# Decision Boundaries in User Journey Event Logs

Paul Kobialka [1*], Einar Broch Johnsen [1], Felix Mannhardt [2], S. Lizeth Tapia Tarifa [1]

[1]Department of Informatics, University of Oslo, Oslo, Norway.
[2]Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

*Corresponding author(s). E-mail(s): paulkob@ifi.uio.no;
Contributing authors: einarj@ifi.uio.no; f.mannhardt@tue.nl; sltarifa@ifi.uio.no;

**Abstract**

Service providers may systematically record and analyze how users interact with a service. User journeys model this interaction from the user's perspective. We can think of user journeys in terms of *multi-party event logs* in which events are triggered by two independent parties, the user and the service provider, both controlling their share of actions. Multi-party event logs enable the automated construction of weighted two-player games, which can be used to analyze the user journeys. However, the games constructed for complex user journeys may get large, which makes visual understanding challenging. To reduce the size of these games, we develop an analysis technique for a game's *decision boundary*, at which the outcome of the game is determined. Decision boundaries identify subgames that are equivalent to the full game with respect to the final outcome of user journeys. The paper formalizes decision boundaries for user journey games with history refinements, and studies properties of decision boundaries that simplify their analysis. The decision boundary analysis from multi-party event logs has been implemented in a modular tool, which can be connected to existing process mining pipelines, and evaluated on benchmark event logs revealing changes in the user journey and the interaction between users and service provider.

**Keywords:** User journeys, event logs, weighted games, decision boundaries.

# 1 Introduction

In a competitive market, a good user experience may be crucial for the survival of service providers [30]. Successful companies enrich their customer offerings with additional services, a practice often referred to as the *servitisation of business* [70]. Until recently, service providers analyzed their services from the *managerial* perspective, neglecting the users' point of view of the service. However, positive user impressions and experience within a service have been shown to yield financial rewards at low-risk for the service providers [30]. User journeys are models that similarly shift the focus of the *analysis* of services from the perspective of service provider to the perspective of the *user* (e.g., [29, 35]), in order to deliver services that value a positive user experience.

*User journeys* model the process of interaction between a *user* (or *customer*) and the services of a company (the *service provider*), from the user's perspective. Since users generally engage with a service to achieve some goal, user journeys are inherently goal-oriented processes. If the users reach the goal, the journey is *successful*, otherwise it is *unsuccessful*. User journeys consist of *touchpoints*; these are the single steps in which the user engages during a user journey, either action events performed by the user or communication events between service provider and user. We assume that users engage in at most one touchpoint of a service at a time; i.e., user journeys do not contain true parallelism.

User journey modeling is similar to the modeling of a business process by means of the activities that constitute a business process and their ordering relations; however, a user journey is composed of user-provider interactions instead of activities performed by a service provider. In contrast to typical process models, current modeling and analysis techniques for user journeys focus on individual journeys, with the aim to improve services from the customers' point of view [35,59]. These techniques are based in a manual generation of models, using interviews and questionnaires to collect user feedback, and have proven highly successful in discovering points of failure and in providing recommendations for service improvement. However, since these manual processes are labor-intensive and slow, they can only be applied to small services and a limited number of users. These limitations hinder the operational use of these techniques for complex digital services [36]: To this aim, analogously to how business processes are analyzed through event logs [2] automated analysis techniques would be required, leveraging the systematically collected event logs from these services. As any data-driven method, a core assumption of such automated analysis is that the collected events are connected to the touchpoints of the journey and relate, at least partially, to the user experience.

The automated analysis of user journeys as processes is of imminent interest to the process mining community, e.g. [13, 15, 37, 65, 66]. So far, user journeys have been directly mapped to regular event logs, e.g., touchpoints are represented by events and are part of a trace representing the journey. However, in contrast to a generic business process, which may include numerous actors and systems, a user journey is a sequence of very specific interactions between two parties: the user and the service provider(s). Some of these interactions are controlled by the user trying to reach their own goal, other interactions are controlled by the service provider(s) trying to guide the user to a successful completion. This invites to a *multi-party* view of the source event log, where

some events are controlled by the user and others by the service provider(s) [39]. Note that this concept of *controlling* an event is different to that of one or multiple resources performing an activity in business processes. Taking such *multi-party* event logs of user journeys as the departure point, this paper focuses on the automatic generation of *user journey games* [41, 42], which model and analyze user journeys as weighted two-player games in which the user and the service provider interleave in choosing the next action. We extend existing techniques for process discovery and treat input events from logs as in the analysis of business processes [2]: each journey is an instance of a process (case) recorded in a sequence of events (trace) and each event represents the occurrence of an activity. In addition, the multi-party view requires that each event carries information on the party controlling the interaction that caused it. Such modeling of user journeys as games provides insights into the users' point of view and into the user's interaction with the service provider, extending and complementing the structural information gained from process discovery on user journey event logs [13].

User journey games constructed for large and complex user journeys can be of considerable size, in practice out-of-reach for both manual generation and analysis. In previous work [41, 42], the analysis of user journey games was demonstrated on an industrial scenario with a small event log (33 sequences) that could — in principle — be manually generated and analyzed. In contrast, in this paper, we consider the construction and analysis of user journey games for challenging event logs, recorded from high-load services, which contain thousands of sequences. Motivated by the size and complexity of the games obtained from event logs at this scale and complexity, we further introduce a novel *model reduction technique* for user journeys — exploiting the generated games — to automatically detect *decision boundaries* for user journeys, and consider *refinement techniques* for the construction of multi-party games based on *k-sequence* and *k-multiset* transition systems. Decision boundaries can be useful for the analysis of user journeys because they identify the parts of the journey (a subset of states) from where the outcome of the journey is determined; i.e., after passing the decision boundary, neither user nor service provider can influence the final outcome of the game if the other player plays strategically. Additionally, we introduce a reduction technique based on decision boundaries, reducing the size of the generated games without discarding information. We further study how decision boundaries relate to techniques that control the cycles and precision of the discovered games.

To demonstrate our proposed method on complex user-service interactions, we apply the method to the event logs of BPIC 2012 [27] and BPIC 2017 [28]. Neither of these event logs include information on which activities are controlled by the user (e.g., completing loan applications), and which are controlled by the service provider (e.g. accepting loan applications). Since no existing analysis technique for event logs could leverage this information, we extend the existing event logs into *multi-party event logs* based on domain knowledge of the party causing the different activities in the considered user journeys. The application of our method to BPIC 2017 demonstrates the feasibility and usefulness of our approach for model reduction. Our results show that we can automatically detect the most critical parts of the game that guarantee successful and, respectively unsuccessful, user journeys. The application to BPIC 2012

additionally demonstrates challenges with ad-hoc recorded data, which were not observed in BPIC 2017, and allowed us to analyze the development of the user journeys over several years.

This paper extends a paper published at the workshop on Event Data and Behavioral Analytics (EdbA 2022) [39]. This paper refines the decision boundary definition and details its theoretical foundation: this paper significantly expands on the details of the model construction and analysis of user journey games for large event logs by formally defining each operation on transition systems, introduces theoretical results on decision boundaries and time horizons, and includes a prototype tool chain for embedding decision boundary analysis into process mining workflows. We have further extended the evaluation of the proposed method by expanding on the evaluation for the BPIC 2017 event log [28], and an additional evaluation dataset, the BPIC 2012 event log [27]. In short, this paper extends the previous workshop paper in every aspect:

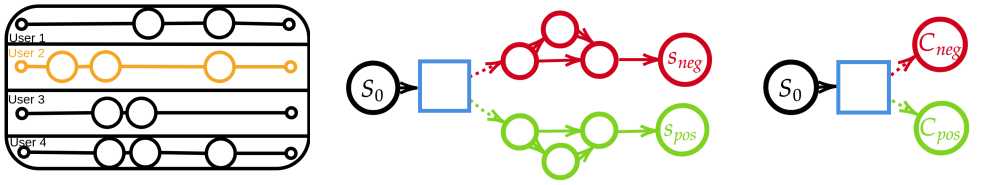
- a refined definition of decision boundaries, supported by novel theoretical results provided in Theorems 1 and 2, and Theorem 3 (Section 5);
- the concept of time-horizons for decision boundaries is introduced and its effects are analyzed both theoretically and experimentally (Section 6);
- a novel open-source prototype tool chain embedding decision boundaries into process mining workflows based on established standards, such as *XES* [33] (Section 7);
- an extended, in-depth experimental evaluation, including in particular an analysis of time-horizon refinements, that additionally confirms the feasibility of our approach on further event logs (Section 8.2); and
- we show how results from our analysis can be supported by manual analysis of the event logs and can improve prediction explanations (Sections 8.3 and 8.4).

***Paper outline:***

Section 2 presents a motivating example. Section 3 discusses related work on the modeling and construction of user journeys. Section 4 introduces technical background material for our work in *user journey games*. The analysis of decision boundaries is presented in Section 5. The steps for mining the decision boundary with process mining methods are described in Section 6. To automate the construction of user journey games from event logs, we have developed a proof of concept tool chain that can be integrated at different stages in standard process mining workflows; described in Section 7. We report in Section 8 on our experiences with using the tool chain to derive user journey games and analyze these on the process mining benchmarks BPIC 2012 and BPIC 2017. Section 9 discusses perspectives on the presented work and Section 10 concludes the paper.

## 2 Motivating Example

Consider an online loan application process where users apply for a loan by interacting with a bank. To understand how users experience this loan application process, the bank wants a user-centric analysis of the user journey that they offer to clients applying for a loan. Today, this analysis is a highly manual process.




(a) Event Log.

(b) User journey game with outcome-determined subgraphs.

(c) Decision boundary with merged clusters.

**Figure 1**: Construction of the decision boundary reduction.

First, to create a user journey map, the bank hires a team of analysts to conduct interviews with a representative group of clients, and analyze and visualize the user journey. Although the resulting visualization may reveal unknown pain points in the user journeys that hinder clients from submitting their loan applications, the analysis process is highly manual, and requires significant effort in conducting interviews with clients and analyzing the results. Furthermore, the selection of clients for interviews may contain bias, the approach is time-consuming and will not scale to a large number of clients.

In contrast to this manual process based on interviews, we consider an approach to automatically derive and analyze user journeys games from multi-party event logs (Figure 1a) by following the idea of process mining, which removes the significant manual effort in conducting interviews with a selected group of users. By automatically constructing games from real event logs, the resulting games reflect the choices of *all users*, which removes the possible selection bias in the manual process described above. Of course, date-driven methods for process discovery introduce other kinds of bias [1]; in particular, data-driven methods are sensitive to data quality issues, or simply the unavailability of data for certain interactions. Still, a large-scale application would be infeasible with interview-based methods. By leveraging information in the multi-party event log whether the user or the service provider controls the interaction, user journey games highlight the independence of the users in the process and hold the service provider responsible for guiding the user; i.e., it is the service provider's challenge to find a strategy in the game such that the user reaches a final state. The event logs reflect the actions of *a large number* of users for a service such as the loan application process, together with the corresponding actions taken by the bank while interacting with these users. The resulting *user journey games* (Figure 1b) aggregate all recorded user journeys and are analyzed to find the best known strategies for the bank that nudge the users toward a successful outcome of the loan application process.

The derived user journey games may become quite large and contain many interactions that do not influence the outcome of the game. Strategies for games of such size can be difficult to understand, and it is not guaranteed that there even exists a strategy for the whole game. The *decision boundary reduction* allows the analysis to focus on the interactions that are crucial for the success of the user journeys and may

reduce the size of the game significantly (Figure 1c). The resulting reduced game will often be easier to analyze, to visualize and to understand.

## 3 Related Work

We discuss related work with respect to the modeling of user journeys and the use of data-driven techniques to build user journey models. We are not aware of prior work on the automatic verification of games [10, 18, 20, 24] to analyze business processes.

Models of user journeys are typically given a graphical presentation. One of the earliest works on the modeling of user journeys is on service blueprinting by Bitner *et al.* [16], which lays out an anticipated model of a service. Several diagrammatic modeling notations for user journey diagrams have been developed [12, 23, 34, 45, 57, 59]. An overview of this line of work is given in two recent surveys by Følstad *et al.* [29] and Tueanrat *et al.* [69]. Available tools for these notations mainly target manual workflows, with experts generating and analyzing the service model by hand. While service blueprinting and diagrammatic models have proven useful for manually analyzing user journeys, e.g. [16, 19, 23, 35, 53, 59], our paper differs from this line of work in proposing an automated method to construct user journey models, applicable to event logs that are recorded for existing digital services.

Methods to analyze user journeys that are based on event logs, exploit events recording the user interactions with a service and its underlying information systems. In particular, process mining techniques have previously been used to analyze user journeys [13, 15, 37, 65, 66]. Due to the sequential nature of user-service interactions, these techniques assume grouped sequences of events as input. For example, Bernard *et al.* explore and discover journeys from events [13] by means of hierarchical clustering to control the granularity of the generated user journey maps [14], and by using genetic algorithms [15], where the complexity of the underlying process is abstracted by generating representative journeys. Our work complements this line of work by introducing decision boundaries as a novel model reduction technique that is independent of the variations present in the recorded traces. To this aim, our work exploits user journey games to capture the complexity of the underlying process.

Recommender systems suggesting how to best maximize key performance indicators for user journeys, have been proposed by Terragni and Hassani [65, 66]; their work analyzes web logs for user journeys to show, e.g., to show how often a particular page is visited. The prediction of the next touchpoint in ongoing journeys has been investigated by Hassani and Habets [37]. Our work is complementary to these approaches since a decision boundary identifies from where in the user journey the service provider can no longer influence the final outcome of the journey. However, a decision boundary does not provide recommendations for how to reach a successful outcome of the journey, or how to optimize the journey toward key performance indicators. Decision boundaries also complement predictive process mining since a guaranteed outcome in a state is a structural property of the user journey with a strategic service provider, and not a dynamic prediction customized to specific users.

The partitioning of event logs into desired and undesired cases or process outcomes has previously been explored in several ways in the context of user journeys.

For example, *deviance mining* classifies cases to investigate deviations from expected behavior [51]. A binary partitioning of event logs into positive and negative cases has been used in, e.g., logic-based process mining [11,44] and error detection [60]. Chesani *et al.* consider positive and negative cases to build a declarative model of successful executions [21]. Slaats *et al.* consider positive and negative cases to express accepted and rejected executions, leveraging the process discovery problem to a binary classification problem [63]. Outcome prediction aims to predict the outcome of a process case based on a partial trace [25,43,64]. Di Francescomarino *et al.* combine clustering with classifiers to predict if any predicate, e.g. a property or temporal logic constraint, will be fulfilled in the execution [25]. Additional approaches and benchmarks are discussed in the surveys by Teinemaa *et al.* [64] and Kratsch *et al.* [43]. However, none of these works consider the users and service providers as independent interacting parties, which enables the game-based analysis considered in our work.

The use of games to analyze user journeys has previously been explored for anomaly detection on mined process models by Saraeian and Shirazi [61] and for explanations in predictive process mining by Galanti *et al.* [32]. In contrast to our work, these works do not use games to account for multiple independent parties interacting in the process model. Thus, in our work, games focus on the interactions between independent parties and the influence of a single party and their actions on the process outcome. User journey games were initially introduced by Kobialka *et al.* [41] to analyze a small service where an optimal strategy to guarantee a successful service existed. The analysis was later extended [42] to a sliding-window analysis where short-term changes in the user journey could be detected via changes in the computed optimal strategies. An active object model, parametric in the user behavior, by Kobialka *et al.* [40] uses strategies from user journey games for prescriptive analysis through simulations. In contrast, this paper is the first to address model reduction for user journey games. For complex services, these models can be of considerable size; therefore, we believe such reductions are important for further analysis and understanding of the user journeys.

# 4 Preliminaries

This section first covers background notation and definitions in Sections 4.1–4.4, provides an introduction to user journey games in Section 4.5 and an example of their analysis in Section 4.6.

## 4.1 From Traces to Multi-Party Event Logs

*Traces* over a set $X$ are denoted by $X^*$. Traces are finite, ordered sequences $\tau = \langle x_0, \ldots, x_n \rangle$ such that $x_i \in X$ for $0 \leq i \leq n$. Let $\varepsilon$ denote the empty trace and $\tau \cdot x$ the append-operator on trace $\tau$, appending element $x$ to the end of $\tau$. For a trace $\tau = \langle x_0, \ldots, x_n \rangle$, we have $\tau \cdot x_{n+1} = \langle x_0, \ldots, x_n, x_{n+1} \rangle$. Let $|\tau|$ denote the length of $\tau$ (so $|\tau| = n + 1$), $\mathit{last}(\tau)$ the last element of $\tau$ (so $\mathit{last}(\tau) = x_n$), and $\mathit{tail}(\tau)$ the list without the first element (so $\mathit{tail}(\tau) = \langle x_1, \ldots, x_n \rangle$). Let $\tau \prec \tau'$ denote that $\tau$ is a prefix (or initial segment) of $\tau'$, so $\tau \prec \tau \cdot x_{n+1}$, and $\tau \preceq \tau'$ the reflexive closure of the prefix relation.

*Multisets* over a set $X$ are denoted by $L = (X, m)$, where the multiplicity function $m : X \to \mathbb{N}$ maps the elements of $X$ to their number of occurrences. The support $\text{supp}(L)$ of the multiset $L$ is its underlying set; i.e., $\text{supp}(L) = \{x \in X \mid m(x) > 0\}$. To ease notation, we write $x \in L$ for $x \in \text{supp}(L)$. We denote by $2^X$ the power set (the set of subsets) and by $\mathcal{B}(X)$ the set of multisets over $X$. Given a trace $\tau$, we let $\alpha(\tau)$ denote its multiset of elements (keeping track of the multiplicity of the elements in $\tau$).

*Event logs* [2] are traces of observations of the actions or events of interest (as given by an alphabet $A$). We formally define event logs as follows:

**Definition 1** (Event logs)**.** *An* event log $L$ *over an alphabet* $A$ *is a multiset of traces over* $A$*:* $L \in \mathcal{B}(A^*)$.

In our work, we assume that event logs are of sufficient quality to reason about the underlying user journeys and connect individual steps relevant to the user's experience in the journey to recorded events in the event log [17].

*Multi-party event logs* are specifically introduced for our analysis of user journeys, extending event logs (Definition 1) by accounting for the parties that trigger the actions reflected in the traces. Note that this differs from the usual notion of resources in an event log: A resource may be associated to an event without triggering or controlling it. Formally, multi-party event logs are defined as follows:

**Definition 2** (Multi-party event logs)**.** *A* multi-party event log *over an alphabet* $A$ *is a tuple* $\mathcal{L} = (L, P, I)$ *where* $L \in \mathcal{B}(A^*)$ *is an event log,* $P$ *is a set of parties, and* $I : A^* \to P$.

In multi-party event logs, each event in a trace has an associated party identified by $I$, which identifies the initiator of that event in the trace. We let $I(\tau)$ denote the initiator of the last event in $\tau$ (which may be influenced by previous actions in $\tau$).

## 4.2 Transition Systems

Let us recall the definition of transition systems (e.g., [52,55]), which have been widely used as a model to analyze processes [3] and are used as basis for our game approach:

**Definition 3** (Transition systems)**.** *A* transition system *is a tuple* $S = (\Gamma, A, E, s_0, T)$ *where* $\Gamma$ *is a set of states,* $A$ *a set of actions (or labels),* $E \subseteq \Gamma \times A \times \Gamma$ *a transition relation,* $s_0 \in \Gamma$ *the initial state and* $T \subseteq \Gamma$ *a set of final states.*

Given a transition system $S = (\Gamma, A, E, s_0, T)$, we say that a trace $\tau \in A^*$ is a *partial trace* of $S$ and (slightly overloading notation) write $\langle s_i, \tau, s_{j+1} \rangle \in E$ to denote the transitive closure of $E$; i.e, $\langle s_0, \varepsilon, s_0 \rangle \in E$ and $\langle s_i, \tau, s_{j+1} \rangle \in E$ if $\tau = \langle a_i, \ldots, a_j \rangle$ and $s_i, \ldots s_{j+1} \in \Gamma$ such that $\langle s_k, a_k, s_{k+1} \rangle \in E$ for $i \leq k \leq j$. A trace $\tau = \langle s_0, \tau, s_n \rangle \in E$ is a (full) trace of $S$ if $\tau$ is a partial trace of $S$ and $s_n \in T$. Full traces can be used to express that a transition system covers all the traces of an event log, defined as follows:

**Definition 4** (Transition systems for logs)**.** *Let* $L \in \mathcal{B}(A^*)$ *be an event log over an alphabet* $A$ *and* $S = (\Gamma, A, E, s_0, T)$ *a transition system.* $S$ *is a* transition system for the event log $L$ *if* $\text{supp}(L) \subseteq \{\tau \mid \langle s_0, \tau, s \rangle \in E, s \in T\}$.

Observe that a transition system for an event log over-approximates the log in the sense that it allows more traces than those in the log, i.e. the language of the transition system contains $\text{supp}(L)$. We can also use a log to reduce a transition system:

**Definition 5** (Log-sliced transition systems)**.** *Let $L \in \mathcal{B}(A^*)$ be an event log over an alphabet $A$ and $S = (\Gamma, A, E, s_0, T)$ a transition system. The transition system $S$* sliced by the event log *$L$ is a tuple $S_L = (\Gamma', A', E', s_0', T')$ where*

$$\begin{aligned}
\Gamma' &= \{s \in \Gamma \mid \langle s_0, \tau, s\rangle \in E, \tau \preceq \tau', \tau' \in L\} \\
A' &= \{a \in A \mid \langle s, a, s'\rangle \in E'\} \\
E' &= \{\langle s, a, s'\rangle \in E \mid \langle s_0, \tau, s\rangle \in E', \tau \cdot a \preceq \tau', \tau' \in L\} \\
s_0' &= s_0 \\
T' &= \{s \in T \mid \langle s_0, \tau, s\rangle \in E', \tau \in L\}
\end{aligned}$$

The new transition system, obtained after log-slicing an existing one, may have fewer states and transitions, because redundant states and transitions may have been removed. Obviously, the log-sliced transition system is still a transition system for the event log. It may be more precise than the existing one, but may still over-approximate the event log.

*History refinements* can be used to make transition systems more precise by including the most recent history in the states [3]. Given a transition system, we consider two different forms of history refinement: a $k$-sequence refinement is a transition system that distinguishes states of the original transition system depending on the $k$ last elements in the trace that leads to a state, maintaining the order of events in the traces, and a $k$-multiset refinement is a transition system that similarly takes the last $k$ events in the traces leading to a state into account but erases the order, thereby further generalizing the $k$-sequence refinement. The two kinds of history-refined transition systems are defined as follows:

**Definition 6** ($k$-sequence transition systems)**.** *Let $k$ be an integer and $S = (\Gamma, A, E, s_0, T)$ a transition system. A* $k$-sequence transition system *is a tuple $S^k = (\Gamma', A', E', s_0', T')$ where*

$$\begin{aligned}
\Gamma' = {} & \{s^\tau \mid \langle s_0, \tau, s\rangle \in E, \quad |\tau| < k\} \cup \{s^\tau \mid \langle s', \tau, s\rangle \in E, \quad |\tau| = k\} \\
A' = {} & A \\
E' = {} & \{\langle s_1^\tau, a, s_2^{\tau'}\rangle \mid \langle s_1, a, s_2\rangle \in E, \quad \tau' = \tau \cdot a, \quad s_1^\tau, s_2^{\tau'} \in \Gamma'\} \cup \\
& \{\langle s_1^\tau, a, s_2^{\tau'}\rangle \mid \langle s_1, a, s_2\rangle \in E, \quad \tau' = \mathit{tail}(\tau) \cdot a, \quad s_1^\tau, s_2^{\tau'} \in \Gamma'\} \\
s_0' = {} & s_0^\varepsilon \\
T' = {} & \{s^\tau \mid s \in T\}
\end{aligned}$$

**Definition 7** ($k$-multiset transition systems)**.** *Let $k$ be an integer and $S = (\Gamma, A, E, s_0, T)$ a transition system. A* $k$-multiset transition system *is a tuple $S^{\{k\}} = (\Gamma', A', E', s_0', T')$ where*

$$
\begin{aligned}
\Gamma' &= \{s^{\alpha(\tau)} \mid \langle s_0, \tau, s\rangle \in E, \quad |\tau| < k\} \cup \{s^{\alpha(\tau)} \mid \langle s', \tau, s\rangle \in E, \quad |\tau| = k\} \\
A' &= A \\
E' &= \{\langle s_1^{\alpha(\tau)}, a, s_2^{\alpha(\tau')}\rangle \mid \langle s_1, a, s_2\rangle \in E, \quad \tau' = \tau \cdot a, \quad s_1^{\alpha(\tau)}, s_2^{\alpha(\tau')} \in \Gamma'\} \cup \\
&\quad \{\langle s_1^{\alpha(\tau)}, a, s_2^{\alpha(\tau')}\rangle \mid \langle s_1, a, s_2\rangle \in E, \quad \tau' = \mathit{tail}(\tau) \cdot a, \quad s_1^{\alpha(\tau)}, s_2^{\alpha(\tau')} \in \Gamma'\} \\
s_0' &= s_0^{\alpha(\varepsilon)} \\
T' &= \{s^{\alpha(\tau)} \mid s \in T\}
\end{aligned}
$$

States $s^\tau$ and $s^{\alpha(\tau)}$ are history-refined variants of states $s$ where the state $s$ can be reached with histories $\tau$ and $\alpha(\tau)$, respectively. We will refer to the integer $k$ as the *horizon* of the history refinement (following the terminology of van der Aalst *et al.* [3]).

**Example** (History-refined transition systems)**.** *Let us illustrate the construction of history-refined transition systems with a horizon of* 2 *for a transition system* $S = (\Gamma, A, E, s_0, T)$ *with* $\{s_0, s_1, s_2, s_3\} \subseteq \Gamma$ *and* $\{a, b, c\} \subseteq A$*. Assume that* $\tau = \langle a, b, c\rangle$ *is a trace of* $S$ *that takes* $s_0$ *via* $s_1$ *and* $s_2$ *to* $s_3$*. The* 2*-sequence states capturing the history of* $\tau$ *are* $\{s_0^\varepsilon, s_1^{\langle a\rangle}, s_2^{\langle a,b\rangle}, s_3^{\langle b,c\rangle}\}$ *and the* 2*-multiset states are* $\{s_0^{\alpha(\varepsilon)}, s_1^{\{a:1\}}, s_2^{\{a:1,b:1\}}, s_3^{\{b:1,c:1\}}\}$.

Observe that a history-refined transition system $S'$ from a transition system $S$ for log $L$ is also a transition system $S'$ for log $L$. The slice of $S'$ by $L$ is also a transition system for $L$, which removes possible non-determinism in the slice of $S$ by $L$.

*Weighted transition systems* associate costs with transitions by combining a transition system with a weight function [68]:

**Definition 8** (Weighted transition systems)**.** *A* weighted transition system *is a pair* $(S, w)$ *where* $S = (\Gamma, A, E, s_0, T)$ *is a transition system and* $w : E \to \mathbb{R}$ *a weight function.*

To compare traces through a weighted transition system, we will be interested in the accumulated sum of weights on the transitions of the (partial) trace. In the context of user journeys, we will refer to the accumulated weights as the *gas* of the user journey, formalized as follows [41, 42]:

**Definition 9** (Gas)**.** *Given a weighted transition system* $(S, w)$ *where* $S = (\Gamma, A, E, s_0, T)$ *and* $w : E \to \mathbb{R}$*, let* $\tau = \langle a_i, \ldots, a_j\rangle$ *be a partial trace of* $S$ *such that* $\langle s_i, \tau, s_{j+1}\rangle \in E$*. The* gas $\mathcal{G}$ *of trace* $\tau$ *is defined as follows:*

$$\mathcal{G}(\tau) = \sum_{k=i}^{j} w(\langle s_k, a_k, s_{k+1}\rangle).$$

Gas quantitatively reflects how moves in the user journey contribute to users reaching their goal. Since moves in the transition system are weighted, they can be accumulated into the gas of the journey. Informally, the gas function captures how much "steam" the user has left to continue the journey; therefore, traces of user journeys with a high gas value are favorable for the user, compare to traces with lower gas.

## 4.3 Weighted Games

To incorporate the two-party perspective of multi-player event logs into transition systems, we extend transition systems to games by making a distinction between controllable and uncontrollable actions [18]:

**Definition 10** (Weighted games)**.** *A game* $G = (\Gamma, A, E, s_0, T)$ *is a transition system such that* $A$ *is partitioned into* controllable *actions* $A_c$ *and* uncontrollable *actions* $A_u$*. A* weighted game *is a pair* $(G, w)$ *where* $G$ *is a game and* $w : E \to \mathbb{R}$ *is a weight function.*

An intuition here is that when seeing the game from the perspective of a player $c$, only the actions in $A_c$ can be controlled while actions in $A_u$ are decided by an adversarial environment. Let the symbol $\lambda$ represent that a player decides to "do nothing" in a particular state. When analyzing games, we look for a *strategy* for player $c$ that guarantees a given property, such as winning the game by reaching a certain state.

**Definition 11** (Strategies)**.** *Let* $G = (\Gamma, A, E, s_0, T)$ *be a game in which* $A$ *is partitioned into* $A_c$ *and* $A_u$*, and let* $p \in \{c, u\}$*. A* strategy *over* $G$ *for* $p$ *is a partial function* $\pi_p : \Gamma \nrightarrow 2^{A_p \cup \{\lambda\}}$*.*

The strategy for a player $p$ can be used to decide which of the actions controllable by $p$ to choose in a given state. Strategies may be non-deterministic and select several possible actions in a given state. Let $\pi_p^2 \oplus \pi_p^1$ denote the strategy such that $\pi_p^2 \oplus \pi_p^1(x) = \pi_p^1(x)$ if $x \in \mathrm{dom}(\pi_p^1)$ and $\pi_p^2 \oplus \pi_p^1(x) = \pi_p^2(x)$ otherwise. The strategies defined here only depend on the current state of the game; these strategies correspond to so-called memory-less strategies in game theory [6, 67]. More expressive strategies that depend on (part of) the trace leading to the current state, may be defined as state-based strategies over $k$-sequence transition systems for a sufficiently large $k$.

## 4.4 Temporal Logic

Temporal properties of transition systems can be expressed using temporal logics. In this paper, we consider *computation tree logic* (CTL) [22], which expresses temporal properties over the traces of the transition system seen as a tree, differentiating between properties that hold for *all* traces and properties that hold for *some* traces. The CTL formulas used in this paper are given by the syntax

$$\varphi \coloneqq \textbf{True} \mid \textbf{False} \mid p \mid \neg\, \phi \mid \phi \wedge \phi \mid \mathsf{AX}\, \phi \mid \mathsf{A{<}{>}}\, \phi,$$

where $p$ is a state formula. Given a transition system $S = (\Gamma, A, E, s_0, T)$ and a state $s \in \Gamma$, these operators can be understood as follows: **True** holds in $s$ while **False** does not. The validity of a state property $p$ can be checked directly on $s$, the *next* operator $\mathsf{AX}\, \phi$ expresses that $\phi$ holds for all successor-states (i.e., states in $\{s' \mid \langle s, a, s' \rangle \in E\}$), and the *eventually* operator $\mathsf{A{<}{>}}\, \phi$ expresses that $\phi$ will hold for some state in all full traces $\langle s, \tau, s' \rangle$ (where $s' \in T$). Negation and conjunction have the obvious interpretations. The validity of a formula $\phi$ in a transition system $S$ at state $s$ is formally captured by a satisfaction relation $S, s \models \phi$ (for the formal definition

of satisfaction for CTL, see, e.g., [22, 38]). Note that our focus here is on reachability properties; this fraction of CTL coincides with *linear temporal logic* (LTL) [38].

### 4.5 User Journey Games

We model user journeys as *user journey games*, which are weighted games with goal states [41, 42]. The intuition is that the successful outcome of a journey does not only depend on the behavior of the service provider — the journey can be seen as a game between service provider and user, where both parties are self-interested and control their share of actions. Thus, a user journey can be modeled as a transition system with final states $T$ and successful goal states $T_s \subseteq T$ such that every journey ending in $t \in T_s$ is successful. Formally, user journey games are defined as follows:

**Definition 12** (User journey games)**.** *A* user journey game *is a weighted game* $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$*, where*

- $\Gamma$ *are states,*
- $A_c$ *and* $A_u$ *are disjoint sets of actions,*
- $E \subseteq \Gamma \times A_c \cup A_u \times \Gamma$ *are the transitions,*
- $s_0 \in \Gamma$ *is an initial state,*
- $T \subseteq \Gamma$ *are the final states,*
- $T_s \subseteq T$ *are the successful final states, and*
- $w : E \to \mathbb{R}$ *is the weight function.*

In user journey games, the transitions model the individual touchpoints of the user journey, $A_c$ the actions initiated by the service provider, $A_u$ the actions initiated by the user and $T_s$ the successful goal states. The weight function, detailed below, associates a weight with every action. Note that states in user journey games are the same as in transition systems; i.e., the states are collections of events observed in the underlying multi-party event log. Connecting to history refined transition systems from Definitions 6 and 7 and user journeys, the states and transitions in the games correspond exactly to the states and transitions that are defined in these refinements. For the touchpoints of a specific user journey, these history refinements define how much of the history of each event in the trace of the user journey is recorded in the resulting game. This enables us to directly relate events in the event log to states in the game without the need of connecting events to states in higher-level process models such as Petri nets [49, 54], often used in process discovery [2].

The analysis of services with a large number of users requires a notion of user feedback [59]. In a user journey game, the weight function $w$ represents the impact that a transition has on the outcome of the game. A method to construct user journey games is described in [41, 42]. When building user journey games from event logs, Shannon entropy [62] and majority voting are combined to estimate a form of user feedback without human intervention. With this notion of user feedback, the more certain the outcome of a journey becomes after a transition, the higher the absolute weight in such transition, a high positive weight for successful journeys and a high negative weight for unsuccessful journeys. Transitions that are not observed in the log are assigned a neutral weight.

The formal properties of a user journey game can be analyzed using a model checker such as UPPAAL STRATEGO [24]; these can express that, e.g., the service provider can guide users to the positive outcome or bound the amount of negative feedback. UPPAAL STRATEGO extends the UPPAAL system [46] by games and stochastic model checking, allowing properties to be verified up to a given confidence level by means of simulations (avoiding the full state space exploration that is necessary for certainty). If a statement holds, an enforcing strategy is computed. For a user-centric analysis of user journeys, we assume that an adversarial environment exposes the worst-case behavior of the service provider by letting the service provider's actions be controllable and the user's actions uncontrollable.

For example, let us define a strategy `pos` for eventually reaching a successful final state in all possible strategic executions of the game:

```
strategy pos = control: A<> positive .
```

The keyword `control` indicates a game with an adversarial environment, where possible (non-deterministic) paths are strategically chosen to guarantee (temporal logic) properties. Here, the property `positive` is a state property expressing successful final states and `A<> positive` a temporal property expressing that such a state will eventually be reached. In case no strategy exists that can guarantee a given property, we let the analysis return an empty function that maps all states to $\emptyset$.

Note that the strategy `pos` might be non-deterministic; i.e., there may be multiple actions at given states in the game that all lead to a successful final state. The strategy `pos` can be further analyzed and refined, e.g., to minimize the expected number of steps or the amount of gas needed to reach a final state within an upper bound on time `T`. The following strategy `minS` is such a refinement of `pos`:

```
strategy minS = minE(steps) [t<= T] : <> positive under pos .
```

Strategies can be stochastically evaluated using a number of runs `N`; e.g., the minimal gas of the refined strategy within an upper bound time `T` can be analyzed as follows:

```
E[t<=T; N] (min: gas) under minS .
```

## 4.6 Analyzing User Journey Games with User Compliance

We illustrate the analysis of user journey games by means of stochastic simulation and model checking, which can help a service provider to evaluate strategies that guide users along their services, before implementing them. We evaluated different strategies for the user journey game for BPIC 2017 until July (this event log is detailed in Section 8.1), using UPPAAL STRATEGO to learn and compare the strategies. In the reported experiments, we consider three strategies that refine the strategy `pos` of Section 4.5. Recall that `pos` guarantees that we reach $s_{pos}$. In strategy `max`, the service provider can guide the user through the service by *maximizing the final gas*, while in strategy `step` the service provider *minimizes the expected number of steps*. Finally, we consider a strategy `both` that combines `pos` and `max`. To enable a comparative

| Query | Strategy strat | | |
|---|---|---|---|
| | max | step | both |
| E[<=30; 100] (max:final-gas) under strat | 75 | 56 | 72 |
| E[<=30; 100] (max:steps) under strat | 14 | 10 | 11 |
| E[<=30; 100] (min:gas) under strat | -30 | -20 | -20 |
| control: A[] gas > -35 under pos | | True | |

**Table 1**: Query results for different strategies.

analysis between the strategies, we assume that users do not abandon their journeys; this assumption ensures that all three strategies reach a positive final state.

Figure 2 shows how the gas value develops for simulations under the different strategies. The simulations reveal that users have to endure a dip in their gas at the beginning of their journey to reach the positive final state. From the user's perspective, it is not optimal to have negative experiences (negative gas) to complete the service successfully. Table 1 displays results for user-centric queries proposed in [42]. The strategy `max` achieves the highest amount of gas, 33% above `step`, but it also causes the largest minimum, 50% more than `step`, within the dip. The strategy `step` reduces the number of taken steps by 30%, and improves the minimum gas by 33%, but it also reduces the final gas by 25%. The combined strategy `both` maximizes the final gas while minimizing the expected number of steps, and yields a comparable high maximum as `max`, while reducing the number of steps by 22% and holding `steps`'s improved minimum. In addition, the service provider can establish a lower bound of $-35$ on gas for reaching $s_{pos}$.

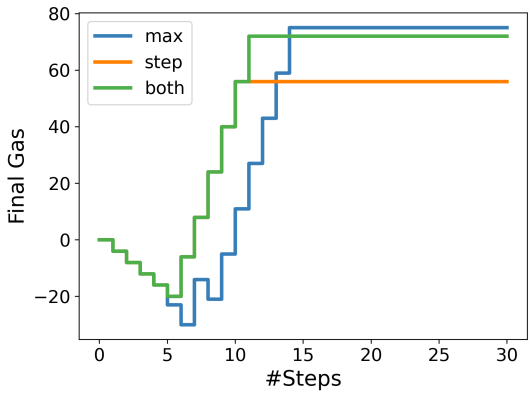



**Figure 2**: Simulations with different strategies.

# 5 Decision Boundaries

*A decision boundary* identifies the states of a user journey game where the eventual outcome of the game will be decided. Finding the decision boundary in a complex game can be useful; for example, although there might be no guarantee of finding a successful strategy `pos` for a game (see Section 4.5), such a strategy may be found for some states in the game. These states may be scattered across the game and therefore hard to identify without automated methods. Moreover, detecting the decision boundary that leads to states from which there is no possibility to recover, can be used to suggest recommendations for service improvement.

The abstraction of the user journey game that we can achieve with decision boundaries, is illustrated by the transition from Figure 1b to Figure 1c. The red and green

parts of the game in Figure 1b show areas of the game in which the outcome is determined. Once a journey reaches a state within those areas, the outcome of the journey is already given. Since all red or green states guarantee the outcome, they can be abstracted away (Figure 1c) — assuming that the parties act strategically.

The decision boundary consists of the last states at which it is possible to influence the outcome of a game and steer it to a positive or negative outcome. This notion can be made precise in terms of the *descendants* of a state (i.e., its set of reachable states), which entail the possibility to reach a positive or negative outcome. The successors and descendants of a state in a user journey game are formally defined as follows:

**Definition 13** (Successors and descendants)**.** *Let* $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$ *be a user journey game and* $s \in \Gamma$*. The successors and descendants of* $s$ *are defined by*

- $\textsc{Successor}(s) = \{s' \mid \langle s, a, s' \rangle \in E \wedge a \in (A_c \cup A_u)\}$
- $\textsc{Descendants}(s) = \textsc{Successor}(s) \cup \bigcup_{s' \in \textsc{Successor}(s)} \textsc{Descendants}(s')$

The decision boundary of a user journey game consists of a set of states after which a positive outcome can no longer be reached, when following one of the successor states. Decision boundaries are formally defined as follows:

**Definition 14** (Decision boundaries)**.** *Let* $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$ *be a user journey game. The* decision boundary *of* $G$ *is a set* $M \subseteq \Gamma$ *such that for all* $s \in M$*:*

- *there exist* $s_{pos} \in T_s, s_{neg} \in T \setminus T_s : \{s_{pos}, s_{neg}\} \subseteq \textsc{Descendants}(s)$,
- $G, s \not\models$ `control: A<> positive`,
- *for all* $s' \in \textsc{Successor}(s)$,
  $G, s' \models$ `control: A<> positive` $\vee\ \textsc{Descendants}(s') \cap T_s = \emptyset$ .

For a state in the decision boundary of a user journey game, there exist reachable final states $s_{pos} \in T_s$ and $s_{neg} \in T \setminus T_s$ and there is no successful strategy for the service provider (who controls the actions $A_c$). Thus, the next transition determines whether the service will reach a state where there is a successful strategy for the service provider or whether the game is lost, excluding the possibility of staying in an outcome-neutral state. The service provider is interested in reaching a positive outcome, not in avoiding a negative outcome. State 4 in Figure 3a is an example of a state in the decision boundary. The following lemma expresses the basic property of decision boundaries that, after leaving the decision boundary, either there exists a winning strategy or the game is lost:

**Lemma 1.** *Let* $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$ *be a user journey game with decision boundary* $M$*. The set* $D = \{s' \mid s' \in \textsc{Successors}(s) \wedge s \in M\}$ *of successors of states in* $M$*, can be divided into two disjoint, non-empty sets:*

$$\{s \in D \mid G, s \models \texttt{control: A<> positive}\} \cap \{s \in D \mid \textsc{Descendants}(s) \cap T_s = \emptyset\} = \emptyset.$$

*Proof.* Assume $s \in M$. By Definition 14, we have $\{s_{pos}, s_{neg}\} \subseteq \textsc{Descendants}(s)$ for some $s_{pos} \in T_s$ and $s_{neg} \in T \setminus T_s$, there is no successful strategy for $s$ (i.e., $G, s \not\models$ `control: A<> positive`), and all successors of $s$ are in either $\{s' \in D \mid G, s' \models$ `control: A<> positive`$\}$ or in $\{s' \in D \mid \textsc{Descendants}(s') \cap T_s = \emptyset\}$. By definition, the intersection of these sets is empty. Now, let us assume that one of the sets is empty.

In this case either $\textsc{Descendants}(s) \cap T_s = \emptyset$ or $G, s \models \texttt{control: A<> positive}$, but then $s \notin M$. Contradiction. So the sets must be non-empty. □

Decision boundaries can be expressed with temporal logic, as shown by the following lemma (where $\overline{\vee}$ denotes exclusive disjunction):

**Lemma 2.** *Let $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$ be a user journey game with decision boundary $M$, $s \in \Gamma$, and $\phi(s)$ and $\psi(s)$ the following temporal propositions:*

- $\phi(s) \coloneqq T_s \cap \textit{Descendants}(s) \neq \emptyset \wedge (T \setminus T_s) \cap \textit{Descendants}(s) \neq \emptyset$
- $\psi(s) \coloneqq (\texttt{control: A<> positive}) \;\overline{\vee}\; (\textit{Descendants}(s) \cap T_s = \emptyset)$

*Then, for all states $s \in \Gamma$, the following property holds:*

- $s \in M \iff G, s \models \phi(s) \wedge \neg\texttt{control: A<> positive} \wedge \texttt{AX}\, \psi(s)$

*Proof.* Assume $s \in M$. By Definition 14, we have $\{s_{pos}, s_{neg}\} \subseteq \textsc{Descendants}(s)$ for some $s_{pos} \in T_s$ and $s_{neg} \in T \setminus T_s$, so $G, s \models \phi(s)$. Furthermore, $G, s \not\models \texttt{control: A<> positive}$, so $G, s \models \neg\texttt{control: A<> positive}$. Finally, $G, s' \models \psi(s')$ for all $s' \in \textsc{Successor}(s)$, so $G, s \models \phi(s) \wedge \neg\texttt{control: A<> positive} \wedge \texttt{AX}\, \psi(s)$.

Now, assume $G, s \models \phi(s) \wedge \neg\texttt{control: A<> positive} \wedge \texttt{AX}\, \psi(s)$. Since $G, s \models \phi(s)$, we have $\{s_{pos}, s_{neg}\} \subseteq \textsc{Descendants}(s)$ for some $s_{pos} \in T_s$ and $s_{neg} \in T \setminus T_s$. Since $G, s \models \neg\texttt{control: A<> positive}$, we have $G, s \not\models \texttt{control: A<> positive}$. Since $G, s \models \texttt{AX}\, \psi(s)$, we have $G, s' \models \texttt{control: A<> positive} \vee \textsc{Descendants}(s') \cap T_s = \emptyset$ for all $s' \in \textsc{Successors}(s)$. Then, by Definition 14, $s \in M$. □

## 5.1 Decision Boundary Detection and Model Reduction

We now consider algorithms for detecting the decision boundary of a user journey game and for model reduction of a user journey game by exploiting its decision boundary. Compared to the workshop paper [39], Algorithm 1 has here been refined to enable the proof of Theorem 3, and Algorithm 2 has been introduced to preserve the transition weights of the initial model. These algorithms make use of a model checker to analyze properties of user journey games. To guarantee that model checking queries will terminate in our algorithms, weighted cyclic games will be approximated by an acyclic directed graph obtained by unrolling all cycles in the game $n$ times (for some integer value $n$). We refer to such an acyclic directed graph as the $n$-bounded (loop) unrolling of the game. For this purpose, we assume given an auxiliary function $\textsc{Acyclic}(G, n)$ which, given a game $G$ and an integer $n$, produces the $n$-bounded unrolling of $G$. (An implementation of this function by means of breadth-first search is given in previous work [41, 42].) Section 5.2 discusses how loop unrolling affects the analysis of the decision boundary in detail.

Algorithm 1 computes the decision boundary for a user journey game $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$ by analyzing the $n$-bounded unrolling of $G$. The algorithm takes as input a user journey game $G$ and a bound $n$ for the loop unrolling. To construct the decision boundary, the algorithm first identifies two subsets of $\Gamma$: $\Gamma_P$, the states for which there is a successful strategy, and $\Gamma_N$, the states for which there is no successful strategy for any descendant. It uses a mapping $R : \Gamma \to \text{Bool}$ to keep track of whether there exists a successful strategy `pos` for a given state. We denote by $G[s]$ the user journey game $G$ in which the initial state has been set to $s$.

**Algorithm 1** Decision Boundary Detection

**Input:** User journey game $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$, loop unrolling value $n$
**Output:** Decision Boundary $M \subseteq \Gamma$

1: $\Gamma_P \leftarrow \emptyset$, $\Gamma_N \leftarrow \emptyset$
2: Initialize mapping $R : \Gamma \to \text{Bool}$
3: **for** State $s \in \Gamma$ **do**
4: Game $G' \leftarrow \textsc{Acyclic}(G[s], n)$
5: Strategy $\texttt{pos} \leftarrow \textsc{Query}(G', s)$
6: Update $R(s) \leftarrow \neg\textsc{Empty}(\texttt{pos})$
7: Set $\Gamma_P \leftarrow \{s \in \Gamma \mid R(s)\}$
8: Set $\Gamma_N \leftarrow \{s \in \Gamma \mid \bigwedge_{s' \in \textsc{Descendants}(s)} \neg R(s')\} \cup (T \setminus T_s)$
9: $M \leftarrow \emptyset$
10: **for** State $s \in \Gamma \setminus \Gamma_P$ **do** ▷ Build decision boundary
11: **if** there exist $s_{pos} \in T_s, s_{neg} \in T \setminus T_s : \{s_{pos}, s_{neg}\} \subseteq \textsc{Descendants}(s)$ **then**
12: **if** $\forall s' \in \textsc{Successors}(s) : s' \in \Gamma_P \ \vee \ s' \in \Gamma_N$ **then** $M \leftarrow M \cup \{s\}$
13: **return** M

For every $s \in \Gamma$, the algorithm updates $R(s)$ (Lines 3–6) by analyzing the game $G'$, which is $G[s]$ unrolled $n$ times. This analysis calls the model checker with $\textsc{Query}(G', s)$, looking for a successful strategy `pos`. The predicate $\textsc{Empty}(\texttt{pos})$ determines whether the model checker found a successful strategy or not. Note that the algorithm only explores the states $\Gamma$ of the original game $G$, not the additional states of the unrolled game $G'$. The result of the model checking for each state $s$ is stored in $R(s)$, after which the sets $\Gamma_P$ and $\Gamma_N$ can be computed (Lines 7–8). Finally, for all states $s \in \Gamma$ that may be in the decision boundary $M$, the inclusion of $s$ in $M$ is decided by looking up successor states in $\Gamma_P$ and $\Gamma_N$ (Lines 10–13). Consequently, the decision boundary contains exactly the states for which the next decision determines the outcome of the game.

The worst-case computational complexity of the algorithm is clearly very high, in particular due to loop unrolling in $\textsc{Acyclic}(G[s], n)$, which increases the state space, and the model checking problem encoded in $\textsc{Query}(G', s)$. However, for our evaluation on real world problems (see Section 8), the algorithm works well because the number of loops is small and the properties considered are reachability properties.

Algorithm 2 computes a reduced user journey game by merging states that can guarantee the same final state, based on the decision boundary computed by Algorithm 1. Since the states in the sets $\Gamma_P$ and $\Gamma_N$ from Algorithm 1 guarantee the outcome of the game, the game can be simplified by abstracting all states in $\Gamma_P$, respectively $\Gamma_N$, into one positive cluster $C_{pos}$ and one negative cluster $C_{neg}$. Algorithm 2 takes as input the user journey game $G$ and the mapping $R$ computed by Algorithm 1. The algorithm then partitions the states $\Gamma$ of $G$ into two sets $\Gamma_P$ and $\Gamma_N$ (Lines 3–4) such that a positive outcome of the game can be guaranteed from any state in $\Gamma_P$ and no positive outcome is possible for states in $\Gamma_N$. From Lemma 1, we know that these sets are non-empty and disjoint.

The function Merge (Algorithm 2, Lines 6 and 7) then computes the corresponding transformation on $G$, in particular transforming the transitions that correspond to

**Algorithm 2** Decision Boundary Reduction

**Input:** User journey game $G = (\Gamma, A_c, A_u, E, s_0, T, T_s, w)$,
mapping $R : \Gamma \to \text{Bool}$ from Algorithm 1
**Output:** User journey game $G' = (\Gamma', A_c, A_u, E', s_0, \{C_{pos}, C_{neg}\}, \{C_{pos}\}, w')$

1: Initialize $G' \leftarrow G$
2: Add states $C_{pos}$ and $C_{neg}$ to $G'$ ▷ States for clustering
3: Set $\Gamma_P \leftarrow \{s \in \Gamma \mid R(s)\}$
4: Set $\Gamma_N \leftarrow \{s \in \Gamma \mid \bigwedge_{s' \in \textsc{Descendants}(s)} \neg R(s')\} \cup (T \setminus T_s)$
5: **for** State $s \in \Gamma'$ **do**
6: **if** $s \in \Gamma_P$ **then** $\textsc{Merge}(G', C_{pos}, s)$
7: **else if** $s \in \Gamma_N$ **then** $\textsc{Merge}(G', C_{neg}, s)$
8: **for** Transition $\langle s, a, C_{pos}\rangle \in E'$ **do**
9: Traces $\tau_{pos} \leftarrow \{\tau \mid \langle s, \tau, s_{pos}\rangle \in E, a \preceq \tau, s_{pos} \in T_s\}$ ▷ Traces to $C_{pos}$
10: $w'(\langle s, a, C_{pos}\rangle) \leftarrow \min_{\tau \in \tau_{pos}} \mathcal{G}(\tau)$
11: **for** Transition $\langle s, a, C_{neg}\rangle \in E'$ **do**
12: Traces $\tau_{neg} \leftarrow \{\tau \mid \langle s, \tau, s_{neg}\rangle \in E, a \preceq \tau, s_{neg} \in T \setminus T_s\}$ ▷ Traces to $C_{neg}$
13: $w'(\langle s, a, C_{neg}\rangle) \leftarrow \max_{\tau \in \tau_{neg}} \mathcal{G}(\tau)$
14: Set $T' \leftarrow \{C_{pos}, C_{neg}\}$
15: Set $T'_s \leftarrow \{C_{pos}\}$
16: **return** $G'$

the merged state. In the reduced game $G'$, we let the weights for the transitions introduced by MERGE approximate the gas of the corresponding paths from the boundary state $s$ in the original game. Transitions connecting states to $C_{pos}$ underapproximate the corresponding gas over paths in $G$ (Lines 8–10), transitions to $C_{neg}$ overapproximate the gas (Lines 11–13). More precise weights can be obtained by considering only partial traces observed in the event log. After the model reduction, the states in $\Gamma'$ that have exactly $C_{pos}$ and $C_{neg}$ as their successor states, form the decision boundary.

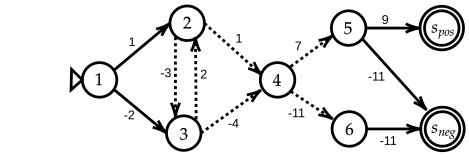
## 5.2 Decision Boundaries and Loop Unrolling

We consider how loop unrolling affects decision boundaries. An example is given in Figure 3, where user actions are depicted as dashed lines and service provider actions as solid lines. The positive and negative final states are $s_{pos}$ and $s_{neg}$. In the original model in Figure 3a, the user can remain in the loop between States 2 and 3. A model in which the user can cycle indefinitely in a loop is unrealistic in user journeys and gives rise to an intractable state space explosion when gas is accumulated in weighted transition systems, which prevents model checking. We remove loops to enable model checking and to keep models more realistic. Figure 3b shows the resulting acyclic model from Figure 3a, where the user can "cycle" in the loop at most one additional round.

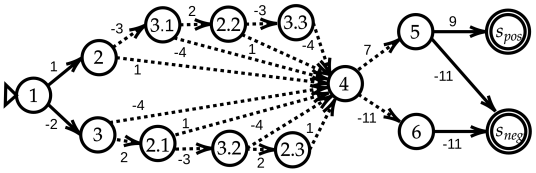
In Figure 3b, there is a strategy that guarantees a successful outcome from State 5, but there is no such strategy from State 6. Thus, both States 5 and 6 can be merged into their respective clusters. Figure 3c shows the game resulting from the model reduction. State 4, marked in blue in Figure 3c, leads to both the positive and negative clusters $C_{pos}$ and $C_{neg}$; thus, it is in the decision boundary according to Definition 14.

A decision taken at State 4 that negatively affects the outcome, cannot be reversed later in the game.

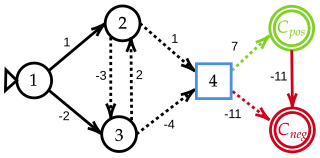
Remark that the traces of an event log do not contain loops, although some cycles may be repeated. Thus, the procedure $\textsc{Acyclic}(G, n)$ provides an abstraction over event logs in the construction of games by deciding how many times loops can be traversed. The choice of value $n$ for the loop unrolling in this procedure determines how many times a game can cycle in the loop. This choice influences numerical evaluations of the user journey game, such as the accumulated gas of a user journey. However, the following theorem shows that it has no impact on the decision boundary detection.




(a) Cyclic model.




(b) Acyclic model.




(c) Game after model reduction.

**Figure 3**: Illustration of unrolling.

**Theorem 3.** *Let* $G = (\Gamma_0, A_c, A_u, E_0, s_0, T, T_s, w)$ *be a user journey game and* $n$ *an integer, the mapping* $R$ *constructed when executing Algorithm 1 on* $G$ *is independent of the value for the* $n$*-bounded loop unrolling.*

*Proof.* Let $G'_i$ denote the game with $i$-bounded loop unrolling of $G$ for some value $i$ and $R_i$ the corresponding mapping constructed by Algorithm 1. By convention, let all states in $\Gamma_0$ be subscripted by 0 (so $x_0 \in \Gamma_0$) and let the new states $\Gamma_i$ in unfolding $i$ be similarly subscripted by $i$ (e.g., $x_i$, $y_i$). Thus, the states $\Gamma$ of $G'_i$ are $\Gamma_0 \cup \ldots \cup \Gamma_i$. Observe that $\Gamma_0$, the domain of $R_i$, is independent of the value $i$ chosen for the loop unrolling; i.e., the states $\Gamma_i$ introduced during loop unrolling $i$ are not in the domain of $R_i$. We say that mappings $R_n$ and $R_m$ are equal, denoted $R_n = R_m$, iff $R_n(s) = R_m(s)$ for all states $s \in \Gamma$. Let $\pi^i_p$ denote a strategy for party $p \in \{c, u\}$ in $G'_i$, where $c$ denotes the controllable and $u$ the uncontrollable player.

Observe that for any $i > 0$, transitions in $G'_i$ are structured in the sense that there will be three types of transitions $\langle x, a, y\rangle$ involving states in $\Gamma_i$: *entry-transitions* where $x \in \Gamma_{i-1}$ and $y \in \Gamma_i$, *internal transitions* where $x, y \in \Gamma_i$ and *exit-transitions* where $x \in \Gamma_i$ and $y \in \Gamma$. Due the recursive nature of the loop unfolding, we further know that if there is an entry-transition $\langle x_{i-1}, a, y_i\rangle$ in $G'_i$, there is a similar entry-transition $\langle x_i, a, y_{i+1}\rangle$ in $G'_{i+1}$, if there is an internal transition $\langle x_i, a, y_i\rangle$ in $G'_i$, there is a similar internal transition $\langle x_{i+1}, a, y_{i+1}\rangle$ in $G'_{i+1}$, and if there is an exit-transition $\langle x_i, a, y_0\rangle$

in $G'_i$, there is a similar exit-transition $\langle x_{i+1}, a, y_0 \rangle$ in $G'_{i+1}$. In the game $G'_i$, the exit-transitions from $\Gamma_i$ will always be to states in $\Gamma_0$ while in games $G'_j$ (where $j > i$), there will be additional exit-transitions from $\Gamma_i$ that are entry-transitions to $\Gamma_{i+1}$.

Now, assume given a game $G'_n$ and the mapping $R_n$ for parties $p \in \{c, u\}$ and some $n > 0$. We need to show that the mapping $R_{n+1}$ for the game $G'_{n+1}$ is such that $R_{n+1} = R_n$. For this purpose, we provide a way to construct strategies for $G'_{n+1}$ from the strategies for $G'_n$. Since strategies consider all opposing actions, the existence of a single strategy is sufficient to guarantee a property [10]. Specifically, we construct strategies $\pi_c^{n+1}$ and $\pi_u^{n+1}$ such that (1) if $\pi_c^n$ guarantees that we reach $T_s$ from $s$ then $\pi_c^{n+1}$ guarantees that we reach $T_s$ and (2) if $\pi_u^n$ guarantees that we reach $T \setminus T_s$ from $s$ then $\pi_u^{n+1}$ also guarantees that we reach $T \setminus T_s$ from $s$.

Consider the game $G'_{n+1}$ with states $\Gamma = \Gamma_0 \cup \ldots \cup \Gamma_{n+1}$ and edges $E$. We define strategies $\pi_p$ for the new states $x_{n+1} \in \Gamma_{n+1}$. For internal and exit-transitions from $\Gamma_{n+1}$, if $\langle x_{n+1}, a, y \rangle \in E$ and $a \in A_p$ $\pi_p^n(x_n) = a$, then let $\pi_p(x_{n+1}) = a$. We then define strategies for the entry-transitions to $\Gamma_{n+1}$. If $\langle x_n, a, y_{n+1} \rangle \in E$ and $a \in A_p$ $\pi_p^n(x_{n-1}) = a$, then let $\pi_p(x_n) = a$. Then, by construction, $\textsc{Query}(G'_{n+1}, x_{n+1})$ if and only if $\textsc{Query}(G'_n, x_n)$. We construct strategies for all states in $\Gamma$ by $\pi_p^{n+1} = \pi_p^n \oplus \pi_p$ for parties $p \in \{c, u\}$. Again, by construction, $\textsc{Query}(G'_{n+1}, x)$ if and only if $\textsc{Query}(G'_n, x)$ for all $x \in \Gamma_0 \cup \ldots \cup \Gamma_N$.

For any state $s \in \Gamma_0$ such that $R_n(s) = \textbf{True}$, we know from $\textsc{Query}(G'_n, s)$ (Line 6) that there exists a strategy $S_n^c$ leading from $s$ to $T_s$ in $G'_n$ and by the construction above there exists a strategy $\pi_c^{n+1}$ leading from $s$ to $T_s$ in $G'_{n+1}$, and $R_{n+1}(s) = \textbf{True}$. Similarly, if $R_n(s) = \textbf{False}$, there exists a strategy $\pi_u^n$ leading from $s$ to $T \setminus T_s$ in $G'_n$ and by the construction above there exists a strategy $\pi_u^{n+1}$ leading from $s$ to $T \setminus T_s$ in $G'_{n+1}$, and $R_{n+1}(s) = \textbf{False}$. Consequently, $R_n(s) = R_{n+1}(s)$ for all $s \in \Gamma_0$.

Finally, consider the case $n = 0$. Assume given $R_1$ for the game $G'_1$. The game $G'_0$, which does not allow any cycle traversal, is obtained from $G'_1$ by removing the states $\Gamma_1$ and all transitions involving these states. We define strategies $\pi_p$ for the states $x_0 \in \Gamma_0$ in $G'_0$ and parties $p \in \{c, u\}$. If $\langle x_1, a, y_0 \rangle \in E$ is an exit-transition in $G'_1$ and $a \in A_p$ for party $p \in \{c, u\}$ and $\pi_p^1(x_1) = a$, then let $\pi_p(x_0) = a$. Then, by construction, $\textsc{Query}(G'_1, x_1)$ if and only if $\textsc{Query}(G'_0, x_0)$. As before, we construct strategies for all states in $\Gamma_0$ by $\pi_p^0 = \pi_p^1 \oplus \pi_p$ for parties $p \in \{c, u\}$ (obviously, the states in $\Gamma_1$ are no longer reachable). By construction, $\textsc{Query}(G'_0, x_0)$ if and only if $\textsc{Query}(G'_1, x_0)$ for all $x_0 \in \Gamma_0$, and it follows that $R_n(s) = R_{n+1}(s)$ for all $s \in \Gamma_0$. □

Since the decision boundary is constructed from the result mapping $R$, it follows from Theorem 3 that the decision boundary is independent of the value chosen for the loop unrolling. In fact, a computationally efficient 0-bounded loop unrolling is sufficient to determine the decision boundary.

# 6 Mining Decision Boundaries

This section explains the mining process for decision boundaries, and particularly discusses how history refinements in transition systems affect the decision boundary.

We build a user journey game from a multi-party event log (see Figure 1 for an overview), following [41,42], but we enrich the construction of the user journey game by

history refinements, to improve the precision of the game and, thereby, the alignment between game and log.

In case the multi-party event log $\mathcal{L} = (L, P, I)$ is lacking, $\mathcal{L}$ can be constructed from a regular event log $L$ by pre-defining parties through an *initiator* function that maps events $a \in A$ in the traces of an event log $L$ to a fixed party in $P = \{c, u\}$, where $c$ denotes the service provider and $u$ the user. For simplicity, we assume that all service providers are all captured by the same party $c$. The party function $I$ then identifies the last initiator in trace $\tau \in L$ by considering the initiator of the last event: $I(\tau) \coloneqq \mathit{initiator}(\mathit{last}(\tau))$.

Given a source log $L_0$, we construct $L$ by inserting an initial event $s_0$ at the beginning of each trace $\tau_0 \in L_0$, and a final event $t \in \{s_{pos}, s_{neg}\}$ at the end of $\tau_0$. To construct history-refined transition systems (Definitions 6 and 7) of horizon $k$, we let $H$ denote the set of states corresponding to a history refinement for all traces $\tau \in L$, then the states $\Gamma$ in $S_L^k$ (respectively $S_L^{\{k\}}$) are given by $\Gamma \subseteq H$ and the transition relation $E$ is constructed over adjacent events in all traces $\tau \in L$. A transition with action $a$ in the transition system, means that the event $a$ has occurred in $\tau$. The resulting history-refined transition systems are transition systems for log $L$ (see Definition 4), and by construction already sliced with $L$ (see Definition 5).

The constructed transition system $S_L^k$, respectively $S_L^{\{k\}}$, is transformed into a user journey game by setting the set of actions controlled by the service provider to $A_c = \{a \in A \mid \mathit{initiator}(a) = C\}$ and the set of actions controlled by the user to $A_u = \{a \in A \mid \mathit{initiator}(a) = U\}$, and computing the weights on the transitions (see Section 4.5). The user journey game is used to compute its decision boundary (Section 5). States that occur after the decision boundary are merged into *successful* and *unsuccessful* states ($C_{pos}$ and $C_{neg}$ in Figure 1c). The result is a reduced game that still preserves all information on the decision structure.

History refinements introduce a trade-off between precision and generalization in the mining process. The horizon $k$ that is chosen for history refinement in the transition system has an impact on how precisely the traces observed in the event log are captured in the transition system. Each state in the transition system reflects a history, i.e., the specific background of traces reaching this state. When deciding on the value $k$, one has to judge whether the observed service incorporates true nondeterminism, the granularity of the observed data is too sparse or further information needs to be considered.

In their seminal work on combining transition systems and state refinements for process mining [3], van der Aalst *et al.* point out that limiting the horizon generally leads to a transition system with fewer distinct states, in which more traces than those observed in the event log traces, are considered valid. Increasing the horizon $k$ yields a transition system with higher precision. Precision is commonly defined as the quotient of valid traces defined by the process model and the observed traces in the event log [2]. Since the set of traces defined by a process model will often be infinite, rendering this quotient is not a useful measurement. Initial proposals for measuring process model precision use a transition system representation of the process model and count the number of unobserved or *escaping transitions* in the transition system [5, 48]. Clearly, when setting the horizon $k$ to the length of the longest trace in the event log and using

the sequence refinement, the transition system only allows exactly the traces in the event log with states being the exact prefixes of the traces in the log.

Conversely, when the horizon $k$ is reduced, the transition system will contain fewer states, each with a less specific history, thus allowing more behavior compared to the observed traces in the event logs. This kind of *generalization* [56] from the traces observed in the event log is often a desired property of discovered process models [3], since we can not always assume that all process behavior has been recorded in the event log. For example, consider a user journey game in which the customer may call the service provider but only two subsequent customer calls have ever been recorded in the event log. By limiting $k$ to a sufficiently small number, the resulting transition system would also consider the customer calling three or more times, which is certainly a useful generalization from the exact observed traces, as valid behavior.

The trade-off between precision and generalization is also visible in the decision boundary. We observed that the number of states within the decision boundary increases when $k$ increases, until $k$ reaches the length of traces leading to the decision boundary, at which point the process model has become a tree of traces, a prefix tree (trie) [31], towards the decision boundary. For lower $k$, many different paths might reach the decision boundary, and for higher $k$ only observed traces reach the decision boundary. The decision boundary reflects the histories leading to decisions that determine the outcome of the user journey. The increased number of states in the model with higher $k$ results in an increased number of states in the decision boundary, each reflecting a different way to reach a the final outcome of the user journey. However, increasing $k$ possibly resolves nondeterminism and thus can remove states from the decision boundary.

# 7 A Tool Chain for Decision Boundary Analysis

The decision boundary computation and the associated model reduction of user journey games have been implemented in a modular tool chain, to facilitate integration in existing process mining tools. For compatibility, the tool chain uses the well-known *XES*[1] standard [33] for logs and the *gexf*[2] standard for transition systems, implicitly treating them as directed, weighted graphs. This allows us to use available analysis tools, e.g., the widely-used network analysis and visualization tool *Gephi*[3] [8] to further investigate the models and display the analysis. To guarantee compatibility and extensibility, we rely on established visualization tools and file formats. The tool chain is written in Python3 and is available online[4] together with installation instructions.

The tool chain consists of several independent programs, each performing a single step in the analysis pipeline for decision boundaries, as illustrated by Figure 1. The detailed tool chain workflow is shown in Figure 4. The program at every step saves its computation in an intermediate model and can thus be independently used from the other steps. The different steps of the tool chain workflow are now described:

[1] https://xes-standard.org/
[2] https://gexf.net/
[3] https://gephi.org/
[4] https://github.com/smartjourneymining/bpi_games/releases/tag/journal_release

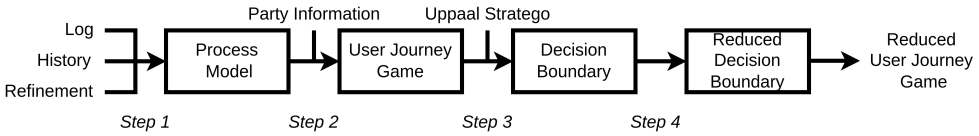



**Figure 4**: Tool workflow.

***Step 1: Process Discovery***

A process model $S_L^k$, resp $S_L^{\{k\}}$ is generated from the input event log $L$, given a specified horizon $k$ and refinement type (sequence or multiset refinement). This step also computes weights and stores them in the intermediate transition system. The output is a weighted transition system, displaying the observed behavior without considering different parties. The modularity of the tool chain allows transition systems generated by third-party tools to be analyzed. The documentation specifies expected fields in the transition system for the next steps.

***Step 2 : User Journey Game Generation***

A user journey game is generated by enriching a process model with party information, provided by the user. We approximate a multi-party event log by assigning events to initiators, assuming that an event is always initiated by the same party. Initiator mappings are provided by the user in an XML file, mapping events to parties. By default, every event not covered in the input file is considered to be controlled by the service provider. This step constructs the party function $I$ and stores it in the intermediate model, transforming the weighted transition system into a weighted game.

***Step 3 : Decision Boundary Construction***

The decision boundary is constructed by means of model checking, as explained in Algorithm 1. Our implementation uses the UPPAAL STRATEGO[5] model checker [24]. The computation of the decision boundary from Section 5 has been slightly optimized to reduce computation time. When the results for all successors of a state are known, the result for the state can be deduced; e.g., if the results of all successors are negative, the parent cannot enforce a positive outcome. In general, not all successors need to be known as long as all reachable final states share the same (positive or negative) outcome. For this reason, our implementation explores the states using a *post-order* traversal strategy and compute the last states first. However, due to loops in the initial user journey game, this optimization cannot be applied to all states, since states that are part of loops still require computation through the loop unrolling procedure and model checking. As with the previous steps, the decision boundary computation is stored in an intermediate model, such that the connection between the decision boundary and other states may be subject to further analysis. States reducible in Step 4 are indicated.

[5] https://uppaal.org/features/#uppaal-stratego

***Step 4 : Decision Boundary Reduction***

Using the computed decision boundary, the user journey game is reduced by merging positive and negative clusters, returning the reduced user journey game, as explained in Algorithm 2.

The tool chain provides functionality for *collective evaluations*: given an event log, all decision boundaries for different history refinements in a specified range can be computed, facilitating decisions on a suitable horizon and type of history refinement.

# 8 Evaluation

We evaluate our decision boundary analysis on two large, real-life datasets from the BPI challenge, to demonstrate the feasibility and possible usefulness of the decision boundary analysis. The BPI challenge is an annual contest organized by the IEEE Task Force on Process Mining[6] that challenges researchers to analyze a real-life event log published along with detailed background information on the process. Since the initial contest, BPIC event logs are frequently used as benchmarks in the process mining community.

The BPI Challenge 2017 (BPIC 2017) [28] provides an event log recording actions ina loan application process from a Dutch financial institution. The BPI Challenge 2012 (BPIC 2012) [27] is provided by the same bank as BPIC 2017, recording an earlier version of the process. Both event logs record activities and interactions between users and a service provider, specifically calls, and are thus promising event logs for user journey analysis. However, for both event logs we needed to make assumptions to complete the missing information for our scenario, in particular to infer the party function $I$, based on domain knowledge about which actions are triggered by which party, and to define which journeys are successful or unsuccessful. The bank that provided these event logs remained anonymous even after the challenge.

BPIC'12 presents interesting challenges not observed in BPIC 2017: (1) the logging is more ad-hoc than in BPIC 2017, (2) the log contains many variants of completed journeys, and (3) user journeys recorded in BPIC 2012 and BPIC 2017 are expected to be very different. We compare the differences between these decision boundary analyses in Section 8.3.

## 8.1 BPIC 2017

The BPIC 2017 event log contains activities from the following groups of activities: Application (A), Offer (O), and Workflow (W) [58]. We define a party function $I$, based on domain knowledge and official information given in the BPIC 2017 forum.[7] We assume that only users can *cancel*, *submit* or *complete* an application, and that users decide whether *calls* take place. Recorded journeys in the log can end with three different states: (1) an offer is accepted, (2) the application is declined, or (3) the application is canceled. We further assume that accepted offers are successful journeys, cancellations are unsuccessful journeys: both parties would prefer a different outcome

[6] https://www.tf-pm.org/
[7] https://www.win.tue.nl/promforum/categories/-bpi-challenge-2017

since the user spent time in the service and the bank invested resources, and declined applications are neither successful nor unsuccessful journeys: users followed the whole process without achieving their goal (the bank has to decline certain offers to protect the users, e.g., from unsustainable debt). We have excluded declined application journeys from the analysis, given their ambiguity.

BPIC 2017 is known to include a substantial change in the service provider's process, called a *concept drift* [4], in July 2016. To investigate how this change impacts the user journey game, we split the log at this month and investigate both parts separately. The first part contains traces until 30.06.2016, while the second part contains traces after 01.08.2016.

### 8.1.1 Generation of the User Journey Game

We now report on the generation of the user journey game for the BPIC 2017 event log, with a focus on the preprocessing of the data. The full implementation is given in the accompanying artefact.[8] We pre-processed BPIC 2017 by discretizing the call durations according to their length, tagging different offers inside one trace, and ignoring incomplete journeys. This was necessary since records of call durations vary between seconds and hours, and several call interactions in one journey consist of repeated adjacent occurrences of events associated with one call. To discretize the duration, we first aggregate repeated and adjacent calls. After the aggregation, we consider calls with a duration under 10 minutes as "short"; between 10 minutes and 4 hours as "long"; and above 4 hours as "super long". Single calls with a speaking time below 60 seconds are omitted in the aggregation. Records of multiple offers can be present in the same journey. One of these offers can be accepted while the remaining are canceled. To simplify journeys, every event associated with an offer or cancellation is ignored after one of the offers is accepted. Offers are automatically canceled if there is no response after 20 days. We differentiate between actively canceled offers and cancellations due to time-out, and ignore incomplete journeys and journeys with declined applications.

We further simplify the event log by removing events that do not influence the journey; e.g., the event *W_Call after offers* is always followed by the event *A_Complete* in the traces, therefore one of them can be removed systematically. We removed outliers and only kept journeys whose variant is present in the corresponding log more than once. Journeys resulting in a cancellation are considered unsuccessful, thus $s_{neg}$ is attached to them; $s_{pos}$ is attached to the successful ones. After preprocessing, we generated the user journey game, following the method discussed in Section 6. We first generated the transition system $S_L^3$, with a sequence history refinement of horizon 3. The party function $I$ and weight $w$ transformed $S_L^3$ into a user journey game. Tables 2 and 3 present the number of loops and states per model. In both event logs, a sequence history refinement of horizon 3 keeps the states minimal while allowing only for feasible numbers of loops.

[8] https://github.com/smartjourneymining/bpi_games/releases/tag/EdbA22

| | Refinement | | | |
|---|---|---|---|---|
| | Sequence | | Multiset | |
| **Horizon** | Loops | States | Loops | States |
| 1 | 103 | 26 | 103 | 26 |
| 2 | 46 | 77 | 88 | 68 |
| 3 | 5 | 152 | 16 | 123 |
| 4 | 0 | 247 | 0 | 191 |
| 5 | 0 | 346 | 0 | 263 |
| 6 | 0 | 445 | 0 | 346 |

**Table 2**: Properties of the model of BPIC 2017 before July 2016.

| | Refinement | | | |
|---|---|---|---|---|
| | Sequence | | Multiset | |
| **Horizon** | Loops | States | Loops | States |
| 1 | 103 | 26 | 105 | 26 |
| 2 | 46 | 82 | 36 | 72 |
| 3 | 5 | 167 | 6 | 139 |
| 4 | 0 | 271 | 4 | 214 |
| 5 | 0 | 384 | 2 | 296 |
| 6 | 0 | 493 | 0 | 372 |

**Table 3**: Properties of the model of BPIC 2017 after August 2016.

### 8.1.2 Decision Boundary

In the following, we drop the assumption from Section 4.6 that users do not quit their journeys. An exhaustive search over all states produces decision boundaries for both BPIC 2017 event logs, using Algorithm 1. Figure 5a shows the decision boundary for the first log, i.e. until July, and Figure 5b for the second log, each with merged clusters.[9] The states $C_{pos}$ and $C_{neg}$ represent their respective clusters. Blue squared states mark the decision boundary. Time-out cancellation transitions are violet, transitions with a positive weight are green, and transitions with a negative weight are red. We illustrate a single successful user journey present in the log with filled states (in Figure 5a), starting from the top right red state, the user traverses via the red and grey filled states to the green filled state, and finally to $C_{pos}$.

We first report on the outcomes of the analysis for the first event log and then for the second one. Our analysis revealed the existence of few paths to successful final states, and that several journeys time-out very far into the application process.

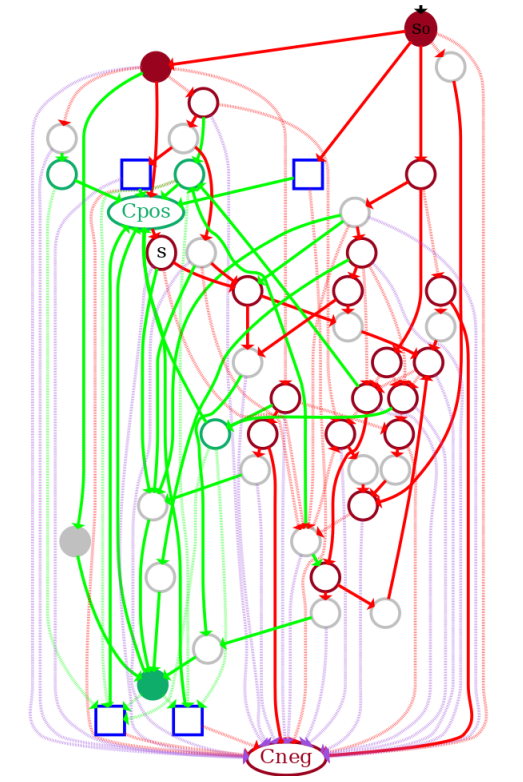
The decision boundary reduction here reduces the number of states from 152 to 50, thus leading to a reduction of 67%. Analyzing the decision boundary reveals that most states are negatively biased and have a direct connection to $C_{neg}$. With uncontrollable user actions, the service provider has no means to guide the user to a successful outcome, except for a few positive states around $C_{pos}$.

Most positive states require long journeys. A detailed analysis reveals that three out of four states in the decision boundary are related to calls, see Appendix A.1. The event *"W_Call incomplete files"* leads to the decision boundary from two states and *"O_Sent (online only)"* (only sending the offer online) from two other states. Additionally, $C_{pos}$ has an outgoing transition to state $s$, leading to follow-up actions after creating the second offer and sending out the first one per mail and online, connecting $C_{pos}$ to $C_{neg}$ and to further states in the middle of the user journey game via the state $s$, indicating that the service provider does not act fully strategically. The figure also reveals many time-out cancellations from various states during the journey, even for paths that are very far progressed into the application process. These are unsatisfactory for both parties.

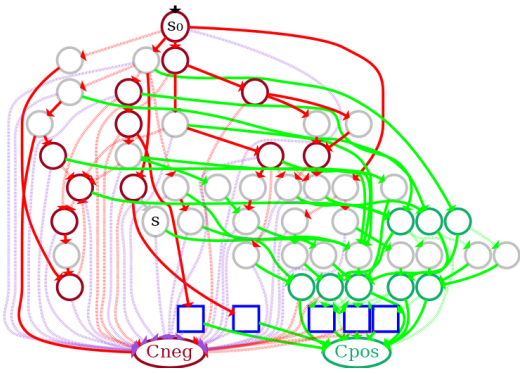
[9] Additional plots with included state names are published in the repository.

We can draw three recommendations for the service provider from the decision boundary analysis: (1) reaching the positive outcome should be simplified, thus the decision boundary could be extended, (2) the behavior in $C_{pos}$ should be more strategic, important decisions should be reached earlier in the process, and (3) well-progressed journeys should be increasingly prevented from time-outs, thus reducing the number of time-outs of progressed journeys.

Figure 5b shows the process model for the second BPIC 2017 data set, the decision boundary reduction reduces the number of states from 167 to 59, thus a reduction of 65%. The figure shows that the process model changed significantly after the concept drift in July 2016. The new decision boundary inherits all states except one and contains one new state, see Appendix A.1. We see that the positioning of the new decision boundary has improved in two parts: (1) it reaches further into the negative part of the game, and (2) it has increased in size. While the previous decision boundary contained only four states, the new decision boundary contains five states. The new decision boundary includes three out of four states from the previous decision boundary and the fourth state lies now before the previous decision boundary. Additionally, the new decision boundary contains two new states: one that was previously in the positive cluster and one new state. State $s$, which was previously reached from $C_{pos}$, now lies before the previous decision boundary. The new decision boundary is earlier in the




(a) BPIC 2017 before July 2016.




(b) BPIC 2017 from August 2016.

**Figure 5**: Decision boundaries (blue) for both BPIC 2017 event logs.

process model: key decisions are taken earlier in the process, and it is static: the service provider acts strategically and provides no possibility of reaching $C_{neg}$ from $C_{pos}$.

We observe that besides its composition, the reachability of the decision boundary also improved. The number of states reaching the decision boundary increased by 22%. We further observe that the amount of timeouts at advanced stages of the user journey has been reduced; users that continue far into the user journey are more prone to finish successfully or to cancel by themselves. The average number of events on shortest paths from start to time-out has been reduced from 7.645 to 7.27, generally improving the user journey. The service provider can now start to investigate the actions related to states in the decision boundary.

## 8.2 BPIC 2012

The BPIC 2012 event log contains activities from the same activity groups as BPIC 2017: Application (A), Offer (O), and Workflow (W). We infer the party function $I$ based on additional information and the analysis of Bautista *et al.* [9]. The party function models the actions of service provider and user. We assume that the user *submits* a (partial) application and might *cancel* it, the user can *decline*, *send back* or *cancel* the resulting offers, and that the user controls the *calls* after offers and regarding incomplete information. Completed journeys end with states as in BPIC 2017: (1) the offer is accepted, (2) the application is denied, or (3) the application is canceled. We use the classification from Section 8.1 of successful and unsuccessful journeys, ignoring declined journeys. We investigate the BPIC 2012 event log from the user perspective and evaluate changes from the users' perspective in comparison to BPIC 2017.

### 8.2.1 Generation of the User Journey Game

It is evident that the bank altered the loan application process and its logging, thus we adapt our preprocessing as follows: the full implementation is accessible in the project repository with the tool, see Section 7. In the first step, we removed incomplete and declined journeys from the log, and aggregated call events to obtain insights about the duration of single calls. We discretized the calls by using *Bayesian-Gaussian Mixture* [7] clustering on the aggregated lengths, ignoring single call instances with a length of less than 60 seconds. Further, we tagged the *creation* and *sending* of offers to gain insights into how many offers were available in a single application.

Moreover, we simplify the log by removing redundant events. Bautista *et al.* [9] observe that *A_PARTLYSUBMITTED* always occurs immediately after *A_SUBMITTED*, *O_DECLINED* always occurs simultaneously with *A_DECLINED*, and that *A_APPROVED*, *O_ACCEPTED*, *A_ACTIVATED* and *A_REGISTERED* occur together in no specific order. Additionally, we discovered that *O_CREATED* is always followed by *O_SENT*. In the cleaned log, we only keep one event per pair.

We augment the log such that every trace starts in a designated start event $s_{start}$ and append an event $s_{pos}$ to the successful journeys and an event $s_{neg}$ to the negative ones. The preprocessing strongly homogenizes the event log: the event log of positive and negative traces consists originally of more than 3000 variants, the cleaned event

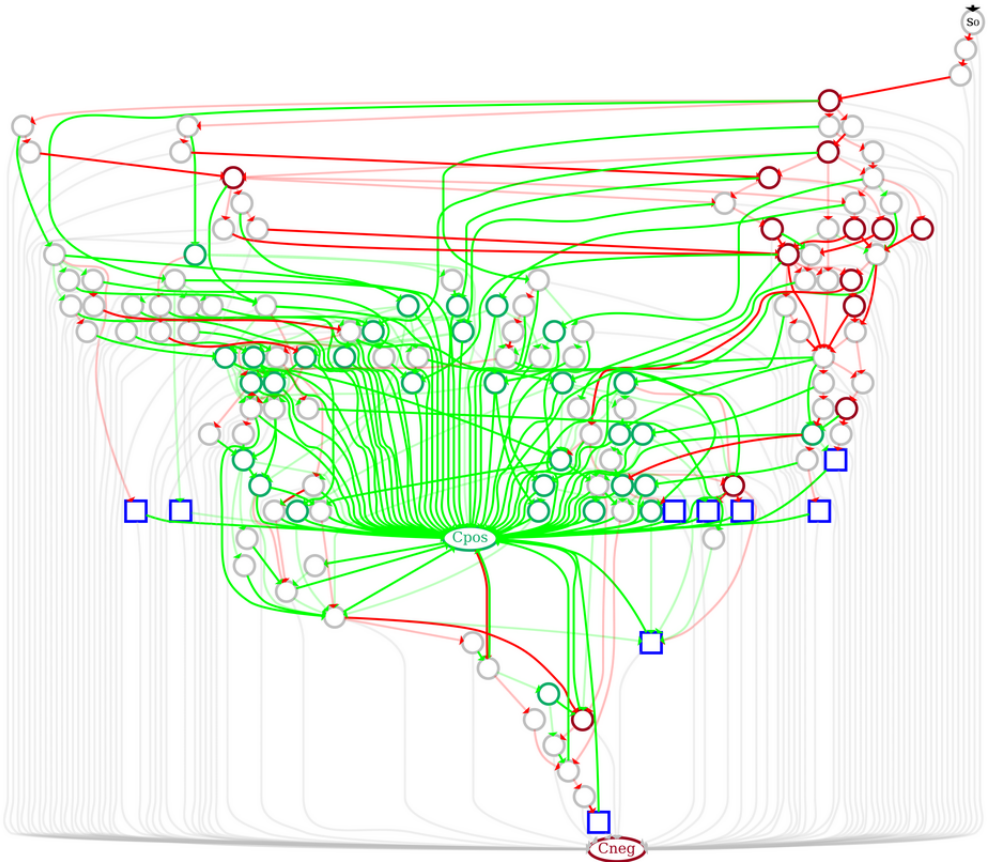



**Figure 6**: BPIC 2012 Decision Boundary.

log contains only 202. Interestingly, BPIC 2012 records a higher ratio of declined journeys than BPIC 2017, after the preprocessing the dataset contains 5053 traces; among them 2807 are negative and 2246 positive. We discard 7635 traces declined at different stages in the process model and 399 unfinished traces. For additional discussions on declined journeys and details regarding distributions in the event log, see [9]. Table 4 presents the number of loops and states for the sequence and multiset refinements under different horizons. With horizon 5, we obtain a feasible number of loops while keeping the number of states as small as possible, resulting in the 5-multiset transition system $S_L^{\{5\}}$. By using the party function $I$ and computing transition weights, $S_L^{\{5\}}$ is transformed into a user journey game.

| | **Refinement** | | | |
|---|---|---|---|---|
| | Sequence | | Multiset | |
| **Horizon** | Loops | States | Loops | States |
| 3 | 3500 | 254 | 175666 | 177 |
| 4 | 94 | 471 | 51746 | 316 |
| 5 | 0 | 706 | 2 | 449 |
| 6 | 0 | 894 | 6 | 560 |

**Table 4**: Properties of the BPIC 2012 model.

| | | 2012 | Until July'16 | From August'16 |
|---|---|---|---|---|
| **Successful** | Incomplete | 1.00 | 0.85 | 0.85 |
| | Information | | | |
| | After Offer | 0.63 | 0.44 | 0.62 |
| **Unsuccessful** | Incomplete | 0.70 | 0.85 | 0.85 |
| | Information | | | |
| | After Offer | 0.60 | 0.08 | 0.05 |

**Table 5**: Average number of calls by category.

### 8.2.2 Decision Boundary

The tool chain from Section 7 computes the decision boundary (with merged clusters) for the processed BPIC 2012 event log, displayed in Figure 6.[10] Blue states represent the decision boundary, the states $C_{pos}$ and $C_{neg}$ aggregate the respective cluster of guaranteed positive and negative states, and transitions with positive weight are colored green while transitions with negative weights are colored in red, transitions to $C_{neg}$ are colored gray to improve visibility. The decision boundary reduction reduces the number of states from 449 to 152, a reduction of 66%.

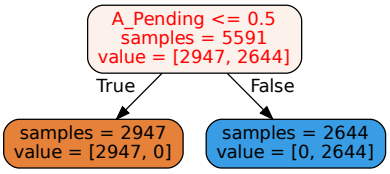
The positive cluster $C_{pos}$ connects to two states by creating new offers, confirming the trend observed between the two BPIC 2017 decision boundaries. When being in $C_{pos}$, the company does not always act strategically and moves to states from where it is no longer possible to guarantee a successful strategy. Analyzing the nine states in the decision boundary, it shows that all states are related to calls: six states contain in total two calls, and three states one call, see Appendix A.2.

Further, we observe that the states in the decision boundary can be grouped into three classes: (1) no cancellations (two states), (2) with one cancellation (three states), and (3) two cancellations (four states). Thus, a large part of the decision boundary is dominated by (several) calls and rejections of offers.

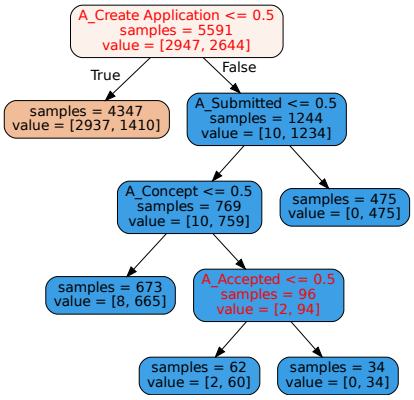
## 8.3 Comparing Decision Boundaries

We highlight insights gained from the decision boundary analysis by comparing the three generated decision boundaries with statistics from the logs. Our analysis indicates that the bank in 2017 is more resourceful and is more skilled in generating suitable offers at the right time than in 2012. The service was successfully improved by sending appropriate offers to users, even sending multiple offers automatically without first requiring a rejection of previous offers. With the help of the decision boundary analysis, a service provider can investigate the impact of process changes on the user journey from the users' point of view. Without decision boundary analysis, deriving these insights was impractical due to the size of the logs, rendering manual analysis prohibitively expensive. Further, existing techniques do not assume the multi-party perspective we introduced through multi-party event logs, highlighting the interaction through communication and actions in the user journey. Decision boundaries provide an automatic reduction, without compromising available information, highlighting important interactions for the success of the service.

[10] An additional plot with included state names is published in the repository.




(a) Decision Tree with Removed Tail 1.




(b) Decision Tree with Removed Tail 12.

**Figure 7**: Decision Trees for BPIC'17-2.

***Conducted calls***

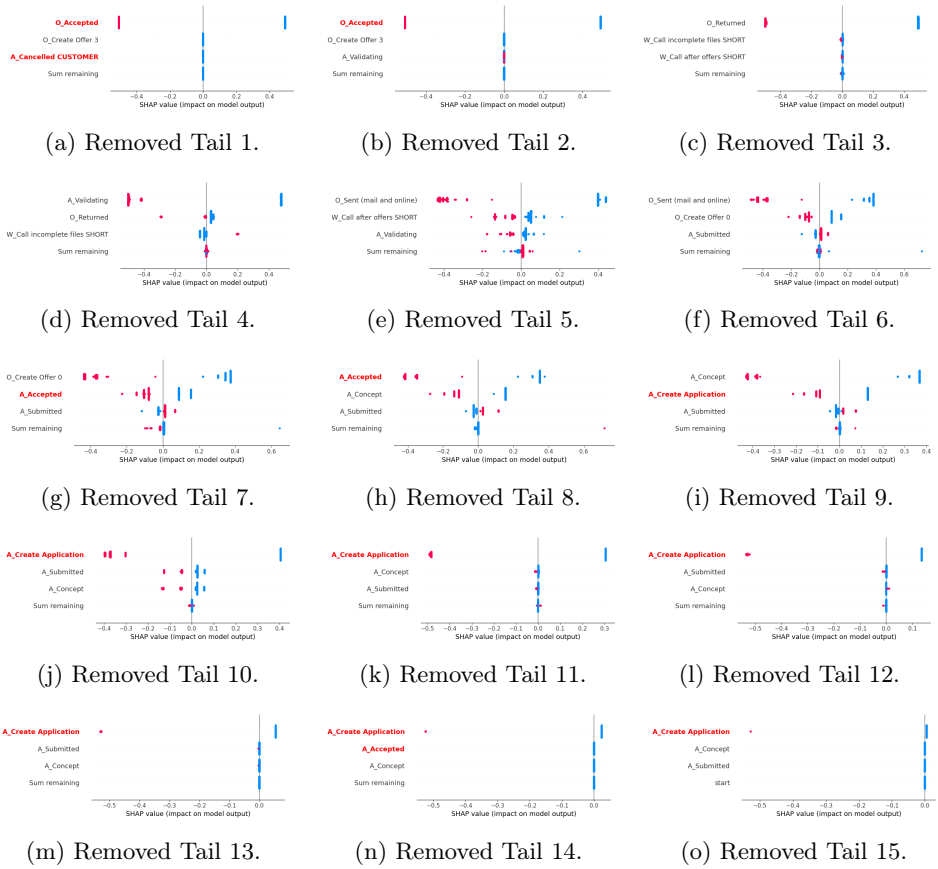
All states in the decision boundary of BPIC 2012 are related to calls, while the decision boundaries of BPIC 2017 both contain one state without any calls. Comparing the number of calls in the BPIC 2012 event log to the BPIC 2017 event logs reveals that the company notably improved its resource handling: calls are more focused. Table 5 presents the average number of calls in positive and negative journeys, showing that most calls are conducted in BPIC 2012. In comparison, the difference between the two event logs from BPIC 2017 is not as big, except for calls after offers. Here, the company increased the average number of calls by nearly 50%. In general, the bank decreased the number of calls regarding incomplete information and calls fewer users after offers in unsuccessful journeys. This observation shows that the company improved the user journeys for both parties, it saves resources by calling users in fewer scenarios and shortens the journey for the users.

***Generated loan offers***

In BPIC 2012, seven out of nine states relate to cancelled offers. Comparing the amount of generated loan offers in BPIC 2012 with the later processes, we observe that the number of proposed offers is reduced. A successful journey in BPIC 2012 receives on average 1.48 loan offers, in both BPIC 2017 event logs successful journeys receive on average only 1.07 offers before the concept drift, and 1.08 offers afterwards. However, in unsuccessful journeys in BPIC 2012, only 0.78 offers are generated on average. In unsuccessful journeys in BPIC 2017, there is only a slight difference, 1.23 resp. 1.16 offers are generated on average.

| Dataset | Removed Tail Length | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Original | 100.0 | 100.0 | 99.79 | 99.79 | 97.5 | 96.07 | 96.14 | 95.92 | 95.99 | 93.71 | 88.05 | 76.04 | 65.45 | 57.44 | 54.65 |
| Reduced | 99.71 | 99.71 | 99.79 | 99.79 | 97.5 | 96.07 | 95.92 | 93.06 | 93.71 | 88.05 | 76.04 | 65.45 | 57.44 | 54.65 | 54.08 |
| Difference | 0.29 | 0.29 | 0 | 0 | 0 | 0 | 0.22 | 2.86 | 2.28 | 5.66 | 12.01 | 10.59 | 8.01 | 2.79 | 0.57 |

**Table 6**: Accuracy of decision trees (in %).




(a) Removed Tail 1. (b) Removed Tail 2. (c) Removed Tail 3.

(d) Removed Tail 4. (e) Removed Tail 5. (f) Removed Tail 6.

(g) Removed Tail 7. (h) Removed Tail 8. (i) Removed Tail 9.

(j) Removed Tail 10. (k) Removed Tail 11. (l) Removed Tail 12.

(m) Removed Tail 13. (n) Removed Tail 14. (o) Removed Tail 15.

**Figure 8**: Beeswarm Plots for Shapley Values.

## 8.4 Using the Decision Boundary Reduction

To further analyse to potential usefulness of our reduction technique, we illustrate how our presented decision boundary reduction could be incorporated into an established process mining pipeline for process outcome prediction. Therefore, we investigate how the outcome of the loan application in the later BPIC'17 event log could be predicted using decision trees. Most predictors heavily rely on features removed by our technique; thus, they rely on highly correlated features that could already imply the journey

outcome. These features obscure the actual performance of the decision tree and do not provide helpful insights for explanations. We measure the importance of individual features in a decision tree using Shapley values [47].

To predict the outcome, we encode each trace as a feature vector in *boolean encoding* [26]. Each dimension of the feature vector corresponds to an event and is set only to 1 if that event was observed in the trace. Based on this tabular data, a decision tree classifier is learned on 80% fo the traces in the event log, and the accuracies measured on the remining 20% for each classifier are reported in Table 6. In the "Reduced" dataset are the events removed by the decision boundary reduction removed from the traces. For better overview, we state the difference between the two in the last row. In our experiments, we iterate over traces of varying lengths by ignoring the tails of each trace. Ignoring a tail of length 1 corresponds to the full trace without the final outcome, and a tail of length 12 ignores the last 12 events. In Figure 7, we depict two learned decision trees. Figure 7a ignores tails of length 1, and Figure 7b ignores tails of length 12. Each node indicates the feature that the node is splitting on. If the feature was removed in the reduction, the node is colored red. Except for tails with a length of $3-6$, all trees contain at least one redundant event, i.e., an event removed by the decision boundary reduction.

To estimate the importance of those events in the decision tree, we compare the Shapley values [47] for the trees in Figure 8. Each subplot displays the Shapley values computed over 300 samples from the training log, for trees learned on traces excluding increasingly long tails, red indicates a high value and blue a low value. In each decision tree, besides tail lengths of $3-6$, a redundant feature is the most or second most influential feature, highlighted in bold red. Meaning, even if a large fraction of the trace is observed, the decision tree still relies on information that determines the journey outcome individually. Note that decision trees trained on the reduced traces achieve a comparable performance to those including redundant states.

Ignoring the states removed by the decision boundary reduction may improve the actionability and consistency of explanations. Decision trees relying on states removed by the decision boundary rely on late and redundant information. Thereby, they might reduce the quality of the explanation for the end-user.

# 9 Discussion: Limitations & Future Work

This section considers limitations to the presented work on identifying decision boundaries for user journey games as well as possible directions for future work.

## 9.1 Limitations

Multi-party event logs, as defined in this paper, assume that every event is mapped to exactly one party. We have not investigated, and therefore not formalized, how the mapping of one event to multiple parties will affect the construction of the user journey game and nor the analysis of the game. However, we do not believe this is a major limitation to our approach: based on Definition 2), a nondeterministic situation can be modelled pessimistically, by assuming that the event is uncontrollable, or optimistically, by assuming that the event is controllable.

Process discovery in this paper has been based on history refinements. We used $k$-sequence refinement for the BPIC 2017 event log and $k$-multiset refinement for the BPIC 2012 event log to construct the transition systems. In our evaluations, these refinements captured the complexity in the logs well, resulting in transition systems with a feasible amount of loops and an acceptable number of states. We have not studied the use of other methods for process discovery that may allow further refinements in the process model, e.g., passive automata learning or discovery algorithms for Petri nets (e.g., [2, 3, 50, 71]).

The analysis of user journey games that we consider in this paper assumes that users can influence their journey through active decisions. A service provider who wants to analyse a service, needs to determine whether events are under their control or under the users' control. In the user journey game, the latter are treated as uncontrollable interactions initiated by the user. User journey games only consider two parties. Consequently, sub-contractors of the service provider are aggregated in the service provider party, which suggests that they collaborate optimally. This assumption shifts the focus of the model to the interaction between users and the service provider. However, other abstractions are possible. For example, user journey games could analyze specific sub-contractors by aggregating the service provider and the other sub-contractors with the users in the user player. Furthermore, the game construction abstracts from how external information might influence the outcome of a user journey. The service provider and its sub-contractors might use information that is not captured in the game to consolidate their decisions, independent of user actions, but the influence of such an external context is at best considered in our model through its possible reflection in the event log.

User journeys are a method within service science research that can give insights into a user's actions and interactions, whereas user experience encompasses a broader concept, that includes not only struggles or (un)successful outcomes but also personal feelings of users while engaging with a service. We acknowledge that our method, as any purely data-driven method, relies on the relevant aspects of user experience being reflectedin the event log. Our method does not claim to predict the future evolution of user journeys or to capture the part of a user's experience that is not recorded in events. To obtain a complete understanding of a user's experience, it needs to be complemented with manual techniques. Under these assumptions, we introduced the concept of *gas* in previous work [41, 42], which has been shown to have a favorable response from domain experts. However, further studies are needed to compare our concept of gas and user experience.

## 9.2 Future Work

User journey games and decision boundaries open many interesting areas of future work. We here focus on three complementary directions.

First, further development of the tool chain. We plan to combine user journey games with established process mining tools, e.g., Petri nets [49,54], to discover process models for behavior leading to states with determined outcome. It would be very interesting to increase automation, both with respect to the manual input needed in the current process and to automate recommendations for service improvements based

on decision boundaries. Hereby, a user study with stakeholders would be of particular interest to evaluate the insights gained through decision boundaries and to compare these insights to traditional process mining techniques.

Second, we plan to capture ambiguities within user actions by means of probabilistic user journey games. Probabilistic decisions of the user and the service provider would make the model more realistic by lifting the assumption that both parties always act optimally. In reality, users do not always perform the worst possible action — and the service providers unfortunately not the best ones! It would be interesting to study probabilistic user journey games and their probabilistic decision boundaries.

Third, we would like to support the analysis of user journey in an online fashion, on real use cases in collaboration with service providers. The automation of the model construction from continuously growing event logs suggests that user journey games could support a continuous user journey evaluation process. principle, we expect that in practice further automation will be required and in particular support for non-deterministic multi-party event logs. Involving service providers in the validation and improvement of the tool will enhance its applicability in the user journey tool landscape.

# 10 Conclusion

This paper presents decision boundary analysis for user journey games, User journey games are weighted games that capture the user's perspective on a user journey between user and service provider. The decision boundary consists of states in the game whose next transition determines the outcome of the game, assuming that both parties act strategically. We show how to build user journey games from user journey event logs and how to compute decision boundaries for these games on two publicly available real-world event logs, demonstrating that the decision boundary can provide clear insights into determining factors of a user journey before and after a concept drift in the logs, and uncovers changes in the workflows and resources of the service provider. We demonstrate the usefulness of the decision boundary analysis for model reduction. The paper further explores the impact of refinements of the user journey game on the decision boundary analysis, especially loop unrolling and history refinement.

We have implemented the decision boundary analysis in a modular tool that can be integrated into existing process mining frameworks. To illustrate such an integration, we have shown how decision boundary reduction can improve the explainability of decision trees for predicting the outcome of user journeys. In this example, we obtain a model reduction from the boundary analysis that reduces the reliance of the decision tree on outcome-determining states.

User journey games with decision boundary analysis enable service analysts to perform an automated, user-centric analysis that complements existing methods in process mining for analyzing the service structure. The resulting games highlight the (worst) possible interactions between a service and its users. The decision boundary explicitly identifies the crucial states in the user journey game, where the outcome of the journey is in fact determined. Further, the decision boundary reduction highlights service improvements from the user perspective where the service provider did not

act strategic. In practice, the insights provided by the decision boundary analysis complement recommender systems on user journeys and statistical insights on user behavior in the service.

**Acknowledgment.** Andrea Pferscher, Ragnhild Halvorsrud and Gunnar Rye Bergersen have provided valuable feedback on this work.

# Declarations

## Ethical Approval

Not applicable.

## Funding

This work is funded by the Research Council of Norway as part of the *Smart Journey Mining* project (grant no. 312198).

## Availability of data and materials

This paper only uses datasets that are already publicly available; these are published in [27] and [28]. All presented results are also open access, available at https://github.com/smartjourneymining/bpi_games/releases/tag/journal_release.

# References

[1] van der Aalst, W.M.P.: On the representational bias in process mining. In: Reddy, S., Tata, S. (eds.) Proc. 20th Intl. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2011). pp. 2–7. IEEE Computer Society (2011). https://doi.org/10.1109/WETICE.2011.64

[2] van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer, 2 edn. (2016). https://doi.org/10.1007/978-3-662-49851-4

[3] van der Aalst, W.M.P., Rubin, V., Verbeek, E., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: A two-step approach to balance between underfitting and overfitting. Software & Systems Modeling **9**(1), 87–111 (Jan 2010). https://doi.org/10.1007/s10270-008-0106-z

[4] Adams, J.N., van Zelst, S.J., Quack, L., Hausmann, K., van der Aalst, W.M.P., Rose, T.: A framework for explainable concept drift detection in process mining. In: International Conference on Business Process Management. Lecture Notes in Computer Science, vol. 12875, pp. 400–416. Springer (2021). https://doi.org/10.1007/978-3-030-85469-0_25

[5] Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Measuring precision of modeled behavior. Inf. Syst. E Bus. Manag. **13**(1), 37–67 (2015). https://doi.org/10.1007/s10257-014-0234-7

[6] de Alfaro, L., Henzinger, T.A., Majumdar, R.: Symbolic algorithms for infinite-state games. In: Larsen, K.G., Nielsen, M. (eds.) Proc. 12th International

Conference on Concurrency Theory (CONCUR 2001). Lecture Notes in Computer Science, vol. 2154, pp. 536–550. Springer (2001). https://doi.org/10.1007/3-540-44685-0_36
[7] Attias, H.: A variational Bayesian framework for graphical models. In: Solla, S.A., Leen, T.K., Müller, K. (eds.) Advances in Neural Information Processing Systems 12, NIPS. pp. 209–215. The MIT Press (1999)
[8] Bastian, M., Heymann, S., Jacomy, M.: Gephi: An open source software for exploring and manipulating networks. Proceedings of the International AAAI Conference on Web and Social Media **3**(1), 361–362 (Mar 2009). https://doi.org/10.1609/icwsm.v3i1.13937
[9] Bautista, A.D., Wangikar, L., Akbar, S.M.K.: Process mining-driven optimization of a consumer loan approvals process. BPI Challenge (2012), https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2012:bautista.pdf
[10] Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., Lime, D.: UPPAAL-Tiga: Time for playing games! In: Damm, W., Hermanns, H. (eds.) Proc. 19th Intl. Conf. on Computer Aided Verification (CAV 2007). Lecture Notes in Computer Science, vol. 4590, pp. 121–125. Springer (2007). https://doi.org/10.1007/978-3-540-73368-3_14
[11] Bellodi, E., Riguzzi, F., Lamma, E.: Probabilistic declarative process mining. In: International Conference on Knowledge Science, Engineering and Management. pp. 292–303. Springer (2010). https://doi.org/10.1007/978-3-642-15280-1_28
[12] Berendes, C.I., Bartelheimer, C., Betzing, J.H., Beverungen, D.: Data-driven customer journey mapping in local high streets: A domain-specific modeling language. In: Pries-Heje, J., Ram, S., Rosemann, M. (eds.) Proc. Intl. Conf. on Information Systems - Bridging the Internet of People, Data, and Things (ICIS 2018). Association for Information Systems (2018), https://aisel.aisnet.org/icis2018/modeling/Presentations/4
[13] Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. In: Franch, X., Ralyté, J., Matulevicius, R., Salinesi, C., Wieringa, R.J. (eds.) Proc. Forum and Doctoral Consortium Papers at the 29th Intl. Conf. on Advanced Information Systems Engineering (CAiSE 2017). CEUR Workshop Proceedings, vol. 1848, pp. 49–56. CEUR-WS.org (2017), http://ceur-ws.org/Vol-1848/CAiSE2017_Forum_Paper7.pdf
[14] Bernard, G., Andritsos, P.: CJM-ab: Abstracting customer journey maps using process mining. In: Mendling, J., Mouratidis, H. (eds.) Information Systems in the Big Data Era - Proc. CAiSE Forum 2018. Lecture Notes in Business Information Processing, vol. 317, pp. 49–56. Springer (2018). https://doi.org/10.1007/978-3-319-92901-9_5
[15] Bernard, G., Andritsos, P.: Contextual and behavioral customer journey discovery using a genetic approach. In: Welzer, T., Eder, J., Podgorelec, V., Latific, A.K. (eds.) Proc. 23rd European Conference on Advances in Databases and Information Systems (ADBIS 2019). Lecture Notes in Computer Science, vol. 11695, pp. 251–266. Springer (2019). https://doi.org/10.1007/978-3-030-28730-6_16
[16] Bitner, M.J., Ostrom, A.L., Morgan, F.N.: Service blueprinting: A practical technique for service innovation. California Management Review **50**(3), 66–94 (Apr

2008). https://doi.org/10.2307/41166446
[17] Bose, R.J.C., Mans, R.S., Van Der Aalst, W.M.: Wanna improve process mining results? In: 2013 IEEE symposium on computational intelligence and data mining (CIDM). pp. 127–134. IEEE (2013). https://doi.org/10.1109/CIDM.2013.6597227
[18] Bouyer, P., Cassez, F., Fleury, E., Larsen, K.G.: Optimal strategies in priced timed game automata. In: Lodaya, K., Mahajan, M. (eds.) Proc. 24th Intl. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2004). Lecture Notes in Computer Science, vol. 3328, pp. 148–160. Springer (2004). https://doi.org/10.1007/978-3-540-30538-5_13
[19] Cateriano-Arévalo, E., Saavedra-Garcia, L., Ponce-Lucero, V., Miranda, J.J.: Applying customer journey mapping in social marketing to understand salt-related behaviors in cooking: A case study. International Journal of Environmental Research and Public Health **18**(24), 13262 (2021). https://doi.org/10.3390/ijerph182413262
[20] Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: PRISM-games: A Model Checker for Stochastic Multi-Player Games. In: Piterman, N., Smolka, S.A. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 185–191. Lecture Notes in Computer Science, Springer (2013). https://doi.org/10.1007/978-3-642-36742-7_13
[21] Chesani, F., Di Francescomarino, C., Ghidini, C., Grundler, G., Loreti, D., Maggi, F.M., Mello, P., Montali, M., Tessaris, S.: Shape your process: Discovering declarative business processes from positive and negative traces taking into account user preferences. In: International Conference on Enterprise Design, Operations, and Computing. pp. 217–234. Springer (2022). https://doi.org/10.1007/978-3-031-17604-3_13
[22] Clarke, E.M., Grumberg, O., Kroening, D., Peled, D.A., Veith, H.: Model checking. MIT Press, 2 edn. (2018)
[23] Crosier, A., Handford, A.: Customer Journey Mapping as an Advocacy Tool for Disabled People: A Case Study. Social Marketing Quarterly **18**(1), 67–76 (Mar 2012). https://doi.org/10.1177/1524500411435483
[24] David, A., Jensen, P.G., Larsen, K.G., Mikučionis, M., Taankvist, J.H.: Uppaal Stratego. In: Baier, C., Tinelli, C. (eds.) Proc. 21st Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015). Lecture Notes in Computer Science, vol. 9035, pp. 206–211. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_16
[25] Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. IEEE Transactions on Services Computing **12**(6), 896–909 (2016)
[26] Di Francescomarino, C., Ghidini, C.: Predictive process monitoring. In: Process mining handbook, pp. 320–346. Springer International Publishing Cham (2022)
[27] van Dongen, B.: BPI Challenge 2012 (2012). https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f, https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204
[28] van Dongen, B.: BPI Challenge 2017 (2017). https://doi.org/10.4121/uuid:

5f3067df-f10b-45da-b98b-86ae4c7a310b, https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884
[29] Følstad, A., Kvale, K.: Customer journeys: A systematic literature review. Journal of Service Theory and Practice **28**(2), 196–227 (Mar 2018). https://doi.org/10.1108/JSTP-11-2014-0261
[30] Fornell, C., Mithas, S., Morgeson, F.V., Krishnan, M.: Customer Satisfaction and Stock Prices: High Returns, Low Risk. Journal of Marketing **70**(1), 3–14 (Jan 2006). https://doi.org/10.1509/jmkg.70.1.003.qxd
[31] Fredkin, E.: Trie memory. Communications of the ACM **3**(9), 490–499 (1960). https://doi.org/10.1145/367390.367400
[32] Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N.: Explainable predictive process monitoring. In: 2020 2nd International Conference on Process Mining (ICPM). pp. 1–8. IEEE (2020). https://doi.org/10.1109/ICPM49681.2020.00012
[33] Gunther, C.W., Verbeek, E.: XES - Standard Definition. BPM Reports, BPMcenter.org (2014)
[34] Halvorsrud, R., Haugstveit, I.M., Pultier, A.: Evaluation of a modelling language for customer journeys. In: Blackwell, A.F., Plimmer, B., Stapleton, G. (eds.) Proc. Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2016). pp. 40–48. IEEE Computer Society (2016). https://doi.org/10.1109/VLHCC.2016.7739662
[35] Halvorsrud, R., Kvale, K., Følstad, A.: Improving service quality through customer journey analysis. Journal of Service Theory and Practice **26**(6), 840–867 (Nov 2016). https://doi.org/10.1108/JSTP-05-2015-0111
[36] Halvorsrud, R., Mannhardt, F., Johnsen, E.B., Tapia Tarifa, S.L.: Smart journey mining for improved service quality. In: Carminati, B., Chang, C.K., Daminai, E., Deng, S., Tan, W., Wang, Z., Ward, R., Zhang, J. (eds.) Proc. IEEE International Conference on Services Computing (SCC 2021). pp. 367–369. IEEE (2021). https://doi.org/10.1109/SCC53864.2021.00051
[37] Hassani, M., Habets, S.: Predicting next touch point in a customer journey: A use case in telecommunication. In: Proceedings of the 35th International Conference on Modelling and Simulation (ECMS 2021). pp. 48–54. European Council for Modeling and Simulation (2021). https://doi.org/10.7148/2021-0048
[38] Huth, M., Ryan, M.: Logic in Computer Science: Modelling and reasoning about systems. Cambridge university press (2004). https://doi.org/10.1017/CBO9780511810275
[39] Kobialka, P., Mannhardt, F., Tapia Tarifa, S.L., Johnsen, E.B.: Building user journey games from multi-party event logs. In: Montali, M., Senderovich, A., Weidlich, M. (eds.) Proc. Workshop on Event Data and Behavioral Analytics (EdbA 2022). Lecture Notes in Business Information Processing, vol. 468, pp. 71–83. Springer (2023). https://doi.org/10.1007/978-3-031-27815-0_6
[40] Kobialka, P., Schlatte, R., Bergersen, G.R., Johnsen, E.B., Tapia Tarifa, S.L.: Simulating user journeys with active objects. In: Active Object Languages: Current Research Trends, Lecture Notes in Computer Science, vol. 14360, pp. 199–225. Springer (2024). https://doi.org/10.1007/978-3-031-51060-1_8

[41] Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: Weighted games for user journeys. In: Proc. 20th International Conference Software Engineering and Formal Methods (SEFM 2022). Lecture Notes in Computer Science, vol. 13550, pp. 253–270. Springer (2022). https://doi.org/10.1007/978-3-031-17108-6_16
[42] Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: User journey games: Automating user-centric analysis. Software & Systems Modeling **23**(3), 605–624 (2024). https://doi.org/10.1007/s10270-024-01148-2
[43] Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J.: Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. Bus. Inf. Syst. Eng. **63**(3), 261–276 (2021). https://doi.org/10.1007/s12599-020-00645-0
[44] Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. In: International Conference on Inductive Logic Programming. pp. 132–146. Springer (2007). https://doi.org/10.1007/978-3-540-78469-2_16
[45] Lammel, B., Korkut, S., Hinkelmann, K.: Customer experience modelling and analysis framework a semantic lifting approach for analyzing customer experience. In: Proc. 6th Intl. Conf. on Innovation and Entrepreneurship (IE 2016). GSTF (Dec 2016). https://doi.org/10.5176/2251-2039_IE16.10
[46] Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. International Journal on Software Tools for Technology Transfer **1**(1-2), 134–152 (Dec 1997). https://doi.org/10.1007/s100090050010
[47] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I.: From local explanations to global understanding with explainable ai for trees. Nature machine intelligence **2**(1), 56–67 (2020). https://doi.org/10.1038/s42256-019-0138-9
[48] Munoz-Gama, J.: Conformance Checking and Diagnosis in Process Mining - Comparing Observed and Modeled Processes, Lecture Notes in Business Information Processing, vol. 270. Springer (2016). https://doi.org/10.1007/978-3-319-49451-7
[49] Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
[50] Muškardin, E., Aichernig, B.K., Pill, I., Pferscher, A., Tappler, M.: AALpy: an active automata learning library. Innovations in Systems and Software Engineering **18**(3), 417–426 (2022). https://doi.org/10.1007/s11334-022-00449-3
[51] Nguyen, H., Dumas, M., Rosa, M.L., Maggi, F.M., Suriadi, S.: Mining business process deviance: A quest for accuracy. In: Meersman, R., Panetto, H., Dillon, T.S., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T.K. (eds.) On the Move to Meaningful Internet Systems: Proc. Confederated International Conferences CoopIS and ODBASE 2014. Lecture Notes in Computer Science, vol. 8841, pp. 436–445. Springer (2014). https://doi.org/10.1007/978-3-662-45563-0_25
[52] Nielsen, M., Rozenberg, G., Thiagarajan, P.S.: Elementary transition systems and refinement. Acta Informatica **29**(6), 555–578 (1992). https://doi.org/10.1007/BF01185561
[53] Panzera, A.D., Bryant, C.A., Hawkins, F., Goff, R., Napier, A., Schneider, T.,

Kirby, R.S., Coulter, M.L., Sappenfield, W.M., Baldwin, J., et al.: Mapping a wic mother's journey: A preliminary analysis. Social Marketing Quarterly **23**(2), 137–154 (2017). https://doi.org/10.1177/1524500417692526
[54] Peterson, J.L.: Petri Nets. ACM Computing Surveys **9**(3), 223–252 (Sep 1977). https://doi.org/10.1145/356698.356702
[55] Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebraic Methods Program. **60-61**, 17–139 (2004). https://doi.org/10.1016/j.jlap.2004.05.001
[56] Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: Bootstrapping generalization of process models discovered from event data. In: CAiSE. Lecture Notes in Computer Science, vol. 13295, pp. 36–54. Springer (2022). https://doi.org/10.1007/978-3-031-07472-1_3
[57] Razo-Zapata, I.S., Chew, E.K., Proper, E.: VIVA: A visual language to design value co-creation. In: Proc. 20th Conference on Business Informatics (CBI 2018). vol. 01, pp. 20–29. IEEE (Jul 2018). https://doi.org/10.1109/CBI.2018.00012
[58] Rodrigues, A.M.B., Almeida, C.F.P., Saraiva, D.D.G., Spyrides, G.M., Varela, G., Krieger, G.M., Dantas, L.F., Lana, M., Alves, O.E., Franca, R., Neira, A.Q., Gonzalez, S.F., Fernandes, W.P.D., Barbosa, S.D.J., Poggi, M., Lopes, H.: Stairway to value: Mining a loan application process (2017), https://www.win.tue.nl/bpi/2017/bpi2017_winner_academic.pdf
[59] Rosenbaum, M.S., Otalora, M.L., Ramírez, G.C.: How to create a realistic customer journey map. Business Horizons **60**(1), 143–150 (2017). https://doi.org/10.1016/j.bushor.2016.09.010
[60] Rubin, V.A., Mitsyuk, A.A., Lomazova, I.A., van der Aalst, W.M.P.: Process mining can be applied to software too! In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 1–8 (2014). https://doi.org/10.1145/2652524.2652583
[61] Saraeian, S., Shirazi, B.: Process mining-based anomaly detection of additive manufacturing process activities using a game theory modeling approach. Computers & Industrial Engineering **146**, 106584 (2020). https://doi.org/10.1016/j.cie.2020.106584
[62] Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), 379–423 (Jul 1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
[63] Slaats, T., Debois, S., Back, C.O.: Weighing the pros and cons: Process discovery with negative examples. In: International Conference on Business Process Management. pp. 47–64. Springer (2021). https://doi.org/10.1007/978-3-030-85469-0_6
[64] Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(2), 1–57 (2019). https://doi.org/10.1145/3301300
[65] Terragni, A., Hassani, M.: Analyzing customer journey with process mining: From discovery to recommendations. In: Proc. 6th International Conference on Future Internet of Things and Cloud (FiCloud 2018). pp. 224–229. IEEE (Aug 2018).

https://doi.org/10.1109/FiCloud.2018.00040
[66] Terragni, A., Hassani, M.: Optimizing customer journey using process mining and sequence-aware recommendation. In: Proc. 34th Symposium on Applied Computing (SAC 2019). pp. 57–65. ACM Press (Apr 2019). https://doi.org/10.1145/3297280.3297288
[67] Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95). Lecture Notes in Computer Science, vol. 900, pp. 1–13. Springer (1995). https://doi.org/10.1007/3-540-59042-0_57
[68] Thrane, C., Fahrenberg, U., Larsen, K.G.: Quantitative analysis of weighted transition systems. Journal of Logic and Algebraic Programming **79**(7), 689–703 (Oct 2010). https://doi.org/10.1016/j.jlap.2010.07.010
[69] Tueanrat, Y., Papagiannidis, S., Alamanos, E.: Going on a journey: A review of the customer journey literature. Journal of Business Research **125**, 336–353 (Mar 2021). https://doi.org/10.1016/j.jbusres.2020.12.028
[70] Vandermerwe, S., Rada, J.: Servitization of business:adding value by adding services. European Management Journal **6**(4), 314–324 (Dec 1988). https://doi.org/10.1016/0263-2373(88)90033-3
[71] Verwer, S., Hammerschmidt, C.A.: Flexfringe: a passive automaton learning package. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 638–642. IEEE (2017). https://doi.org/10.1109/ICSME.2017.58

# A Decision Boundary States

## A.1 BPIC'17

The process models for BPIC'17-1 and BPIC'17-2 capture a history of the last 3 states as a sequence in each state. The decision boundary of BPIC'17 before July contains the four states:

- A_Validating - O_Returned - W_Call incomplete files SHORT,
- A_Validating - O_Returned - W_Call incomplete files LONG,
- A_Accepted - O_Create Offer 0 - O_Sent (online only), and
- W_Call after offers SHORT - O_Create Offer 1 - O_Sent (online only).

The decision boundary of BPIC'17 after August contains the five states:

- A_Validating - O_Returned - W_Call incomplete files SHORT,
- A_Validating - O_Returned - W_Call incomplete files LONG,
- W_Call after offers SHORT - O_Create Offer 1 - O_Sent (online only),
- O_Sent (mail and online) - O_Create Offer 1 - O_Sent (online only), and
- W_Call incomplete files SHORT - O_Returned - W_Call incomplete files SHORT.

## A.2 BPIC'12

The process model for BPIC'12 captures a history of the last 5 states as a multiset in each state. The decision boundary of BPIC'12 contains the nine states:

- {"A_FINALIZED": 1, "O_CREATED0": 1, "O_SENT_BACK": 1, "W_Nabellen incomplete dossiers#1": 1, "W_Nabellen offertes#0": 1},
- {"O_CANCELLED": 2, "O_CREATED4": 1, "W_Nabellen offertes#0": 1, "W_Nabellen offertes#2": 1},
- {"A_ACCEPTED": 1, "A_FINALIZED": 1, "O_CREATED0": 1, "O_SENT_BACK": 1, "W_Nabellen incomplete dossiers#1": 1},
- {"O_CANCELLED": 1, "O_CREATED2": 1, "O_SENT_BACK": 1, "W_Nabellen incomplete dossiers#0": 1, "W_Nabellen incomplete dossiers#2": 1},
- {"O_CANCELLED": 1, "O_CREATED2": 1, "O_SENT_BACK": 2, "W_Nabellen incomplete dossiers#2": 1},
- {"O_CANCELLED": 2, "O_CREATED5": 1, "W_Nabellen incomplete dossiers#0": 2},
- {"O_CANCELLED": 1, "O_CREATED3": 1, "O_SENT_BACK": 1, "W_Nabellen incomplete dossiers#2": 1, "W_Nabellen offertes#0": 1},
- {"O_CANCELLED": 2, "O_CREATED4": 1, "W_Nabellen offertes#0": 2}, and
- {"O_CANCELLED": 2, "O_CREATED4": 1, "O_SENT_BACK": 1, "W_Nabellen offertes#0": 1}