clear Invoice' is directly followed by itself, we can find cases where an invoice was paid twice. We again filter out multiple invoices since two cleared invoices could also be correct for cases with two invoice receipts. As a result, we identified 626 cases of duplicate payments. Fig. 22 shows an example case with a duplicate 'Clear Invoice' activity for a single invoice.

Activity	Resource
Create Purchase Order Item	user_103
Cancel Goods Receipt	user_104
Vendor creates debit memo	NONE
Cancel Invoice Receipt	batch_00
Clear Invoice	user_111

Fig. 23. Example Case of a payment without a valid invoice (ID 4507002401_00010).

Another possible compliance issue revolves around cancelled invoices that are still cleared. We can identify these cases with a follower filter, where activity 'Cancel Invoice Receipt' is directly followed by the activity 'Clear invoice'. In a total of 2,492 cases the invoice was cleared directly after a cancel event occurred. We identified two different kinds of cases with different problems: Fig. 23 shows a sample case, where a cancelled invoice is cleared without ever receiving a valid invoice are goods receipt. We can filter for these cases by applying another follower filter, where 'Clear Invoice' is never eventually followed by another 'Record Invoice Receipt' event. 769 out of 2,492 cases represent this case. The other kind of cases eventually receives a correct invoice that is cleared later on, for example shown in Fig. 24. This kind

Activity	Resource
Create Purchase Order Item	user_000
Vendor creates invoice	NONE
Vendor creates debit memo	NONE
Record Goods Receipt	user_000
Record Invoice Receipt	user_007
Cancel Invoice Receipt	user_006
Clear Invoice	user_006
Record Invoice Receipt	user_006
Clear Invoice	user_002

Fig. 24. Example Case of receiving a correct invoice after clearing a cancelled invoice (ID 2000000185_00001).

of cases make up the remaining 1,723 out of 2,492 cases. The second type of wrongly cleared invoice is less severe since the error is later corrected in the case. However, it is possibly beneficial to implement security procedures to prevent clearing invoices right after an invoice cancellation.

4.4 Compliance issues regarding wrong invoice values

Compliance issues resulting from wrong invoice values can be identified with a Follower filter. Particularly, we filter for cases, where one of the events 'Create Purchase Order Item', 'Record Goods Receipt' or 'Record Invoice Receipt' are eventually followed by one of the events 'Record Goods Receipt', 'Record Invoice Receipt' or 'Clear Invoice' and that require a different value for the cumulative net worth of the invoice. As a result, we retrieve 1,529 cases (<1%), where invoice values are wrong. We further investigated different case variants, when mismatching invoice values occurred. The most common case is depicted in Fig. 25, where the values of several goods receipts are summed up in one invoice receipt. The invoice for these cases, however, is then cleared for the value of a single goods receipt, which either indicates a wrong invoice value or a wrong payment amount.

Activity	Resource	Cumulative net worth (EUR)	User
Create Purchase Order Item	batch_06	595.0	batch_06
Record Goods Receipt	batch_06	595.0	batch_06
Record Goods Receipt	batch_06	595.0	batch_06
Record Goods Receipt	batch_06	595.0	batch_06
Record Goods Receipt	batch_06	595.0	batch_06
Record Service Entry Sheet	NONE	595.0	NONE
Record Service Entry Sheet	NONE	595.0	NONE
Record Service Entry Sheet	NONE	595.0	NONE
Record Service Entry Sheet	NONE	595.0	NONE
Vendor creates invoice	NONE	595.0	NONE
Record Invoice Receipt	user_007	2382.0	user_007
Clear Invoice	user_002	595.0	user_002

Fig. 25. Example Case for multiple goods receipts with a summary invoice (ID 4507001872_00001) .

Another common case can be found in Fig. 26, where the goods receipt value is exactly twice the amount of the corresponding invoice and payment of the case.

Activity	Resource	Cumulative net worth (EUR)	User
Create Purchase Order Item	user_036	88.0	user_036
Vendor creates invoice	NONE	88.0	NONE
Record Goods Receipt	user_029	177.0	user_029
Record Invoice Receipt	user_024	88.0	user_024
Clear Invoice	user_002	88.0	user_002

Fig. 26. Example Case for wrong goods receipt (ID 4507000855_00080) .

One last common case with wrong invoice values is shown in Fig. 27, where multiple goods receipts show values unrelated to the PO item during creation of the case. The invoice then similar to the case in Fig. 25 sums up the goods receipt values, but assumes that the goods receipt values are equal to the value during creation of the PO. The payment, however, is then again equal to the amount during the creation of the invoice. While the amount of compliance issues from wrong invoice values is low, further analysis should be conducted with more domain knowledge.

Activity	Resource	Cumulative net worth (EUR)	User
Create Purchase Order Item	batch_06	185.0	batch_06
Record Goods Receipt	batch_06	1295.0	batch_06
Record Goods Receipt	batch_06	9067.0	batch_06
Record Service Entry Sheet	NONE	185.0	NONE
Record Service Entry Sheet	NONE	185.0	NONE
Vendor creates invoice	NONE	185.0	NONE
Record Invoice Receipt	user_008	370.0	user_008
Clear Invoice	user_002	185.0	user_002

Fig. 27. Example Case for unrelated multiple goods receipt values (ID 4507001538_00001) .

4.5 Identifying Customers with the most rework cases

One question of the challenge was to identify the customers that cause the most amount of rework. From a compliance perspective, we consider a case to cause rework, if it contains at least one of the rework activities from Table 2. We can filter for all cases with rework by applying an attribute filter with the option to maintain only cases that contain at least one of the six mandatory rework activities. 8% of the cases contain rework activities. We consider the vendors to be the customers of the process. Thus, we can identify the customers causing the most amount of rework by using the statistics view of Disco for the attribute 'Vendor'. Fig. 28 shows the summary statistics after applying the filter and especially the top 5 vendors with the highest amounts of frequencies. The amount of rework cases of the top 5 vendors (>3000 cases) is rather significant considering an average of only 162 cases per vendor with a standard deviation of 559 cases.

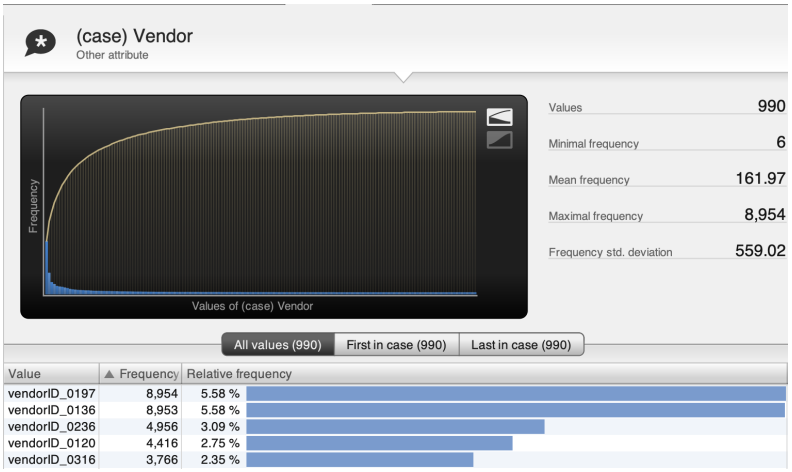

Fig. 28. Top 5 vendors with rework cases.

5 Conclusion

In this paper, we analysed the event log of a Purchase-to-Pay (P2P) process from a large multinational company operating from The Netherlands, provided for the BPI Challenge 2019. Our aim was threefold; (i) to derive a collection of process models explaining the P2P process recorded in the even log; (ii) to analyse the throughput of the P2P process; and (iii) to identify compliance issues present in the event log that conflict with the high-level control-flow descriptions.

First, we analysed the event log and the activities recorded within. We identified four different data flows, and discovered the respective as-is BPMN process models. From our understanding of the data, we proposed a set of to-be BPMN process models that capture not only the reference behavior but integrate also the extra behavior observed in the event log.

Then, we developed a technique to match each *Goods Receipt* with the corresponding *Invoice*, and this latter to its *Clearance* event. Based on these matchings, we estimated the throughput times and the throughput in terms of total invoices cleared per day and total net worth. We learned that the throughput times between *Goods Receipt* and *Invoice Receipt* in more than 50% of the cases did not exceed 10 days. Whilst the throughput times between *Invoice Receipt* and *Clear Invoice* exceed most of the times the 40 days. Regarding the throughput, we found that 508 payments are processed with an average net worth of 4.9 millions per day.

Lastly, we analysed compliance issues with respect to the high-level process control flows descriptions. We provided a categorization of incomplete cases and recommended to implement follow-ups for open cases that go beyond the usual payment deadlines of invoices. For the set of completed cases, we gave a more detailed analysis of compliance issues by finding a set of exceptional cases and by identifying the top five customers causing the most amount of rework. When analysing compliance with regards to the control flow, we found cases with double payments, payments for previously cancelled invoices and wrongly removed payment blocks for invoices. Finally, we identified a small set of cases with wrong invoice values and gave some characteristics of the problematic cases.

References

1. A. Adriansyah, J. Munoz-Gama, J. Carmona, B. van Dongen, and W. van der Aalst. Alignment based precision checking. In *BPM*. Springer, 2012.
2. A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy. Split miner: automated discovery of accurate and simple business process models from event logs. *KAIS*, 2018.
3. W. van der Aalst. *Process Mining - Data Science in Action*. Springer, 2016.
4. B.F. van Dongen. *Dataset BPI Challenge 2019. 4TU.Centre for Research Data.* https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1.