Automate does not always mean Optimize
Case Study at Logistics Company

Jan Suchy, Milan Suchy, Michal Rosik, Agnes Valkova

GRADIENT ECM, Kosicka 56,
82108 Bratislava, Slovakia
{Jan.Suchy, Milan.Suchy, Michal.Rosik, Agnes.Valkova}@gradientecm.com

Abstract. Dynamic growth of digitized information creates space for systematic collection of data related to business processes. Extraction of this data is an enormous challenge because many systems exist, which store data in many different formats. With the use of advanced analytics techniques, it is possible to present collected data in an "as-is" view of processes and find bottlenecks, loops, delays or deadlocks. In this paper, we present the methodology to extract business-related events from given processes of a logistic company. The logistic company has over the years fully automated their Purchase Order and Invoice Approval processes driven by BPM system. Logistics is always about optimization and cost reduction. The company asked us, whether it was possible to optimize processes furthermore. We analyzed the BPM system and deployed processes to develop connector for event data extraction. Process mining techniques were used to reconstruct processes from event logs. As a result, we identified bottlenecks, over allocated employees, and suppliers' characteristics.

Keywords: BPM system, Event log, Process Mining, Performance Measurement of Processes, Process Analysis

1 Introduction

In today's day and age, most industries automate their processes via workflow systems [1]. Efforts to capture and automate the desired behavior in industry processes can in one aspect bring many benefits, and on the other hand, may hide all the ineffective behaviors and instances when automating. Thus, when automating processes, especially those that bring many gains, it is vital to monitor these processes to see what truly takes place. With the following analysis of behaviors that takes place, it is possible to enhance and effectively change business needs and eliminate risks.

To capture current conditions of processes can be quite the complicated and complex matter. What would be needed would be a team of people that will carefully monitor all resource processes for each activity and record individual steps that provide solutions to their problem areas. The team will be able to create a visible and interactive process model for dedicated time frames for the people being tracked pertaining to that dedicated activity. The Process Model can be represented in a graph-based modeling language such as, Petri nets [2], BPMN [3], or YAWL [4]. As a result, the model carried

out holds all subjective views by the process analysts whom contributed. On the other hand, modern technology is available that can reconstruct and visualize processes with an objective view, in a fraction of the time. This technology is known as Process Mining. To reconstruct a process using Process Mining technology, it is important to acquire data recorded regarding all processes and transform them into structures needed for the reconstruction process to occur. With Process Mining, it is possible to reconstruct processes rapidly faster and to see the "as-is" reality of a process. In turn, a common and objective view is seen for any process. Process Mining allows you to discover what truly goes on in your organization, a clear reality check, that way you can see any flaws or inefficiencies needed to be worked out to enhance your processes and the overall outcome.

Our goal was to show a precise reality for what really goes on in the Purchase Order and Invoice Approval processes. At first, we become familiar with the specifications of the processes given by the company. We then define the structure in which we recorded the data. After familiarizing ourselves with the architecture of the BPM system, a connector was developed, which extracts raw data and creates structured event logs. Event logs are then imported into the Process Mining tool, Minit. Basic statistics and their characteristics of processes are then introduced.

Lastly, we focused on key analytical human activities, resources, and suppliers that take place. After seeing this information, gathered knowledge and optimization points were made for both processes.

2 Process definition

To analyze and discover optimizations, we were provided Purchase and Orders Invoice Approval processes. Processes are implemented in a process-driven application and driven by a workflow engine. The Process Owner provides us BPM models and specifications of certain processes through which we have become familiar with the individual steps and process attributes. The main objective in this part of work, was to identify the flow of processes so that we could validate extracted process models.

2.1 Purchase Order process

The process describes the creation and approval authority in cases where orders are made. The system provides the users the ability to create a new order in the form of editable structured forms. Thereby, allowing the user to fill any number of items ordered. Ordered items are defined by a set of attributes. After having created a new order, it is automatically launched into the Purchase Order process. The system then selects, based on the entered data in a tree of authorized users/group of the lowest in authority levels to approve the Purchase Order. Subsequently, the system assigned the approver/approvers tasks by informing them by email notifications. The approver may then approve orders or reject them. After the approval process, the system can decide, based on the financially authorized limit, if the level is sufficient enough to approve of the order made, and if not, there will be a reselection process from the existing tree of

authoritative users/group of higher authorities. This process is repeated until all authorized approvals are met. In the instance that the order process is deemed approved in the system, the author will be notified of the outcome or status change to "Approved". If an order has been rejected, the system will change the status to "Declined" and a notification will be sent out to the appropriate author as well as the reasons for refusal. After a rejection of a process order takes please, the order ends. With approved orders, the system identifies whether it is a cyclic order[1]. If this is not a cyclic order, it continues and assigns tasks to those related to the ordering process, which has been designated by the author of orders during its formation. In this part of the process, the user will have established and forwarded approved orders with its suppliers. The process thereby continues, to confirm receipts of the ordered goods/services. The system will then assign tasks to those employees whom were authorized throughout the formation of the order process. These employees are then given a chance to confirm a completed delivery order, confirm any partial deliveries, or cancel the order in the meantime while a customer had cancelled the order. When a partial delivery takes place, the employee can wait for a delivery of any missing parts of the ordered goods, or they can declare the order as a partially delivered. Subsequently, the system will treat the order as a "Closed" order and the process will come to an end.

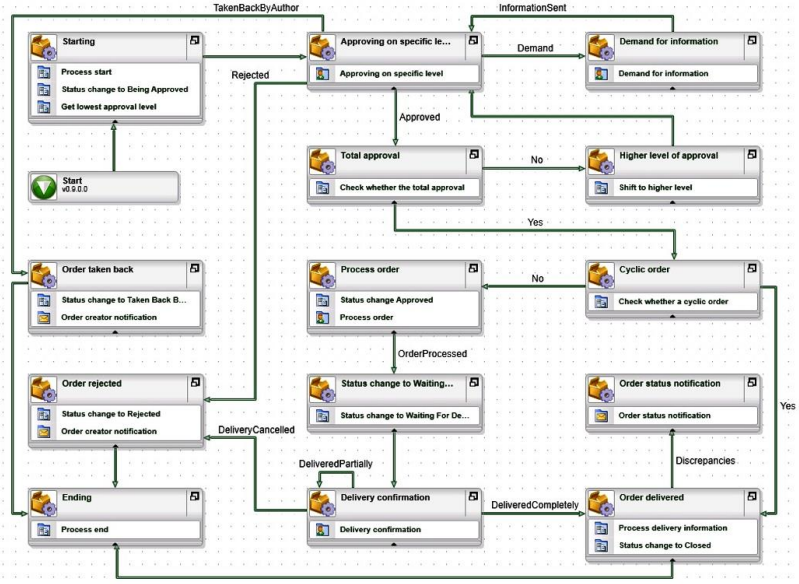

Fig. 1. Process diagram of the Purchase Order process.

[1] If the process is identified as a cyclic order, the system will automatically declare this order as a delivered and the process will come to an end. Cyclic orders are characterized by regular repetition.

Figure 1 indicates the BPMN process diagram of the Purchase Order process. The Process creates 4 human tasks and 16 system tasks.

2.2 Invoice Approval process

The process is automatically prompted, as soon as an invoice is scanned/digitized. In the beginning, the System automatically assigns invoices to the correct order. If the order was not found, to be assigned, the system will generate a task for manual entry of the order. This task is assigned to a group known as "Accountants" whereby, their goal is to find the order and pair it together with its invoice. After assigning the order, the system automatically compares the sum total amounts. If the sum amounts do not match, the system will check the correctness of the cost center and continues to process the invoice in the approval process.

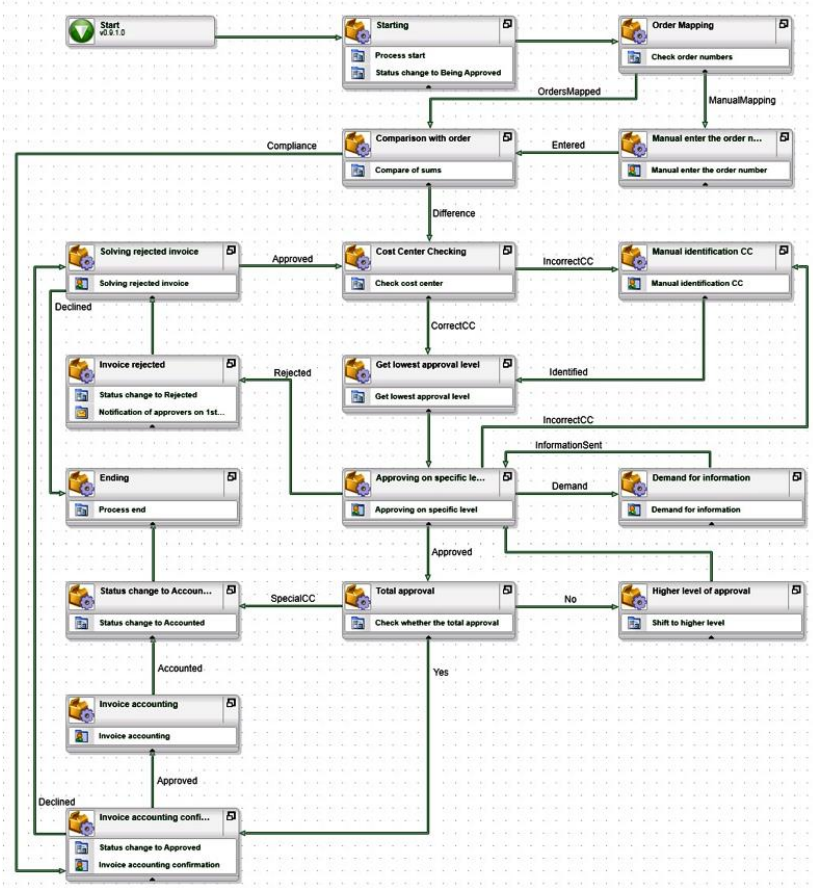

Fig. 2. Process diagram of the Invoice Approval process.

The approval process has the same procedures as the Purchase Order Process. Subsequently, the system will generate tasks to confirm the accounted invoice. In this part of the process, the employee will check all other information pertaining to the invoice. If they did not find any discrepancies in the invoice, the will generate tasks to complete the accounting part/accounts payable process of the invoice, whereby the process will come to an end. However, if an employee finds irregularities, the process will continue on and the system will generate tasks to solve any discrepancies. At this point, the employee may reject the invoice and the process ends, or, obtaining any missing data will then enable the system to return the invoice to a re-approval state. Figure 2 indicates the BPMN process diagram of the Invoice Approval process. The process creates 7 human tasks and 13 system tasks.

3 Event log

In order for us to analyze the previously described processes and see what takes place within them, we need to extract data from databases, as well as, structure them. Data extraction is a general challenge because data may be located not only within the database, but may also be found in different formats of data sources (e.g., message logs, flat files, transaction logs, document management systems, ERP systems, etc.). Our main objective is to analyze the data obtained from a process-oriented perspective. In this section, we discuss information that should be present in such event logs.

Table 1. Fragment of the event log.

Case ID	Event ID	Activity	Start timestamp	End timestamp	Event Type	Resource
1	45678	Order delivered-	26.11.2014 12:51	26.11.2014 12:51	1	System
1	45679	Closed	26.11.2014 12:51	26.11.2014 12:51	1	System
1	45670	Waiting	23.10.2014 13:58	23.10.2014 13:58	1	System
1	45680	Approved	23.10.2014 13:25	23.10.2014 13:25	1	System
1	45681	Process order	23.10.2014 13:25	23.10.2014 13:58	2	USER2358
2	45682	Process start	23.10.2014 10:36	23.10.2014 10:36	1	System
2	45683	Being Approved	23.10.2014 10:36	23.10.2014 10:36	1	System
2	45684	Lowest level	23.10.2014 10:36	23.10.2014 10:36	1	System
2	45685	Process end	25.11.2014 7:18	25.11.2014 7:18	1	System
2	45686	Approving	23.10.2014 10:36	23.10.2014 11:13	2	USER2358
2	45687	Approving	23.10.2014 11:13	23.10.2014 13:18	2	USER0357

Table 1 show a fragment of an event log. Typical information needed to analyze are presented. The main assumption is that event log contains data related to a single process. Each event of the case is related to a single process instance frequently marked as *case*. We see within Table 1, Event ID (45678-45681) is related to Case 1. Another

important factor, is that every event must be related to activity. In Table 1, you can clearly see that events refer to activities like *Process Order*, *Being Approved* or L*owest level*. In order for process analysis to take place, it is important to define the minimal requirements which the log has to contain, Case ID and Activity. If the log does not contain a timestamp, it is important to secure the correct chronological sequence at the very first record stage of events. Table 1 also indicates additional information per event, through which we can see all events that have a timestamp. For a log that has events and their timestamps recorded, it is not necessary for the events to be chronologically sorted. With the help of timestamps, we can order events within a case. Without right ordered events we would not be able to detect casual dependencies in process models. The number of timestamps recorded per event can be further analyzed from a performance perspective. If an event has a recorded timestamp, it is possible to further examine the duration between implemented events, where the activity alone has a duration value of zero. However, on the other hand, it is possible to examine the duration of performed events, where their individual durations between events is zero. It is possible to analyze at one time interval the individual duration of that process instance, referred to as a throughput time. If there is an event with a recorded timestamp, included both a start and end time, we can measure the duration of the event entirely. Other than the duration of events, we can further analyze to include examinations of events between implementation, seen as waiting time. Table 1 also includes attribute resources, which distinguish the personnel whom was dedicated to specific activity. Attributes can be further examined in two levels, an event level attribute and a case level attribute. A case level attribute holds information regarding concrete process instances. Simply said, attribute values are noted for all events corresponding to its case. Event level attributes hold information that pertains to events within a case, in simple terms, values of this attribute are within a case within an event that may vary.

To be able to reason in regards to logs, and to precisely specify the requirements for event logs, various notions were formalized [5].

Definition 1 (Event, attribute) Let E be the *event universe*, i.e. the set of all possible event identifiers. Events may be characterized by *various attributes*, e. g., an event may have a timestamp, correspond to an activity, is executed by a particular person, has associated costs, etc. Let AN be a set of attribute names. For any event $e \in E$ and name $n \in AN$: $\#_n(e)$ is the value of attribute n for event e. If event e does not have an attribute named n, then $\#_n(e) = \perp$ (null value).

Definition 2 (Case, case attribute, trace, event log) Let C be the *case universe*, i.e., the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in C$ and name $n \in AN$: $\#_n(c)$ is the value of attribute n for case c ($\#_n(c) = \perp$ if case c has no attribute named n). Each case has a special mandatory attribute *trace*: $\#_{trace}(c) \in E^*$. $\underline{c} = \#_{trace}(c)$ is a shorthand for referring to the trace of a case. We assume $\#_{trace}(c) \neq \langle\rangle$, i.e., traces in a log contain at least one event.

A *trace* is a finite sequence of events $\sigma \in E^*$ such that each event appears only once, i.e., for $1 \leq i < j \leq |\sigma|$: $\sigma(i) \neq \sigma(j)$.

An *event log* is a set of cases $L \subseteq C$ such that each event appears at most once in the entire log, i.e., for any $c_1, c_2 \in L$ such that $c_1 \neq c_2$: $\partial_{set}(\underline{c}_1) \cap \partial_{set}(\underline{c}_2) = \emptyset$.

If an event log contains timestamps, then the ordering in a trace should respect these timestamps, i.e., for any $c \in L$, i and j such that $1 \leq i < j \leq |\underline{c}|$: $\#_{time}(\underline{c}(i)) \leq \#_{time}(c(j))$. Events and cases are represented using *unique* identifiers. An identifier $e \in E$ refers to an event and an identifier $c \in C$ refers to a case. This mechanism allows us to point to a specific event or a specific case. This is important as there may be many events having identical attributes, e.g., start events of some activity a may have been recorded for different cases and even within a case there may be multiple of such events. Similarly, there may be different cases that followed the same path in the process. These identifiers are just a technicality that helps us to point to particular events and cases. Therefore, they do not need to exist in the original data source and may be generated when extracting the data from different data sources.

Extracted data from various data sources are needed to be saved in a suitable format. One format which is noted to be the standard for storing and exchanging event logs is MXML (Mining eXtensible Markup Language). Using MXML, it is possible to store event logs such as the one shown in Table 1 using an XML-based syntax.

The second format is XES (eXtensible Event Stream) [6]. XES is the successor of MXML. Based on many practical experiences with MXML, the XES format has been made less restrictive and truly extendible.

The most widely used format used is also, .CSV (Comma separated values). This format is less restrictive than XES. .CVS in comparison to XES only enables stored data, it is not possible to create your own extensions.

When extracting event logs, we come across many challenges [5]. One of the challenges is known as, *correlation* events, i.e., events showing the need to be related to each other. Dealing with legacy and a variety or interconnected systems, additional efforts are needed to correlate events; see [7] for an example of an approach to correlate events without any a-priori information. Events need to be ordered per case. In principle, such ordering does not require timestamps. However, when merging data from different sources, one typically needs to depend on *timestamps* to sort events (in order of occurrence). In extracted data, we can come across existing cases, which are still running. Therefore, it is important to realize that event logs typically just provide a *snapshot* of a longer running process. Another challenge is the *scoping* of the event log. Information systems may have thousands of tables thus, making it imperative to see which tables incorporate relevant data. Domain knowledge is needed to locate the required data and to scope it. *Granularity* of logged events are seen within the system. Some systems produce low-level events, there exist several approaches to preprocess these type of events. For instance, frequently appearing low-level patterns can be abstracted and merged into a new event, which represent the performed activity [8].

4 Event log extraction

We designed and implemented a connector that serves to extract event logs of process-driven application that ensures a proper process monitoring functionality. All data is present in relational databases. For definition of processes, activities, events, cases, and case level attribute data, used design-time parts from the database. Activity, events and their metadata distinguish the monitored processes and we were able to extract data from runtime parts within the database.

Design-time data. The design-time data contains all process definitions. For individual processes defined, information is available regarding the date of implementation, actual running versions, as well as, historical record of previously implemented versions. In addition, all process have defined their activity sessions/lines and their metadata. For individual activity, corresponding events are defined, which are of two types, either human or system. Information is also available about the rights to a user whom are involved in user groups within a process. Important information was that which involved the setup process concerning the amount of detail that will be logged, where the most necessary values represent complete logging. Throughout the logging process of all activities, important for analysis were attribute values pertaining to process instances regarding individual process activities.

Runtime data. The runtime data contains all the data for the running instances of processes and their components. Each released process instance contains unique identifiers. For activities carried out in the process, unique identifiers were recorded and the foreign key for a process instance in which it took place in. Individual events are recorded within activities that contain unique identifiers that contains foreign key to an associated activity instance (which is specific to when an event instance occurs). For individual events, timestamps are recorded and the human activity type holds the employee that performed event. Metadata related to a process, activity, and event instances are linked via specific foreign keys.

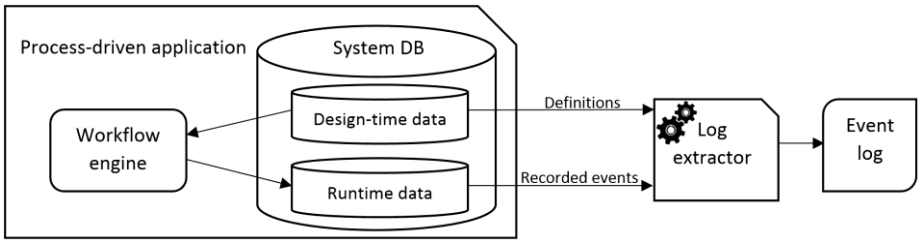

Fig. 3. Scheme of event log extraction.

Figure 3 demonstrates a scheme of processes extracted within an event log. Through the information about processes gained from design-time areas of the database, we were able to extract data from runtime parts of the database. Table 2 contains a list of attributes which are needed to reconstruct individual processes. Attributes “Resource”

and “Event Type” are expandable attributes for reconstructed social networks and in respect to the server or human activity.

Table 2. Base extracted attributes.

Attribute name	Orders	Invoices	Description
Case ID	✓	✓	Process instance identifier
Activity	✓	✓	Activity + event name
Start timestamp	✓	✓	Event start time
End timestamp	✓	✓	Event end time
Event Type	✓	✓	1-server, 2-human
Resource	✓	✓	Resource name

Extracted data were accounted for both processes as well as supplementary information regarding suppliers as follows: supplier name, supplier city, supplier state as case attributes. For the Invoice Approval process, user comments data were extracted to identify the most frequent reason for refusal of invoices. An additional case attribute was a case status, which supported in seeing the differences between completed, running, error, and delete process instances. Analyzed data is provided only in regards to actual versions of processes.

Logs were stored in .CSV files. If we were to rate the quality of logs, then based on maturity levels [9], five stars would be their assessed rating. The reason for this quality is, logs are derived from a BPM system. Events are recorded in an automatic, systematic, reliable, and safe manner.

5 Process mining techniques

For reconstructed processes, we used a process mining technique. This technique is able to extract knowledge from event logs. The idea of process mining is to discover, monitor and improve processes. Process mining includes discovery process models, conformance checking (comparing model and log), social networking, organizational structure mining, case prediction, and history-based recommendation.

As a process mining tool we choose Minit. With Minit we can import datasets for a wide range of possibilities in analyzing processes precisely. With this tool, we were able to analyze process models in a logistic company and their statistical characteristics. What was of great importance in our analysis was to analyze the social network by which we could bring out the fine details of each zoned resource in all processes. The greatest advantage was also that Minit has the capability to produce custom charts through which we could analyze data and compare and contrast their characteristics based on any needs.

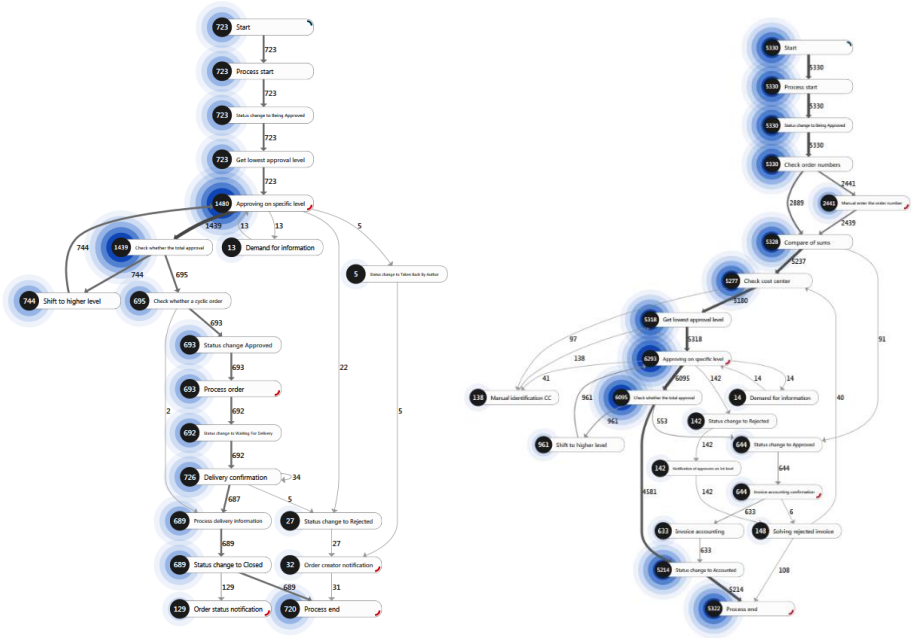

Fig. 4. Purchase Order process map (left side) and Approve Invoice process map (right side).

After importing event log into Minit**,** identified process maps were created. Figure 4 shows a Purchase Order process map and Invoice Approving process map.

5.1 Process discovery

Our overall goal was to reconstruct process models from an event log and analyze them. At first, we began to identify the main process flows and their deviations. Social networks and selected activity processes were analyzed. Statistical findings and analysis included only for completed process instances in case attributes with a reached value "completed". Table 3 contains basic statistics of reconstructed processes.

Table 3. Overwiew of processes characteristics.

	Orders	**Invoices**
Timeframe	166 days 7 hours	48 days 3hours
Cases	720	5322
Events	12 328	65 985
Activities	20	20
Event attributes	3	3
Case attributes	17	20
Variants	28	54
Resources	42	45

5.2 Control flow identification

Process main streams were identified via the most numerous variant of processes. The variant of the processes is defined by unique activity sequences (cases followed same paths in the process). We observed how many percent in overall behavior captured in the most numerous variants.

Purchase Order. In the approval process of orders, noted were 5 of the most numerous variants which accounted for 88% of the overall behavior. Table 4 indicates the basic characteristics of chosen variants. *Variant 1* describes the behavior when an order is approved in the second level, "Approving on a specific level", for every process instance exercised, which took place twice. Orders are approved without the carrying out additional activity. An order is approved and then carries out the processing of orders and its goods to be delivered complete. *Variant 2* describes the behavior regarding when a purchase order is first approved without the approval of a higher ranked individual. Purchase Orders are approved without carrying out any additional activities. The approval of the Purchase Order thereafter is completed upon delivery of all goods. *Variant 3* describes the behavior in which a Purchase Order is approved in the third level. "Approving on specific level" takes place three times for every process instance. Purchase Orders are approved without carrying out any further activities. The approval of the Purchase Order thereafter is completed upon delivery of all goods. *Variant* 3 and *Variant* 4 describe behavior when Purchase Orders are approved in second (*Variant 4*) and third (*Variant 5*) level, "Approving on specific level". For these specific levels, each process instance is still performed two to three times. Orders are approved without performing any further activity. These Purchase Orders are approved, and processing takes place, however, goods delivered are carried out with discrepancies.

Table 4. Characteristics of the most numerous variants in the approval process of purchase orders. Table includes how many cases and events are included in the monitored variants with information of their mean duration.

Variant ID	Cases coverage	Events per case	Events coverage	Mean duration
Variant 1	36 %	17	35 %	10d 23h 15min
Variant 2	20 %	14	16 %	5d 21h 51min
Variant 3	18 %	20	21 %	12d 21h 34min
Variant 4	11 %	18	12 %	8d 5h 55 min
Variant 5	3 %	21	4 %	7d 9h 2min

Invoice Approving. Four of the most numerous variants were described having 89% of the overall behavior in the Invoice Approval process. Table 5 indicates the basic characteristics of the selected variants. *Variant 1* describes the behavior where an invoice is successfully mapped comparing the difference in prices found. Followed by the cost center which verifies data. The invoice is approved in first level without the need of a higher authority to approve of the invoice, and no further activities take place at this stage. All invoices in this variant are directed to a special CC (Cost Center).

Variant 2 describes the behavior where the invoice does not have a purchase order labeled followed by one that is manually labeled to complete the invoice's missing data. When prices are compared a difference is found. The CC checking takes place, where it correctly defines which center will further take action to process the order. At the first stage, the invoice does not need the approval of a higher authority and with no further activity. All invoices in this variant are directed to a special CC. *Variant 3* describes the behavior where a purchase order number pertaining to the invoice is not labelled and is then followed by a manually labelled order. Through price comparisons a difference is found and an inspection is followed by the CC checking, where it correctly defines which center will process further. The invoice at the second level is approved and without the need of a higher authority or further activity. All invoices in this variant are redirected to a special division of the CC. *Variant 4* describes the behavior of an invoice that has an unlabeled order number and then follow a manually labeled order number. When prices are compared a difference is found. CC is correctly selected. Invoices are approved in second level. Invoices are approved without further activity and all invoices in this variant are not directed to the CC, they are approved and sent for accounts payable.

Table 5. 5 Characteristics of the most numerous variants of the Invoice Approval process. This table indicates how many cases and events have been obtained in the monitoring of variants obtaining mean duration values.

Variant ID	Cases coverage	Events per case	Events coverage	Mean duration
Variant 1	50 %	11	44 %	1d 3h 54min
Variant 2	28 %	12	27 %	1d 11h 46min
Variant 3	8 %	15	9 %	2d 22h 48min
Variant 4	4 %	18	6 %	8d 10h 45 min

The most frequent behavior and performance properties were disclosed. Both processes have a common bottleneck where multi-level approvals take place. Presented for both processes was the performance aspect to identify the differences between processes. In both, a remarkable growth in throughput time cases were noticed where multiple approvals took place. In the Purchase Order process, throughput times in second level approvals were 1.8x higher and 2.2x higher in the third level, compared to an approval on the first level. In the Invoice Approval process, throughput times in second level were 2.5x higher and 4.5x times here in a third level, compared to an approval on the first level.

5.3 Points of interest in Purchase Order process

In this section, we tend to the possibilities of optimizing the Purchase Order process. We identify areas in which we see processes that hold statistical values or performance problems. We purely focused on the approval of purchase orders, the people involved in the process and their social network. Revealed characteristics showed suppliers and reoccurring problems.

Approving of specific level.

Activity seen in this section pertains to the approval of purchase orders. All orders must be approved where the approver can at first instance be able to do so, or request additional information. A problem was discovered in the distribution of work and the duration range between resources for approving where the approver approved 288x order in 20 hours. The approver with the second highest number of approvals was 208x by which their average time needed to approve of order was 8 minutes.

For 17 process instances, research found procedural irregularities where one user approved on more levels where rules state that the approver cannot approve at multiple levels.

An actual instance was revealed to show that for one supplier, 23 orders were approved, by which the order had the same sum amounts where all 3 users carried out the same order of processing. Seeing this procedure take place allowed a cyclic procedure to be implemented to save time in process.

Resources in the process.

In the Purchase Order process, over allocated resources were discovered. Resources in the process are aimed to provide to multiple tasks, where they carry out all or some human activity. Unclear defined working tasks for employees that are not implemented, show strained and overworked employees comparing on another. Within the process, 42 resources were seen in total having 5 employees perform 55% of all human activities. Of these 5 employees, they performed 57% of all process instances. Process owners were advised to relieve these over allocated resources. There were employees discovered having communicated with a larger number/group of employees to process purchase orders. Also, it was seen that a pair of resources shifted their work among other areas of work in 27% and 20% of process instances. Over allocated resources are clearly seen through the social network of the Purchase Order process in Figure 5, where you can see the size of them through the width in the echo/halo effect. Their communication is also seen quantified compared to others, where comparisons can be seen through the social network to caution their high frequencies seen by the thickness in paths.

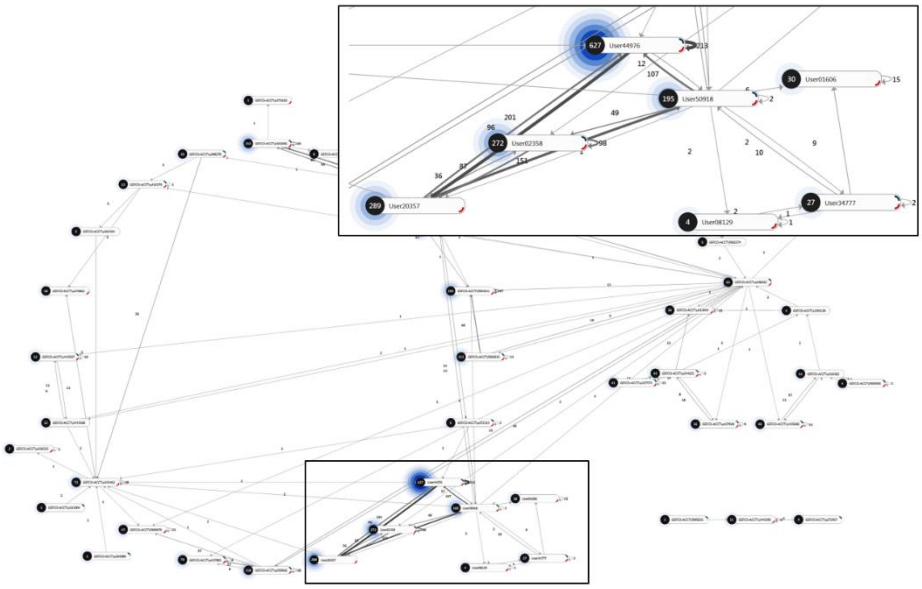

Fig. 5. Social network for Purchase Order process with displayed event count metrics.

When creating purchase orders for individual resources, insufficient knowledge regarding the working process was seen. First off, the slowest employee performed activity 30x slower than the fastest employee. Also, different duration ranges were observed. The highest duration range was 17 seconds compared to 21days and 18 hours that was seen for one of the resources. Among the resources whose activity created an order that took longer than average was seen to take place for over allocated resources.

Suppliers' characteristics.

Through "Delivery confirmation" activity, we were able to identify discrepancies in the delivery of goods/services for individual suppliers together with the information regarding the amount of discrepancies that took place. Contractors were also identified, that reported discrepancies in all their deliveries of goods/services. In addition to these findings, we were able to predict the time it would take to deliver goods/services for individual contractors together with information regarding the overall completeness. With this information, process owners knew how to accurately plan/schedule orders for the delivery of their goods/services.

5.4 Points of interest in Invoice Approving process

In this section, we tend to the Invoice Approving process. We investigated activity that took place in "Manual enter order number" and saw that it accounted for approximately 50% of all process instances. Focusing on the "Solving rejected invoices" activity, we applied text mining techniques to see the most common causes of rejected invoices.

Manual enter order number.

Invoices with incorrect or missing labeled IDs, are identified by the system. For unidentified invoices, the system allocates tasks among different staff to manually enter correct order numbers. We identified suppliers that needed the most manual entries of order numbers. Suppliers were introduced to process owners. Based on this information, it was possible to reduce the number of invoices that had to be manually labeled. Results showed that for identified suppliers, the process owner was able to save up to 7 hours in processing the invoice.

Solving rejected invoice.

It is possible to reject an invoice at the first level of approval or at the stage of payment processing. Rejected invoices are directed to human activity that examines why the invoice was rejected. In this part of the examination, it is possible to end the activity (Decline) or reveal reasons why it was declined and begin the process again to obtain an (Approved) stage for activity that monitors and controls the sum amounts on an invoice. For this activity, the process owner allocated 3 employees to complete the tasks. It was planned to equally divide tasks among the employees. The reality showed that all three employees differed in how much each of them were dedicated to a task, one performed 56% in all tasks, another at 34%, and the third at only 10% of the overall number of events. The person with the highest number of events in this activity obtained the shorted average duration time in solving rejected invoices at only taking 13 hours. The employee with the lowest percentage of their participation in events took them an average duration time of 15 days. These "reality checks" were presented to the process owner regarding the most efficient employee and their working manners to other employees so that they too, may effectively work. Through analysis of this information, an interesting finding was made, one that showed if an employee devoted less than 2 days to problems that were detected, 85% of invoices were declines and only 15% were approved. A ratio of 50% approved and 50% declined were obtained in the case of an employee whom devoted themselves to more than 2 days in sorting out rejected invoices.

In order to establish the most common grounds for refusal regarding invoices, an expansion of the event log was made to contain "User comments". Data was extracted and reserved for this activity alone, by which all other activity in logs contained an empty value. Through text mining techniques [10], obtained were the most common causes of rejection. The most frequently seen phrase was "wrong verification" at 29%, and another "wrong amount" at 24%. The lesser frequently seen phrases were "missing date" and "invoice in the wrong language". With these identified phrases, the process owner was able to eliminate these shortcomings to better enhance the overall process to run smoothly.

6 Conclusion

For valuable analysis of processes, it is important to have high quality logs. The biggest challenge is the pre-processing of data to create quality logs from different systems. It was vital to grasp all events and information that were recorded in the midst of all operations in processes. The higher quality of information present in logs, the higher quality the details will be regarding the processes. Thereby, we can gain insight by using process mining techniques to create high quality analysis and evaluations.

In our work, we greatly focused on the extraction, pre-processing, and analysis of data that was stored in BPM systems. We defined the structure for stored data and defined the attributes attached to analyzing the processes. Newly created process logs were then imported into a process mining tool. Subsequently, we were able to introduce a process model and its statistics based on the extracted processes. Consequently, we introduced characteristics and points for improvement for individual human activity. We also devoted our time to the employees in the Purchase Order process studied and found over allocated employees to dedicated tasks via the social network feature in the process mining tool.

One of the possibilities that was not provided in this work, was the view of how to analyze within a BPM system. Our studies showed the analysis mainly regarding human activity. On the other hand, in that same regard, we can analyze the server activity. With this information, it is possible to detect performance characteristics, communication or discovery of bottlenecks pertaining to individual systems in a process. By modifying logs or using selected filters in the process mining tool, we were able to select and analyze parts of processes that needed to be most looked at.

References

1. van der Aalst, W.M.P., van Hee, K.: Workflow Management: Models, Methods, and Systems, MIT Press (2004).
2. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers (1998).
3. Wohed, P. et al.: On the Suitability of BPMN for Business Process Modelling. Conf. Business Process Management (2006).
4. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems (2005).
5. W.M.P. van der Aalst.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011).
6. C.W. Günther.: XES Standard Definition. www.xes-standard.org (2009).
7. D.R. Ferreira and D. Gillblad. Discovering Process Models from Unlabelled Event Logs. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors, Business Process Management (BPM 2009), volume 5701 of Lecture Notes in Computer Science, pages 143–158. Springer, Berlin (2009).
8. R.P.J.C. Bose and W.M.P. van der Aalst. Abstractions in Process Mining: A Taxonomy of Patterns. In U. Dayal, J. Eder, J. Koehler, and H. Reijers, editors,

Business Process Management (BPM 2009), volume 5701 of Lecture Notes in Computer Science, pages 159–175. Springer, Berlin (2009).
9. IEEE Task Force on Process Mining. Process Mining Manifesto. In BPM Workshops, volume 99 of Lecture Notes in Business Information Processing. Springer, Berlin (2011)
10. Krzysztof J. Cios, Roman W. Swiniarski, Witold Pedrycz, Lukasz A. Kurgan.: Data Mining: Knowledge Discovery Approach. Springer (2007), 453-465