ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΑΡΙΑ ΦΩΤΕΙΝΗ ΑΛΑΤΣΑΚΗ
CSD4584

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Comparative Analysis of Process Mining Algorithms Using Graphs

Συγκριτική Ανάλυση Αλγορίθμων Εξόρυξης Διαδικασιών με χρήση Γραφημάτων

Επόπτης Καθηγητής : Κωνσταντίνος Στεφανίδης
Επιβλέπων : Ξενοφών Ζαμπούλης
Μέλος της Επιτροπής Παρακαλούθησης : Γρηγόρης Τσαγκατάκης

Contents

Abstract

This thesis presents a data-driven framework for discovering and comparing process models derived from ethnographic observations of real human workflows. The study focuses on transforming qualitative field data—recordings of manual and cultural activities such as weaving, polishing, and restoration—into quantitative event logs suitable for process mining analysis . Initially, raw textual observations are preprocessed and converted into the standardized .xes (eXtensible Event Stream) format, enabling their integration into the ProM process mining platform. Within ProM, two foundational process discovery algorithms—the Alpha Miner and the Inductive Miner —are applied to extract formal models of the observed workflows in the form of Petri Nets.

Each resulting model is exported as a .pnml (Petri Net Markup Language) file, which encodes the underlying structure of the process graph, including its activities, places, and arcs. A custom Python parsing script was developed to automatically convert these PNML files into simplified, human-readable .txt representations, summarizing the detected activities, transitions, and Activity□Activity relations. This textual abstraction enables systematic comparison between algorithms in terms of model completeness, structure, and behavioral accuracy.

Results show that the Alpha Miner tends to generate concise, linear models capturing the dominant workflow sequence, whereas the Inductive Miner produces more detailed and sound process structures that capture concurrency and optional behaviors. The developed methodology establishes a reproducible pipeline—from ethnographic data collection to automated model extraction and textual graph analysis—that bridges qualitative observation with computational process discovery.

Introduction

Digitization and computational analysis have transformed the way we observe, document, and interpret human activity. In cultural and ethnographic research, where the focus lies on understanding complex manual or creative workflows—such as weaving, restoration, or polishing—data-driven methods now enable the systematic exploration of behavioral patterns that were once only described qualitatively. Process mining, a field at the intersection of data science and workflow analysis [1] , provides a framework for discovering, monitoring, and improving real processes by extracting models from recorded event data.

Traditional ethnographic studies emphasize rich qualitative narratives but often lack formal, analyzable models of process behavior. Conversely, process mining relies on structured event logs but is rarely applied to the fluid, human-centered workflows found in cultural production. Bridging these two domains requires translating observational data into event-based representations that computational algorithms can analyze while preserving the semantic meaning of human actions.

Recent developments in the ProM process-mining tool and algorithms such as Alpha Miner and Inductive Miner [2], [3] have made it possible to automatically infer formal process models from event logs. These models, typically represented as Petri Nets [6], capture the causal and temporal relationships between activities in a process. However, when applied to real-world, ethnographically sourced data, challenges arise: logs may be incomplete, noisy, or ambiguous, and the resulting models can differ significantly depending on the chosen discovery algorithm.

This thesis introduces a structured workflow for transforming ethnographic activity data into analyzable process representations and for comparing the results of different process discovery algorithms. The proposed pipeline involves:

1. **Data Preprocessing and Conversion** — Raw observational notes, initially formatted as plain-text activity logs, are cleaned and converted into .xes files suitable for analysis in ProM.
2. **Process Discovery** — Within ProM, the Alpha Miner and Inductive Miner algorithms are applied to the same event logs to generate distinct Petri Net models representing the underlying workflow.
3. **Model Extraction and Simplification** — Each Petri Net is exported as a .pnml file. A custom Python script parses these XML-based models, extracting transitions, places, and arcs, and outputs human-readable .txt summaries that encode the relationships between activities.
4. **Comparative Analysis** — The resulting textual and graphical representations are analyzed to assess model complexity, completeness, and structural differences between the two algorithms.

By combining qualitative ethnographic observation with quantitative process-mining analysis, this work aims to evaluate how different discovery algorithms reconstruct real human workflows. The Alpha Miner, as a foundational approach, provides concise representations ideal for structured, noise-free processes, while the Inductive Miner offers robust and sound models that can handle concurrency and variation.

Through this comparative study, the thesis contributes to the understanding of how automated process discovery techniques can be adapted for ethnographic and cultural research, offering a reproducible tool that bridges human-centered observation and computational modeling.

The remainder of this thesis is organized as follows:
Section 2 reviews related work in process mining, workflow discovery algorithms, PNML representation, and the application of process mining to ethnographic data.
Section 3 describes the methodology, including data preprocessing, event log conversion to the XES format, process discovery using the Alpha Miner and Inductive Miner algorithms, model export in PNML, and the development of a Python-based PNML-to-text conversion script.
Section 4 presents the experimental results and discussion, including synthetic test cases (loop, choice, and parallel patterns) and real-world applications to the glass-blowing and wood-turning datasets. Comparative analysis between the two algorithms highlights differences in model completeness, structure, and behavioral accuracy.
Section 5 concludes the thesis with a summary of findings, key insights, limitations, and suggestions for future work on process mining in ethnographic and cultural contexts.

2. Related Work

2.1 Process Mining and Workflow Discovery

Process mining is a discipline that bridges data science and process modeling by extracting structured process models from event logs that record real executions of activities over time. The main goal is to discover how processes actually occur in practice, rather than how they are prescribed. Since its formalization by van der Aalst and colleagues [1] , process mining has become a foundational tool in business process management and has recently extended to domains such as healthcare, manufacturing, and cultural workflows.Within process mining, three primary tasks are distinguished: process discovery, which automatically builds a model from event logs; conformance checking, which compares real behavior to an existing model; and enhancement [2], which uses insights from event data to improve the process. This thesis focuses exclusively on process discovery , applying algorithms to ethnographically derived data to extract visual representations of real human workflows.

2.2 Process Discovery Algorithms

The two process discovery algorithms applied in this work are Alpha Miner [2] and Inductive Miner [3].

The Alpha Miner is the first and most widely taught discovery algorithm. It identifies causal, parallel, and sequential relations between activities based on their order of occurrence in event logs. Its output is a Petri Net graph that formally represents the control flow of the observed process. However, Alpha Miner is sensitive to noisy or incomplete data and often fails to capture complex or concurrent behavior.

The Inductive Miner , on the other hand, uses a recursive divide-and-conquer approach to partition the event log into smaller sublogs and construct a hierarchical process tree. The key advantage of Inductive Miner is that it guarantees soundness—the generated model can always be executed without deadlocks or livelocks—and performs well even with real-world, unstructured data.

In this study, both algorithms were applied using the ProM tool, allowing a direct comparison of the structural differences in the Petri Nets generated from the same ethnographic dataset.

2.3 Model Export and PNML Representation

The models produced by process mining algorithms can be exported in the .pnml format [6] , a standardized XML-based notation that encodes the structure of Petri Nets, including places, transitions, and arcs. PNML enables interoperability between tools and serves as a convenient format for further analysis or visualization outside ProM.

In this project, the PNML files produced by the Alpha and Inductive Miner algorithms were used as the basis for automated text conversion. This step was implemented through a custom Python script, which reads the PNML structure, extracts transitions and their relations, and exports the results as .txt files listing all detected activities and Activity□Activity connections. This textual representation allows for future quantitative comparison between the models without requiring direct use of ProM or visual inspection.

2.4 Process Mining for Ethnographic Data

Although process mining was originally developed for enterprise systems, recent work has explored its application in human-centered and ethnographic contexts [8]. In these settings, the focus shifts from performance optimization to understanding patterns of human activity, decision-making, and cultural workflows. Ethnographic data often consist of manually recorded observations or transcriptions of sequential human actions, which can be converted into event logs compatible with process-mining tools.

Applying algorithms such as Alpha and Inductive Miner to ethnographic datasets allows researchers to visualize how real activities unfold over time, identify variations or repetitions, and compare different representations of the same observed process. The work presented in this thesis adopts this approach, serving as a first step toward modeling and comparing human workflows using automated process discovery techniques.

3. Methodology

3.1 Overview

The methodology of this work focuses on the application of process mining techniques to ethnographic data, with the goal of automatically discovering and representing human workflows as Petri Net–based models. The approach consists of four main stages:

1. **Data Preparation** – converting the provided ethnographic logs into the standardized XES format;
2. **Process Discovery** – applying the *Alpha Miner* and *Inductive Miner* algorithms using the *ProM* tool;
3. **Model Export** – saving the generated Petri Nets in PNML format;
4. **Model Conversion and Analysis** – using a Python-based parser to extract key elements from PNML files and export them as textual summaries.

This pipeline enables a clear and reproducible workflow for analyzing the structure of real-world processes while maintaining compatibility across software tools.

3.2 Data Preparation

The raw dataset provided for this study consisted of text-based descriptions of sequential human activities recorded during ethnographic observation. Each record corresponded to an event, including an activity name and its occurrence order. To enable process mining, the data were converted into the .xes format , which is the standard input for the ProM tool [1]. The conversion process involved mapping each activity line to an XES event with a unique case identifier and timestamp. This ensured that ProM could interpret the data as valid process executions suitable for algorithmic analysis.

3.3 Process Discovery in ProM

The ProM tool was used to perform process discovery. Two discovery algorithms were selected based on their complementary characteristics and frequent use in the literature: Alpha Miner and Inductive Miner.

- The Alpha Miner reconstructs causal relations between activities by analyzing their direct succession patterns in the event log. It produces a Petri Net model that visually represents the control-flow logic inferred from the data.
- The Inductive Miner, in contrast, applies a hierarchical decomposition strategy that guarantees a sound, block-structured model even when the data contain noise or missing events.

Each algorithm was executed separately on the same XES log, generating distinct Petri Nets that reflect different structural interpretations of the same process. The resulting models were visualized within ProM for verification.

3.4 Exporting Models in PNML Format

The Petri Nets generated by each algorithm were exported from ProM in the .pnml format. PNML is an XML-based representation [6] that encodes the model's structural components —places, transitions, and arcs—along with any associated labels.
This format was chosen because it preserves the full topology of the discovered model and can be easily parsed by external scripts for analysis or visualization.

Each algorithm's PNML file was stored separately for further processing.
For example:

- `alpha.pnml` — model discovered by Alpha Miner
- `inductive.pnml` — model discovered by Inductive Miner

3.5 Text Conversion and Structural Extraction

To enable model comparison and further analysis outside ProM, a Python-based parser was developed to transform PNML files into human-readable text files.

The script performs the following steps:

1. Reads the PNML file as raw XML.
2. Extracts all places, transitions, and arcs from the file.
3. Identifies the start and end nodes based on PNML annotations or naming conventions.
4. Generates a summary listing all transitions (activities) and the directed edges (activity → activity) that define their relationships.

The output is a `.txt` file that lists:

- The number of activities, places, and arcs;
- The set of detected start and end places;
- A complete mapping of transitions and connections.

This textual format serves as a simplified but complete representation of the process model, enabling direct comparison between algorithms without relying on graphical visualization.

3.6 Summary of Current Stage

At this stage of the project, the following milestones have been achieved:

- Successful conversion of ethnographic data into XES format;
- Execution of Alpha Miner and Inductive Miner in ProM;
- Export of resulting Petri Nets to PNML files;
- Development of a Python script for PNML-to-text transformation.

The next step will involve defining a method to compare the textual or graph-based representations produced by the two algorithms and extending the analysis to larger and more complex datasets provided by the supervisor.

Section 4 – Results and Discussion

This section presents the results of the process discovery experiments performed using the Alpha Miner and Inductive Miner algorithms [2],[3] within the ProM tool.
Three synthetic event logs were designed to represent fundamental workflow patterns—Loop, Choice, and Parallel—each stored in XES format and executed through both algorithms. The discovered models were then exported in PNML form and analyzed in text to evaluate structure, correctness, and behavioral accuracy.

4.1 Loop Scenario (A → B → C)

The *loop.xes* log simulated a process where activity B may repeat one or more times between A and C.

Alpha Miner Result:
The Alpha Miner produced a fragmented Petri net, linking only A and C while leaving B disconnected. Because Alpha Miner relies strictly on direct succession relations [2], it fails to model cyclic or repeated behavior and cannot represent loops within its basic net structure.

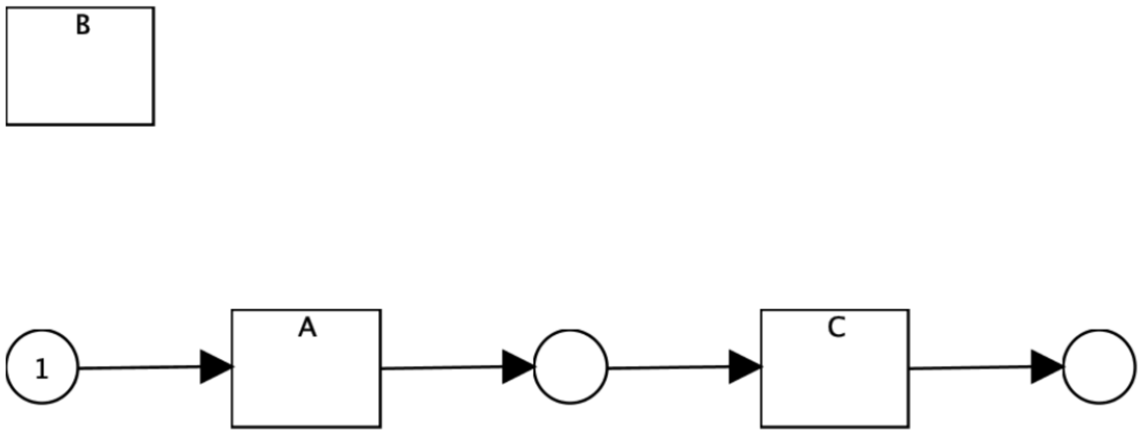


Figure 1: The Alpha Miner cannot generalize repeated sequences

Inductive Miner Result:

The Inductive Miner correctly recognized the iterative pattern [3] and generated a block-structured model that allows multiple occurrences of B before continuing to C. This demonstrates the algorithm's capability to detect repetition and to build semantically valid loop constructs.

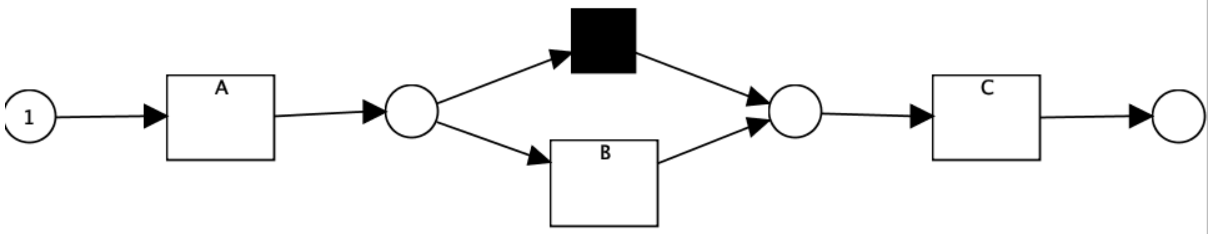
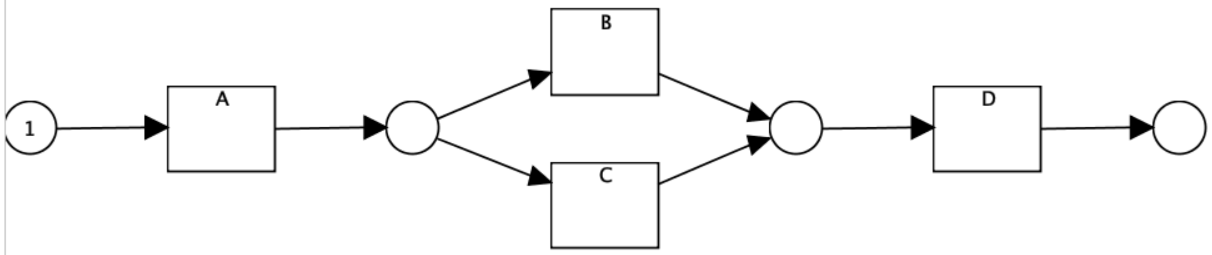


Figure 2: Inductive Miner reliably reconstructs loop semantics.

4.2 Choice Scenario (A → B or C → D)

The *choice.xes* log modeled an exclusive choice where, after A, either B or C is executed before the process converges at D.



Both algorithms produced identical and correct models. Each identified the branching point after A and the merge before D, resulting in a simple, well-structured choice pattern. In straightforward, non-overlapping decision structures, Alpha Miner and Inductive Miner perform equivalently, since no concurrency or iteration analysis is required.

4.3 Parallel Scenario (A → [B and C in parallel] → D)

The *parallel.xes* log represented a concurrent pattern where B and C can execute independently and must both complete before D.

Alpha Miner Result:

Produced two separate paths from A to D but did not encode synchronization, effectively treating the parallelism as an alternative flow.

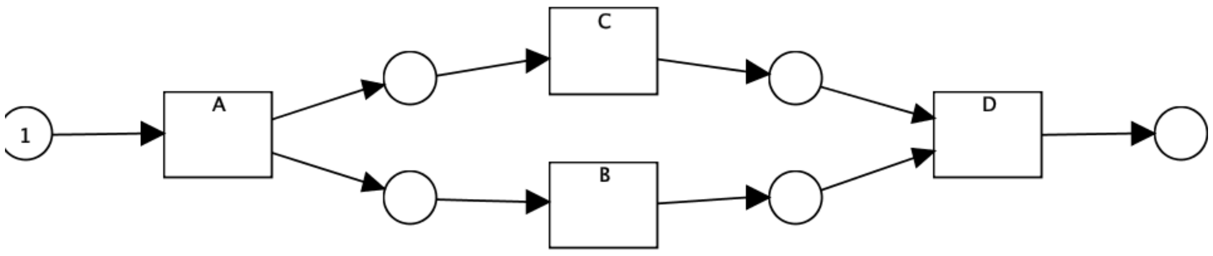


Figure 3: The Alpha Miner oversimplifies it into sequential behavior.

Inductive Miner Result:

Generated a fully block-structured Petri net with AND-split and AND-join gateways (displayed as black squares in the visualization). The model explicitly represents concurrency, allowing B and C to occur in any order or simultaneously, and ensures both must finish before D.

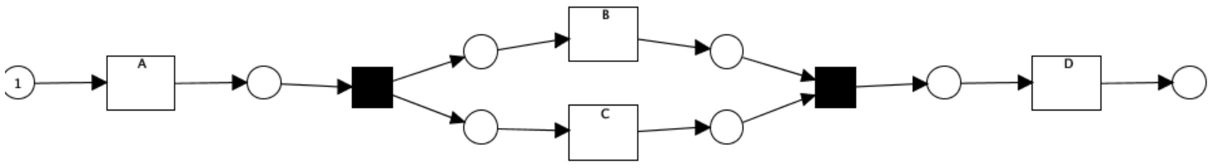


Figure 4: The Inductive Miner successfully captures concurrent execution.

4.4 Comparative Summary

Workflow	Alpha Miner	Inductive Miner	Evaluation
Loop	Fails – no loop detected	Correct loop identified	Inductive Miner superior
Choice	Correct	Correct	Both similar
Parallel	Partial, no true concurrency	Accurate AND-split/join	Inductive Miner superior

The Inductive Miner consistently yields block-structured, sound, and semantically meaningful Petri nets, especially when handling repetition and concurrency. The Alpha Miner, while effective for simple acyclic and choice-based workflows, lacks robustness for more complex behavioral relations. These results align with the theoretical properties of both algorithms [2],[3]: Alpha Miner is pattern-driven and limited by direct-follows relations, whereas Inductive Miner recursively detects and composes constructs that preserve model soundness.

4.5 Real-World Dataset: Glass-Blowing Process

The dataset provided by the supervisor consisted of detailed textual descriptions of a traditional glass-blowing workflow. Each record corresponded to an observed activity (e.g. Getting a blowpipe, Gathering glass from the furnace, Shaping the glass, Heating the object) and was linked to a higher-level production instance. After preprocessing, the file was transformed into a standardized event log (Glass.xes) and imported into ProM for process discovery [1], [2], [3] using both the Alpha Miner and Inductive Miner algorithms.

Alpha Miner Result

The Alpha Miner produced a concise and straightforward Petri Net that captures the main sequential flow of the glass-blowing process. It successfully identifies the dominant order of operations, outlining the core progression from material gathering and heating to shaping and finishing. However, because the Alpha Miner relies solely on direct-succession relations between activities, it is unable to represent repetitive, concurrent, or optional behaviors present in the real process. Cycles such as the repeated *Heating* and *Shaping* stages are collapsed into a single linear sequence, resulting in a simplified depiction of the workflow. While this makes the model clear and easily interpretable, it also means that the inherent variability and dynamic structure of the actual glass-blowing process are not fully captured.

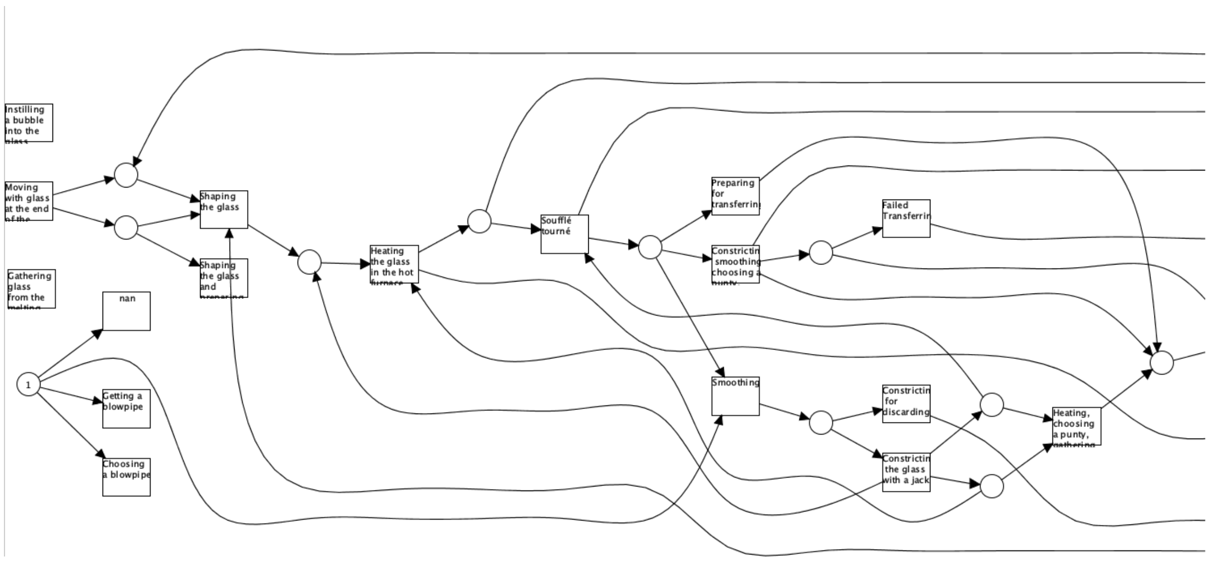

Instilling a bubble into the
Moving with glass at the end
Gathering glass from the
nan
1
Getting a blowpipe
Choosing a blowpipe
Shaping the glass
Shaping the glass and
Heating the glass in the hot
Soufflé tourné
Preparing for transferrin
Constrictin smoothing choosing a
Failed Transferrin
Smoothing
Constrictin for discarding
Constrictin the glass with a jack
Heating, choosing a punty,

Figure 5a: Alpha Miner model – left section

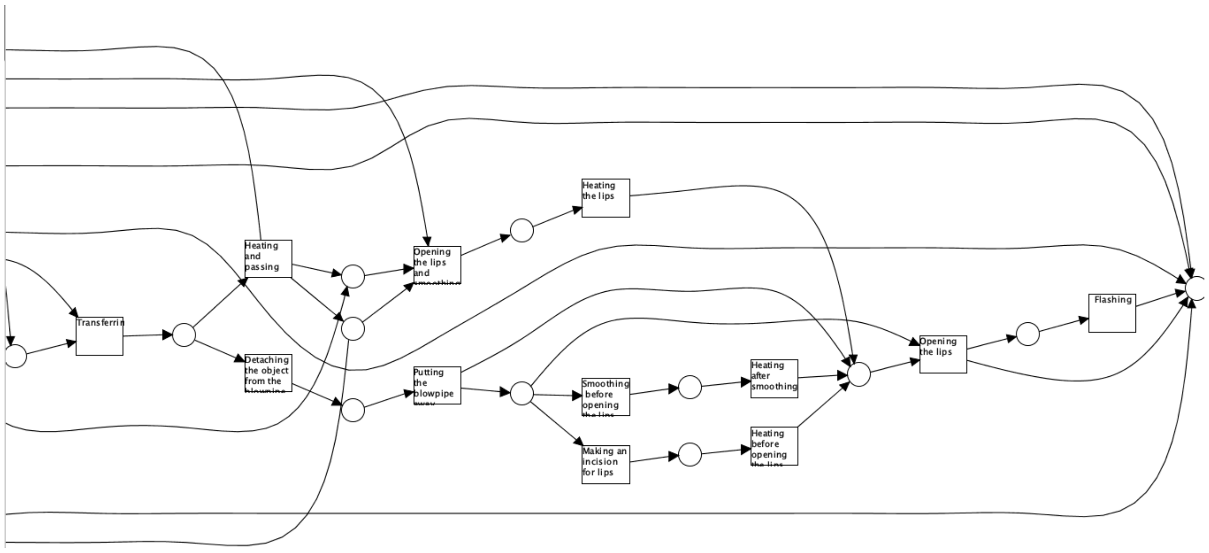


Figure 5b: Alpha Miner model – right section

Inductive Miner Result

The Inductive Miner produced a well-structured and sound Petri Net that accurately reflects the logic of the glass-blowing workflow. Unlike the Alpha Miner, which tends to linearize activities, the Inductive Miner identifies the true hierarchical organization of the process through its recursive decomposition approach. The resulting model correctly captures repeated sub-cycles, such as the alternating *Heating* ↔ *Shaping* stages, and includes optional branches representing activities like *Opening the lips and smoothing* or *Preparing for transferring*. The Inductive Miner preserves both behavioral completeness and model soundness [6]. Overall, it generates a semantically coherent and behaviorally faithful representation of the real workflow, successfully distinguishing between sequential, parallel, and iterative sections of the production process.

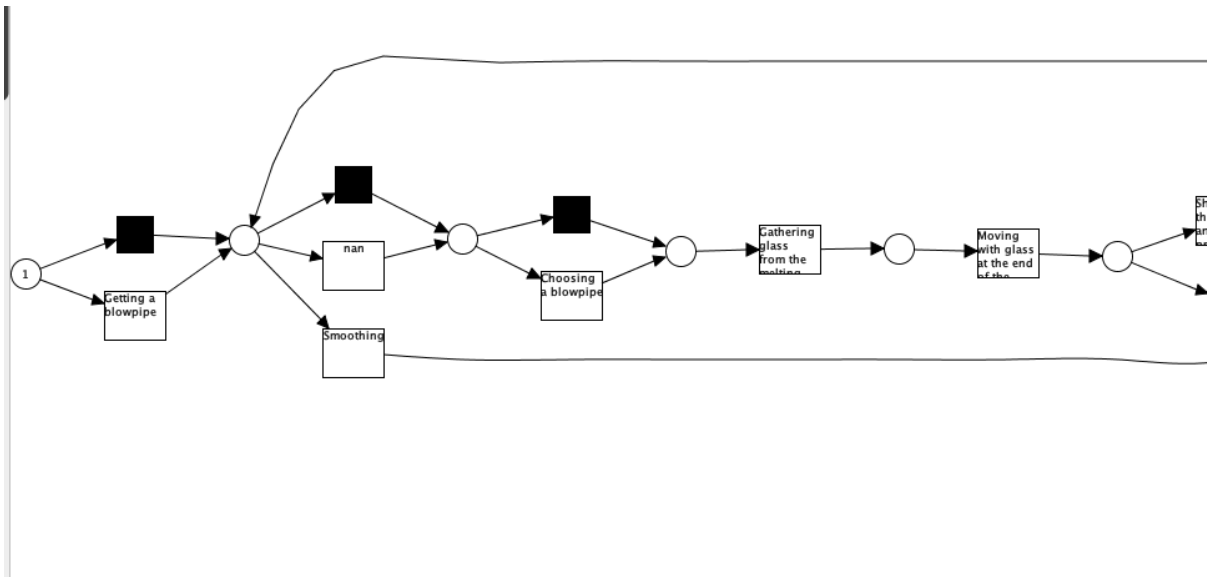

1
Getting a blowpipe
nan
Smoothing
Choosing a blowpipe
Gathering glass from the melting
Moving with glass at the end of the

Figure 6a: Inductive Miner model – start point

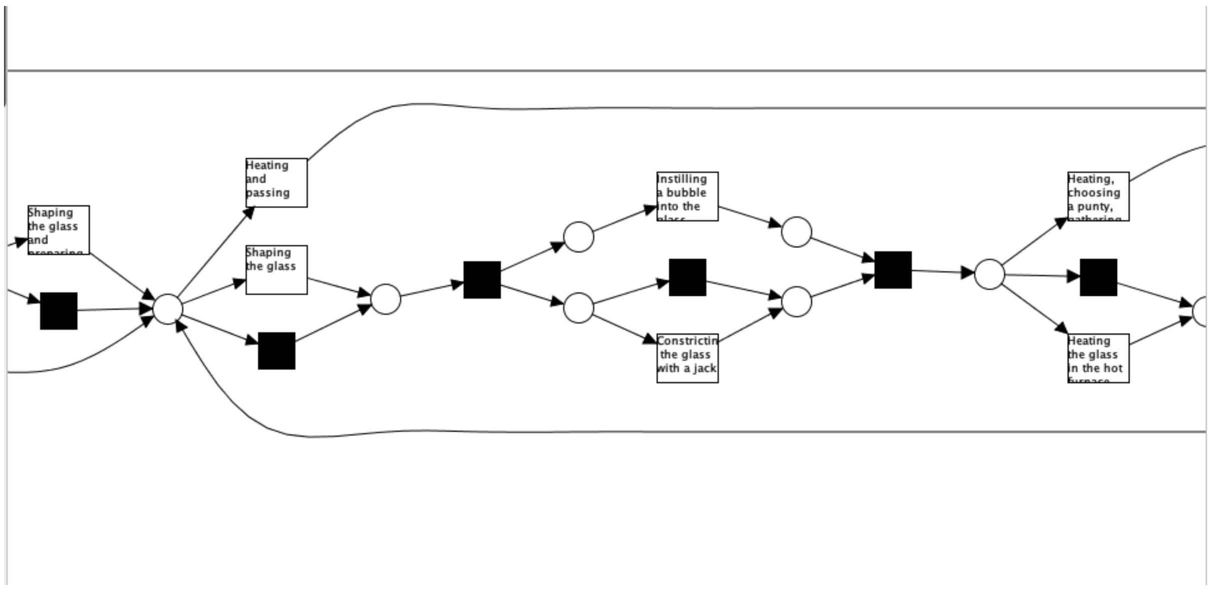

Shaping the glass and preparing
Heating and passing
Shaping the glass
Instilling a bubble into the glass
Constricting the glass with a jack
Heating, choosing a punty, gathering
Heating the glass in the hot furnace

Figure 6b: Inductive Miner model – main sequence

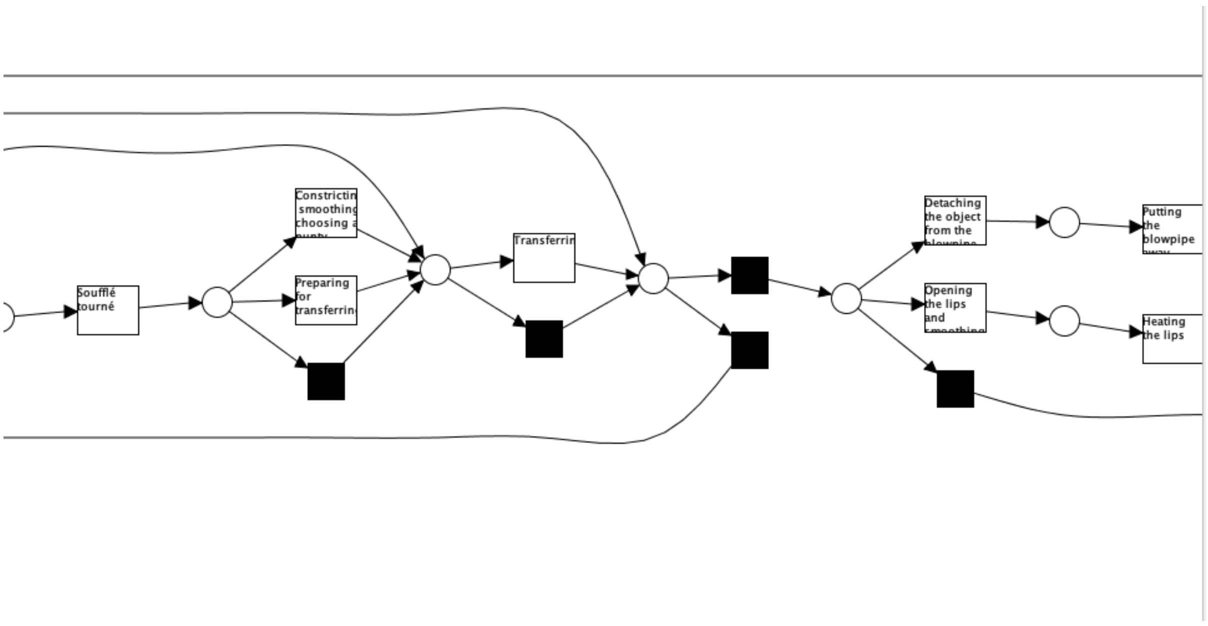

Soufflé tourné
Constrictin smoothing choosing a
Preparing for transferrin
Transferrin
Detaching the object from the
Putting the blowpipe
Opening the lips and
Heating the lips

Figure 6c: Inductive Miner model – middle point

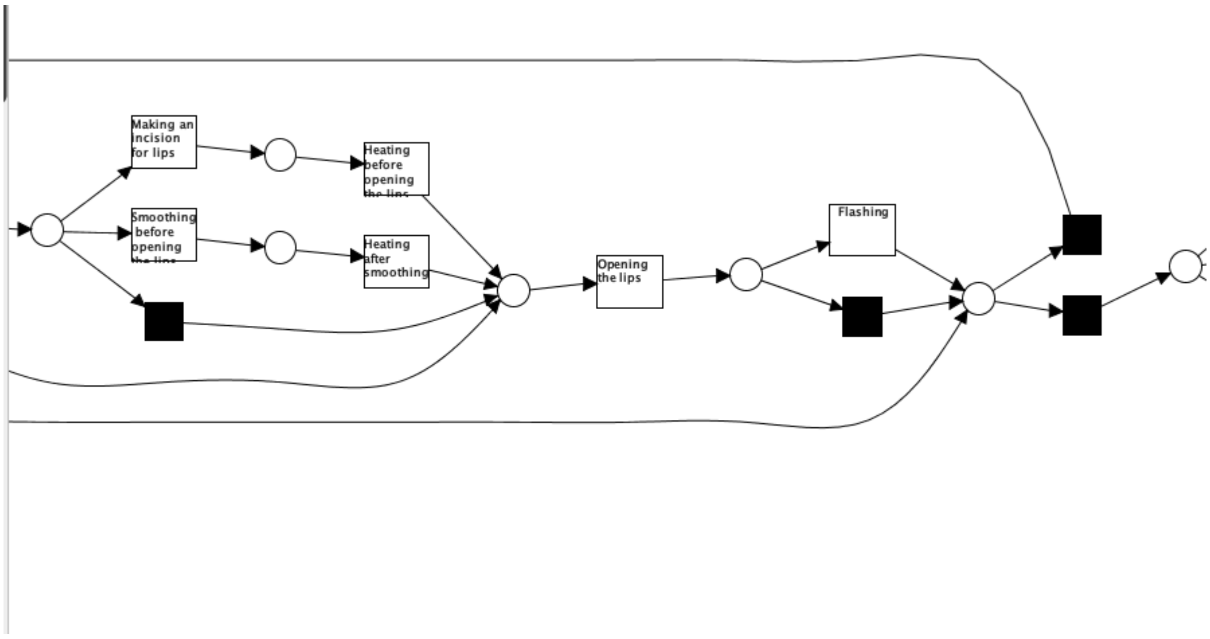


Figure 6d: Inductive Miner model – right section

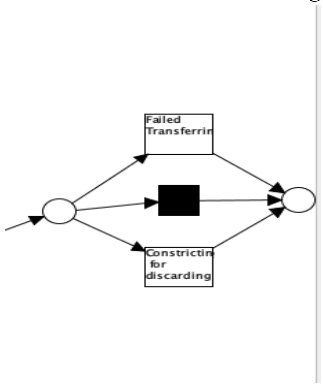


Figure 6e: Inductive Miner model – end point

Conclusion

Both algorithms reconstruct the essential order of activities within the glass-blowing process, but they differ significantly in expressive power. The Alpha Miner offers a high-level skeleton of the workflow, suitable for summarization or educational visualization, yet it omits loops and concurrency that occur in practice. The Inductive Miner, by contrast, produces a behaviorally complete representation, reflecting the true cyclical nature of glass production. These results confirm the theoretical strengths of Inductive Miner in handling unstructured, human-centered workflows while illustrating the simplicity and limitations of the classical Alpha Miner approach.

4.6 Real-World Dataset: Wood-Turning Process

The second dataset provided by the supervisor represents a wood-turning production workflow, documenting the key stages of material preparation, shaping, and finishing. Each activity was recorded in a structured text file, then converted into a standardized event log and imported into ProM for process discovery. Both the Alpha Miner and Inductive Miner algorithms were applied to the same event log to analyze how each captures the logic of the process.

Alpha Miner Result

The Alpha Miner generated a simple, sequential model that outlines the main operational flow of the wood-turning process. It clearly represents the linear relationship between the primary activities—starting from material setup, followed by shaping, and ending with finishing. However, as observed in previous experiments, the Alpha Miner is limited to direct-succession behavior, which prevents it from modeling repetitions or concurrent steps that may occur during the actual manufacturing process. This leads to a clear but simplified representation, capturing the dominant flow but omitting iterative refinements and alternative tool-handling sequences.

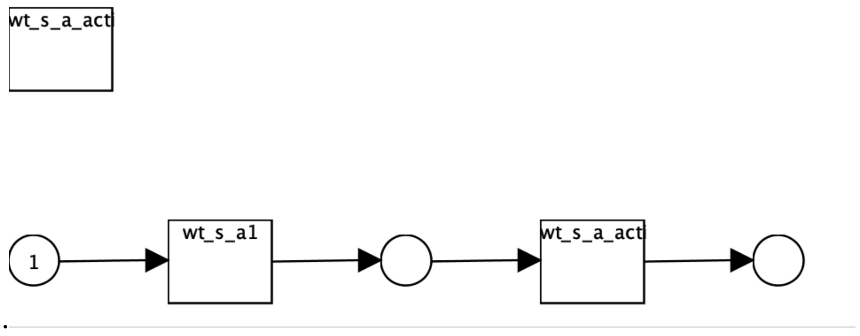


Figure 7: Alpha Miner model- The model shows a clear sequential pattern without explicit loops or branching.

Inductive Miner Result

The Inductive Miner produced a structured and sound Petri Net that better reflects the logical organization and optional behaviors within the wood-turning workflow.
It correctly identifies branching control-flow structures that represent parallel or optional shaping operations, and it includes invisible transitions that define synchronization between activities. The resulting model offers a hierarchical, semantically accurate representation of the process, correctly distinguishing between sequential actions, optional tool preparations, and iterative refinements. Compared to the Alpha Miner, this model provides a more realistic and complete depiction of the workflow dynamics observed during production.

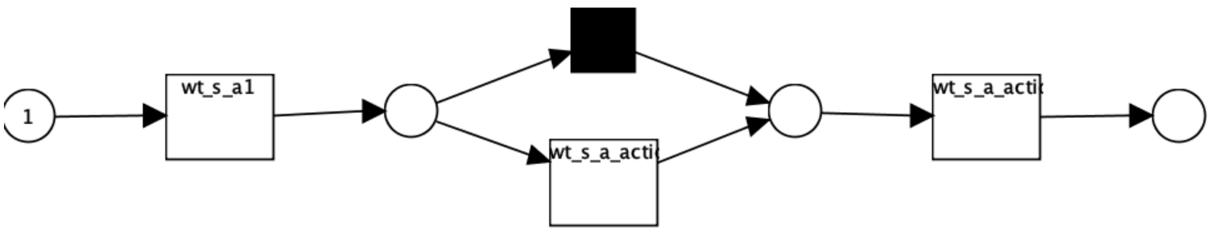


Figure 8: Inductive Miner- The model displays structured branching and synchronization through transitions.

4.7 Real-World Dataset: Glass Video Segmentation

The third dataset provided by the supervisor corresponds to real observations of a traditional glass-blowing workflow, recorded in the "Glass video segmentation" sheet. Each "BEGINNING OF CRAFT" and "END OF CRAFT" marker represents one complete case of the process (seven in total). Each case contains 12 – 32 recorded activities, such as *heating, gathering, shaping, transporting, attaching the workpiece,* and *cooling*. After preprocessing, the file was converted into an XES log (`glass_video_segmentation.xes`) and imported into ProM for process discovery using both the Alpha Miner and Inductive Miner algorithms.

Alpha Miner Result

The Alpha Miner produced a highly entangled and visually complex graph, with many overlapping arcs and repeated transitions. This result reflects the algorithm's sensitivity to noise, spelling inconsistencies (e.g., *transporting* vs. *trasporting*), and variable event orders. Because the dataset includes concurrent and iterative activities, Alpha Miner attempts to connect all observed sequences directly, leading to a tangled network rather than a clean, structured model. Although it technically represents all relations, the resulting model lacks interpretability and fails to capture loops or optional behavior.

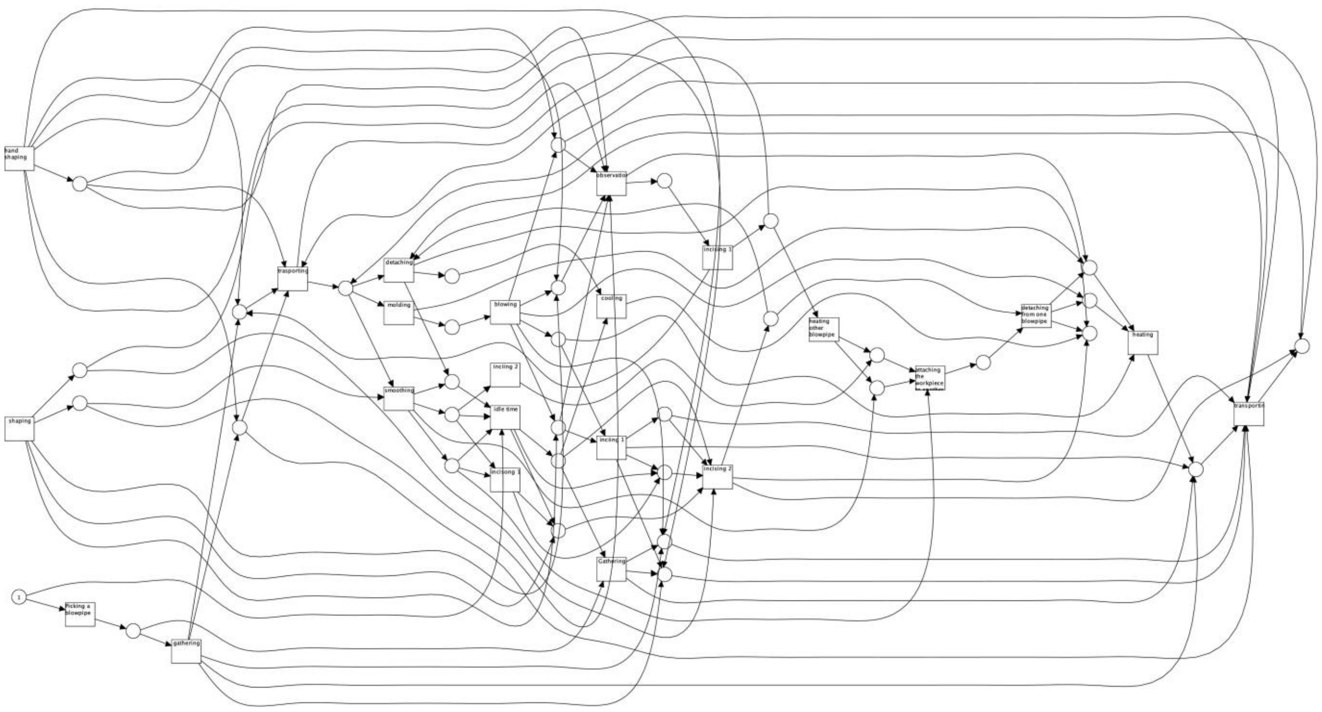


Figure 9: Alpha Miner model – Glass video segmentation.

Inductive Miner Result

In contrast, the Inductive Miner generated a sound, block-structured model that accurately represents the logic of the glass segmentation workflow. It successfully captured repetitive loops (e.g., *heating* ↔ *hand shaping*), optional sub-processes (e.g., *transporting* → *molding* vs. *transporting* → *shaping*), and maintained behavioral correctness. The discovered Petri net is both interpretable and faithful to the observed process, clearly showing concurrency and variation that the Alpha Miner failed to express.

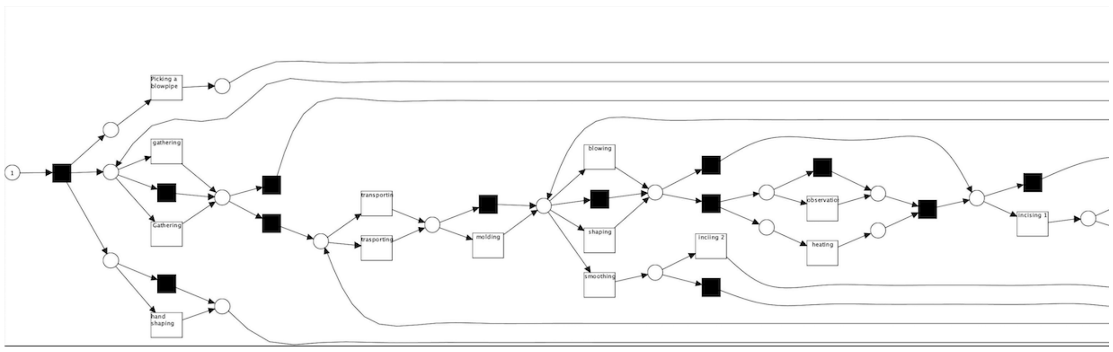


Figure 10a: Inductive Miner model – Glass video segmentation –Left section.

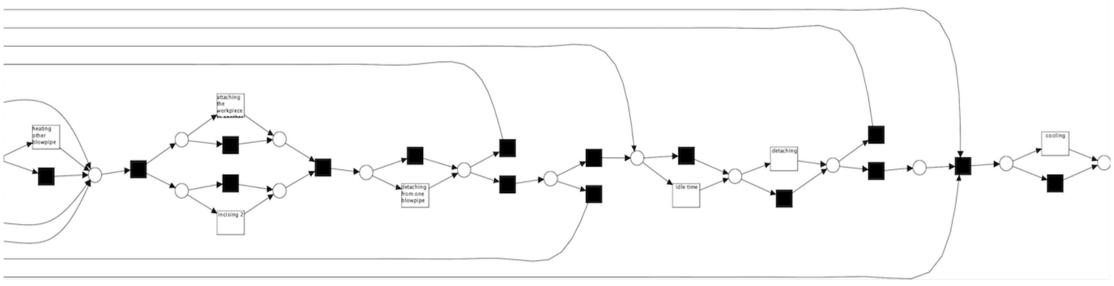


Figure 10b: Inductive Miner model – Glass video segmentation –Right section.

Table below summarizes the key structural and behavioral differences observed between the Alpha Miner and Inductive Miner models for the glass segmentation dataset.

Criterion	Alpha Miner	Inductive Miner	Interpretation
Model Structure	Flat and unstructured; highly tangled Petri Net	Hierarchical and block-structured	Inductive produces a clear, modular process model
Handling of Loops	Fails to detect iterative behavior	Explicitly models repetition as loop blocks	Inductive correctly captures heating ↔ shaping cycles
Parallelism / Concurrency	Linearized or merged into a single path	Distinguishes parallel and concurrent activities	Inductive preserves real-world concurrency
Interpretability	Low – visually complex and difficult to read	High – semantically coherent and traceable	Inductive Miner preferred for analysis
Best Use Case	Simple, structured processes	Real-world, human-centered workflows	Confirms Inductive Miner's suitability for ethnographic data

References

[1] W.M.P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[2] W.M.P. van der Aalst, T. Weijters, and L. Maruster, “Workflow Mining: Discovering Process Models from Event Logs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[3] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst, “Discovering Block-Structured Process Models from Event Logs—A Constructive Approach,” in *Application and Theory of Petri Nets and Concurrency (Petri Nets 2013)*, LNCS vol. 7927, Springer, 2013, pp. 311–329.

[4] C. Geertz, *The Interpretation of Cultures*. Basic Books, 1973.

[5] A.J.M.M. Weijters, A.K. Alves de Medeiros, and W.M.P. van der Aalst, “Process Mining with the Heuristics Miner Algorithm,” *Technische Universiteit Eindhoven*, 2006.

[6] ISO/IEC 15909-2:2011, *Systems and Software Engineering — High-level Petri Nets — Part 2: Transferring, Exchanging and Archiving Petri Nets (PNML format)*, International Organization for Standardization, 2011.

[7] C.W. Günther and W.M.P. van der Aalst, “Fuzzy Mining—Adaptive Process Simplification Based on Multi-perspective Metrics,” in *Business Process Management (BPM 2007)*, LNCS vol. 4714, Springer, 2007, pp. 328–343.

[8] M. Seidel, J. Mendling, and J. Reijers, “Ethnographic Studies in Process Management: Trends and Challenges,” *Information Systems Journal*, vol. 28, no. 1, pp. 9–42, 2018.

[9] A. Ardito, A. Lanz, M. Leotta, F. Ricca, and G. Scanniello, “Using Process Mining to Understand Software Development Processes: A Systematic Literature Review,” *Information and Software Technology*, vol. 151, 2022.

Annex A – Implementation Guide and Screenshots

A.1 Workflow Overview

The implementation of the process mining pipeline followed four main stages:

1. **Data Preparation:**
   a. Raw text logs from ethnographic observations were formatted as .txt files.
   b. Each line represented an activity or event with its order of occurrence.
   c. These were converted into .xes files using a Python script.
2. **Process Discovery in ProM:**
   a. Imported the .xes files into ProM 6.
   b. Executed Alpha Miner and Inductive Miner on the same log.
   c. Verified model soundness and exported the results as .pnml (Petri Net Markup Language) files.
3. **Model Export and PNML-to-Text Conversion:**
   a. Each .pnml file was processed using a custom Python XML parser.
   b. The script extracted transitions, places, and arcs, and generated a readable .txt summary.
   c. Example command:

```
python pnml_to_text.py alpha.pnml alpha_summary.txt
python pnml_to_text.py inductive.pnml
inductive_summary.txt
```

4. **Comparison and Visualization:**
   a. The textual outputs were compared side-by-side to analyze completeness and structure.
   b. Screenshots of each Petri Net were taken directly from ProM for inclusion in the report.

A.2 Screenshots

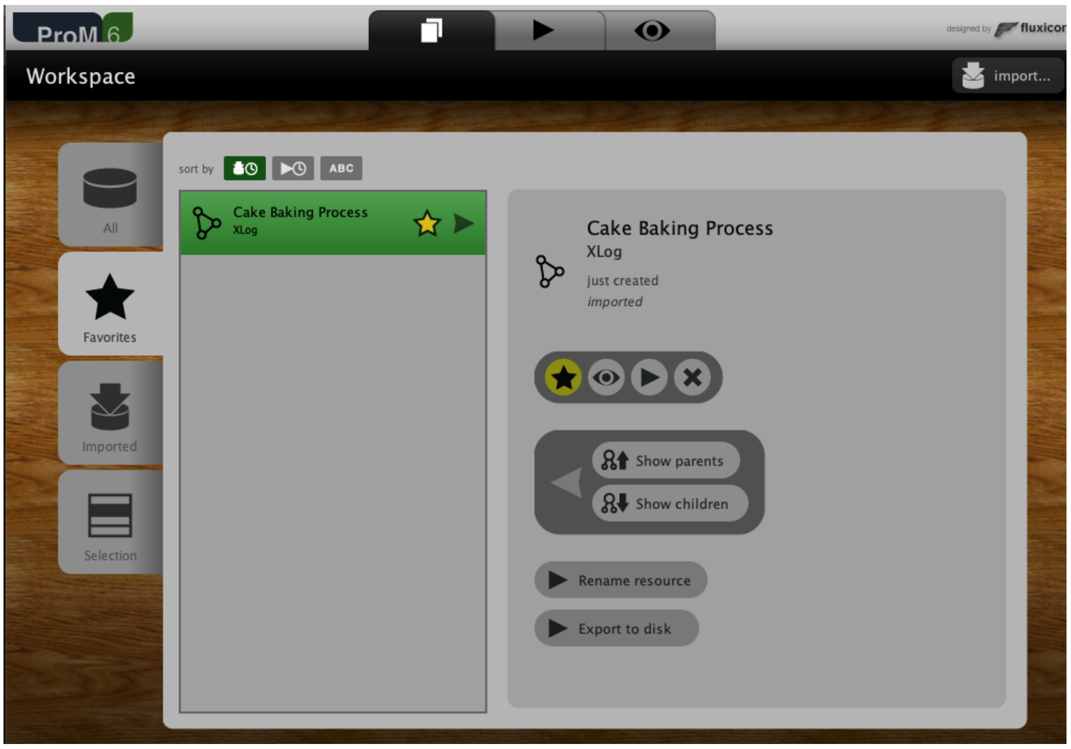


Figure A1: ProM configuration for cake_baking.xes.

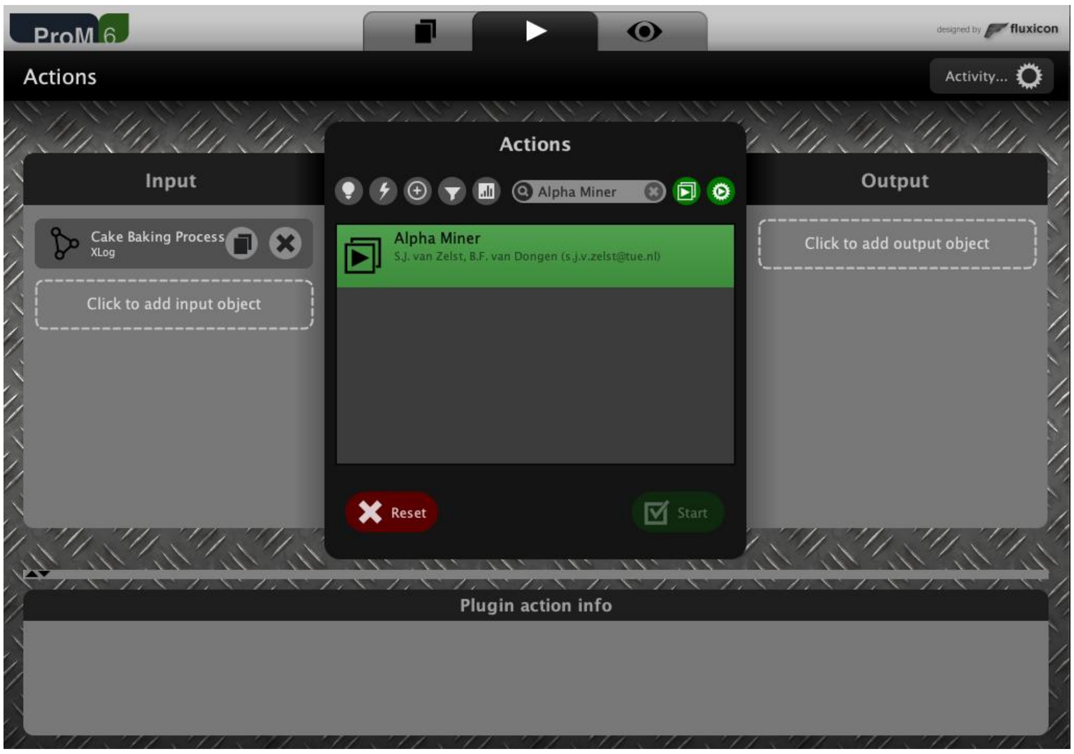
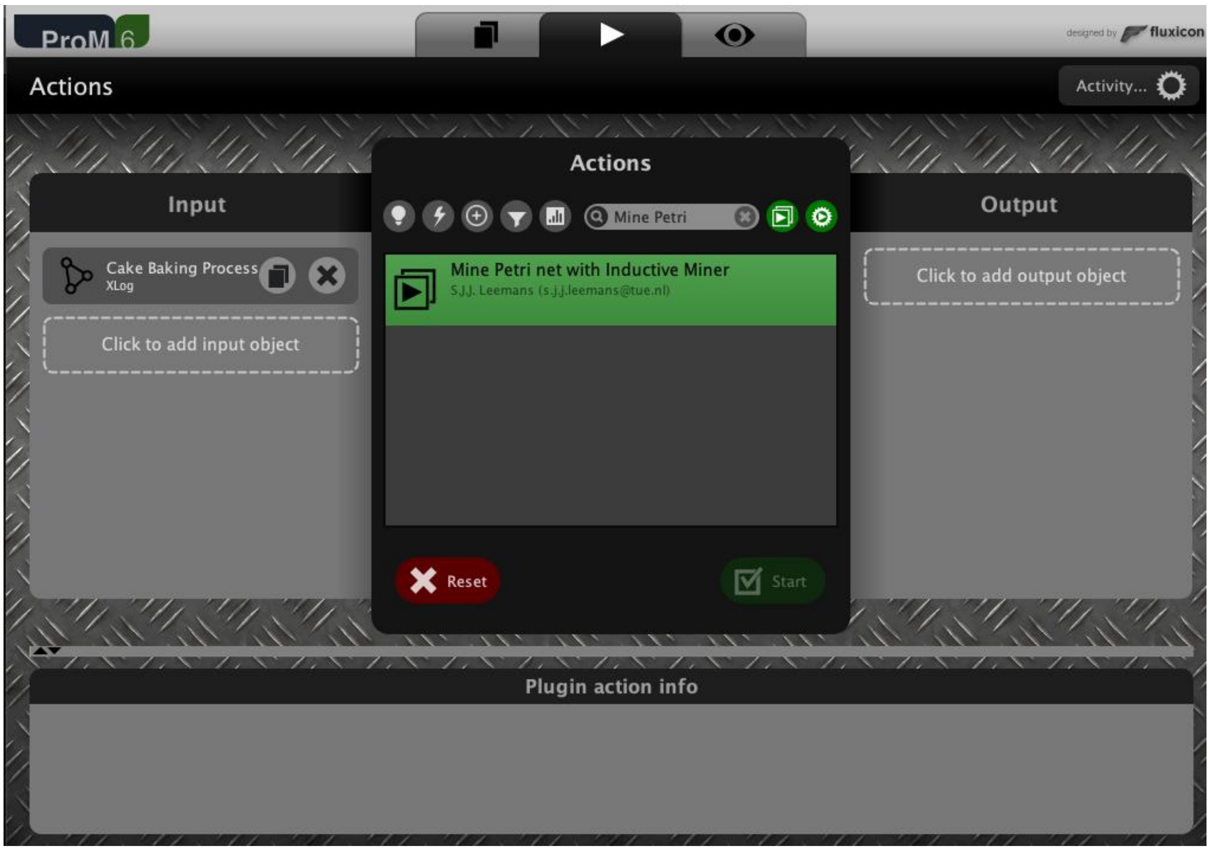
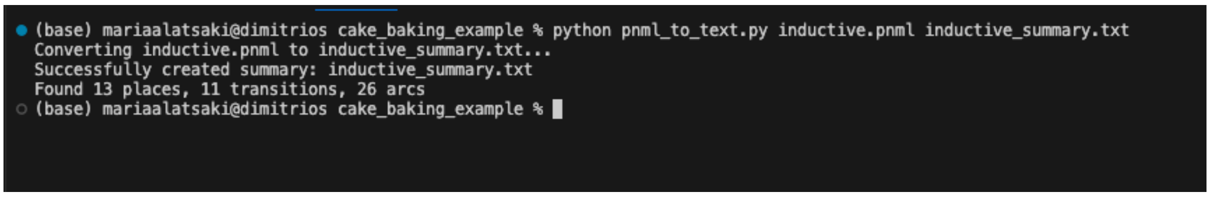
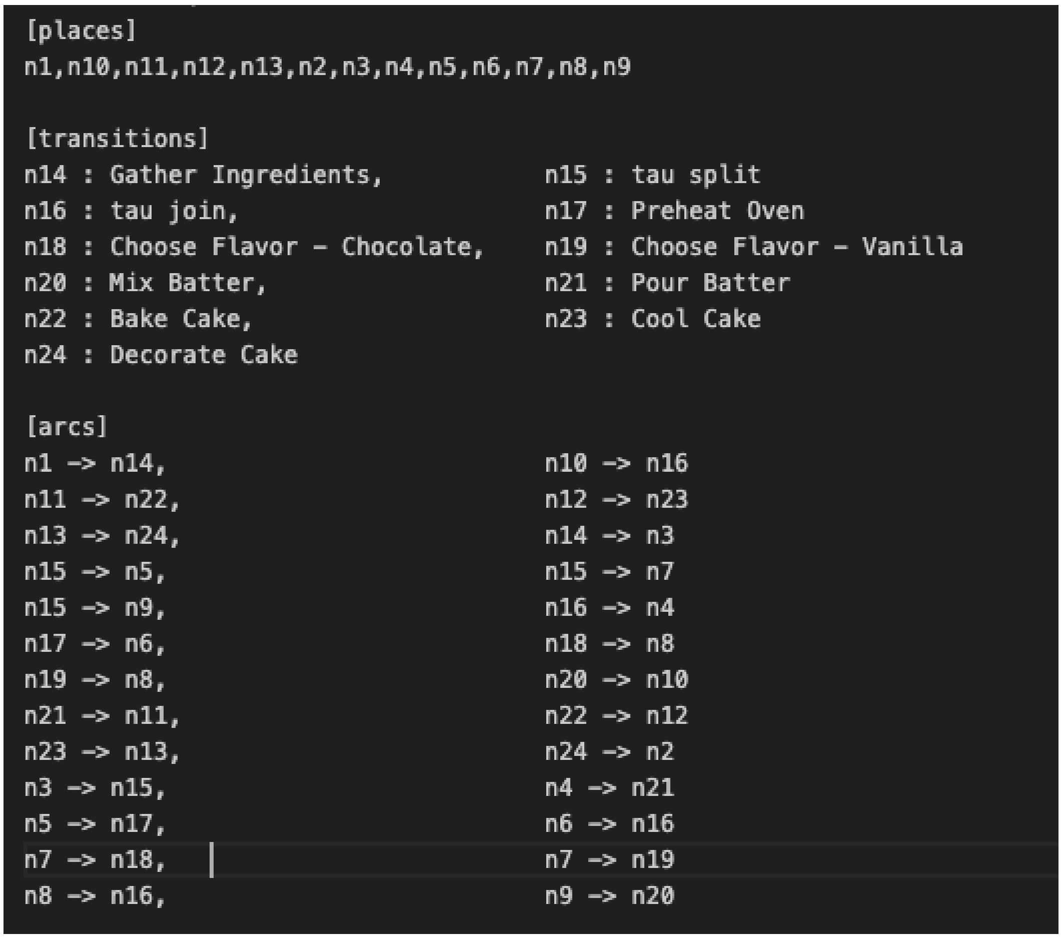


Figure A2: Open log panel (cake_baking.xes selected) + Alpha Miner plugin panel (default settings are fine).


ProM 6
designed by fluxicon
Actions
Activity...
Input
Cake Baking Process
XLog
Click to add input object
Actions
Mine Petri
Mine Petri net with Inductive Miner
S.J.J. Leemans (s.j.j.leemans@tue.nl)
Reset
Start
Output
Click to add output object
Plugin action info
Figure A2: Open log panel (cake_baking.xes selected) + Inductive Miner plugin panel (default settings are fine).

After generating Petri Net models from the cake_baking.xes event log using both the Alpha and Inductive Miner algorithms, the resulting PNML files were converted into structured .txt files. This conversion process extracts the key elements of the process graph—namely the places, transitions, and arcs—and organizes them into a clear, human-readable text format for easier analysis and comparison of the two discovered models.


```
● (base) mariaalatsaki@dimitrios cake_baking_example % python pnml_to_text.py inductive.pnml inductive_summary.txt
  Converting inductive.pnml to inductive_summary.txt...
  Successfully created summary: inductive_summary.txt
  Found 13 places, 11 transitions, 26 arcs
○ (base) mariaalatsaki@dimitrios cake_baking_example % █
```
Figure A3: Running `pnml_to_text.py` on `inductive.pnml`.



```
[places]
n1,n10,n11,n12,n13,n2,n3,n4,n5,n6,n7,n8,n9

[transitions]
n14 : Gather Ingredients,           n15 : tau split
n16 : tau join,                     n17 : Preheat Oven
n18 : Choose Flavor - Chocolate,    n19 : Choose Flavor - Vanilla
n20 : Mix Batter,                   n21 : Pour Batter
n22 : Bake Cake,                    n23 : Cool Cake
n24 : Decorate Cake

[arcs]
n1 -> n14,                          n10 -> n16
n11 -> n22,                         n12 -> n23
n13 -> n24,                         n14 -> n3
n15 -> n5,                          n15 -> n7
n15 -> n9,                          n16 -> n4
n17 -> n6,                          n18 -> n8
n19 -> n8,                          n20 -> n10
n21 -> n11,                         n22 -> n12
n23 -> n13,                         n24 -> n2
n3 -> n15,                          n4 -> n21
n5 -> n17,                          n6 -> n16
n7 -> n18,                          n7 -> n19
n8 -> n16,                          n9 -> n20
```

Figure A4: Extracted activities and Activity→Activity relations from PNML.

A.3 Data and Source Files

All process model files and datasets generated during this thesis are available in a public GitHub repository for reproducibility and further study.

GitHub Repository: https://github.com/Alatsakimaria/Thesis

The repository contains:

- .xes event logs used for process discovery (e.g., loop.xes, choice.xes, parallel.xes, glass_video_segmentation.xes),
- .pnml Petri Net models generated by the Alpha Miner and Inductive Miner algorithms,
- corresponding .txt summaries exported via the PNML-to-text parser,

These files allow anyone to reconstruct the discovered process graphs and reproduce the analyses presented in this report.

Annex B – Real-World Example: Cake-Baking Process

To illustrate the principles of process mining on a relatable example, a synthetic real-world process was modeled — the preparation of a chocolate or vanilla cake.

B.1 Description of the Process

The cake-baking process includes a variety of sequential, parallel, and conditional actions that correspond to typical workflow patterns:

1. **Prepare Ingredients (Start Event)**
   Gather flour, sugar, eggs, butter, milk, and flavor ingredients.
2. **Preheat Oven (Parallel)**
   The oven can be preheated *while* mixing ingredients — a parallel process.
3. **Choose Flavor (Branching)**
   a. If *Chocolate* → add cocoa powder.
   b. If *Vanilla* → add vanilla extract.
   → This represents a choice (XOR split).
4. **Mix Batter (Loop)**
   a. Continue mixing until no lumps remain, representing a loop (iteration) condition.
5. **Pour Batter and Bake**
   Sequential step after mixing is complete.
6. **Cool and Decorate (End)**
   Cooling must finish before optional decoration, representing sequential dependency.

B.2 Example in Process Mining Format

Activity	Type	Description
Gather Ingredients	Start	Beginning of process
Preheat Oven	Parallel	Can run simultaneously with mixing
Choose Flavor	Branch	Chocolate or Vanilla path
Mix Batter	Loop	Repeat until smooth
Pour Batter	Sequential	Next after mixing
Bake Cake	Sequential	Must follow pouring
Cool Cake	Sequential	Before decoration
Decorate Cake	Optional	End activity

B.3 Expected Petri Net Representation

- **Parallel paths:** Preheat Oven and Mix Batter occur concurrently.
- **Choice:** XOR gateway between Chocolate and Vanilla.
- **Loop:** Feedback arc from Mix Batter to itself.
- **Sequential flow:** Baking and cooling form a linear continuation.

When the Alpha Miner is applied, produced a linearized version missing the loop and parallelism.
When the Inductive Miner is applied, correctly represented all three: parallelism, choice, and repetition — consistent with the findings of this thesis.

B.4 Visualization

Alpha Miner Result – Cake-Baking Example

The Alpha Miner, when applied to the cake_baking.xes event log, produced a simplified, linear model that captures the dominant sequential flow of the process (Figure A5). The model begins with Gather Ingredients and proceeds through Preheat Oven, Choose Flavor, Pour Batter, Bake Cake, Cool Cake, and Decorate Cake. While this representation correctly identifies the main order of operations, it omits the parallelism and iteration present in the actual process. For example, the Preheat Oven and Mix Batter activities, which can occur concurrently, appear strictly sequential in the Alpha Miner model. Likewise, the repetitive Mix Batter steps are not represented as a loop but instead collapsed into a single transition. This behavior is consistent with the algorithm's known limitation of relying solely on direct-succession relations, resulting in linearized models that lose information about concurrency and repetition.

Overall, the Alpha Miner provides a clear but oversimplified view of the cake-baking workflow, making it suitable for structured or deterministic processes but less effective for capturing human or flexible tasks such as cooking.

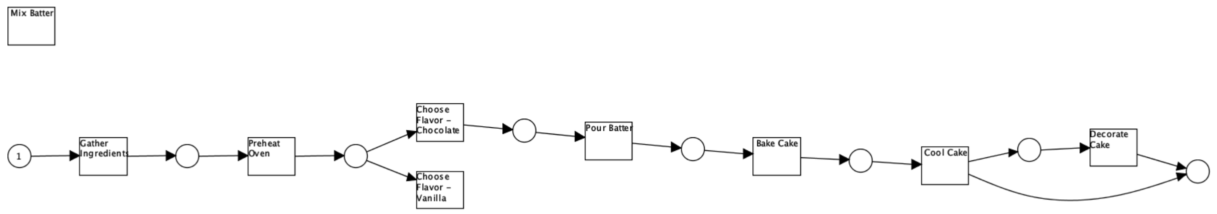


Figure A5: Alpha Miner process model for the cake-baking workflow.

Inductive Miner Result – Cake-Baking Example

The Inductive Miner, when applied to the cake_baking.xes event log, successfully reconstructed the main structure and logic of the process. Unlike the Alpha Miner, which linearized all activities, the Inductive Miner correctly identified and represented the parallel, branching, and iterative elements inherent to the cake-baking workflow. Specifically, the model captures the parallel execution between Preheat Oven and Mix Batter, allowing both tasks to occur simultaneously before proceeding to the next stage. It also distinguishes the exclusive choice (XOR) between Choose Flavor – Chocolate and Choose Flavor – Vanilla, accurately reflecting the recipe's decision point. The resulting model is block-structured and behaviorally sound, ensuring that every activity can be reached and completed without deadlocks or inconsistencies.

This output demonstrates the Inductive Miner's superior ability to model realistic, flexible human workflows compared to the Alpha Miner. The discovered Petri net not only reflects the intended logic of the cake-baking process but also generalizes its execution semantics by incorporating concurrency and decision-making.

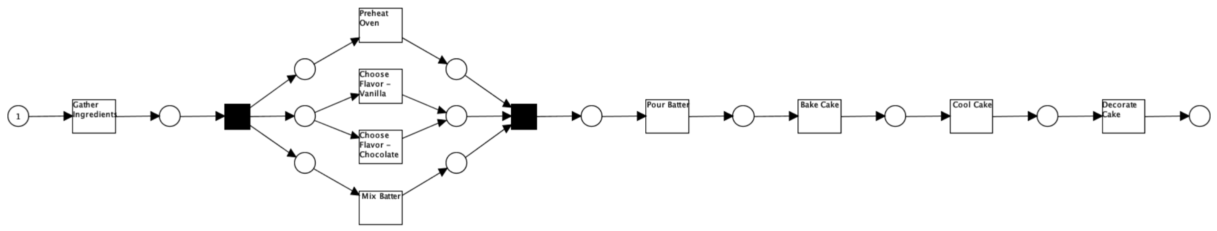


Figure A6: Inductive Miner process model for the cake-baking workflow.

Acknowledgements

I would like to express my deepest gratitude to **Mr. Xenophon Zabulis**, the supervisor of this thesis, for his invaluable guidance, constructive feedback, and continuous support throughout this work. His insights and encouragement were essential to the successful completion of this thesis.

Maria Foteini Alatsaki