Process Mining: Research in Banking Operations

Felipe Rojos, Francesca Lucchini, José Manuel González, Diego Espinoza, Jonathan Lee, Juan P. Salazar Fernández, and Michael Arias

Computer Science Department, School of Engineering
Pontificia Universidad Católica de Chile, Santiago, Chile
[farojos,flucchini,jugonzalez]@uc.cl,diesmori@gmail.com,[wllee,jpsalaza,m.arias]@uc.cl

Abstract. This paper analyses a loan application process of a Dutch Financial Institute proposed by the BPIC Challenge 2017 using process mining tools and techniques. Several filters and performance and organizational analysis are executed in order to observe underlying patterns that may explain some of the bottlenecks of the process, time spent on certain activities and their acceptance rate. Also, we proposed a recommender system to assist client profiling. Our results also include observations regarding to customer communication.

Key words: Performance, Organizational analysis, Process discovery, Social networking, Recommender systems

1 Introduction

1.1 Context

The Business Processing Intelligence Challenge (BPIC) 2017 [1] is a well-known contest that promotes the use of process mining and other business intelligence tools to analyze a process. This year, the focus is on the loan application process of a bank. The institute that provided the datasets is particularly interested in:

1. What are the throughput times per part of the process? In particular, the difference between the time spent in the company's systems waiting for processing by a user and the time spent waiting on input from the applicant.
2. What is the influence on the frequency of incompleteness to the final outcome?
3. How does the conversion compare between applicants for whom a single offer is made and applicants for whom multiple offers are made?

This year's challenge is related to the one in 2012 [2]. Since the bank incorporated to their process ideas submitted in 2012, the datasets provided in 2017 are an enriched version of those of that year. An important difference is that the company switched systems and now supports multiple offers for a single application (in contrast to 2012, where a work-around was clearly visible in the log).

There are two datasets: (1) the Application Dataset that represents the whole process and (2) the Offer Dataset that brings detailed information on Offers and their state.

When a client applies for a loan, a group of tasks must be executed before the client receives the loan. It is necessary to build up a client profile based on the client's background and documents requested by the bank. Then, the bank should validate these documents, determine the amount and the risk of the loan, and call the client to make the corresponding offer until an agreement is found. Therefore, behind every loan there is a complex process.

All banks are interested in understanding this process. The challenge is to study it by answering the questions previously mentioned. Many tools of many different kinds can be used in order to achieve this goal, and among them, there is one that has been gaining popularity nowadays: Big Data Analysis.

This paper contribution is the use of different Big Data tools in the Business Intelligence area, especially process mining tools. The obtained results pointed to difficulties in the interaction with clients and client's profiling. Because of this, we propose a solution to this problems based on building client trust and client profiling using a recommender system.

The paper is structured as follows: Section 2 gives detailed information on the tools used to analyze the process. Section 3 describes the event log of the dataset and event log data preprocessing. Section 4 includes analysis related to the BPIC 2017 questions and our proposed approaches. Finally, the conclusions can be found in section 5.

2 Tools

We mainly focused in the use of different process mining tools and techniques. In particular, process mining focuses on answering three central questions [3]:

- ***discovery***: What is the process that resides on the event log?
- ***conformance***: How is the entity's (the bank in this case) idea of the process, compared to the model discovered by analyzing the event log?
- ***enhancement***: How we can use the process model obtained, in order to understand the system and analyze it, such as determining performance of different activities on it, make predictions based on characteristics of the cases, etc?

Different tools were used to study the dataset and obtain valuable observations to both the questions of the challenge and our proposed approach. The following list gives a general description of each tool and its main purpose in the analysis of the data.

1. *Disco* [4]: a commercial software from Fluxicon that focuses mainly on process discovery, filtering and performance analysis. A free academic license was provided in order to face the BPI Challenge. It was mostly used to understand the process, filter the log and visualize it.

2. *Celonis*[1]: a commercial tool for process mining. Its main features include process discovery and visualization, conformance checking features and social networking analysis. A free academic license was provided too. It was useful for performance analysis, studying bottlenecks of the process and social network analysis.

3. *ProM 5.2/ ProM 6.6* [5]: an open source framework for process mining algorithms. It was used as a complement of *Disco* for conformance analysis. Additionally, it was used to visualize the process as a Petri Net and discover bottlenecks.

4. *Python*[2]: a high-level programming language. Python was used to create simple scripts that filtered and obtained simple statistics from the event log. It also had a third use: programming the recommender system.

5. *Excel*[3]: a spreadsheet application made by Microsoft Corporation. It was used to store statistics, sort information and create charts.

3 Data Description

3.1 The Log

The data used for the analysis was published for the BPIC 2017. As mentioned before, it is composed by two event logs (1) the Application event log that represents the whole process and (2) the Offer event log that brings detailed information on Offers and their state.

After downloading the application event log and importing it to *Disco* as a .XES file, we found that the complete log contains 31,509 different cases, each one of them corresponds to an application from a client, requesting a loan. Also, there are 561,671 events and 4,047 variants of the process.

Each case corresponds to a list succession of events that occurred during a loan request and is labeled with a unique *Case ID*. Each event is a row in the event log conformed by five main values: Event ID, Case ID, Activity and start and end timestamps. Along with these information, every case has other attributes, such as who was the user of the bank that performed the different activities (resource), what was the loan goal and the requested amount, how much money did the bank offer, etc.

As described in the BPI forum, there are three types of activities. The ones with identified with the prefix *A_* are related to the state of the application itself. The ones with the *O_* prefix correspond to offer events and the *W_* is reserved

[1] http://www.celonis.com
[2] https://www.python.org
[3] https://products.office.com/excel

Table 1: Description of the Application Event Log Activities

Activity	Description
A_Accepted	Loan application has been accepted by the bank and now offers can be provided for the client
A_Cancelled	Application has been canceled because client canceled or a response is not received after a fixed period of time
A_Complete	Offers have been sent, bank waits for a response from the client
A_Concept	An automatic first assessment has been done to an Application
A_Create Application	A new application has been created
A_Denied	Endpoint of the application process; client does not meet the criteria and the loan is refused
A_Incomplete	Required documents are incomplete or have mistakes
A_Pending	Endpoint of the application process; all requirements are met and the client receives the loan
A_Submitted	A client submitted a new loan application from the website
A_Validating	All documents have been received and are being checked
O_Accepted	Offer has been accepted by the client
O_Cancelled	Offer has been canceled because offer was sent to applicant who did not in reply in time
O_Create Offer	
O_Offer Created	Offer has been created. Occurs sequentially with the previous one
O_Refused	Client refused the offer or the bak refused the offer
O_Returned	A response about the offer has been received from the client
O_Sent (mail and online)	Offer has been sent through land mail and through an online channel
O_Sent (online only)	Offer has been sent through an online channel
W_Asses potential fraud	Check if an application might be a fraud
W_Call after offers	Call the client after sending offers
W_Call incomplete files	Call the client because there are incomplete documents
W_Complete application	A pre-accepted application is completed
W_Handle leads	Manage incomplete application submission
W_Personal loan collection	Loan has been handed over to a different part of the bank
W_Shortened completion	Short completion of an application
W_Validate application	Application is being validated

for those who represented “workflow” events or transitions in the process, which are activities executed by the bank employees.

Table 1 presents a list of the activities and a brief description based on the ones described in [6].

3.2 Preprocessing

Figure 1 shows the resulting model after importing the application event log in *Disco*. Every blue box in the image represents an activity and every line, a possible path that a case has made from one event to another. It is possible to observe the complexity of the process.

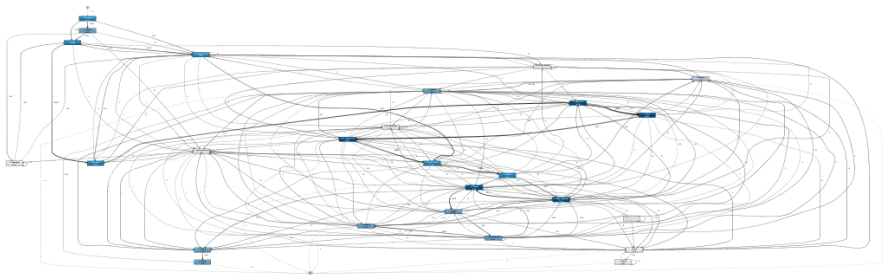

Fig. 1: Process map obtained using *Disco*

Considering the mentioned complexity, we decided that was necessary to pre-process the log. In order to do that, we use *Disco*, and we proceeded as is described below:

1. Sequences:
   Groups of activities that always occurred in sequence and the time of execution between them was less than a few seconds were identified and grouped. Among those groups we found:
   - *A_Accepted, A_Submitted* and *A_Concept*
   - *O_Created* and *O_Create Offer*
   - *O_Accepted* and *A_Pending*

   We summarized these activities by selecting one that we thought represented the whole group, and those correspond to: *A_Concept*, *O_Create Offer* and *A_Pending*. So, for example, in every trace that initially presented the activities *O_Accepted* and *A_Pending*, now will only appear *A_Pending*, with a time of execution corresponding to the sum of both.
2. End points:
   The end points of the process are *A_Accepted/A_Pending*, *A_Denied* and *A_Cancelled*. Given this, cases that ended in the activities: *O_Sent*, *W_Personal loan collection* or *W_Shortened* completion were removed. We decided to do so, because it can be seen that the three of them don't correspond to a real completion of the process and the frequency of these cases was insignificant.

Also, in order to reduce the number of variants in the log, we decided to remove the least frequent variants. The criteria for elimination was to have a frequency under 100.

3. We didn't consider the traces that presented fraud analyses, because they represented a significant bottleneck in the process, but were very rare (301 cases).

After the described pre-process, we obtained a simplified log that is compared with the original in Table 2:

Table 2: Original vs Preprocessed Log

Log	Number of cases	Number of variants
Original	31,509	4,047
Preprocessed	31,194	3,544

It can be seen that the resulting log contains up to 96% of the original cases, but has 500 less variants, which reduces the number of events considerably, and so, the complexity of the data.

4 BPI Question's Analysis

In this section we refer to the three questions proposed by the BPIC and our two proposed approaches: bus factor analysis and a recommender system for client profiling.

4.1 Initial Analysis: Mainstream bottlenecks and their causes

Before answering the questions proposed by the BPIC, we need to classify the existing bottlenecks of the process.

As has been pointed out in the BPIC, there's a concern over time spent working and time spent waiting for clients' responses. We will analyze each bottleneck, label it and try to discover the cause. We hipothesize that bottleneck's causes might be related to not having enough employees and lack of client's trust. We will provide some solutions in line with the results we found.

We imported the log with the mainstream variants into *ProM 5.2*. Then, we generated a Petri Net from the model mined with the plug-in *Heuristic Miner*. After that, we did a performance analysis using *ProM 5.2* with the *Performance Analysis with Petri Net* plug-in, the log and the Petri net. We refer as bottleneck as anything that takes over 8 hours (approximately aworkday).

We found that there were 6 bottlenecks and decision points:

1. The place in the Petri net after W_Complete, A_Concept or A_Incomplete application leading to A_Accepted.

2. The place after A_Complete or O_Sent (mail and online) leading to A_Cancelled
3. The place after O_Returned leading to the final assessment[4]
4. The place after A_Incomplete leading to W_Validate application.
5. The place after A_Complete leading to O_Create offer
6. The place after A_Validating leading to A_Denied or W_Call Incomplete Files

We labeled these into two groups: (1) *Client dependent*: long waiting times associated with clients' responses (2) *Work-flow dependent*: long waiting times associated with internal elements of the process, such as an activity's nature or resources. All Client-dependent bottlenecks are related to waiting feedback from the client. So, the first two bottlenecks in the list are client dependent. The end-point activities of these elements are related to the end of the process determined by the client. Either they accept an offer or cancel the whole process(explicitly or after a fixed period without an answer). In the worst case scenario, the client takes about 4 days to accept (50% of the cases or 9,429 cases) and about 31 days to cancel (41% of the cases or 7,763 cases). However, we can't put the blame on clients, we must ask: do clients trust the bank? In our previous analysis we shown that clients who talk with the same employees about different offers are 5% more likely to accept an offer. It may seem small, but if that 5% of acceptance will reduce average process time, since cancellation takes around 8 times more. One recommendation could be to encourage employees to discuss offers with the same clients. The remaining four bottlenecks are related to a process of assessment and/or validation. Possible causes might include the employee's workload. The questions that arise are: is the workload well distributed among employees? Is enough people working on these processes? Is the process itself too complicated? Adding more workers can or cannot speedup the process?

4.2 Proposed question 1: throughput time

As we mentioned before, the question proposed by the BPIC is to compare the time spent by bank users vs. the time spent by the clients in the different activities. We used *Disco* and *Celonis* to preprocessed the application event log and to analyze the throughput times and the performance of the different cases.

General information: We found that the mean throughput time of the cases is 22 days. Also, we discovered the most time demanding activities, which are shown in Table 3.

Table 3: Mean Activities Throughput Times

Activity	Case frequency	Mean duration
W_Validate application	69.41 %	22 h, 51 m.
W_Call incomplete files	47.61 %	21 h, 1 m.
W_Complete application	94.08 %	6 h, 2 m.
W_Handle leads	11.56 %	20 m. 20 s.

[4] final assessment may lead to A_Denied, W_Complete application, W_Call incomplete files or O_Accepted

It can be seen that the longest activities correspond to *validate client's information* and *call because of incomplete files*, which correspond to bank tasks. But each one of them lasts in less than a day (fewer than 5% of the expected time in process).

In addition, Table 4 shows the longest transitions.

Table 4: Mean transitions throughput times

Transition	Case frequency	Mean duration
A_Complete → A_Cancelled	25.45 %	27.4 d
A_Complete → W_Validate application	58.67 %	8.8 d
O_Sent (online only) → W_Validate application	2.62 %	4 d

Unlike the activities, the most critical transition, *A_Complete → A_Cancelled*, corresponds to a client task and it is even longer than the expected time in process. For the second and third longest transitions, it is not clear if they are clients or bank actions.

Considering the time that takes the transition from *A_Complete* to *A_Cancelled* and because a quarter of the cases passes through it, we propose that it is the client the one that is causing the greatest loss of time. So, we decided to focus our attention on clients events, in order to find: bottlenecks in activities executed by them, similarities between the different variations, the existence of possible patterns that explain the final outcome and any other useful information.

Our first approach was to split the data in three groups: A, B and C, according to the last activity that the cases executed in the process (end point). The first one corresponds to those cases which ended up in a cancellation, the second one, in a success, and the third one, the rest of the cases. By doing so, we found that the loans that end up being canceled or rejected, are the longest ones. This is detailed next.

Groups considering end points: Table 5 shows the mean time of the process for each group.

Table 5: Groups Considering End Points

Group	End point	Mean time on process	Number of cases
A	*A_Cancelled* and *O_Cancelled*	25.7 d	18,355
B	*A_Pending*	16.2 d	12,722
C	The rest of the cases	16.5 d	3,669

It can be noticed, as we said before, that the cases which end up canceling are the longest ones. In fact, they tend to stay in process almost 10 days more than the other cases.

This result encouraged us to analyze if it was possible, somehow, to predict whether a client will cancel the contract or not. So, setting aside the groups by end points, we decided to split the full data in four new groups, according to the mean time spent on process. The main result that we obtained using this idea was an underlying correlation between the requested amount and the time in process. These, along with other findings are detailed next.

Groups considering throughput times: The groups were constructed as it's exposed on Table 6.

Table 6: Groups Considering Throughput Times

Group	Throughput time (days)	Number of cases	Percentage of the log
1	0 to 12	8630	28 %
2	12 to 20	7547	24 %
3	20 to 32	8032	26 %
4	Greater than 32	6930	22 %

The idea was to analyze the data contained in each group in to order to find dissimilarities between them. For example, it is expected that group 4 contains a larger number of cases that went from *A_Complete* to *A_Cancelled* than the other groups, since cancellation time takes around 30 days. We found the following outcomes:

- Critical activities *W_Validate application*, *W_Complete application*, *W_Call incomplete files* and *W_Handle leads* were common on every group. In other words, for every group those activities create bottlenecks in the process. But, it's worth to mention that their duration varied from one group to another. In fact, for the shorter cases the activity *W_Validate application* was the most critical, but as the throughput time increased, so did the activity *W_Complete files*, which was the most critical in the third and fourth group. This lead us to conclude that the cases which end up canceling are related to those ones that completing files takes more than a day.
  Figure 2 shows a process map extracted from *Disco* in which appears the longest activities for the first group (left) and the fourth group (right).
- The critical transitions varied from one group to another and, in the case of *A_Complete* $\rightarrow$ *A_Cancelled*, we found that in the first two groups its frequency was less than 1,000 cases and its mean duration was at most 4.5 days. But, in the last two groups we obtained a frequency over 3,000 cases and a mean duration over 30 days.
- We analyzed the case attributes in the four groups, in order to determine if there are significant differences between them:
  - Loan goal: Table 7 shows the frequency on each group of the four more common loan goals:

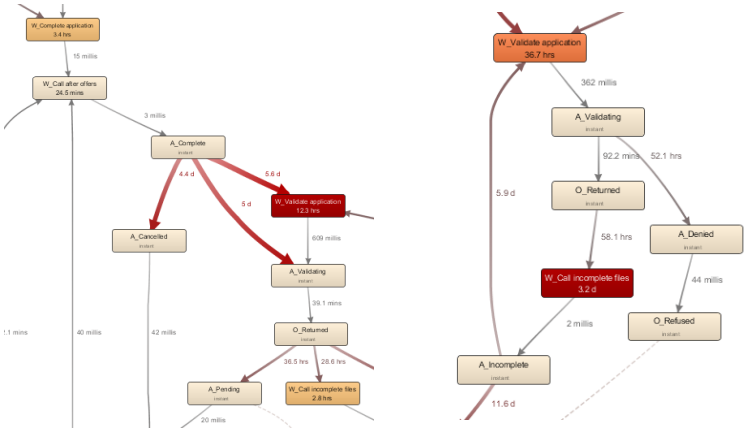

Fig. 2: Bottlenecks Groups 1 and 4

Table 7: Loan Goal Frequency on Throughput Times Groups

Loan goal	Group 1	Group 2	Group 3	Group 4
Car	32.62 %	27.05 %	24.63 %	26.19 %
Existing loan takeover	13.91 %	19.34 %	20.9 %	20.73 %
Home improvement	20.41 %	26.38 %	25.79 %	25.15 %
Other	8.84 %	9.5 %	9.32 %	9.15 %

The first group presents a greater amount of cases whose loan goal is a car, than the other groups, while has less applications whose goal is an existing loan takeover or a home improvement.

– Requested amount: As mentioned before, we found that the requested amount had a positive correlation with the time spent in the process, as can be appreciated in Figure 3.
The chart shows that when the loan requested amount becomes bigger it can be expected that the throughput time does so. In fact, the slope of the adjusted line is greater than 0.5 as can be seen in equation (1).

$$y = 0.57x + 19.42 \tag{1}$$

Also, it's worth mentioning that for the line, the statistic $R^2 = 0.69$. This allows us to conclude that, for example, if a client's loan requested amount is between 5,000 and 10,000 dollars, and another is between 20,000 and 25,000, the time in process of the second one will be approximately $3 \cdot 0.57$ days greater than the first one.

Considering the original question, we can conclude that is the client the one that is causing the biggest delays in the process. It would be interesting for the bank being able to predict as much as possible the outcome of a client and the requested amount would be useful in doing so. Also, we found that the cases that

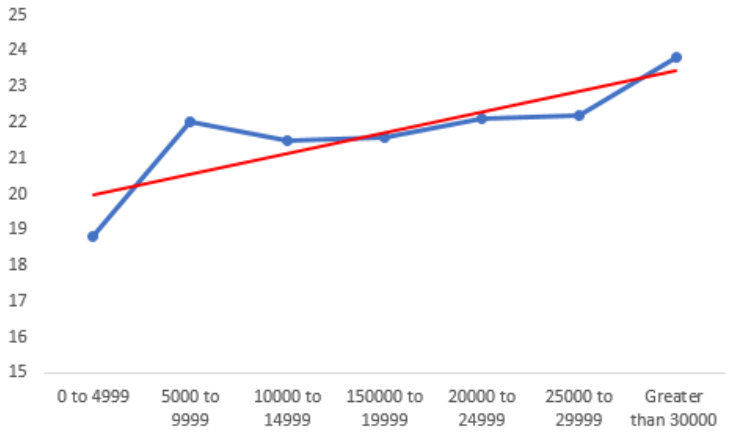

Fig. 3: Loan requested amount vs. throughput time

has problems in completing files are related to the ones which end up canceling, so we delve into this fact in question 2.

4.3 New Approach: Bus Factor

Considering the fact that the process has several bottlenecks and there is aneed to discover possible sources, we proposed a new approach: the Bus Factor or Truck Factor. It measures risk related to information and employees' skills, since both are not equally shared by all members of the organization. Then, the bus factor measures how many people can "be hit by a bus or be removed from the process before it stops working. The bigger the bus factor the better, because this means that there isn't a monopoly in skills or relevant information.

The Bus Factor is usually applied to software development processes and it's defined as "*number of key developers who would need to be incapacitated, i.e. hit by a bus, to send the project into such disarray that it would not be able to proceed*" [7]. However, this definition can be extrapolated to any process or activity. Our version of the Bus Factor is focused on each activity and how many users execute it; in the end, is a simple sum of all resources that perform each activity. This will be used to find possibly risky activities that will be subject to an organizational evaluation.

The higher the Bus Factor is the better. Also, to get deeper results, we can enhance this result with an analysis of how frequently each user executes an activity and their throughput time. Ideally, the workload, considering amount and difficulty, must be evenly shared between employees.

We implemented a Python script to calculate the Bus Factor for each activity. This script received as input the Applications event log. Then, it iterated over all rows filling a set associated to each activity with all of its resources. We considered 2 categories: (1) full event log, (2) log with variants with over 100 occurrences (Mainstream Variants). Results are shown in Table 8.

Table 8: Bus Factor for each activity

Activity	**BF (Full)**	**BF (Mainstream)**
A_Accepted	113	105
A_Cancelled	108	94
A_Complete	113	103
A_Concept	114	105
A_Create Application	111	104
A_Denied	99	26
A_Incomplete	101	69
A_Pending	40	39
A_Submitted	1	1
A_Validating	127	117
O_Accepted	40	39
O_Cancelled	133	94
O_Create Offer	120	105
O_Created	120	105
O_Refused	97	26
O_Returned	47	41
O_Sent (mail and online)	114	104
O_Sent (online only)	104	0
W_Assess potential fraud	25	0
W_Call after offers	119	104
W_Call incomplete files	117	104
W_Complete application	119	107
W_Handle leads	84	65
W_Personal Loan collection	2	0
W_Shortened completion	9	0
W_Validate application	128	117

We can see that the process of the challenge has one activity, *A_Submitted* that is only executed by one user, *user_1*, the system. In this case, the Bus Factor is 1, which is risky, since if the system stops working, all variants with it can not be executed. At the same time, the table shows activities with over 100 users executing it. This means, several people could stop executing this activity and related variant could still be carried out.

As can be seen, most activities have around 100 users executing them, so most of them have a high Bus Factor. We will focus on the ones that have between 2 and 50 users in the *Mainstream Variants* category, because the only activity with 1 user is an activity of the system and when they have 0 users, that means that activity is not included in mainstream variants. These activities are listed in Table 9. The results obtained using the social analysis feature of Celonis revealed that most activities with a small user set have the event user frequency as is shown in Figure 4, and an event throughput time as shown in Figure 5.

Table 9: Bus Factor: risky activities

Activity	**BF (Mainstream)**
A_Denied	26
A_Pending	39
A_Submitted	1
O_Accepted	39
O_Refused	26
O_Returned	41

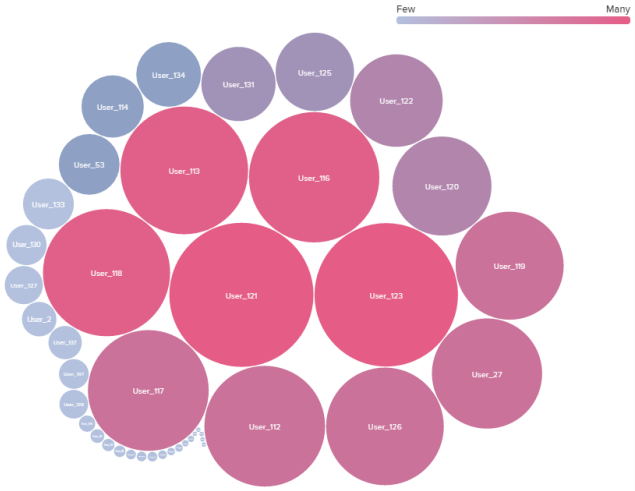

Fig. 4: Event frequency by User

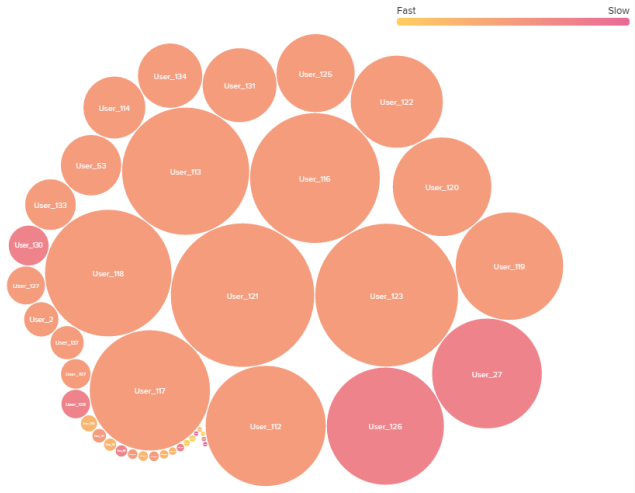

Fig. 5: Event Throughput by User

In Figure 4 circles represents a user and its diameter is defined by how frequently the user executes that activity, a trait also represented with colors: darker color (red) means a higher event count and a lighter color (light blue), a lower event count. Figure 5 represents throughput time with colors: a lighter color (yellow, orange) represents a higher throughput compared to average throughput and a darker color (red), shows a lower throughput.

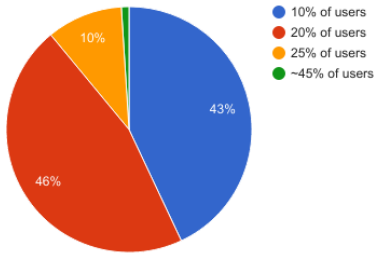

Fig. 6: Load Balance by Percentage of total users

As we can see in Figure 6, these activities usually have a small group of users, around 30% that is in charge of 80% of event occurrences. Also, as can be seen in Figure 5, the throughput time is similar between users, this suggests us that they probably have a similar level of expertise on the execution of the activity. In addition, it is shown that there are no users taking simple cases in order to solve them quickly and get a higher frequency of execution. This helps us determine that, even if there is a workload imbalance and the bus factor as a simple sum is low, the process will still continue even if the main 30% of resources are removed.

In Sub-section 4.1, we inquired about the bottlenecks surrounding process of assessment and/or validation. According to the obtained results of the Bus Factor Analysis, we can rule out the possibility that the bottlenecks are caused by the workload of the employees. This means that for all the employees it takes too much time to make full revision of the profile of a client. This suggest that the process is complex and might need auxiliary techniques to reduce the time it takes. A solution might be to make initial profile assessment using Big Data tools. Also, a recommender system might be useful to speed up the evaluation of clients' profiles.

4.4 New Approach: Recommender System

Currently, the bank has a vast event log, with over 30,000 traces with 4,047 variants. Each time there is a new applicant, this case is evaluated only by itself

and the previous history of similar cases might not be considered. In addition, if a new employee is hired, chances are this person does not have experience and does not understand completely already completed cases. Because of this, our goals are to aument the knowledge of cases and, most importantly, mitigate related to the assessment of clients' profiles. In order to do so, we defined a variant prediction system, that from a given incomplete case, returns an array of the most similar variants.

This system works as follows: first, uses recommender systems techniques to calculate similarity between all variants, building a similarity matrix. Second, when an incomplete case comes, we calculate similarity between incomplete case with all variant. If we find a variant with similarity equal to one, that means we found a perfect match and the program ends. Otherwise, we calculate similarity between all variants with the incomplete case. This is useful since we want to able to predict to which set of variants an incomplete trace belongs to; with this set, we calculate an estimated success rate.

Variant1	Variant2	Similarity
A_Create Application	A_Create Application	1
A_Submitted	A_Submitted	1
A_Concept	A_Concept	1
W_Complete application	W_Complete application	1
A_Accepted	A_Accepted	1
O_Create Offer	O_Create Offer	1
O_Created	O_Created	1
O_Sent (mail and online)	O_Sent (mail and online)	1
W_Call after offers	W_Call after offers	1
A_Complete	A_Complete	1
A_Cancelled	W_Validate application	0.90909
O_Cancelled	A_Validating	0.83333
	O_Returned	0.78493
	W_Call incomplete files	0.7429
	A_Incomplete	0.70601
	W_Validate application	0.6527
	A_Validating	0.60908
	O_Accepted	0.5855
	A_Pending	0.56412

Fig. 7: Similarity comparison between 2 variants

For example, if we compare *variant_1* (11 events) with *variant_2* (19 events) (see Figure 7), the first sequence of the first 10 events are equal in both, resulting in a similarity equal to 1, but as we add more events, the divergence reduces the similarity.

Similarity is obtained from a frequency vector for each variant of the activities (26 variables) of the first k events for each variant. Using the vectors, cosine similarity is calculated to measure distance, but others could be used like Jaccard and binary cosine, among others. However, in order to obtain a robust recommendation, calculating distance is not enough, since it does not consider the order of the activities.

In order to face this problem, We proposed to include the footprints or alignment conformance checking technique to reflect the sequence of events in the similarity value. The alignment consist on taking two cases and comparing their activities step by step. The comparing process begins with a counter in 0. We used the alignment in such a way that when we find an alignment match we added one to the counter. After that, we divide the counter by the maximum event length of the compared variants. For example, we calculated the alignment between variant 1 and 2 in Level 15, as a result, we obtained:

$$\frac{\text{matches}}{\text{total events}} = \frac{10}{15} = 0.66$$

Taking the cosine similarity and alignment, we added both and normalized the result (divide by 2), which gives us the similarity shown in Figure 7.

Success rate estimate and error metrics: Since we already defined how to calculate similarity between two cases, we proceed to estimate the success rate of an incomplete case. We propose a success rate estimate as a powerful tool to discriminate cases. With this, the bank can focus their resources into increasing the success rate of cases with high chance of success.

First, we calculate the success rate for all complete variants. Then, we calculate the success rate for the incomplete case using the set of the most similar variants (we took only five, but it can be any other number). In order to calculate the success rate of the similarity set, we calculated the weighted average between their success rates. The weight associated to each variant is its frequency in the event log.

The recommender system was validated with two of the most popular error metrics used in rating prediction:

1. MAE: Mean Absolute Error measures the error between the predicted rating (the result obtained from the recommender system) r_p and the real rating r_r chosen by each user, using absolute value.

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |r_p - r_r|$$

2. RMSE: Root Mean Square Error measures the error between the predicted rating (the result that recommender gives) r_p and the real rating r_r chosen

by each user,using root square. This metric was used in the Netflix Prize.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(r_p - r_r)^2}$$

We divided randomly the dataset into a training set and a test set (3,047 variant for training and 1,000 for testing). Utilizing the testing set, we carry out the following experiments:

1. **Complete**: predicts success rate using all events for all variant in test set.
2. **Incomplete. 5**: prediction uses at most the first 5 event for of each variant in test set to obtain results.
3. **Incomplete. 10**: prediction uses at most the first 10 event for of each variant in test set to obtain results.
4. **Incomplete. 15**: prediction uses at most the first 15 event for of each variant in test set to obtain results.

Table 10 shows the obtained results:

Table 10: RMSE and MAE in different levels

Error Metric	**Complete**	**Incomplete. 5**	**Incomplete. 10**	**Incomplete. 15**
MAE	0.33282	0.26672	0.41863	0.40045
RMSE	0.52592	0.48391	0.589987	0.58012

As can be seen, there is an anomaly with the result *Incomplete. 5*, which shows better results than the others. This should not be the case since it uses less information. Considering this, to get better results it might be necessary to include meta data, for example requested amount, into the recommender system. Other improvement could be the use of a more flexible version of alignment because if two variants with the sequences: a,b,c,d and e,a,b,c,d, the alignment we used will return 0. A more flexible approach would return a bigger number. Another option, is to include the percentage of similarity as weight when calculating the weighted average of a set's success rate. Finally, a RMSE greater than 0.5 shows that our approach of recommendation is poor [8].

4.5 Proposed question 2: effect of information incompleteness

The second question corresponds to: What is the influence on the frequency of incompleteness to the final outcome? The hypothesis proposed by the bank is that if applicants are confronted with more requests for completion, they are more likely to not accept the final offer.

In other words, they want to know about the quitting rate of the credit-getting process, resulting from the incompleteness of data by the customer. It is important to note that obtaining all the necessary data is a step prior to the

approval or rejection of a bank loan. We tried to answer this question by filtering cases that presented more than one data request instance.

Using the preprocessed data log, and after filtering for those cases that pass more than once through the *W_Validate Application* activity, a set of cases is obtained that represent the clients that must re-send data because they are incomplete. It should be noted that a few cases in this segment are reviewed for possible fraud, reaching the activity *W_Assess Potential Fraud*, but the vast majority enters into a cycle of activities involved in receiving and reviewing the applicant's data.

The filtering was done in *Disco*, by selecting the cases that entered the activity *A_Incomplete*, and that didn't pass through the activities *O_Accepted* or *O_Refused*. After obtaining the cases in question, we observed the behaviour of activities performed per case to determine the behaviour of cycles performed before leaving the data delivery cycle. As can be seen in Figure 8, they leaving the cycle as an accepted, refused or abandoned request on behalf of the applicant.

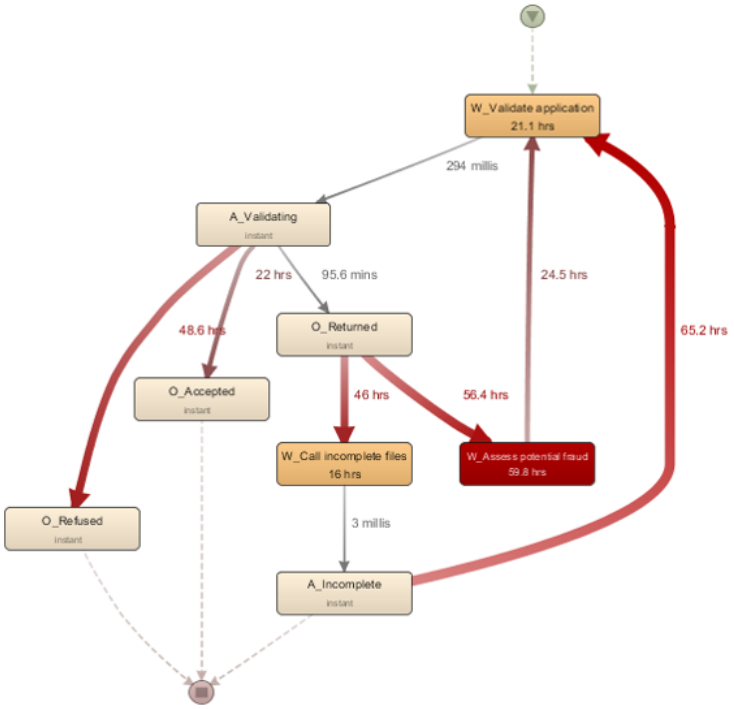

Fig. 8: Activities cycle in resend data cases.

After filtering the data, as shown in Figure 9, we could appreciate a clear trend of the applicants to quit from the process in each resending cycle. Counting the amount of activities done in each case, we found that they are grouped in

peaks after four activities: most of the applicants quit after 5 activities, then there's another peak of quit after 9 activities, then after 13, after 17 and so on. These peaks represent the cycles, because most of the applicants who quit, do it after another *A_Incomplete* activity. The most remarkable thing is the constant rate of quitting between the peaks: two of three cases are withdrawn in every peak, and after five peaks/cycles there are no more cases left.

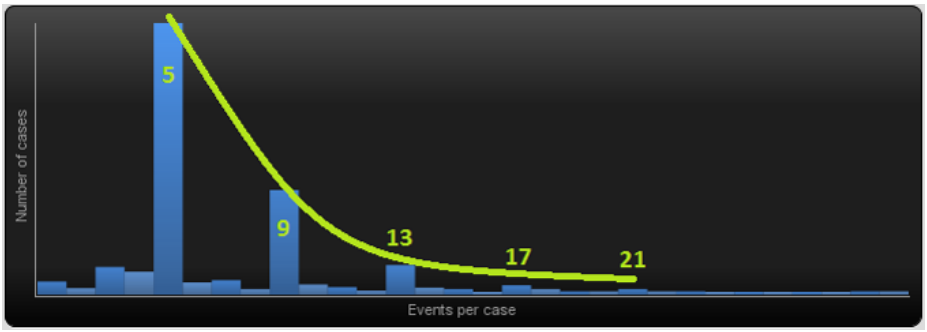

Fig. 9: Cases peaks and quitting rate.

Considering the high rate of resignation, 2/3, after each cycle of requests, it becomes necessary to stop the leakage of clients because of incomplete or poorly entered applications. In addition, each time the case enters the cycle, it faces a bottleneck of many days. In a simplified version of our cycle made with *ProM 5.2* (see Figure 10), we were able to confirm that avoiding to re-enter this cycle is important to get these applicants who leave the process after a new request.

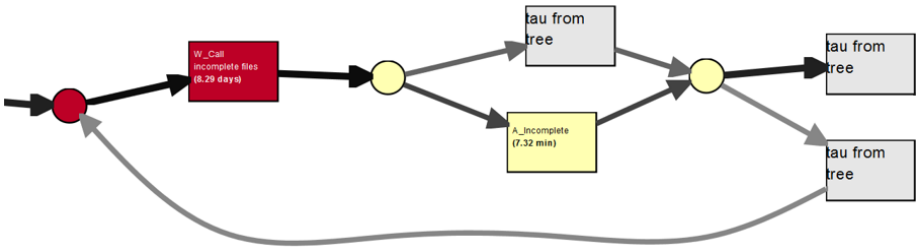

Fig. 10: Simplified version of the cycle, that shows the bottleneck at asking for more data.

It is the task of the bank's staff, and the institution itself, to improve the way they obtain data. This task includes not only changing the application forms, but facilitating the client sending data and documents throughout the process. It is suggested to minimize the amount of data requested at the beginning of

the application to bank loans and, once entered in the process, request new antecedents to continue forward. Specifically, just request basic data at the beginning to start the review cycle. If officials require more data to evaluate a possible fraud, then the applicant is requested to provide the necessary information. Otherwise, the applicants will not quit by an excess of requests of resending at this stage. Then, after *O_Accepted*, the applicant will be able to provide more information to the bank. However, if the applicant requests new offers, he will have to attach the necessary information in each new application.Specifically, just request basic data at the beginning to start the review cycle. If officials require more data to evaluate a possible fraud, then the applicant is requested to provide the necessary information. Otherwise, the applicants will not quit by an excess of requests of resending at this stage. Then, after *O_Accepted*, the applicant will be able to provide more information to the bank. However, if the applicant requests new offers (returning to *O_Create_Offer*, he will have to attach the necessary information in each new application.

4.6 Proposed question 3: multiples conversations vs single conversation

The third proposed question relates to clients who requested for multiple offers and if it is relevant if they are made in single or multiple conversations. Also, if there is a difference between client who receive single and multiple offers.

With only the process model, is not easy to distinguish which cases are inside the multiple offer category. To solve this, we used *Disco* to obtains all cases where an offer is created more than once (the activity *O_Create* is repeated), implying multiple Offer IDs for one trace.

We defined "different conversations" as the following concept: within the same trace, the repeated activity of creating an offer is executed by different users. A Python script was implemented to execute this filter. This script selects the first user that execute *O_Create*, after that selects next and compare.

Finally, Figure 13 shows that multiple offers with multiple conversations only have 1% of acceptance improvement when compared with Figure 12; the difference is so small that could be considered as insignificant. But, when compared with Figure 11 (multiple offers within one conversation, same User_ID), the acceptance rate increases in a 6% in cases with multiple offers in comparison with a single offer. The details about the number of cases in distinct scenarios are shown in table 11.

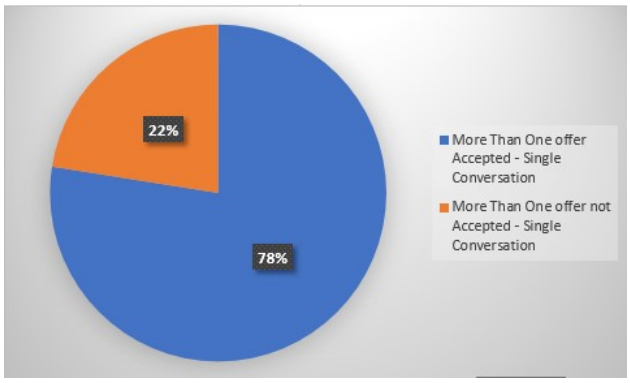

Fig. 11: Multiple Offer, single conversation: acceptance and rejection comparison

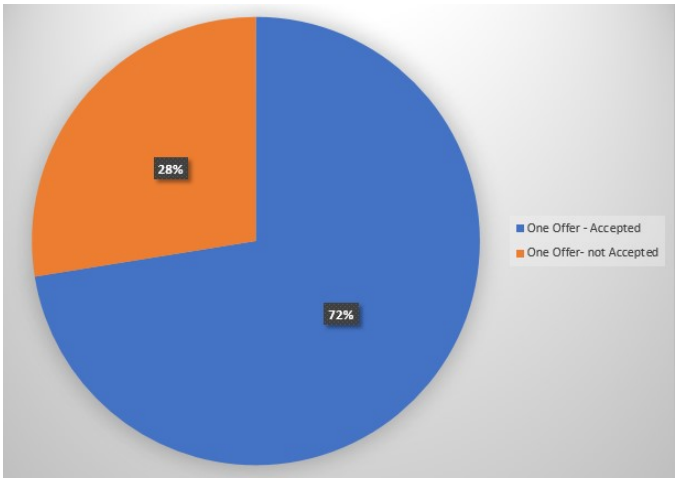

Fig. 12: Single Offer : acceptance and rejection comparisons

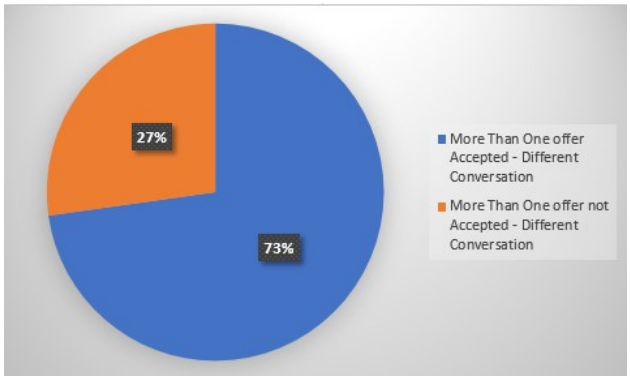

Fig. 13: Multiple Offer, multiple conversation : acceptance rate and rejection rate comparisons

Table 11: Number of cases in different scenarios

Scene	Number of cases
More Than one offer	547
One offer	22,948
One Offer - Accepted	6,628
One Offer- not Accepted	320
More Than One offer Accepted	376
More Than One offer not Accepted	171
More Than One offer Accepted - Different Conversation	820
More Than One offer Accepted - Single Conversation	556
More Than One offer not Accepted - Different Conversation	429
More Than One offer not Accepted - Single Conversation	72

5 Conclusions

During our analysis of the BPIC 2017 we found interesting insights about the process: the clients were the ones causing the most delays in the process, there was a high correlation between the incompleteness of the application and the final outcome. Additionally, we believe that the idea of comparing traces in order to predict the outcome of a client is a very interesting one and because of that is that we implemented a recommender system[5]. The recommender system uses all previous results and analysis in order to facilitate user profiling and to mitigate work-flow bottlenecks. Although we haven't used it to analyze the

[5] `https://github.com/farojos/BPI2017_script`

process as much as we would liked to, we encourage the bank to explore this approach. We expect this information to be useful understanding the system and its particularities.

It is crucial for every enterprise or institution to manage event logs, or some kind or registry for their operations. It would be very hard to obtain all the information we gathered and the solutions we proposed without event logs. Specifically, loan-obtaining logs across a large quantity of cases and a period of time. Nowadays, medium and big enterprises are making big efforts to track customers' choices and behavior, using benefit programs, their own credit cards, or just asking for personal data in every sale because we, the process miners, can improve business intelligence. With the used tools, we got many answers that couldn't have been reached at the same time span that we got, and with the same accuracy or capacity to anticipate customers' trends.

Finally, we could manage to simplify, study and analyze the event log provided by the BPI Challenge 2017, in an effort to answer the corresponding questions and the new ones that we found after mine the whole process log given. We expect that those tools will gain in popularity on a near future and will become an indispensable part of business management.

References

1. Business Process Intelligence Challenge (BPIC) 2017. Retrieved from http://www.win.tue.nl/bpi/doku.php?id=2017:challenge
2. Business Process Intelligence Challenge (BPIC) 2012. Retrieved from http://www.win.tue.nl/bpi/doku.php?id=2012:challenge
3. Wil M.P. van der Aalst. (2016). Process Mining - Data Science in Action. Springer.
4. Günther, C. W., and Rozinat, A. (2012). Disco: Discover Your Processes. BPM (Demos), 940, 40-44.
5. Van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., and Van Der Aalst, W. M. (2005, June). The prom framework: A new era in process mining tool support. In ICATPN (Vol. 3536, pp. 444-454).
6. Arjel D. Bautista, Lalit Wangikar and Syed M. Kumail Akbar : Process Mining-Driven Optimization of a Consumer Loan Approvals Process - The BPIC 2012 Challenge
7. V. Cosentino, J. L. C. Izquierdo and J. Cabot, "Assessing the bus factor of Git repositories," 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Montreal, QC, 2015, pp. 499-503. doi: 10.1109/SANER.2015.7081864
8. Veerasamy, R., Rajak, H., Jain, A., Sivadasan, S., Varghese, C. P., & Agrawal, R. K. (2011). Validation of QSAR models-strategies and importance. International Journal of Drug Design & Discovery, 3, 511-519.