Chair of Information Systems and Business Process Management
TUM School of Computation, Information and Technology
Technical University of Munich

First Exercise - Business Process Prediction, Simulation, and Optimization

Data and Process Science on the BPIC-17 Event Log

Mertcan Besler✉

✉ mertcan.besler@tum.de

November 20, 2025

Abstract — This report analyzes the BPI Challenge 2017 (BPIC-17) event log, a real-life dataset capturing the loan application process of a Dutch financial institution. Using Python and PM4Py, the log is preprocessed, explored, and examined through descriptive statistics, process discovery, and conformance checking.
α–algorithm, Heuristic Miner and Inductive Miner, three process discovery algorithms, are applied to model the control-flow behaviour of the event log. The resulting models exhibit structural complexity, which substantially limits their interpretability for analytical purposes. Based on these insights, a simplified BPMN model is derived that emphasises an balance between fitness and interpretability. Advanced analyses, including a process variability analysis and a root cause analysis, show that applicant-driven interactions with wide throughput-time dispersion, along with cancellation-related behaviour, are the primary contributors to long case durations and to increased variability in throughput times. The results integrate descriptive, diagnostic, and predictive perspectives and provide a foundation for subsequent simulation and predictive modelling on the BPIC-17 event log.

1 Introduction

The Business Process Intelligence (BPI) Challenge, organized by the Process Mining Group at Eindhoven University of Technology, annually releases real-life event logs to benchmark process mining methods and foster reproducible research [7]. This report analyzes the BPI Challenge 2017 (BPIC-17) event log, a real-life dataset from the financial services domain capturing the end-to-end loan application process of a Dutch financial institution.[1] The event log records all applications filed in 2016 and their handling until February 2, 2017, including the creation and communication of offers and the validation of customer documentation. It distinguishes three perspectives: application state changes (`A_`), offer state changes (`O_`), and workflow items (`W_`). Moreover, the event log provides timestamps, resources, and lifecycle transitions that enable process discovery, conformance checking, and performance analysis [2]. From a process perspective, a case begins when an application is submitted either online by the customer or manually entered by bank staff. After an initial automated or semi-automated assessment, the bank prepares and sends one or more offers to the applicant. The applicant subsequently provides the required documentation, which initiates a validation phase and, if documents are incomplete, further follow up by the bank. The process concludes with one of three outcomes: the loan is granted, the application is denied, or the case is cancelled by the applicant.
Prior reports highlight that most applications are submitted online and that the majority of cases receive a single offer with a long tail of cases involving multiple offers. They also note that the document validation phase introduces the most pronounced loops in the control flow [3, 4]. A further report by KPMG supports these observations with their analysis on the customer experience dimension Time and Effort. It identifies an average end to end lead time of about three weeks and substantial waiting after offers are sent. It also finds higher conversion when multiple offers are provided, which motivates recommendations for faster multi channel follow ups and clearer upfront instructions [1]. Rodrigues et al. reconstruct a high quality AS IS model with a fitness of approximately 97.28% and a precision of approximately 98.72%. They demonstrate that the longest waiting times typically occur during applicant document submission and the subsequent validation cycles. The outcome distribution is dominated by pending cases at about 55%, followed by cancelled at about 33% and denied at about 12% [4]. Another report by Povalyaeva et al. examines the density of client interaction. It shows that

[1]Challenge website: https://ais.win.tue.nl/bpi/2017/challenge.html.

shorter time to first offer and regular follow ups correlate with fewer cancellations. It also finds that the design of offers, including factors such as the number of terms, first withdrawal and monthly cost, materially influences acceptance. More frequent incomplete file calls alone do not increase acceptance likelihood [3]. Overall, the existing reports position BPIC-17 as a validated basis for studying data and process science methods in a realistic financial setting and for linking model based insights to customer experience outcomes [1,3,4].
Building on these findings, this report first outlines the technical setup and summarizes the core log statistics. It then derives and evaluates process models created by three discovery algorithms together with conformance checking, followed by advanced analyses for process variability and root cause investigation.

2 Approach and Results

This section outlines the methodological approach, the analysis workflow and results. It establishes a reproducible setup for processing, analyzing and interpreting the BPIC-17 event log and prepares the foundation for subsequent process simulation, predictive modelling and optimization.

2.1 Technical Setup and Preliminaries

The technical analysis of the BPIC-17 event log was conducted in a Jupyter Notebook environment using Python 3.10. All notebooks, models and supplementary implementations are publicly accessible in the project's GitHub repository.[2] The core processing and modelling tasks were implemented with the process mining library PM4Py[3], which provides functionalities for event log import, transformation, process discovery and conformance checking. For data manipulation and tabular analysis, the `pandas`[4] library was used. The BPIC-17 event log was imported in the IEEE XES format[5] and converted into a tabular pandas DataFrame to support descriptive statistics, case-level computations and preprocessing. All computations were executed locally on a Windows 11 workstation equipped with an AMD Ryzen 7 5800H processor, 16 GB of RAM and SSD storage. For process model construction and refinement, the Signavio Academic Edition[6] was used to manually derive a BPMN model based on insights from discovery algorithms, prior reports and domain-driven reasoning. The Celonis Academic Alliance Edition[7] was used to provide the process intelligence graph as a comparative reference point refined model creation.

2.2 Basic Analysis

The BPIC-17 event log follows the standard structure of an information system event log, where each row represents a single recorded event and each column corresponds to an attribute associated with that event. As shown in Table 1, these attributes span two levels: event-level attributes, which describe properties of the event itself (such as the activity name, lifecycle transition, timestamp, resource, and event origin), and case-level attributes, which describe properties of the underlying loan application (such as the loan goal, requested amount, credit score, and offer-related financial information). The derived data types show that most attributes are categorical objects, which is typical for administrative and financial processes that rely heavily on identifiers, status labels, and reference values [6]. These categorical attributes can be further distinguished into categorical text fields and categorical Boolean fields. The Boolean categories include the attributes Accepted and Selected. Numerical attributes of type float correspond primarily to monetary amounts, term lengths, and credit-scoring values, while the timestamp attribute is the only field stored as a true temporal datatype. This heterogeneity across categorical, numerical, and temporal data types must be considered during preprocessing, process discovery,

[2]https://github.com/mertcanbesler/Business-Process-Prediction-Simulation-and-Optimization---BPIC-17
[3]PM4Py: A Python library for process mining, developed by Process Intelligence Solutions, https://processintelligence.solutions/pm4py.
[4]pandas: Python library for data analysis, https://pandas.pydata.org/.
[5]IEEE eXtensible Event Stream (XES) Standard, https://xes-standard.org/.
[6]SAP Signavio Process Manager (Academic License), https://www.signavio.com/de/academic-and-research-alliances/.
[7]Celonis Academic Alliance, https://www.celonis.com/academic-alliance/.

and conformance checking, as each type influences model construction and analytical outcomes differently.

Table 1 Structure of the BPIC-17 event log with automatically derived data types and representative example values. The data types were computed based on the loaded XES event log, and the example values correspond to a real event instance[8].

Collumn	Data Type	Example Value
Action	Categorical Object	Created
org:resource	Categorical Object	User_52
concept:name	Categorical Object	O_Create Offer
EventOrigin	Categorical Object	Offer
EventID	Categorical Object	Offer_148581083
lifecycle:transition	Categorical Object	complete
time:timestamp	datetime64[ns, UTC]	2016-01-02 11:29:03.994000+00:00
case:LoanGoal	Categorical Object	Existing loan takeover
case:ApplicationType	Categorical Object	New credit
case:concept:name	Categorical Object	Application_652823628
case:RequestedAmount	float64	20000.0
FirstWithdrawalAmount	float64	20000.0
NumberOfTerms	float64	44.0
Accepted	Categorical Object	True
MonthlyCost	float64	498.29
Selected	Categorical Object	True
CreditScore	float64	979.0
OfferedAmount	float64	20000.0
OfferID	Categorical Object	Offer_148581083

Additional basic structural and temporal characteristics of the BPIC 17 event log are summarized in Table 2. The dataset comprises 31,509 loan application cases and a total of 1,202,267 recorded events. The number of process variants (15,930) indicates a high degree of behavioural variability, which reflects that the loan application process is executed through many different paths. The event log contains 26 distinct event labels, each representing a unique activity performed within the process. Across all cases, the mean case length is 38.16 events, with a standard deviation of 16.72 events, indicating noticeable variation in the number of events required to complete a case. The mean case duration is 21.90 days, while the median duration is 19.09 days. Reporting both values is appropriate, since real process throughput times are typically right skewed [6]. The presence of cases lasting up to 286 days demonstrates substantial outliers, which increase the mean but do not affect the median to the same extent. The relatively high standard deviation of 13.17 days further confirms considerable heterogeneity in case completion times. The event log contains twelve categorical event attributes, reflecting the presence of descriptive fields, identifiers and status values that are characteristic of administrative and financial information systems. The mean inter event time is approximately 14 hours and describes the average temporal distance between two consecutive events within a case. This value provides insight into the typical waiting periods or idle phases encountered throughout the process. The event level rework rate was computed following the definition by van der Aalst [6]. It is calculated as the number of repeated executions of the same activity within a case, beyond the first occurrence, divided by the total number of completed events. The resulting rework rate of 14.66% shows that repeated execution of activities constitutes a meaningful proportion of the observed behaviour. This provides insight into iterative patterns that may influence throughput times, resource utilisation and the behaviour of simulation or predictive models.

[8]All computations of this table can be reproduced in the accompanying Jupyter notebook `01_load_explore.ipynb`.

Table 2 Basic statistics of the BPIC-17 event log [9]

(a) Event log volume and variant complexity

Metric	**Value**
Total number of events	1,202,267
Total number of distinct process variants	15,930
Total number of distinct event labels	26
Total number of categorical event attributes	9
Mean inter-event time	14 hours, 8 minutes, 43 seconds
Event-level rework rate	14.66%

(b) Case-level performance and duration characteristics

Metric	**Value**
Total number of cases	31,509
Total number of case labels	4
Mean case length	38.16 events
Standard deviation of case length	16.72 events
Mean case duration	21 days, 21 hours, 35 minutes, 25 seconds
Median case duration	19 days, 2 hours, 6 minutes, 20 seconds
Standard deviation of case duration	13 days, 4 hours, 3 minutes, 41 seconds
Minimum case duration	0 days, 0 hours, 3 minutes, 21 seconds
Maximum case duration	286 days, 1 hour, 44 minutes, 18 seconds

2.3 Process Model Creation and Validation

This section presents the creation and validation of process models for the BPIC-17 event log. Process discovery algorithms were applied, and the resulting models were evaluated using all relevant pm4py quality metrics. Based on the insights gained from conformance, simplicity, and generalisation analyses, a refined BPMN model was constructed that balances fitness, interpretability, and structural simplicity. The evaluation covers both the discovered models and the final refined model.

2.3.1 Process Disocery Algorithms on BPIC-17 Event Log

In Order to model the BPIC-17 Event log and understand the behavioural data space contained in the log, three process disovery algorithms were used and implemented [10]
The α-algorithm [5] was applied to reconstruct a Petri net based on the present directly-follows relations. By analysing the ordering of activities across all traces, the algorithm derives causal relations, parallel behaviour, and exclusive choices, and uses these relations to construct a corresponding workflow net. The resulting Petri net model is shown in Figure 1. The initial part of the model displays defined and sequential progresses through a series of standardised intake activities such as application creation and submission. These steps appear consistently across the traces, resulting in a linear and interpretable start of the workflow. After the submission of an application, the structure becomes more complex. In the central part of the model, numerous activities are connected in a highly intertwined manner, forming a broad parallel block. The α-algorithm captures these behavioural relations directly, which results in a dense network of concurrent and alternative paths in this region of the model. Towards the end of the process, the model converges again into a more streamlined

[9] All computations of this table can be reproduced in the accompanying Jupyter notebook `02_basic_metrics.ipynb`.

[10] All process discovery algorithms and the corresponding process models presented in this section can be reproduced in the accompanying Jupyter notebook `03_model_creation_validation.ipynb`.

structure. Several possible closing activities lead the workflow towards its final marking, indicating that despite the complexity of the intermediate phase, the process eventually resolves into a limited set of outcome paths.
The Heuristics Miner algorithm [8] was applied to the BPIC–17 event log to discover a Petri net that incorporates not only directly-follows relations but also their observed frequencies. Based on these frequencies, the algorithm derives dependency relations between activities and infers splits and joins that reflect the most dominant routing patterns in the log. The resulting Petri net, as generated with PM4Py, is shown in Figure 2. During this construction, the algorithm also introduces silent τ-transitions , which do not correspond to an actual activity in the log. These τ-transition are used to structure the workflow, for example to model complex branching behaviour or to separate overlapping dependencies when no explicit event marks these points in the process [8]. Structurally, the model again starts with a sequential intake phase. The creation, submission and handling of an application appear as a straightforward chain of transitions leading from the initial place into the core of the process. This part of the model closely resembles the structure obtained with the α-algorithm and reflects the standardised execution of the early process stages. In the central part of the model, the Heuristics Miner reveals a wide range of alternative and partially overlapping behavioural patterns. Several branches emanate from the main path, representing different processing, assessment, and communication activities that may occur in parallel or in varying orders. The model includes both solid and dashed arcs, with solid arcs indicating strong dependencies that appear frequently in the log, while dashed arcs represent weaker relations observed less consistently. The presence of silent transitions in this region helps to organise the routing logic by connecting or separating behaviour that cannot be represented through observable events alone. Several alternative closing steps lead towards a shared final marking, highlighting that the process resolves into a limited number of completion outcomes despite its complex intermediate section.
The Inductive Miner algorithm [6] was applied to the BPIC-17 event log to discover a Petri net in a block-structured manner. In contrast to purely relation-based approaches such as the α-algorithm, which derive models solely from directly-follows relations, the Inductive Miner recursively partitions the event log into subsets that correspond to high-level process constructs such as sequences, choices, parallel blocks, and loops, and then translates the resulting process tree into a sound workflow net. In contrast to the Heuristics Miner, which uses directly-follows relations with frequency-based dependency measures, the Inductive Miner follows a decomposition-based strategy that guarantees a structured and sound result. During the translation from the process tree to the workflow net, silent transitions (τ) are introduced to realise the routing logic of these constructs, for example to model XOR- and AND-splits or to close loop fragments. The Petri net obtained in this way is shown in Figure 3. The left-hand side of the model represents the intake phase of the process and is again characterised by a sequential structure: the creation, preparation, and submission of a loan application are modelled as a linear fragment leading from the initial place into the main body of the workflow. After this initial segment, the model branches into several clearly separated subprocess blocks. Each of these blocks groups together related activities and is internally represented by a structured combination of sequences, choices, and loops. Several parallel and alternative paths are present, but they are organised into distinct regions with explicit split and join points rather than forming a single entangled area. Loop behaviour is represented as dedicated fragments that allow certain groups of activities to be repeated before the workflow continues. Towards the right-hand side of the model, the various subprocess blocks gradually converge. The closing activities are grouped into a limited number of branches that lead to the end of the process.
Although the three discovered Petri nets differ in structure, they all exhibit a high degree of visual complexity, making them difficult to interpret in the business context of loan application processing. The models generated by the Heuristics Miner and the Inductive Miner contain many places, τ-transitions, long-range arcs, and several nested loops. Such structural patterns are associated with the so-called “spaghetti process” appearance, which reduces interpretability for domain experts [6]. The model discovered by the α-algorithm is structurally smaller, yet it also contains densely connected regions and does not provide an easily communicable representation of the underlying process. These observations motivate a systematic evaluation of the discovered models and underline the need for a more compact final representation.

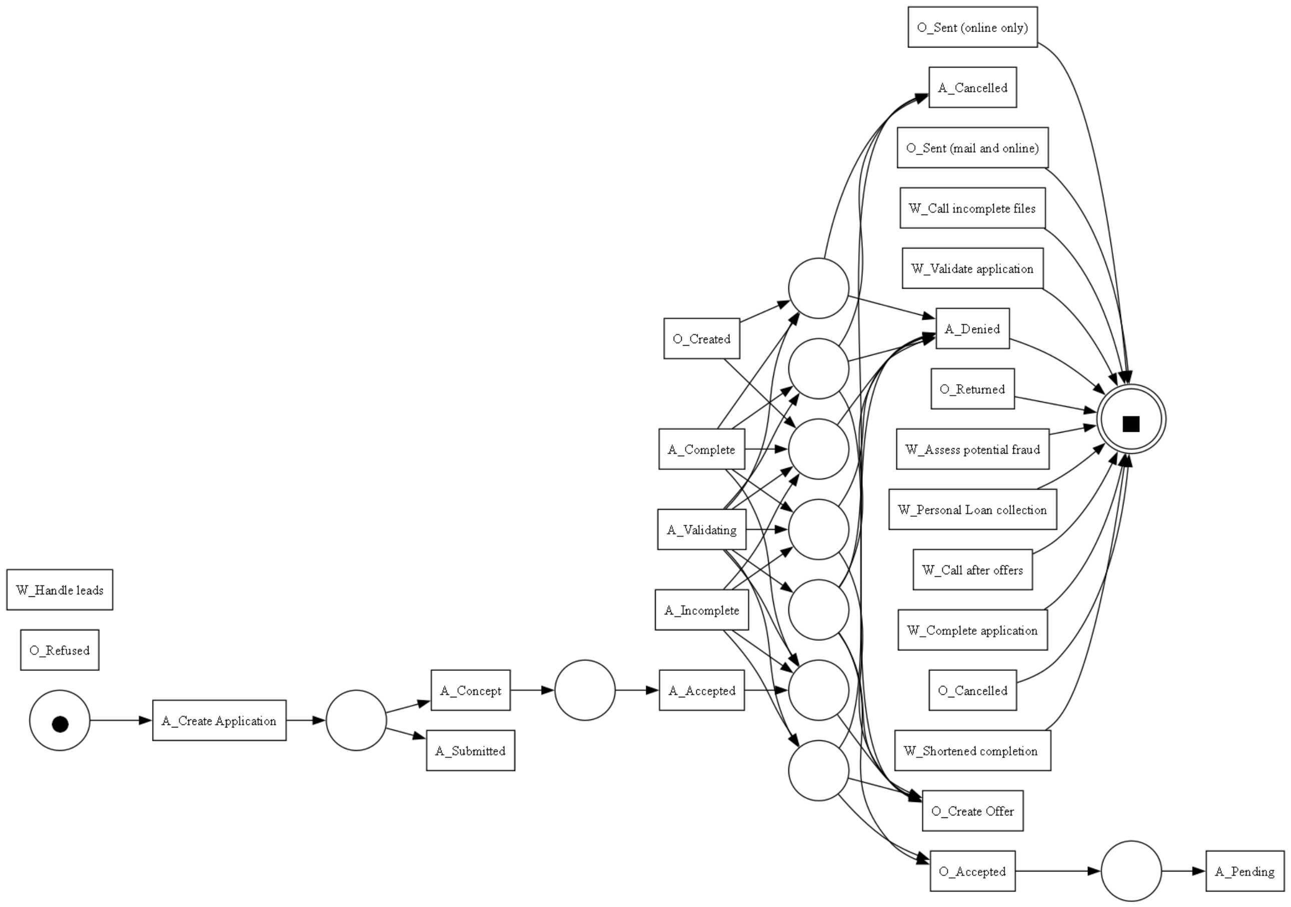

Figure 1 Petri net discovered by the α-algorithm using PM4Py. An enlarged version of this figure is provided in Appendix 6.

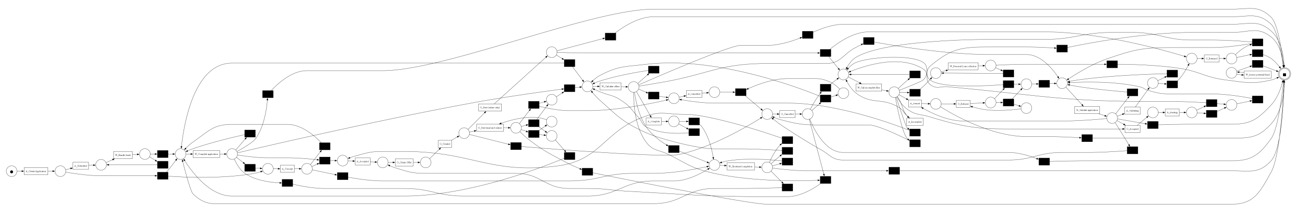

Figure 2 Petri net discovered by the Heuristics Miner using PM4Py. An enlarged version of this figure is provided in Appendix 7.

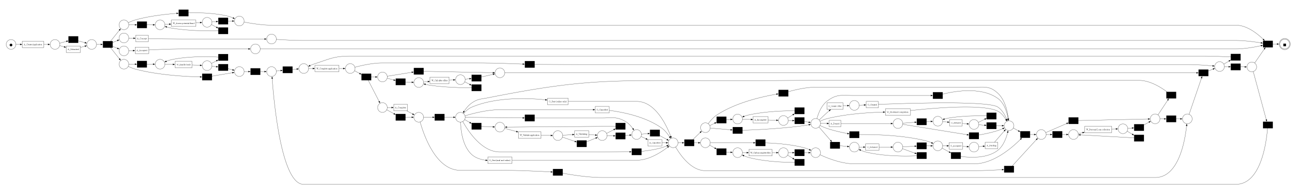

Figure 3 Petri net discovered by the Inductive Miner using PM4Py. An enlarged version of this figure is provided in Appendix 8.

2.3.2 Quality metrics

To assess the behavioural quality of the discovered nets, the standard PM4Py metrics for fitness, precision and generalization were applied [2]. Fitness was computed via token-based replay and quantifies how well the model reproduces the traces in the event log. Precision penalises model behaviour that was not observed in the log. Generalization evaluates the extent to which the model avoids overfitting to specific cases.
In addition to these built-in metrics, two simplicity metrics were implemented to allow for an additional structural comparison of the discovered models[11]. Both metrics are based on structural properties of the Petri nets and were computed directly from the discovered models. The first measure, *Node Count Simplicity*, is defined as

$$S_{\text{nodes}} = \frac{1}{1 + (|\text{places}| + |\text{transitions}|)},$$

meaning that models with fewer nodes obtain higher simplicity values. The second measure, *Arc Density Simplicity*, is given by

$$S_{\text{arcs}} = \frac{1}{1 + \left(\frac{|\text{arcs}|}{|\text{places}|+|\text{transitions}|}\right)},$$

which reflects how densely the nodes are connected. A lower arc-per-node ratio therefore corresponds to higher simplicity. Both metrics are normalised to the interval $(0, 1)$ by construction.
Table 3 summarises the quality metrics obtained for all three discovery algorithms. The Inductive Miner achieves a fitness of 100.00%, indicating that all traces in the BPIC–17 log can be replayed. Its precision is comparatively low at 14.07%, showing that the model permits behaviour not present in the log. Its generalization of 94.85% suggests moderate overfitting. The Heuristics Miner achieves a fitness of 95.46%, a considerably higher precision of 67.11%, and a generalization value of 92.89%. The α-algorithm yields the lowest fitness (38.26%) and precision (9.04%), while its generalization is the highest at 98.24%, reflecting its tendency to infer behavioural relations even from limited evidence. The α-algorithm achieves the highest Node Count Simplicity value (0.025641), indicating that it produces the model with the smallest number of nodes among the three algorithms. However, the Inductive Miner (0.006993) and the Heuristics Miner (0.007937) both generate substantially larger nets, consistent with the visualisations in Figures 2 and 3. The Arc Density Simplicity values lie in a narrow interval between 0.4058 and 0.4356, indicating that the differences in structural complexity are primarily driven by the number of nodes rather than by differences in connectivity structure. Overall, the combination of conformance and simplicity metrics confirms that the discovered models represent the event log behaviour, but their size and structural density limit their interpretability for business stakeholders.

Table 3 Quality metrics of the discovered process models.

Model	Fitness (%)	Precision (%)	Generalization (%)	S_{nodes}	S_{arcs}
Inductive Miner	100.00	14.07	94.85	0.006993	0.435583
Heuristics Miner	95.46	67.11	92.89	0.007937	0.405844
α-Algorithm	38.26	9.04	98.24	0.025641	0.408602

2.3.3 Refined BPMN Model of the BPIC-17 Event Log

Based on the insights obtained from the three discovery algorithms, the official BPIC-17 reports [1, 3, 4], and the Celonis Process Intelligence Graph (see Figure 4), a refined BPMN model was constructed. The initial sequence of activities (Create Application $\rightarrow$ Submitted $\rightarrow$ Handle Leads) reflects the behaviour consistently identified by the Inductive Miner and the Heuristics Miner and represents the dominant pattern observed in the event log. After the activity Complete Application, the process becomes more heterogeneous, with multiple alternative paths and rework loops; therefore, these parts of the model were simplified to minimize structural complexity. Rarely occurring detours and exceptional behaviours visible in the Heuristics Miner model were

[11]The full implementation is provided in Appendix 1.

either aggregated or omitted when they contributed only marginally to the overall replay fitness but would have introduced substantial additional structure. Alternative case outcomes such as Accepted, Denied, Cancelled, and Returned were modelled using exclusive gateways. Repeated validation and communication steps were represented as loops wherever these patterns were clearly supported by the event data. The resulting BPMN model is shown in Figure 5.

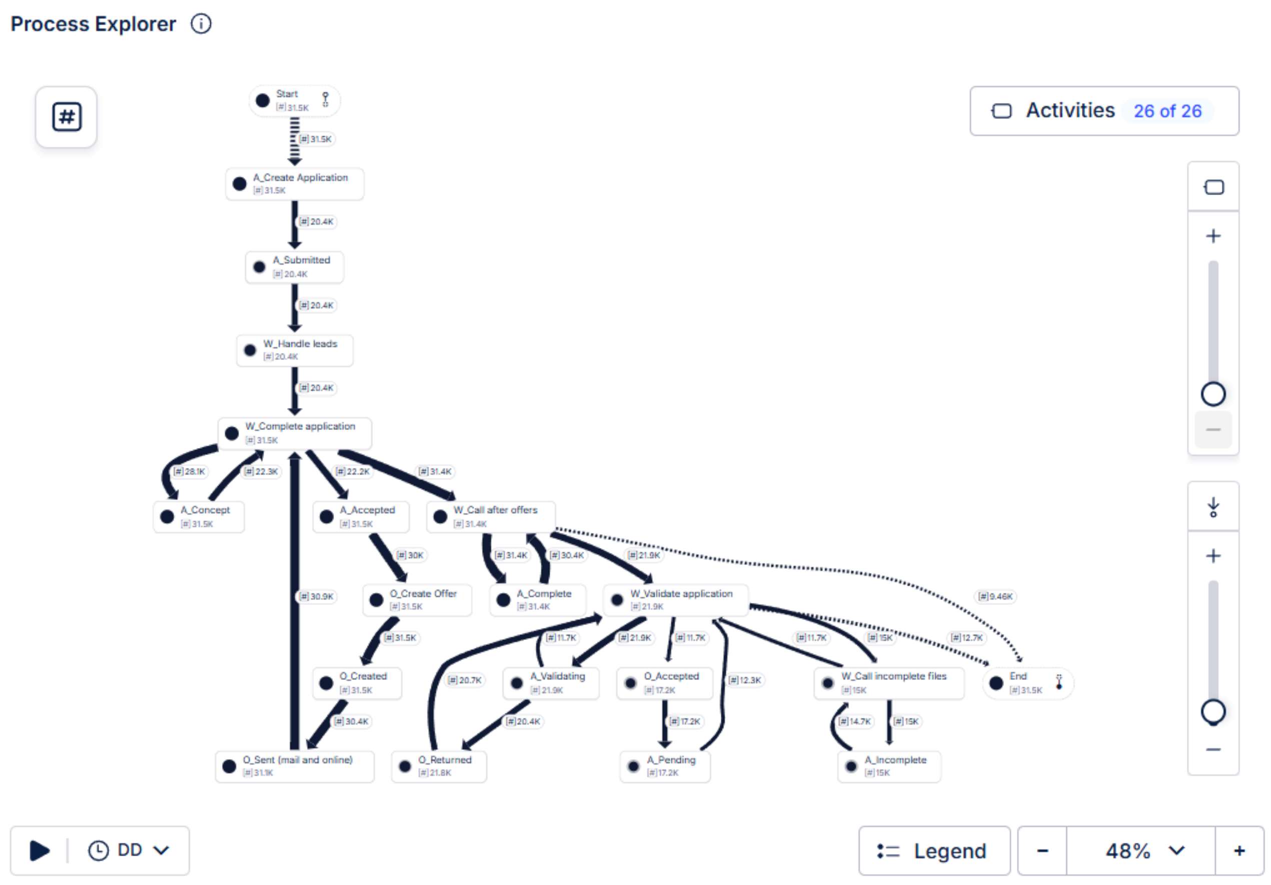

Figure 4 Celonis Process Intelligence Graph generated from the BPIC-17 event log. The graph is automatically created within the Celonis environment upon uploading the event log in the XES format.

As described in Section 2.3.2, the refined model was iteratively evaluated using the relevant pm4py quality metrics. Intermediate versions of the model were repeatedly validated with token-based replay. Whenever fitness fell below a target threshold of 80%, missing behaviour observed in the log was incorporated by refining gateways or reinstating activities with relevant frequency. Conversely, branches that contributed only minimally to replayability but required a substantial number of additional nodes were removed or consolidated. This ensured that the final model remained representative of the dominant process behaviour without becoming overly complex. After finalisation, the BPMN model was converted into a Petri net using pm4py to compute the relevant quantitative metrics as reported in Table 3. The resulting values are summarised in Table 4. The final model achieves a fitness of *94.06%*, which is close to the value obtained by the Heuristics Miner (95.46%) and considerably higher than that of the α-algorithm (38.26%). Its precision of *48.18%* lies between the Inductive Miner (14.07%) and the Heuristics Miner (67.11%), indicating a balanced degree of behavioural permissiveness. Generalization remains high at *97.51%*, similar to the value of the α-algorithm (98.24%) and above the values produced by the other two techniques.
Both custom simplicity metrics show clear improvements over the discovered nets. Node Simplicity (S_{nodes} = 0.011111) is higher than for the Inductive Miner (0.006993) and the Heuristics Miner (0.007937), reflecting a smaller structural footprint. Arc Simplicity (S_{arcs} = 0.465969) exceeds the values of all automatically discovered models, indicating a sparser flow relation. These improvements align with the design objective of constructing

a model that captures the essential behaviour of the loan application process while remaining interpretable for domain experts.

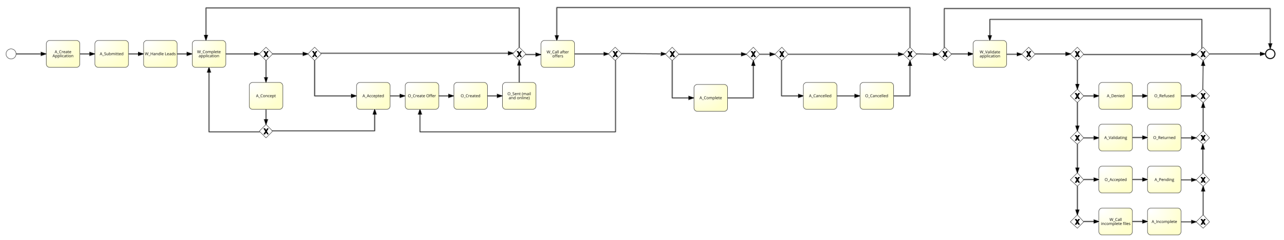

Figure 5 Final BPMN process model constructed from discovery results, domain knowledge, and iterative conformance analysis. An enlarged version of this figure is provided in Appendix 9.

Table 4 Quality metrics of the final BPMN model (computed analogously to Table 3).

Model	Fitness (%)	Precision (%)	Generalization (%)	S_{nodes}	S_{arcs}
Manual BPMN	*94.06*	*48.18*	*97.51*	*0.011111*	*0.465969*

2.4 Advanced Analysis

To complement the structural and conformance-oriented analysis, a process performance–focused temporal analysis of the BPIC–17 event log was conducted [12]. The analysis consists of two parts: a process variability analysis identifying activity-to-activity connections with both high frequency and wide throughput-time distribution, and a root cause analysis examining which behavioural and organisational factors are associated with long case durations.

2.4.1 Process Variability Analysis

This analysis is based on the hypothesis that a small subset of activity-to-activity transitions (*connections*) accounts for a disproportionately large share of throughput-time variability in the process. Such connections are particularly relevant for prediction and simulation models because transitions that occur frequently and exhibit highly variable throughput times introduce uncertainty into overall case duration. Identifying these transitions therefore helps determine which parts of the model require explicit stochastic timing assumptions to reproduce realistic waiting-time dynamics.

Approach and Results

The event log was ordered by case identifier and timestamp. For each event, the throughput time until the subsequent event in the same case was computed. These inter-event durations were then aggregated for each directly-following connection ($A \rightarrow B$), resulting in the following key statistics:

- $Case_{\%}$: the percentage of all cases in which the connection occurs, indicating how widespread the transition is,
- μ_{TPT}: the mean throughput time of the connection,
- σ_{TPT}: the standard deviation of throughput times,
- $\max(\sigma_{\text{TPT}}) \approx 956.93$h: the largest observed standard deviation across all connections, used for normalising σ_{TPT}.

[12] All computations of the advanced analysis can be found in the accompanying Jupyter notebook `04_advanced_anaylsis.ipynb`.

To jointly evaluate how *frequent* and how *variable* a connection is, a normalised comparative measure was defined:

$$C_{\text{relevance}} = \frac{\sigma_{\text{TPT}}}{\max(\sigma_{\text{TPT}})} \cdot Case_{\%}.$$

The resulting connection relevance score $C_{\text{relevance}}$ is designed to highlight connections that combine high throughput-time variability with high case coverage. A larger value of $C_{\text{relevance}}$ indicates that the connection contributes more strongly to process-wide temporal variability. The ten most relevant connections according to the computed score $B_{\text{relevance}}$ are listed in Table 5.

Table 5 Top 10 connections ranked by the comparative connection relevance score $C_{\text{relevance}}$.

Connection	$Case_{\%}$	μ_{TPT}	σ_{TPT}	$C_{\text{relevance}}$
1. W_Call after offers → W_Call after offers	99.53	1d 21h 49m	3d 2h 32m	0.0775
2. W_Call after offers → A_Cancelled	27.10	23d 3h 6m	8d 6h 15m	0.0561
3. W_Complete application → A_Accepted	70.61	0d 9h 2m	1d 8h 41m	0.0241
4. W_Call incomplete files → W_Call incomplete files	47.61	0d 12h 33m	1d 23h 30m	0.0236
5. W_Call incomplete files → O_Accepted	15.18	5d 8h 27m	5d 23h 38m	0.0228
6. W_Complete application → W_Complete application	78.23	0d 6h 49m	1d 1h 15m	0.0223
7. W_Validate application → W_Validate application	69.41	0d 10h 37m	1d 4h 47m	0.0209
8. W_Call after offers → O_Create Offer	11.92	5d 15h 13m	6d 18h 53m	0.0203
9. A_Concept → W_Complete application	70.66	0d 19h 38m	0d 22h 55m	0.0169
10. W_Validate application → O_Accepted	37.25	0d 17h 43m	1d 11h 31m	0.0138

Interpretation

The results show that the most substantial sources of process variability arise from applicant-facing communication and follow-up interactions. This observation is consistent with previous analyses of the BPIC–17 process (see [4]). The self-loop *W_Call after offers → W_Call after offers*, which occurs in 99.53% of all cases, exhibits a standard deviation of more than three days. This wide dispersion indicates that applicant response behaviour is inconsistent, making this connection a major contributor to throughput-time variability. The transition *W_Call after offers → A_Cancelled* reinforces this finding. With an average throughput time of 23 days and a standard deviation exceeding 8 days, cancellations often occur only after prolonged periods of inactivity. These delays are externally driven and therefore not controllable through internal process optimisation. As a result, they must be explicitly incorporated into predictive and simulation models. In contrast, internal processing steps such as *W_Validate application* or transitions leading to *O_Accepted* show substantially smaller standard deviations despite occurring frequently (e.g. 37.25% of all cases). Their comparatively low variability indicates that these parts of the process are executed in a more standardised and stable manner and therefore do not substantially contribute to system-wide temporal dispersion. Overall, the analysis demonstrates that applicant-facing interactions are characterised by non-standardised timing behaviour, resulting in wide throughput-time distributions. Moreover, this is an indicator of process inefficiency [6]. For predictive and simulation models, these high-variance connections must be represented using suitable stochastic timing assumptions. In particular, models should account for (i) heavy-tailed waiting-time distributions, (ii) high variability driven by external response behaviour, and (iii) the possibility of repeated follow-up loops before a case progresses or is cancelled. Incorporating these characteristics is essential to realistically capture the observed variability in case duration and to produce reliable forecasts of process performance.

2.4.2 Root Cause Analysis

While the process variability analysis highlights where wide throughput-time distributions occur, it does not yet clarify why certain cases become substantially longer than others. This analysis therefore examines the assumption that extended case durations arise from specific behavioural and structural patterns within the process. In particular, it is assumed that long cases are not merely the result of a high number of executed events, but are associated with prolonged inactivity, the presence of escalation and cancellation activities, repeated rework and the influence of automatic system-triggered events. Identifying these mechanisms is essential for determining which behavioural dynamics must be explicitly incorporated into prediction and simulation models.

Approach and Results

To investigate this hypothesis, a case-level feature set was constructed. For each case, the total throughput time in days, the mean waiting time per event, the total number of events, the number of distinct resources involved and the number of rework occurrences were computed. In addition, the presence of selected activities, in particular cancellation and escalation steps, was encoded using binary indicators. Following the findings of Rodrigues et al. [4], events executed by the resource *User_1* were classified as automatic system-generated events, whereas all remaining events were treated as manually executed. Based on this classification, the proportion of automatic and manual events per case was derived. A Pearson correlation analysis was then performed to quantify the statistical association between these features and case throughput time. Table 6 reports the selected correlations. The mean waiting time per case shows the highest correlation with throughput time (approximately 0.75). The presence of *A_Cancelled* and *O_Cancelled* correlates positively (around 0.43). Rework frequency correlates at approximately 0.34. The proportion of automatic events is positively correlated (around 0.32), whereas the proportion of manual events shows a negative correlation (around −0.32). The activity *O_Accepted* also displays a negative correlation with throughput time.

Table 6 Correlation of selected case-level features with throughput time.

Feature	**Correlation with throughput time**
Mean waiting time per case (hours)	0.746
Activity present: A_Cancelled	0.427
Activity present: O_Cancelled	0.425
Rework count	0.336
Proportion of automatic events	0.323
Activity present: W_Personal Loan collection	0.152
Activity present: W_Handle leads	0.144
Proportion of manual events	−0.323
Activity present: O_Accepted	−0.314

Interpretation

For prediction and simulation, the root cause analysis yields two central implications. First, the identified correlations specify which case-level features must be included as explanatory variables in predictive models of throughput time. These include mean waiting time, the presence of cancellation-related activities and the proportion of automatic event creation, as these features exhibit the strongest statistical association with long case durations.Second, the analysis clarifies which behavioural mechanisms must be explicitly incorporated into simulation models to reproduce the empirical dynamics of the process. Applicant responsiveness must be represented through empirically grounded stochastic waiting-time distributions rather than fixed delays, because prolonged inactivity is strongly associated with extended case durations. Escalation and cancellation behaviour must be encoded using rule-based triggers that activate when waiting thresholds are exceeded. Rework must

be modelled using empirically estimated recurrence probabilities, since repeated execution of the same activity contribute to throughput time.
By integrating these mechanisms, simulation models can reproduce not only the average behaviour of the process but also the long-tailed throughput-time distribution, which arises from extended waiting periods and repeated escalation cycles. This aligns with prior findings on the BPIC–17 process, which similarly report that most bottlenecks stem from applicant-driven delays in submitting required documents rather than from inefficiencies in internal processing [4].

3 Conclusion

This report presents a comprehensive data- and process-science analysis of the BPIC–2017 loan application process. After establishing a reproducible Python-based technical environment, the event log was examined to reveal its structural complexity, pronounced temporal variation and recurrent rework patterns. Three process discovery algorithms were applied and evaluated systematically, showing that although the Inductive Miner achieves perfect fitness, all automatically discovered models exhibit considerable structural complexity that limits interpretability for domain experts. To address this, a simplified yet behaviourally faithful BPMN model was manually constructed, iteratively validated, and shown to provide a more balanced representation by combining reduced modelling complexity with strong fitness and generalisation. The advanced analyses further deepen the understanding of the process. The process variability analysis identifies those activity-to-activity connections that contribute most strongly to process-wide temporal variability, showing that high-variance delays arise predominantly in applicant-facing communication loops. The root cause analysis complements these findings by demonstrating that long case durations are driven primarily by extended waiting periods, escalation and cancellation behaviour, repeated rework and a high proportion of automatic system-triggered events. Together, these analyses confirm that throughput time in BPIC–2017 is shaped far more by applicant-driven delays than by inefficiencies in internal processing.
Overall, this report provides an empirically grounded characterisation of the structural and temporal behaviour of the process. The findings establish a solid foundation for future work in process simulation and predictive modelling by identifying the specific mechanisms such as stochastic waiting times, escalation triggers and rework dynamics.

References

[1] Liese Blevi, Lucie Delporte, and Julie Robbrecht. Process mining on the loan application process of a dutch financial institute (bpi challenge 2017, professional winner), 2017. KPMG Technology Advisory.

[2] Josep Carmona, Boudewijn Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking: Relating Processes and Models*. 2018.

[3] Elizaveta Povalyaeva, Ismail Khamitov, and Artyom Fomenko. Bpic 2017: Density analysis of the interaction with clients (bpi challenge 2017, student winner). In *BPI Challenge 2017 Reports*, 2017. National Research University Higher School of Economics.

[4] Ariane M. B. Rodrigues, Cassio F. P. Almeida, Daniel D. G. Saraiva, Felipe B. Moreira, Georges M. Spyrides, Guilherme Varela, Gustavo M. Krieger, Igor T. Peres, Leila F. Dantas, Mauricio Lana, Odair E. Alves, Rafael França, Ricardo A. Q. Neira, Sonia F. Gonzalez, and William P. D. Fernandes. Stairway to value: mining a loan application process (bpi challenge 2017, academic winner). In *BPI Challenge 2017 Reports*, 2017. Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

[5] W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[6] Wil M. P. van der Aalst. *Process Mining: Data Science in Action*. Springer, 2016.

[7] Boudewijn F. van Dongen, Jan Claes, Andrea Burattin, and Jochen De Weerdt. BPI Challenge 2017: Loan Application Process of a Dutch Financial Institute. https://ais.win.tue.nl/bpi/2017/challenge.html, 2017. Accessed: 19.11.2025.

[8] A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible heuristics miner (fhm). BETA Working Paper Series WP 334, Eindhoven University of Technology, Eindhoven, 2010.

Declaration of Generative AI Use

In the preparation of this report, generative AI tools were used exclusively for linguistic refinement, including grammar correction, stylistic polishing, and improvements in clarity of expression. At no point were generative AI systems used to produce, modify, or influence the scientific content, including the formulation of arguments, methodological choices, data analyses, computations, interpretations, or conclusions. All scientific results and all domain-specific reasoning presented in this work were developed independently by the author, who assumes full responsibility for the accuracy and integrity of the content.

A Appendix

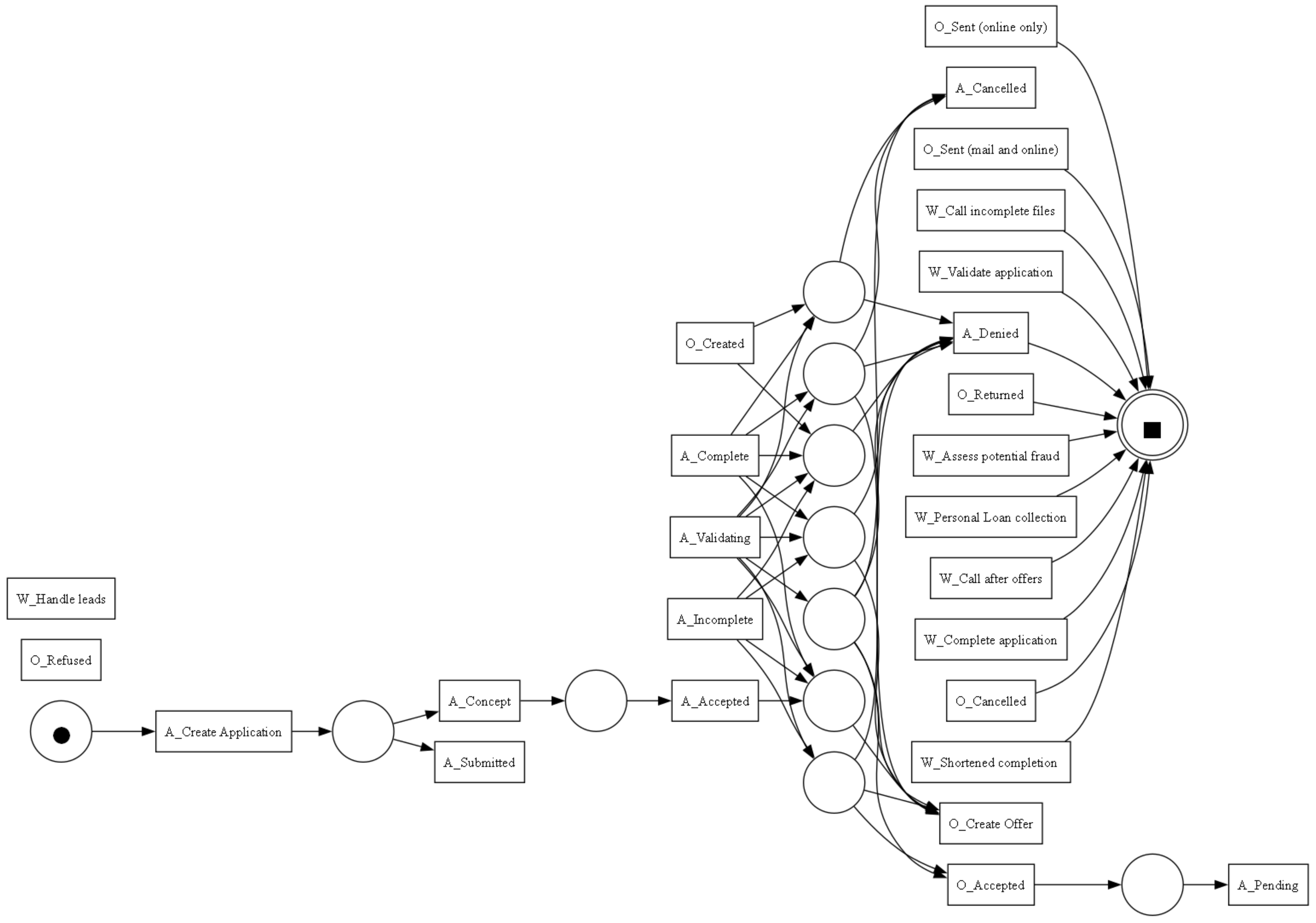

Figure 6 Full-page view of the Petri net discovered by the α-algorithm using PM4Py.

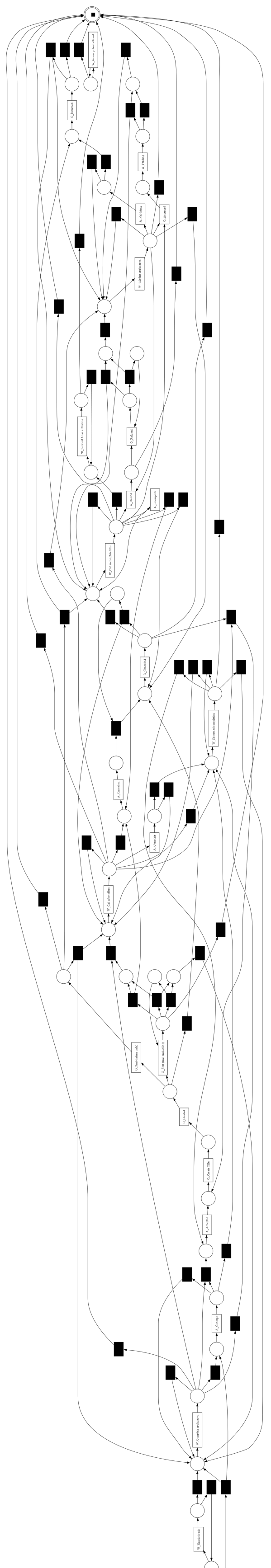

Figure 7 Full-page view of the Petri net discovered by the Heuristics Miner using PM4Py.

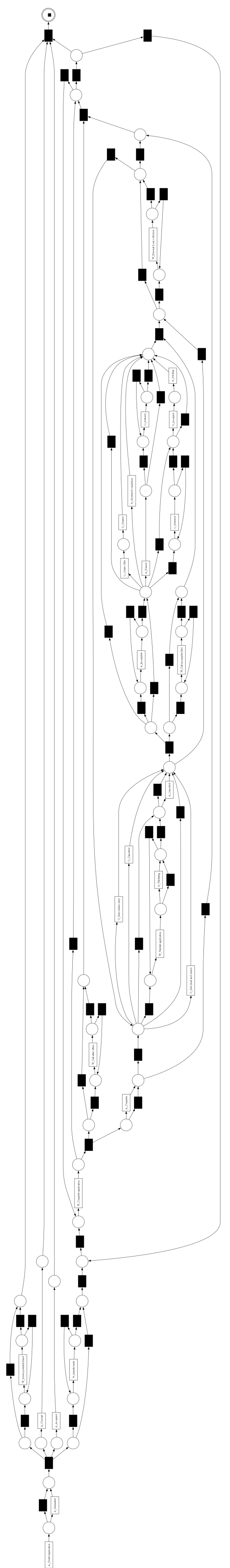

Figure 8 Full-page view of the Petri net discovered by the Inductive Miner using PM4Py.

```python
def compute_simplicity_metrics(net):
    # basic structural counts
    num_places = len(net.places)
    num_transitions = len(net.transitions)
    num_arcs = len(net.arcs)

    node_count = num_places + num_transitions

    # metric 1: Node Count Simplicity (fewer nodes -> simpler model)
    simplicity_node_count = 1 / (1 + node_count)

    # metric 2: Arc Density Simplicity (fewer arcs per node -> simpler model)
    arc_node_ratio = num_arcs / node_count if node_count > 0 else float("nan")
    simplicity_arc_density = (
        1 / (1 + arc_node_ratio) if arc_node_ratio > 0 else float("nan")
    )

    return {
        "num_places": num_places,
        "num_transitions": num_transitions,
        "num_arcs": num_arcs,
        "node_count": node_count,
        "arc_node_ratio": arc_node_ratio,
        "simplicity_node_count": simplicity_node_count,
        "simplicity_arc_density": simplicity_arc_density,
    }


# compute simplicity for all three models
simplicity_im = compute_simplicity_metrics(net_im)
simplicity_hm = compute_simplicity_metrics(net_hm)
simplicity_alpha = compute_simplicity_metrics(net_alpha)

# summary table
simplicity_summary = pd.DataFrame(
    [
        {"model": "inductive", **simplicity_im},
        {"model": "heuristics", **simplicity_hm},
        {"model": "alpha", **simplicity_alpha},
    ]
)
```

Listing 1 Implementation of the two structural simplicity metrics.

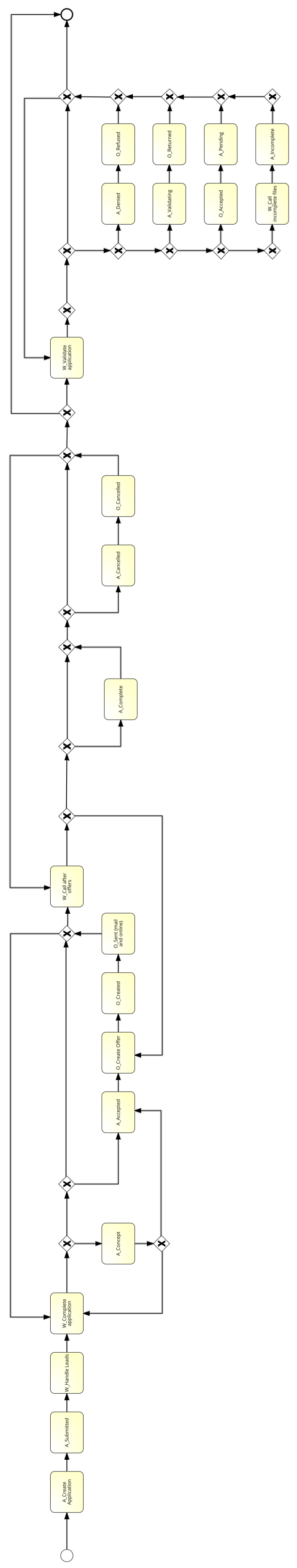

Figure 9 Full-page view of the final BPMN process model.