Mining Event Log Data to Improve a Loan Application Process

Frank Berger
frank.berger@exploras.ch

Exploras GmbH, Bahnhofstrasse 78, 5001 Aarau, Switzerland

Abstract. The event data of a loan application process is analysed. Process mining techniques are applied to discuss the process and to answer specific questions of the process owner. Opportunities for process improvement are discussed and recommendations are made. Various KPI, such as manual processing time and process cost are calculated, and the relationship between different KPI is explored. Finally, process mining is combined with statistical process control techniques to analyse relevant KPI over time, and to identify additional improvement areas. The combination of the two techniques turns out to be very powerful and provides a basis for continuous process improvement.

Keywords: BPIC'17, process mining, activity based costing, statistical process control, time series analysis, XmR-charts, process improvement

1 Introduction

This paper takes up the international Business Process Intelligence Challenge (BPIC) 2017 [1]. The challenge is presented by the organizers of the 13th international workshop on Business Process Intelligence 2017 in Barcelona, Spain. It is based on a real live event log of the loan application process of a dutch financial institute [2]. Participants are asked to analyse the data using any appropriate tools and techniques. They should focus on the process owner's questions and provide any other relevant insights into the underlying process. Questions about the data could be posted in an online forum. Beyond that, no domain knowledge was available to participants of the challenge.

The event log contains loan applications filed through an online system in 2016, until February 1st 2017. The financial institute providing the event log will be referred to as "the bank" in this paper. In fact, two log files were made available by the bank: an application event log and an offer event log. However, the offer log is just a subset of the application log. The data sets are public and fully anonymised. In total, the application log contains 31'509 loan applications (or cases) with 42'995 offers and ∼1.2 Mio events. There are three types of events: Application state changes, Offer state changes and Workflow events. These types represent three intertwined sub-processes. The first letter of each event name in the log identifies the sub-process it originated from.

2 Overview of the process

2.1 Big picture overview with the dotted chart

To get a big picture overview of the process, a dotted chart [3] was created using ProM-Lite. A dotted chart shows all process events as dots on a timeline. ProM-Lite is an open-source process-mining software popular in academic research. Fig. 1 shows the time line on the x-axis and the cases on the y-axis. The volume of

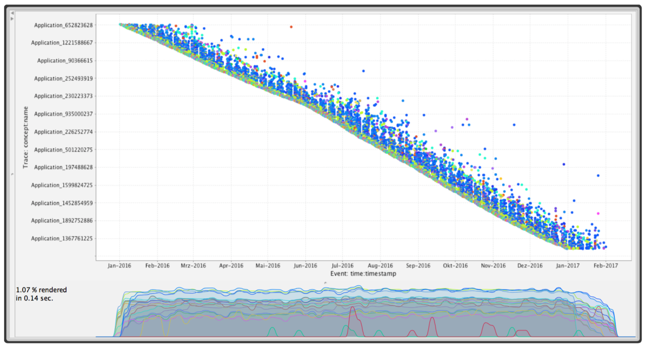


Fig. 1. Dotted chart [3] of all process events in the event log

cases is fairly stable, with a slight increase around June 2016. When zooming in, a weekly pattern appears, with high activity occurring from Mondays to Fridays, reduced activity on Saturdays and almost no activity on Sundays. Setting the x-axis of the dotted chart to indicate the time since the start of the case, and sorting cases according to their duration, gives us the distribution of case durations. The resulting Fig. 2 reveals two broad case patterns: the densely populated top

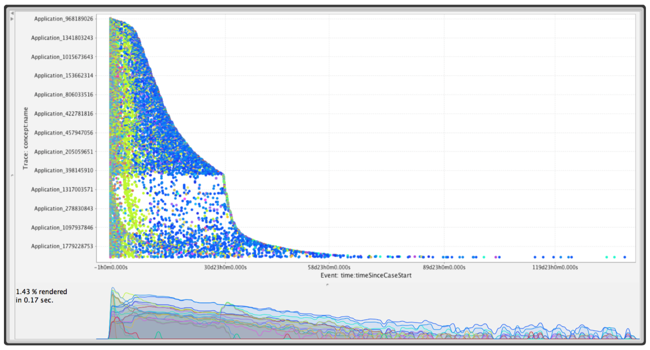


Fig. 2. Distribution of the case durations: two broad patterns can be observed

part of the graph, accounting for about 60% of the cases, shows an increase in case duration, with a slight S-curve up until 30 calendar days. We then see a very sparsely populated second category of cases, that all have a similar duration between 30 and 33 calendar days. They account for roughly one third of all cases. The remaining ∼10% bottom part of the graph actually looks like a seamless continuation of the first category, yet displaying a very rapid increase in case duration, ending with a few extreme outliers. In the first category the pareto principle seems to be at work, while the second category looks as if the cases had been consciously terminated after ∼30 days of little or no activity.

2.2 A_events process description

Every application walks through a series of statuses, represented by the A_events. According to the process owner's description, the A_events can be interpreted as follows:

A_Submitted: a customer has submitted a new application from the website. A new application can also be started by the bank, in that case this state is skipped.

A_Concept: the application is in the concept state, that means the customer just submitted it (or the bank started it), and a first assessment has been done automatically. An employee calls the customer to complete the application.

A_Accepted: after the call with the customer, the application is completed and assessed again. If there is a possibility to make an offer, the status is accepted. The employee now creates 1 or more offers.

A_Complete: the offers have been sent to the customer and the bank waits for the customer to return a signed offer along with the rest of the documents (payslip, ID, etc.).

A_Validating: the offer and documents are received and are checked. During this phase the status is validating.

A_Incomplete: if documents are not correct or some documents are still missing, the status is set to incomplete, which means the customers needs to send in documents.

A_Pending: if all documents are received and the assessment is positive, the loan is final and the customer is paid.

A_Denied: if somewhere in the process the loan cannot be offered to the customer, because the application doesn't fit the acceptance criteria, the application is declined, which results in the status 'denied'.

A_Cancelled: if the customer never sends in his documents or calls to tell he doesn't need the loan, the application is cancelled.

Process flow of the A_events Since Celonis is sponsoring the challenge, their process mining tool is used to discover the overall process flow. If we include all event types, this results in a complex model, even when filtering out less frequent activities and connections. To simplify the model, we filter out all offer-related O_events and workitem-related W_events. The remaining A_events give a nice big picture overview of the application process (Fig. 3). We can see that 35% of all applications skip the A_Submitted event. According to the above description, such applications are started by the bank. They are probably coming in via a different channel, such as an advisor in a branch, and not the customer-facing online system. After reaching A_Complete, 9'307 (30%) of the applications are

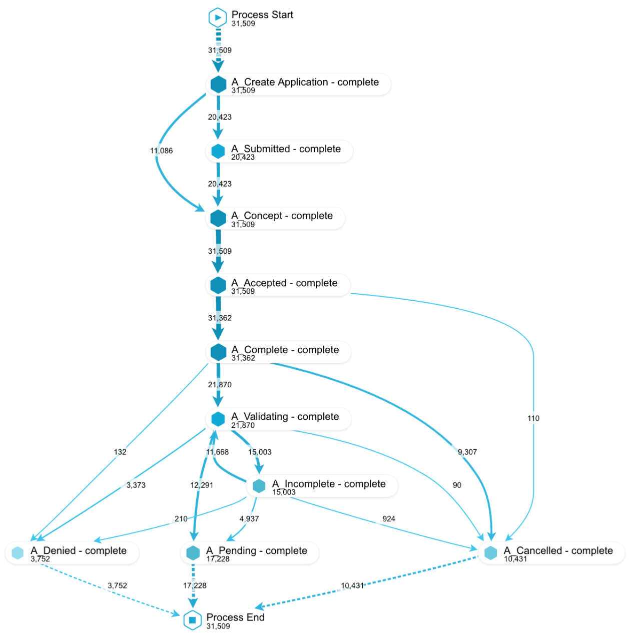


Fig. 3. A_events process model

branching off to A_Cancelled. This means that the application is cancelled, because the customer never sends in his documents, even after receiving an offer.

The throughput time view shows a median duration of 31 days for these cases. This matches the above observation in the dotted chart. These are lost customers who showed initial interest, but then never sent in the required documents. A quick filtering shows that lost customers occur in both channels, although the proportion in the online channel is a bit higher (34% vs 22%). This makes intuitive sense, because we expect a human interaction to create a higher level of commitment than a pure online application submission. However, other explanations are possible, as we shall see in a later analysis.

Out of the 21'870 applications that send back documents, 15'003 (∼70%) are incomplete. The status is set to A_Incomplete and the customer has to send in documents again. This loop can occur more than once. Finally, one of the three end states is reached: "A_Pending" for accepted applications (55%), "A_Cancelled" if the customer is no longer interested (33%), and "A_Denied" if the bank declines the application (12%).

2.3 O_events process description

To get an overview of the offer flow we conduct a second analysis focusing on O_events (Fig. 4). We can see that most A_events from the above analysis have

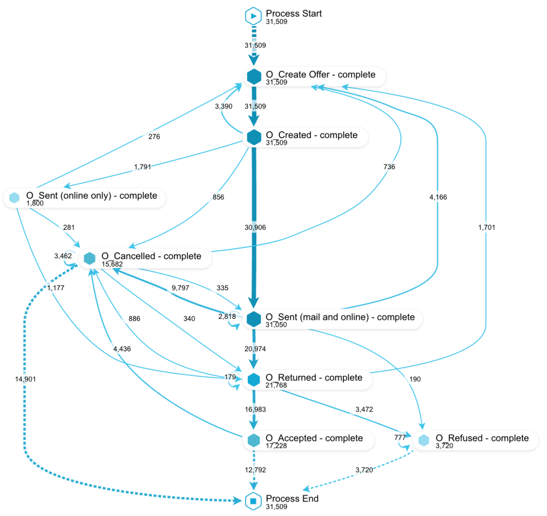


Fig. 4. O_events process model

a corresponding O_event representing the offer states. As an example, the three final outcomes are O_Accepted, O_Cancelled and O_Refused, just as their corresponding A_events with slightly different names. We can also see how multiple offers can be created for one application. This is evidenced by the loop from "O_Created" back to "O_Create Offer", and the subsequent self-loops at "O_Sent (mail and online)". Since only one offer can be accepted for a single application, all other offers have to be discarded. This is confirmed by the large number of self-loops at the "O_Cancelled" and "O_Refused" events.

2.4 W_activities process description

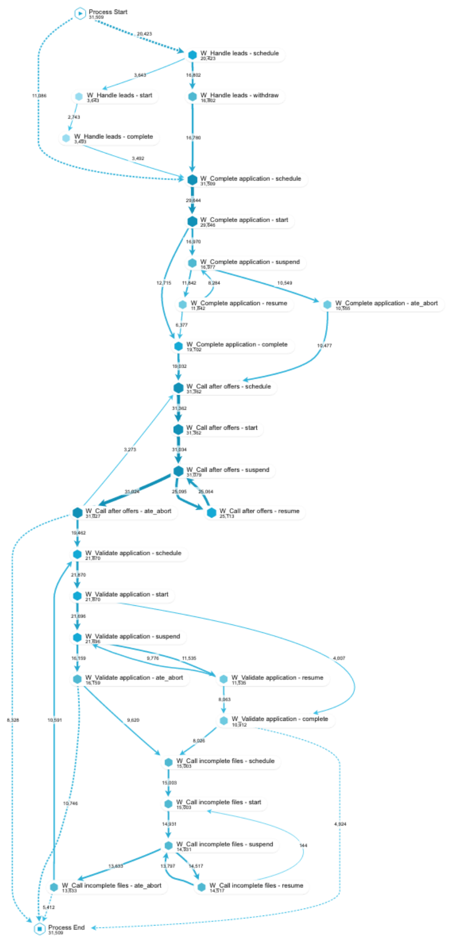

Process Start
31,509
20,423
W_Handle leads - schedule
20,423
3,643
16,802
11,086
W_Handle leads - start
3,643
W_Handle leads - withdraw
16,802
2,743
W_Handle leads - complete
3,493
16,780
3,492
W_Complete application - schedule
31,509
29,644
W_Complete application - start
29,646
16,970
W_Complete application - suspend
16,977
12,715
11,842
8,284
10,549
W_Complete application - resume
11,842
W_Complete application - ate_abort
10,555
6,377
W_Complete application - complete
19,102
10,477
19,032
W_Call after offers - schedule
31,362
31,362
W_Call after offers - start
31,362
3,273
31,034
W_Call after offers - suspend
31,079
31,024
25,095
25,064
W_Call after offers - ate_abort
31,027
W_Call after offers - resume
25,113
19,462
W_Validate application - schedule
21,870
21,870
W_Validate application - start
21,870
21,696
W_Validate application - suspend
21,696
16,159
9,776
11,535
4,007
W_Validate application - ate_abort
16,159
W_Validate application - resume
11,535
8,063
8,328
10,591
9,620
W_Validate application - complete
10,912
8,026
W_Call incomplete files - schedule
15,003
15,003
10,746
W_Call incomplete files - start
15,003
14,931
4,924
W_Call incomplete files - suspend
14,931
144
13,833
13,797
14,517
W_Call incomplete files - ate_abort
13,833
W_Call incomplete files - resume
14,517
5,412
Process End
31,509

Fig. 5. W_activities process model

Finally, we analyse the W_activities. These are workitems for call agents to process. Just as the offer-related events, workitems are associated with a corresponding application state, indicated by the A_events. Unlike the application- and offer-related events, W_activities can have up to seven different lifecycle transitions. This adds an additional level of complexity. However, we still get a good sense of what's going on after filtering out less frequent activities and connections (Fig. 5, 7–9). For example, we can see that the lifecycle transitions of most workitems follow a typical pattern:

schedule → start → (suspend) → (resume) → complete or ate_abort

W_activities can run through the "suspend-resume"-cycle several times. This matches the transactional model for activities in the XES standard definition [4] (Fig. 6). We can therefore assume that the actual manual work is carried out

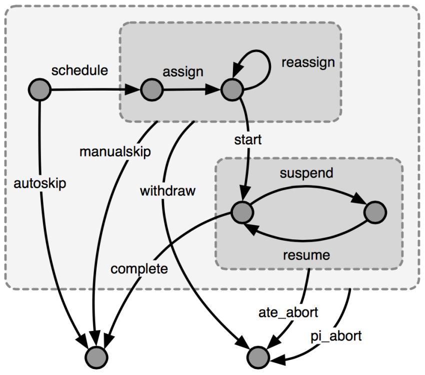


Fig. 6. XES standard transactional model for activities [4]

between the following pairs of consecutive events:

– start → suspend
– start → complete
– resume → suspend
– resume → complete

All other pairs of consecutive events represent waiting time in this model. This insight will be useful later on, because it allows to calculate manual activity durations and therefore the labour cost of processing an application.

In Fig. 5 we can also re-recognize the 11'086 applications that are started by the bank, skipping the "W_Handle leads" activities. For the 20'423 applications coming in through the e-channel, a first assessment should be done automatically, according to the description of the A_events. However, we can see that 3'643 (18%) of them still require manual processing (Fig. 7). The subsequent "W_Complete application" activity represents a phone call to the customer to complete the application, according to the process owner. The large number of "suspend-resume"- loops suggests that there are many unsuccessful attempts to reach customers by phone (Fig. 8).

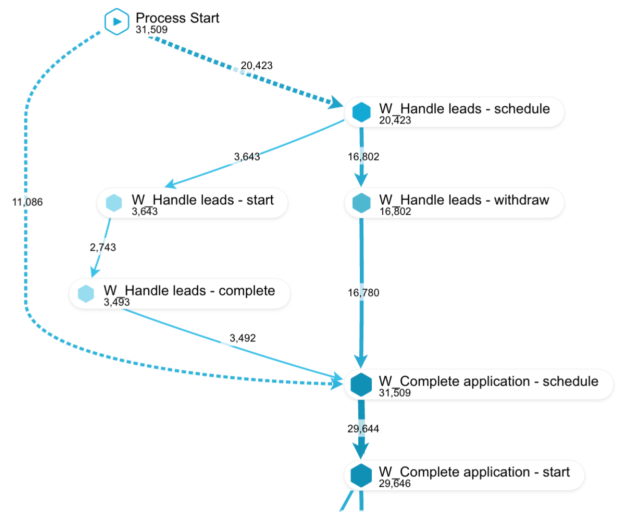

Process Start
31,509
20,423
11,086
W_Handle leads - schedule
20,423
3,643
16,802
W_Handle leads - start
3,643
W_Handle leads - withdraw
16,802
2,743
W_Handle leads - complete
3,493
16,780
3,492
W_Complete application - schedule
31,509
29,644
W_Complete application - start
29,646

Fig. 7. 3'643 (18%) of the applications need a manual first assessment

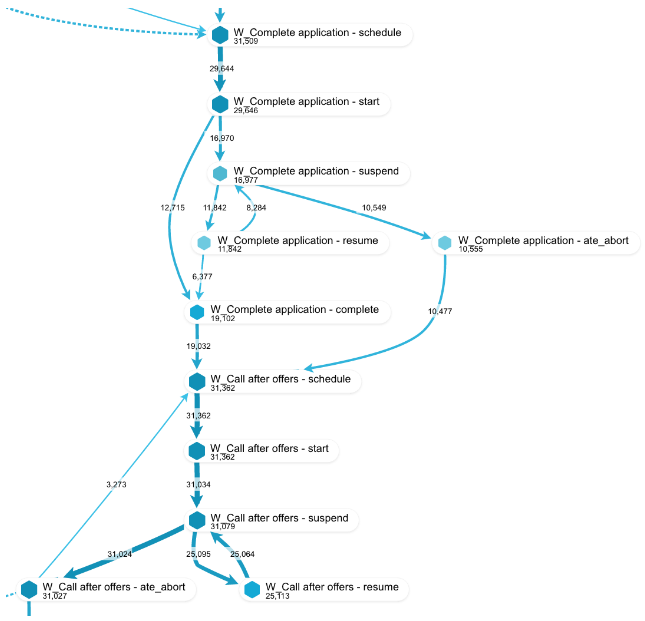

W_Complete application - schedule
31,509
29,644
W_Complete application - start
29,646
16,970
W_Complete application - suspend
16,977
12,715
11,842
8,284
10,549
W_Complete application - resume
11,842
W_Complete application - ate_abort
10,555
6,377
W_Complete application - complete
19,102
10,477
19,032
W_Call after offers - schedule
31,362
31,362
W_Call after offers - start
31,362
3,273
31,034
W_Call after offers - suspend
31,079
31,024
25,095
25,064
W_Call after offers - ate_abort
31,027
W_Call after offers - resume
25,113

Fig. 8. Completing applications and calls after offers

Next in Fig. 8, we can see the "W_Call after offers" activity. According to the process owner, the customer is called after a few days of sending the offer to remind him about it, ask if he received everything, still has any questions, is planning to accept the offer, etc. Again, the very large number of "suspend-resume"-cycles suggests many unsuccessful call attempts (Fig. 8).

Finally, we can see the activities associated with validating incoming offers and calling customers again in case of incomplete documents (Fig. 9). Here too, the map suggests many unsuccessful attempts to reach customers by phone.

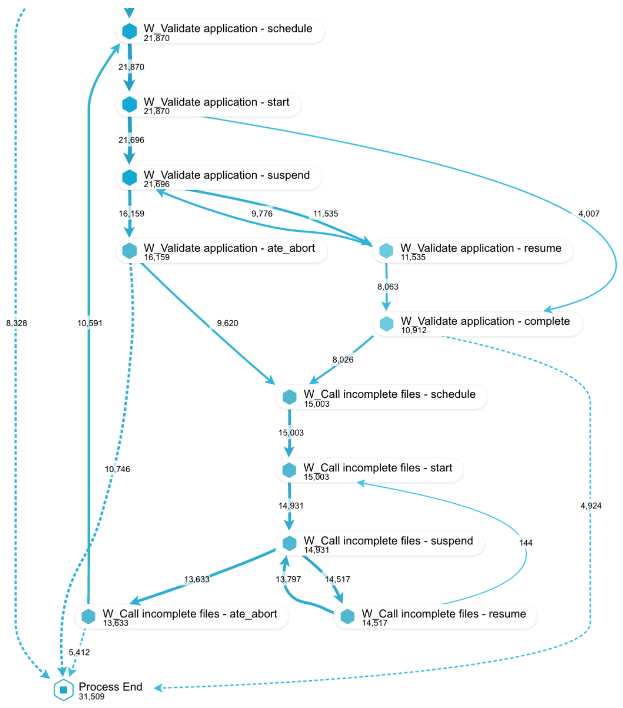

W_Validate application - schedule
21,870
21,870
W_Validate application - start
21,870
21,696
W_Validate application - suspend
21,696
16,159
9,776
11,535
4,007
W_Validate application - ate_abort
16,159
W_Validate application - resume
11,535
8,063
8,328
10,591
9,620
W_Validate application - complete
10,912
8,026
W_Call incomplete files - schedule
15,003
15,003
10,746
W_Call incomplete files - start
15,003
14,931
4,924
W_Call incomplete files - suspend
14,931
144
13,633
13,797
14,517
W_Call incomplete files - ate_abort
13,633
W_Call incomplete files - resume
14,517
5,412
Process End
31,509

Fig. 9. Validating incoming offers and calling customers in case of incomplete files

3 Answering questions of the process owner

3.1 Throughput times

Question 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?"*

To answer this question, we go back to the process description with application-related A_events. We choose the Throughput Time view with a trimmed mean, to remove the most severe outliers. As we can see in Fig. 10, the longest delay is between A_Complete and A_Cancelled (29 days). These are the 30% of

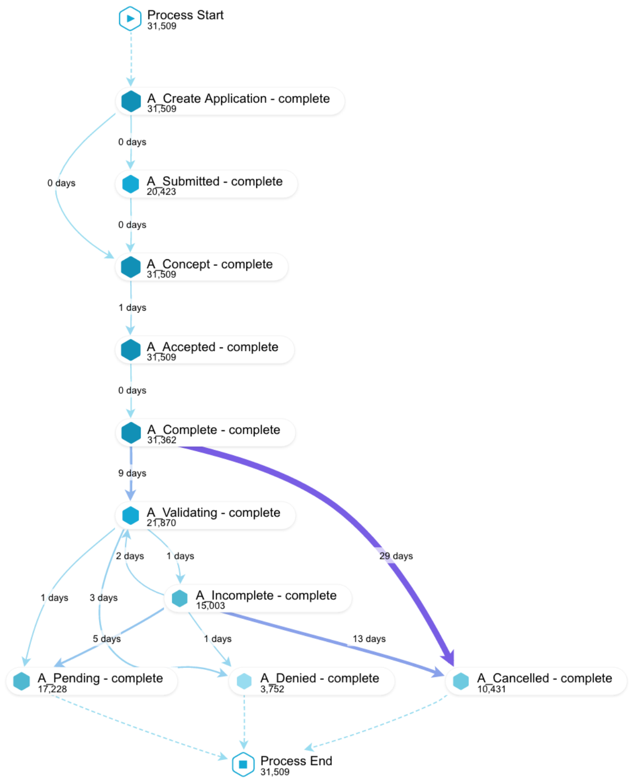


Fig. 10. Throughput times between the A_events

customers who never send in their documents. There might be a rule in the workflow system that cancels such applications after one month. This delay is clearly spent waiting for input from the applicant. The same is true for the path leading from A_Incomplete to A_Cancelled (13 days). The path from A_Complete to A_Validating (9 days) represents waiting for the customer to return a signed offer. The same is true for the delay between A_Incomplete and A_Validating (2 days), when the company is waiting for the applicant to provide additional documents. When adding more details by relaxing the filter, we can see that the bank also waits for customer's input in the delay between A_Incomplete and A_Pending (5 days). So overall, customers take quite some time returning their documents, if they react at all.

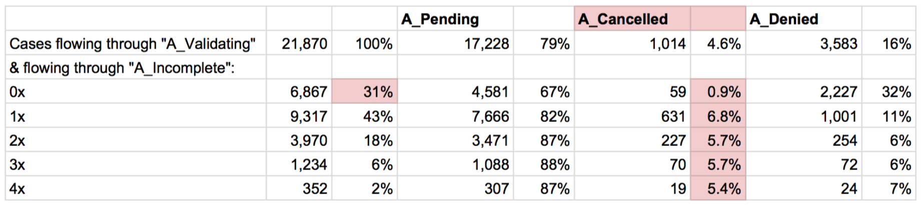
The overall throughput time of the entire process can be seen in the process overview tab of Celonis. The trimmed mean is 21 days for all cases. However, that figure alone is less interesting, because there is such a wide variation (see Fig. 2), and because most delays are caused by customers.

The one relevant delay that seems due to waiting for an internal user is A_Validating → A_Incomplete. That's the time between validating the received offer and recognising that the documentation is incomplete. This path often occurs more than once in an application, in total 23'052 times. Although the trimmed mean is only 1 day, there is quite some variation: for example, the delay is longer than 4 days in 16% of all occurrences and longer than 5 days in 10% of occurrences. Some of this will be due to weekends in between, and the average delay is short, compared to the ones caused by customers. So overall, responsiveness of call agents does not seem like a big problem. However, this doesn't mean that all is perfect. The long waiting times for applicant's inputs are not ideal, and are a result of the process design, as will be discussed later.

3.2 Frequency of incompleteness

Question 2: *"What is the influence of the frequency of incompleteness on the final outcome? The hypothesis here is that if applicants are confronted with more requests for completion, they are more likely to not accept the final offer."*

As we have seen in the previous Fig. 3, 21'870 applications had reached the A_Validating state, meaning that they returned their documents after receiving an offer. The above question can be answered with a conditional filter for the A_Validating state and a subsequent rework filter on A_Incomplete. One can play with the number of occurrences of A_Incomplete and look up the number of cases for each of the three possible outcomes. The result can be seen in Fig. 11. The table shows that only 31% of applicants, that had sent back the offer,



			A_Pending		A_Cancelled		A_Denied	
Cases flowing through "A_Validating"	21,870	100%	17,228	79%	1,014	4.6%	3,583	16%
& flowing through "A_Incomplete":								
0x	6,867	31%	4,581	67%	59	0.9%	2,227	32%
1x	9,317	43%	7,666	82%	631	6.8%	1,001	11%
2x	3,970	18%	3,471	87%	227	5.7%	254	6%
3x	1,234	6%	1,088	88%	70	5.7%	72	6%
4x	352	2%	307	87%	19	5.4%	24	7%

Fig. 11. Final outcome depending on the frequency of incompleteness

had zero requests for completion. The remaining ~70% had to resend documents at least once. The answer to the above question can be seen in the A_Cancelled column. Only 0.9% of customers with zero requests for completion did not accept the final offer and cancelled. One request for completion resulted in a 6.8% cancellation rate. That didn't change significantly, as the number of completion requests rose further. The cancellation rate remained stable at 5-6%. The above hypothesis can therefore be partly confirmed, with additional clarity: as soon as there is at least one completion request, the cancellation rate increases by 5-6 percentage-points. Further requests don't create additional cancellations. In absolute numbers these were ∼1'000 applications in 2016, or ∼3% of the grand total. There are two plausible explanations for the increased cancellation rate: a request for completion could be perceived as a bureaucratic burden by some applicants, leading them to lose interest. Another explanation could be that some of these applicants realize that they probably won't qualify for a loan, and therefore pull out of the process.

Note, that looking at the successful outcome in the column A_Pending only, may lead to a false conclusion: with more requests for completion, the acceptance rate by the bank steadily increases from 67% to 88%, making that appear more successful. Yet this increase merely mirrors the steep decline of the refusal rate by the bank (A_Denied) from 32% to 6%. An obvious explanation is, that the clearly non-qualifying applicants have been refused already in the first validation round. Refusing non-qualifying applicants is important from a risk management perspective. So the target function for the bank is not to maximise the rate of A_Pending, but to minimise the rate of A_Cancelled.

What should the bank do? We don't know how many of the 1'000 "lost" applications are due to which of the above explanations. It's therefore safe to assume, that not all of that potential can be captured by avoiding requests for completion. These 1'000 applications also need to be put into perspective. As discussed in the A_events process description, the vast majority of cancellations occurred much earlier in the process: 9'307 (30%) of applicants never sent in their documents in the first place. This is a much bigger lever to avoid cancellations.

Reducing requests for completion On the other hand, avoiding completion requests also makes sense from a process point of view. Correcting or completing documents represents rework for both the applicant and the bank. In the lean philosophy, this is a form of waste. Here, this happens with 70% of the relevant applications, which clearly seems too high. A specific recommendation would require more domain knowledge about the customer interface and the communication towards the applicants. However, a general improvement strategy in such a situation is to "front-load" the process. This means, that all the necessary information exchanges should be moved towards the beginning of the process. The bank should make it easy for customers to provide all required information at the beginning, in the right form. This also includes so-called mistake-proofing mechanisms. For example, online forms with free text, or requesting paperwork through postal mail, leave much room for mistakes. Obviously, what can go wrong, will go wrong. It is recommended to use classical process improvement methods such as FMEA (failure mode and effects analysis) and root-cause analysis of completion requests to define the right improvement measures. A possible short-term measure could be to improve the customer communication, while medium-term the bank might decide to redesign the front-end of the online system to prevent mistakes. A longer-term measure could be moving to a fully digitized application process with real-time detection of incomplete information and mistakes. Process mining as part of an ongoing improvement cycle can help verify the impact of such process improvements and reduce requests for completion.

3.3 Number of offers per application

Question 3: *"How many customers ask for more than one offer? (where it matters if these offers are asked for in a single conversation or in multiple conversations). How does the conversion compare between applicants for whom a single offer is made and applicants for whom multiple offers are made?"*

Two offers per application: To operationalise these questions, we inspect some applications with multiple offers. Conceptually, a clear distinction between single and multiple conversations can be made if there are exactly two offers for an application. In this case, the typical pattern of O_events looks as follows:

O_Create Offer → O_Created → O_Create Offer → O_Created → O_Sent (mail and online) → O_Sent (mail and online).

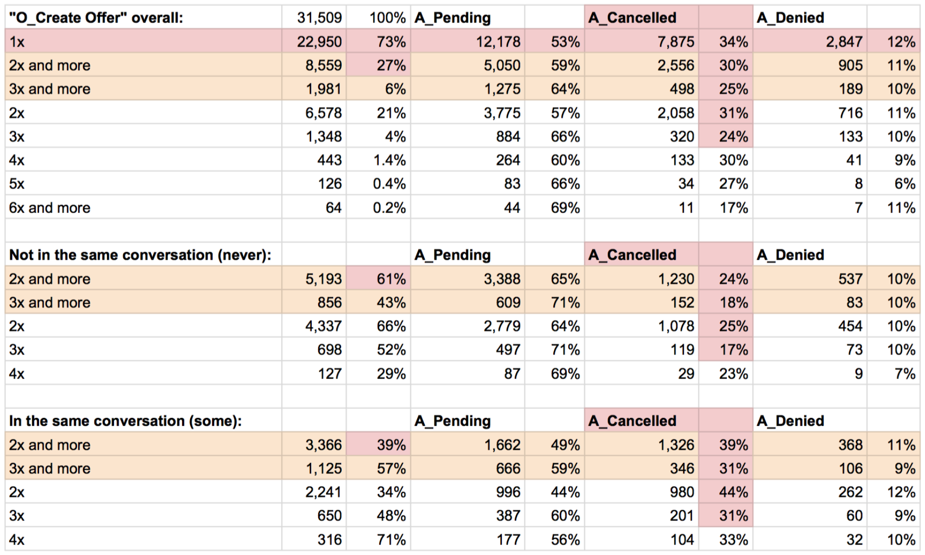
First, two offers are created, and then they are sent out one after another. The event order and the timestamps suggest that the two offers are created in a single conversation. Such a pattern can be interpreted as a customer asking for two offers in the first phone call. This pattern can be filtered with a process flow filter in Celonis, demanding a direct succession of O_Created → O_Create Offer.

If two offers are created in multiple conversations, then there are other events and also a time delay between the two offer creation steps. The interpretation would be that a customer finds out that he actually wants a different loan structure after receiving a first offer. Such a pattern can be filtered in a similar way, with slightly different filter settings. As before, we then look up the number of cases for each of the three final loan decisions.

Three and more offers per application: If there are three or more offers per application, the distinction between single and multiple conversations gets blurred. That's because most such cases have two offers in the first conversation, and a third offer later on. To have the two patterns mutually exclusive and collectively exhaustive, we stipulate the following definition:

- Different conversations = The different offers were *never* created in the same conversation (none of them).
- Same conversation = *Some* of the different offers were created in the same conversation.

Playing with an additional rework filter provides the numbers for each scenario and answers the first part of question 3. The result can be seen in Fig. 12. Numbers not adding up exactly are due to unfinished cases. 27% of customers asked for more than one offer. Of these, 61% are created in different conversations and 39% are created in the same conversation. With more offers, the number of cases falls rapidly. Fig. 12 provides both cumulated and individual frequencies. For example, only 6% asked for three or more offers.



"O_Create Offer" overall:	31,509	100%	A_Pending		A_Cancelled		A_Denied	
1x	22,950	73%	12,178	53%	7,875	34%	2,847	12%
2x and more	8,559	27%	5,050	59%	2,556	30%	905	11%
3x and more	1,981	6%	1,275	64%	498	25%	189	10%
2x	6,578	21%	3,775	57%	2,058	31%	716	11%
3x	1,348	4%	884	66%	320	24%	133	10%
4x	443	1.4%	264	60%	133	30%	41	9%
5x	126	0.4%	83	66%	34	27%	8	6%
6x and more	64	0.2%	44	69%	11	17%	7	11%
Not in the same conversation (never):			A_Pending		A_Cancelled		A_Denied	
2x and more	5,193	61%	3,388	65%	1,230	24%	537	10%
3x and more	856	43%	609	71%	152	18%	83	10%
2x	4,337	66%	2,779	64%	1,078	25%	454	10%
3x	698	52%	497	71%	119	17%	73	10%
4x	127	29%	87	69%	29	23%	9	7%
In the same conversation (some):			A_Pending		A_Cancelled		A_Denied	
2x and more	3,366	39%	1,662	49%	1,326	39%	368	11%
3x and more	1,125	57%	666	59%	346	31%	106	9%
2x	2,241	34%	996	44%	980	44%	262	12%
3x	650	48%	387	60%	201	31%	60	9%
4x	316	71%	177	56%	104	33%	32	10%

Fig. 12. Final outcome depending on the number of offers made

Impact on conversion For the second part of question 3, we need to define "conversion". At first glance, a conversion is defined as the successful outcome A_Pending. However, this can lead to false conclusions, if different conversion rates are just the result of different refusal rates (as we have seen in question 2). Even though this effect is small here, it is safer to focus on minimizing cancellations by customers (A_Cancelled), just as we did in question 2. Applications with one offer have a cancellation rate of 34%. If more offers have been created in *different* conversations, the cancellation rate falls steadily, which is good (25% for 2 offers, 17% for 3 offers). One has to be careful though, as the absolute numbers in the denominator of these proportions get smaller, leading to higher uncertainty. One possible interpretation of the falling cancellation rate could be that customers increase their commitment, if they ask for additional offers after receiving one. Or the other way round, committed applicants tend to ask for additional offers. In any case, if customers ask for additional offers after receiving one, this is a good sign and reduces their chances to defect.

However, if several offers have been created in the *same* conversation, their cancellation rate actually tends to be higher compared to one offer, which is bad (44% for 2 offers, 31% for 3 offers). A quick significance test suggests that the differences to the previous rates are not due to chance. Various interpretations of this counterintuitive result are possible: these applicants might have considered several competing offers from more than one bank from the beginning. Or, if every single offer involves a lot of paperwork, they might also feel overwhelmed and have a higher tendency to pull out of the application process.

In any case, the safe recommendation is that call agents should not proactively suggest multiple offers during a conversation. However, if the customer explicitly asks for it, or if he has already received one, an additional offer will be beneficial.

4 Other interesting trends and dependencies

4.1 Opportunities for process improvement

As we have seen in the W_activities process description, the application process is not very smooth. Outbound calls at three different stages, little predictability of customer's actions and a lot of rework are designed into the process. For example, we might question why an employee has to call the customer every time to complete the application, after it has been submitted. An ideal process would make sure that customers provide all required information at the beginning. As an example, customers could provide salary information and upload a scanned payslip right at the start. This would also eliminate the problem of unsuccessful call attempts. This is another manifestation of the strategy to "front-load" the process, as we have seen in the answer to question 2. Designing outbound calls into a transactional service process disrupts the flow and tends to create inefficiencies.

Closer inspection of the throughput times reveals that the first "W_Call after offers" activity occurs at the same time as the previous call to complete the application. As these must therefore happen in the same call, this first instance isn't really a "call after offers". Maybe the system generates a first "W_Call after offers"-event automatically, or call agents have to select that activity as a standard procedure. For most applications, another "W_Call after offers" activity occurs 4 days later. This must therefore be a real call after offers. According to the process owner's description, this call reminds customers of the offer, offers to answer any questions, etc. On one hand, this could be considered a form of waste. If everything is well-designed and self-explanatory, there is no need to proactively ask for any questions. On the other hand, there is also a sales element involved. "Nudging" customers with a human interaction may reduce defections and therefore increase conversion. The question remains however, if "successful" follow-up calls are just windfall gains, and customers would have sent back the signed offer anyway. After all, the calls after offers didn't prevent the 30% of customers that never sent back anything. To get more clarity on the real benefit of calls after offers, the bank might consider conducting an A/B-test. Calls after offers could be omitted for a test group, and their cancellation rate compared to the control group.

Longer term, the bank might consider a fully digitized process that offers loan decisions to customers in near real-time. This would of course eliminate all of today's calls, not just calls after offers. Right now, only the customer-facing part of the process seems to be digitized by the online system. All subsequent activities still look like a traditional paper-based process squeezed into a workflow system, with heavy "back and forth". As today's consumers increasingly adopt a "now and here"-mentality, end-to-end digitization is probably the way forward. A claim such as "Get a loan in 24 hours" could well become a standard industry benchmark in the future. In any case, such a value proposition would be a competitive advantage for the bank and could reduce the 30% of defections. Of course, this requires many details to be solved, and strong resistance is to be expected from the bank's legal department. For example, a physical signature from customers may still be required. The challenge will be to avoid that creativity gets crumbled by seemingly unchangeable ways of doing business.

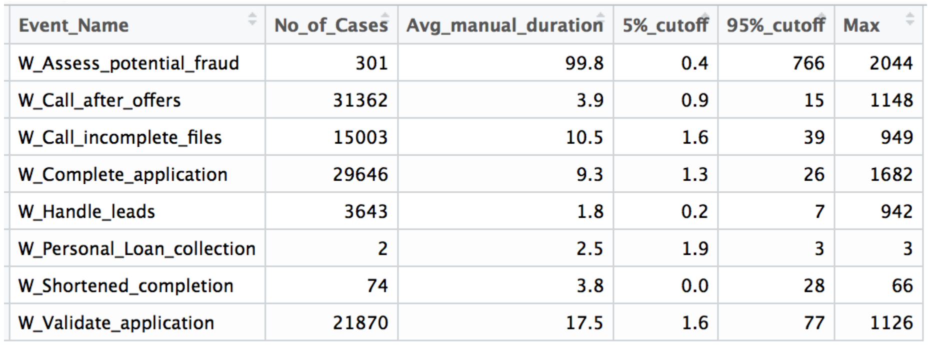
The third stage of outbound calls are the requests to complete documents. In the answer to question 2, this 70% rework rate has already been discussed in depth, and what the bank could do about it.

4.2 Calculating process cost

So far, the design of the process has been discussed without considering its cost. As we have seen in the W_activities process description, the transactional model for activities helps us calculate the manual activity durations. Process mining therefore makes it possible to conduct activity-based costing that is based on real process data. To extract the KPI from the event log, RStudio has been used, an open-source software for data analysis. As R is a programming language, its flexibility is well-suited for the considerable data-wrangling involved in the calculations.

As discussed in the W_activities process description, there are four different pairs of consecutive lifecycle transitions that represent manual work. The time difference between all four relevant pairs was calculated for every workitem of a case. The resulting durations were then aggregated by workitem for every case and added as new variables. Exploratory analysis revealed that the resulting distributions of the manual activity durations have some extreme outliers. In a few cases, a single manual activity seems to have taken more than 16 hours and finished in the middle of the night. Even without any domain knowledge, this clearly doesn't make sense. Closer inspection of these outlier cases suggests that this must be due to a data quality problem. In these cases, call agents most likely forgot to change the status of a started or resumed activity to "complete" or to "suspend". The status change was only done hours later by a different user or by the system. To make the average duration of the workitems more robust to data quality problems, a trimmed mean was used to calculate the averages. The result can be seen in Fig 13.

For example, "W_Assess potential fraud" has a trimmed mean of 99.8 minutes. The trimmed mean was calculated by cutting off the top 5% and the bottom 5% of values. The 5% cut-off was stipulated based on exploratory analysis of the duration distribution. While this seems roughly right, it could be further refined with better domain knowledge of the work performed. The "No_of_Cases" column, combined with the Avg_manual_duration, reveal that there are only four major activities that occur often, and take enough time, to be relevant. Ordered by their cost impact, these are the following:



Event_Name	No_of_Cases	Avg_manual_duration	5%_cutoff	95%_cutoff	Max
W_Assess_potential_fraud	301	99.8	0.4	766	2044
W_Call_after_offers	31362	3.9	0.9	15	1148
W_Call_incomplete_files	15003	10.5	1.6	39	949
W_Complete_application	29646	9.3	1.3	26	1682
W_Handle_leads	3643	1.8	0.2	7	942
W_Personal_Loan_collection	2	2.5	1.9	3	3
W_Shortened_completion	74	3.8	0.0	28	66
W_Validate_application	21870	17.5	1.6	77	1126

Fig. 13. Average processing duration of manual activities in minutes (trimmed mean)

W_Validate applications: In the 21'870 cases when this activity occurred, it took 17.5 minutes on (trimmed) average. The maximum value was 1'126 minutes, which clearly doesn't make sense. The table also shows the upper and lower cut-off value for the trimmed mean. Here, all values between 1.6 and 77 minutes were included in calculating the mean. Some of the 17.5 minutes are indirectly driven by the large number of requests for completion, because this requires subsequent re-validation. As the answer to question 2 has shown, almost 70% of customers that had sent back the offer, had to resend documents at least once. The completion requests are therefore not just an annoyance for customers, but also a major cost driver. A later analysis will quantify this impact.

W_Complete application: This workitem occurred in 29'646 cases and took 9.3 minutes on average.

W_Call incomplete files: This workitem shows the direct cost of incomplete files. It occurred in 15'003 cases and took 10.5 minutes on average. Again, the trimmed mean discards the extreme outliers caused by the presumed data quality problems.

W_Call after offers: Despite being short (3.9 minutes on average), this activity occurred in almost all cases (31'362). It therefore has the fourth largest impact on process cost.

All other workitems are much less relevant from a cost perspective, because they only occur in few cases. One such example is "W_Assess potential fraud". Despite its long average duration the cost impact is limited, as it only occurs in 301 cases.

The total duration of manual work in a case is 33 minutes overall, calculated by the same trimmed mean method. Note, that this cannot be inferred directly from the numbers in Fig. 13, due to the different weightings, and due to the non-linear trimming effect. If we assume an hourly labour cost of EUR 40, the average processing cost of an application would be around EUR 20. The yearly manual processing cost of all cases started in 2016 would therefore be around EUR 600'000.

4.3 Relationships between different process KPI

The observations from the discovered process model are a good starting point to define process KPI, such as return-rates, completion requests, cost and others. R allows to further analyse these KPI and their relationships in a very flexible way. Some of the KPI could be obtained from the case-level attributes. Others were created by a method called feature engineering and had to be calculated from the application event log. Feature engineering is the process of creating new variables from the data, that could potentially help predict or explain an outcome. The following variables and KPI have been defined:

– Application type (new credit or limit raise)
– Loan goal (a categorical variable, indicating different loan uses such as car, home improvement, etc.)
– Returned (a calculated binary variable, indicating whether the customer had returned any offer in that application)
– Number of completion requests (calculated at the case level)
– Number of offers made (calculated at the case level)
– E-Channel (a calculated binary variable, indicating whether an application came in through the online channel, or was started by the bank)
– Decision (a calculated categorical variable, indicating the loan decision with values Accepted, Denied and Cancelled)
– The manual duration of each W_activity, and the total manual duration at the case level

It is a good idea to guide an analysis by some overall objective, to avoid getting lost in all the possible combinations. In this case study, the following objectives were used to guide the analysis:

– Maximising returned offers
– Minimising cancellations
– Minimising manual processing cost

Minimising cancellations is favoured over maximising "Accepted" for the reasons discussed in question 2. An increased "Accepted"-rate may lead to false conclusions, if it merely mirrors a decreased "Denied"-rate. Waiting times were not included as an overall objective, since they are mostly caused by customers, and are a result of the process design.

Most cancellations are from non-returned offers: When comparing cancellations and returned offers, it becomes obvious, that the vast majority of cancellations have never been returned at all. Once a customer has returned an offer, his application has a very high chance of becoming accepted (Fig. 14). Therefore, the return-rate is an important lever for improvement.

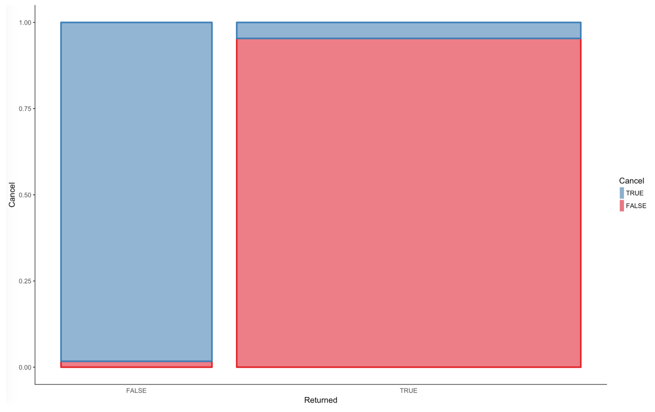


Fig. 14. Most cancellations are from applications with non-returned offers

Return rate, application type, loan goal and channel: There is some relationship between returned offers and the application type, the loan goal and the channel (Fig. 15). Limit raises had a higher return rate than new credit,

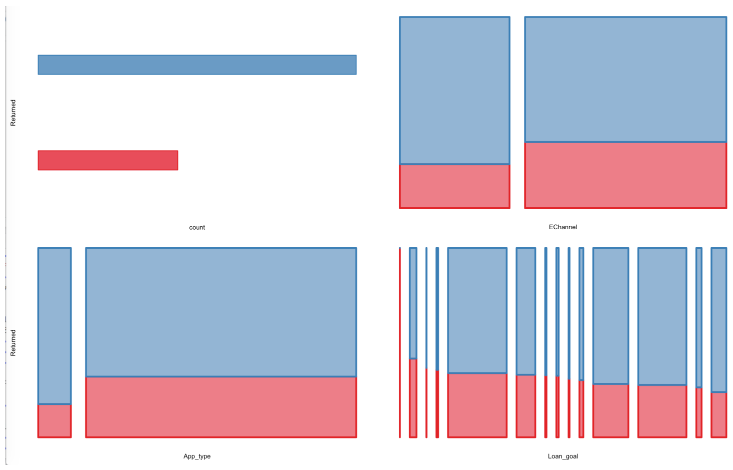


Fig. 15. Return-rate depending on the channel, application type and loan goal

and applications started by the bank had a higher return rate than those in the online channel. There is also a relationship between the application type and the channel: applications for limit raises were exclusively started by the bank, while applications for new credit were mostly filed through the online channel (Fig. 16). In other words, the online channel is exclusively used for new credit. This is also the most likely reason for the online channel's lower return rate. It is probably not the channel itself that directly leads to a different return-rate, as presumed in the previous section "Process flow of the A_events". Rather, the difference can be explained by an indirect effect: the two channels have a different mix of application types, which themselves have a different return rate. It seems plausible, that customers applying for new credit are less committed than those applying for a limit raise. However, while all these relationships are real, Fig. 15 shows that they are too weak to give any actionable recommendation with regard to the overall objectives.

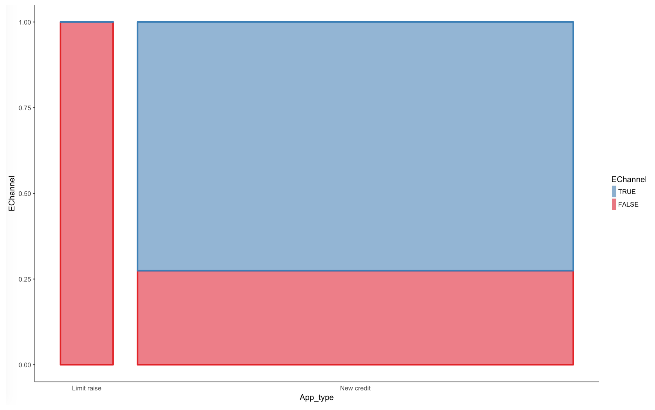


Fig. 16. The online channel is exclusively used for new credit

Completion requests drive the cost of validation: As expected, the manual processing time for validating offers increases with the number of completion requests. Fig. 17 shows the distribution of the validation time per case as multiple boxplots on the y-axis for every number of completion requests on the x-axis.

The bars within the boxes show the median processing time in minutes, the lower and upper end of the boxes indicate the 25% and 75% quartiles. The width of the boxes indicates the case volume in every bucket. To make the graph more readable, the top 5% value range has been trimmed to eliminate the worst outliers caused by the presumed data quality problems. While there is considerable variation, one can clearly seen how the median processing time, and therefore the cost, increases with every round of completion requests. The decreasing width of the boxes also shows how the number of cases gets smaller with more requests for completion. These observations are consistent with the recommendation in question 2 to reduce the number of completion requests.

Number of offers drives the cost of validation: We can also see that the manual processing time for validating offers increases with the number of offers. Fig. 18 shows the same multiple boxplots for the validation time, but this time

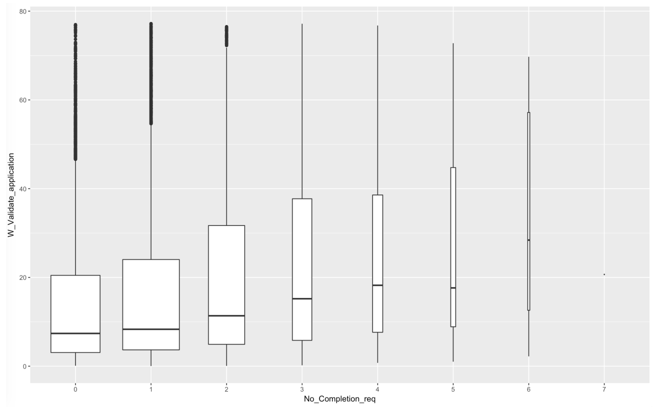


Fig. 17. Completion requests drive the validation processing time (in minutes)

the x-axis indicates the number of offers. Again, we clearly see how the median

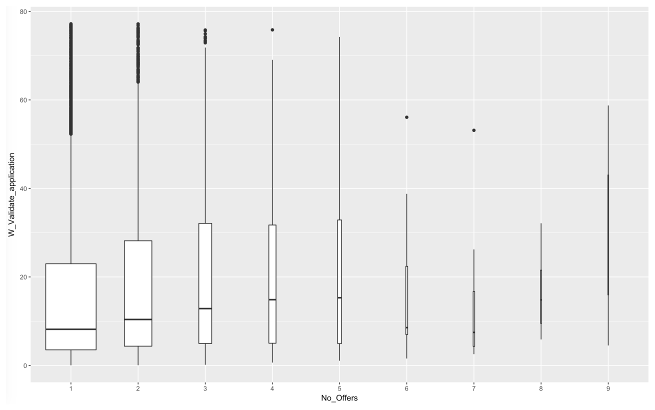


Fig. 18. Number of offers drives the validation processing time (in minutes)

processing time for validating offers increases with the number of offers. This was to be expected, and is consistent with the recommendation in question 3, to only create additional offers reactively, but not proactively. The decreasing median at 6 and 7 offers might be due to the noise in those small sample sizes, and does not change the message.

Other KPIs on the offer level, such as the credit score, requested amount, offered amount, first withdrawal amount, monthly cost and number of terms, were also analysed. However, all of these relationships were found to be either trivial or of little help for the stated overall objectives. After all, we are looking for action-oriented insights that go beyond descriptive statistics.

4.4 Analysing KPI over time

So far, process models, process cost and other KPI have been analysed for the entire data set. Any process changes over time cannot be discovered with this method. They are simply lumped together in one big data sample. Any summary statistics of the entire data set make the implicit assumption, that the process is homogeneous and does not change over time [5]. This implicit assumption also applies to many published process mining case studies. However, in reality that assumption rarely holds true. Practitioners of established process management disciplines such as lean, six sigma and quality management, have long used a simple, yet powerful method: plotting process data over time. This gives a visual impression of the process behaviour over time. Understanding the root cause of a process change provides an opportunity to improve the process. This is a fundamental principle of continuous improvement. Every process has some variation, some of it being purely random. W. Shewhart showed that there are two types of variation [6]: common cause (random) variation and special cause variation. The former is pure noise, and the latter are the signals that have an assignable (special) cause. This has vast implications on the management of processes and management KPI. Reacting to changes, that represent common cause variation, is actually counterproductive. In contrast, understanding the root-causes of signals is the key to process improvement. The question is then, how to separate the signals from the noise.

In many cases, a simple run chart with a centre line already provides enough evidence of a signal, and hence a process change. Combining process mining with time series analysis is therefore a promising method to discover actionable insights. Again, RStudio has been used to calculate and plot process KPI on a run chart. For this case study, a weekly granularity has been chosen for the data points. This level of granularity has proved useful in practice to detect process changes. All KPI were calculated on a case basis, and then mapped to the calendar week of the start of that case. Finally, weekly averages for each KPI were calculated. Different mappings of KPI to calendar weeks are possible. This method has been chosen, because it can be applied to all KPI types, and because it allows to easily relate any signals to the cases that started in that week.

Some examples of simple run charts can be seen in Fig 19. This shows the weekly averages of six measures, together with a centre line, which is the overall average of that measure. Even without applying any advanced techniques, it becomes immediately clear that there was a structural change in the number of applications (n) around week 23. The increase is so strong, that it had already showed up in the dotted chart in a previous section. There must be a specific cause for the increased demand, such as a new product launch or targeting additional customer segments. Using the centre line as a visual reference, the number of applications seems to decrease again at the end of 2016. The second plot shows that the proportion of new credit out of all applications has increased as well. The proportion of online applications (EChannel) in the third plot seems fairly stable over time, at least based on the simple run chart. The average

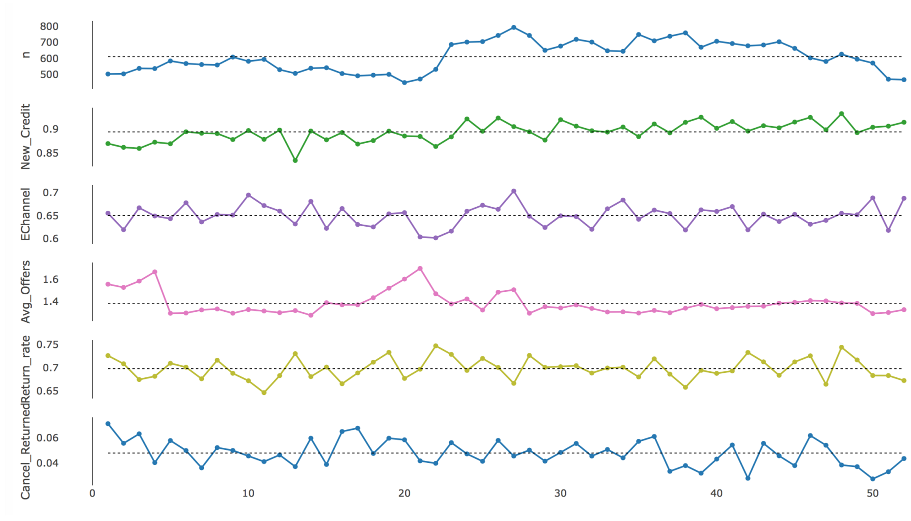


Fig. 19. 1) No. applications, 2) Proportion new credit, 3) Proportion online applications, 4) No. offers, 5) Return rate, 6) Cancellation-rate for returned applications

number of offers per application shows an unusual pattern around week 20, and also at the beginning of the year. The visual context of the variation already gives enough evidence, that the elevated number of offers per application around week 20 must have an assignable cause. The fifth plot is the return rate. This is the proportion of applications that had reached the A_Validating step, which means an offer had been signed and returned by the customer. The last plot in Fig. 19 is the proportion of returned applications, that had later been cancelled by the customer (Cancel-Returned). The simple run charts of the last two measures don't show any clear signals. This means that the elevated number of offers around week 20 did not seem to have any positive impact on the return rate, or the cancellations after returning.

4.5 Analysing KPI using statistical process control

Run charts with a centre line provide a good visual context of the variation over time, and help detect the strong and obvious signals. However, a much more powerful method exists to separate signals from the noise in process data: Statistical Process Control (SPC). SPCs main tool is the control chart, often called a "process behaviour chart". Statistical process control has been developed by W. Shewhart [6] and made popular by W. E. Deming [7], who famously helped Japanese manufacturers become quality leaders starting from the 1950s. Control charts are based on sound theory and are widely used in industry to monitor and improve manufacturing processes. However, they are just as useful to discover the "voice of the process" in service processes and management KPI. The chart to be used in this case is the "Individuals and moving range chart"

(ImR-chart, more often called XmR-chart [8]). The XmR-chart consists of two parts: the chart for individual values, and the chart for moving ranges between consecutive individual values. Again, RStudio has been used to create an XmR-chart for the return-rate (Fig. 20). The upper individuals chart with its centre line is the same as its corresponding run chart before in Fig 19. The two dashed lines symmetrical to the centre line are the "Upper Control Limit" (UCL) and the "Lower Control Limit" (LCL). They are sometimes referred to as "Natural Process Limits". These limits are calculated from the data points and reflect the amount of natural variation in the data. Any points outside the limits are a reliable indicator of a special cause. There are three additional rules to detect process changes [8]. The lower moving range chart also contains two dashed lines: the average moving range and the "Upper Range Limit" (URL). Combined with the individuals chart, this also helps detect signals and process changes. With proper calculation and interpretation, the XmR-chart is very robust. It neither results in many false alarms, nor does it miss many signals [9].

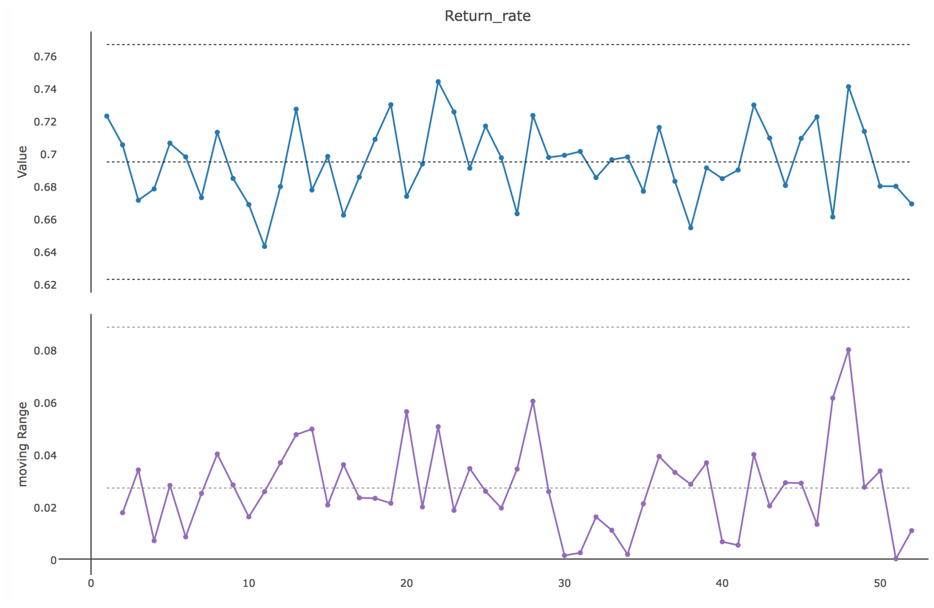


Fig. 20. XmR-chart for the return-rate: all common-cause variation

Interpreting XmR-charts for process KPI In Fig. 20, none of the detection rules indicate any signal in the return-rate. That KPI is stable and predictable between the two control limits. All variation is common-cause, and therefore indistinguishable from noise. This is actually rather unusual. Most processes show some kind of special cause variation, allowing for a targeted investigation of root causes, and subsequent process improvement. But even though the return-rate doesn't display any signals, the control-chart can still be used as a baseline for controlled experiments. It is therefore an important part of a continuous Plan-Do-Check-Act improvement cycle: an improvement idea is developed (Plan) and tested in reality (Do). Any impact will immediately show up in the control chart and can be quantified (Check). Depending on the impact, the improvement idea is then rolled out or gets rejected (Act). It is recommended, that the bank uses this chart as an ongoing tool to test potential improvement ideas to reduce the application return-rate.

When applied to the other measures in Fig. 19, their XmR-chart will confirm the qualitative findings from their run charts: a clear and strong increase of applications, an increase in the share of new credit applications, a stable online channel rate, a signal in the number of offers per case in the beginning of the year and around week 20, and a stable cancellation-rate of applications with returned offers.

Interpreting XmR-charts for processing durations The story gets more interesting, when we investigate the time series of the number of completion requests, and the processing durations of the four major manual activities. Fig. 21 shows their respective run charts. The first one (Avg_Cmpl_req) shows a steady increase in the number of completion requests per returned case, starting in week 11. The visual context of the centre line already shows that this is not just random variation. Plotting the same values on the more rigorous XmR-chart confirms that there must be an assignable cause for the increase. In addition, variation also seems to increase starting in week 33. The bank should definitely reverse this ascending trend. It is recommended to use domain knowledge, to investigate the root-cause of the increasing number of completion requests. One possible explanation might be stricter review criteria of applications. Knowledge of the root-cause will help reverse the trend and redesign a "front-loaded" process, as described in question 2.

The second run chart (Time_Validate) is very counterintuitive. It shows the average manual processing time per case to validate applications, using the same trimmed mean as before. We can see a steep decline after week 26. The XmR-chart confirms this clear structural change of the process. While this reduction is good, one would actually expect the opposite, because of the steady increase in the number of completion requests per application. As we have seen in the previous section, more requests for completion lead to longer manual processing times. A quick drill-down confirms that this relationship also holds true, when splitting the data at week 26, and analysing the two sets before and after the process change separately. This means, there must have been a strong root cause starting in week 26, that over-compensated the effect of increasing completion requests, and cut in half the average time required for manual validation from ∼25' to ∼12'. If that root-cause can be identified using domain knowledge, the resulting insight might help to reinforce the desired effect, or to achieve the same reduction with other manual activities. A further drill-down suggests that

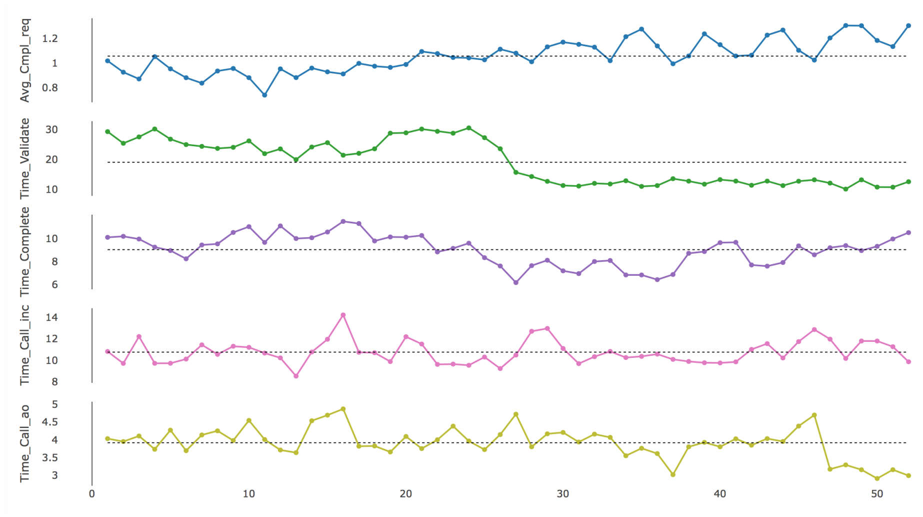


Fig. 21. 1) No. completion requests, 2) Time to validate applications, 3) Time to complete applications, 4) Time to call for incomplete files, 5) Time to call after offers

the reduction of the validation time was not primarily driven by a change of employees. Rather, it could be observed within most individual users. Therefore, potential root-causes could have been a policy change, better training, or an improved system support. Note, that the steep decline coincides with the strong increase of the application volume around week 23 (Fig. 19). So, a more trivial explanation could be that employees simply took less time for validation, in order to cope with the higher volume.

In the third run chart (Time_Complete) it is difficult to visually separate the signals from the noise. We therefore use an XmR-chart to inspect the time to complete applications (Fig. 22). As we can see, this shows a more erratic behaviour. There is a structural shift in week 26 to a shorter processing time, just as we have seen in the time to validate applications. However, starting from around week 40, there is an upward trend again. SPC allows to recalculate the control limits of a chart, if there is a sustainable process change. In this case, recalculating the limits doesn't change the message of the chart. There must be an assignable cause for both process changes. As before, it is recommended to use domain knowledge to identify the root causes of the changes. This would help stabilize the process and reduce variation.

The time to call incomplete files (Time_Call_inc) is more stable, but still shows three different signals (Fig. 23). Again, this is an opportunity to investigate the root causes by combining analytical insights with domain knowledge. Understanding the root causes allows to reduce variation and processing cost in a continuous improvement effort. The same is true for the last measure, the time for calling customers after sending them an offer (Time_call_ao). Three different signals can be identified that must have an assignable cause (Fig. 24).

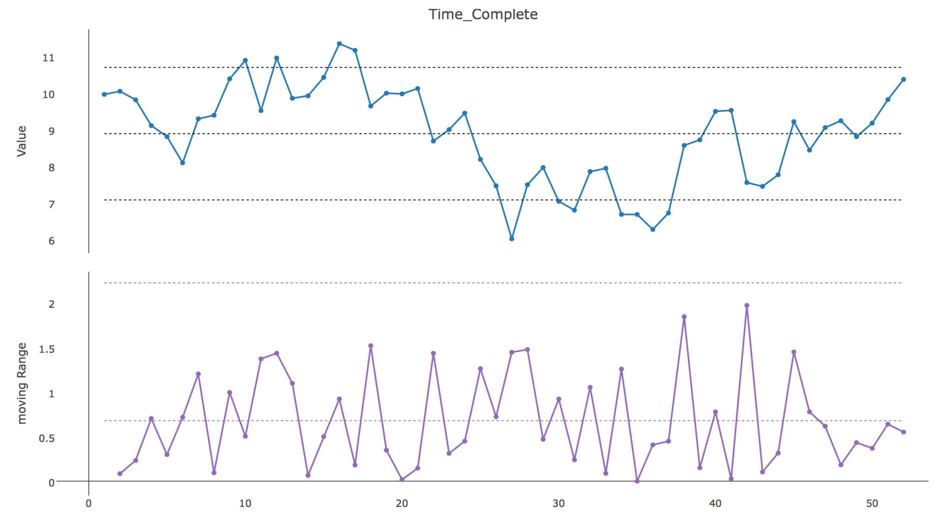


Fig. 22. XmR-chart for the time to complete applications: several signals around week 26 and week 40

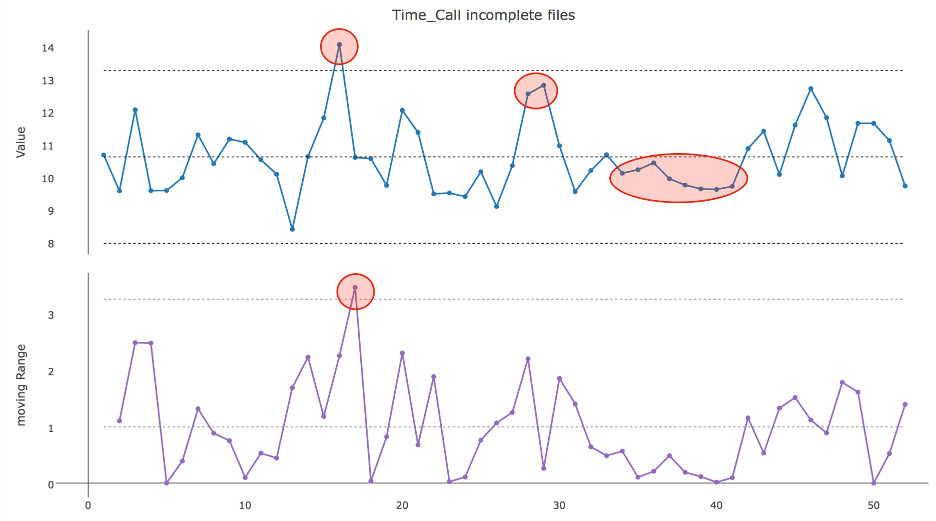


Fig. 23. XmR-chart for the time to call incomplete files: three different signals

5 Summary of recommendations

This last section summarises the main recommendations made in the preceding parts of this paper.

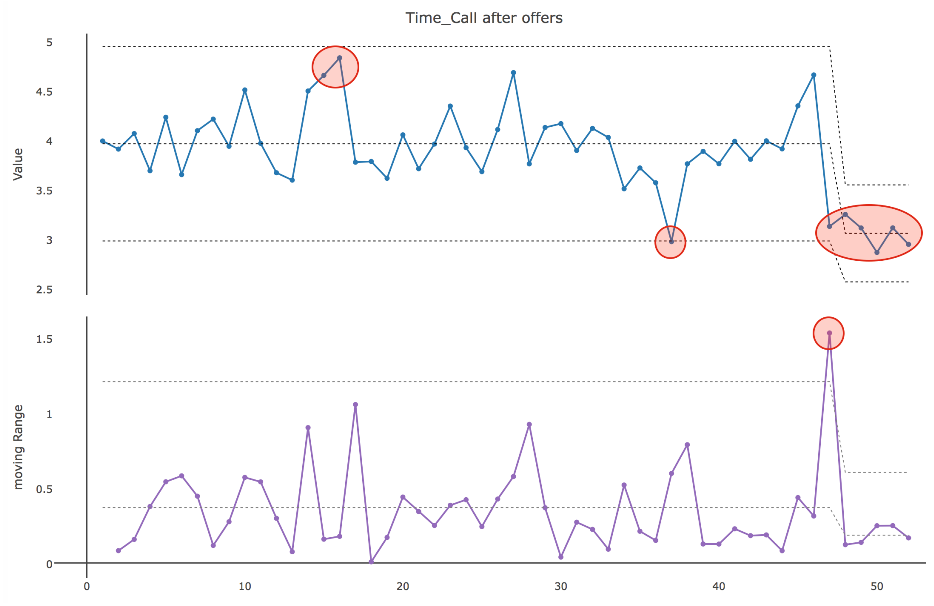


Fig. 24. XmR-chart for the time to call after offers: three different signals

5.1 Standardise handling of multiple offers

Do not proactively suggest multiple offers during a conversation. However, if the customer asks for it, provide an additional offer. This increases the chances of a successful outcome.

5.2 Redesign the application process

- Investigate the real benefit of calls after offers by conducting an A/B-Test. Omit these calls for a test group and compare the cancellation rates with the control group. If the difference is not strong enough, eliminate that process step.
- Redesign a "front-loaded" application process, with all the necessary information exchanges moved towards the beginning of the process. Strive for a process design that completely avoids requests for completion and other calls that don't add value from a customer's perspective.
- Consider developing a fully digitized application process, in which no paper has to be sent to and from customers. Customers can upload scans of their documents and missing information is automatically detected at the front-end. However, still provide human assistance and let customers choose their preferred channel of interaction (e.g. text messaging via popular messaging apps, call, chat, etc.). Strive for a process design that permits most loan decisions to be made within 24 hours.

5.3 Investigate root-causes of insights using domain knowledge

- Understand the root-cause of the 30% lost customers that did not return any documents after receiving an offer. Develop hypotheses based on domain knowledge, test improvement ideas in a pilot and monitor the return-rate using a control chart.
- Investigate the root-cause of the steadily increasing number of completion requests per returned application and reverse that trend.
- Investigate the root-cause of the drop in the processing time for manual validation around week 26. If possible, transfer that positive effect to other manual activities.
- Investigate the exact reason for the presumed status change errors that lead to the outliers in manual processing time. Have the workflow system ensure that human error is not possible when changing the status of workitems. This is important if the bank wants to calculate reliable KPI.
- Investigate the root-causes of other signals found, by discussing the XmR-charts of relevant KPI with domain experts.

5.4 Establish continuous improvement

Establish process mining and statistical process control as standard management tools. Discuss the combined insights in regular management meetings and use them as a basis for continuous improvement of the application process. Extend this to other operational processes of the bank.

References

1. BPIC'17: http://www.win.tue.nl/bpi/doku.php?id=2017:challenge
2. van Dongen, B.F. (2017) BPI Challenge 2017. Eindhoven University of Technology. Dataset. http://dx.doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
3. Aalst, W.M.P. van der: Process mining: Data science in action, Second Edition. Springer, Heidelberg (2016)
4. Guenther, C.W., Verbeek, E.: XES Standard Definition, v 2.0. Eindhoven University of Technology, (March 28, 2014)
5. Wheeler, D.: The Six Sigma Practitioner's Guide to Data Analysis, Second Edition. SPC Press, Inc., Knoxville, Tennessee, (2010)
6. Shewhart, W.: Economic Control of Quality of Manufactured Product. Republished by American Society for Quality, Milwaukee, Wisconsin, (1931)
7. Deming, W.E.: Out of the Crisis. MIT Press, Cambridge, Massachusetts, (2000)
8. Wheeler, D.: Understanding Variation - the Key to Managing Chaos. SPC Press, Inc., Knoxville, Tennessee, (1993)
9. Wheeler, D.: Advanced Topics in Statistical Process Control, Second Edition. SPC Press, Inc., Knoxville, Tennessee, (2004)