Building User Journey Games from Multi-party Event Logs *

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

[1] University of Oslo, Oslo, Norway
{paulkob,sltarifa,einarj}@ifi.uio.no
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
f.mannhardt@tue.nl

Abstract. To improve the user experience, service providers may systematically record and analyse user interactions with a service using event logs. User journeys model these interactions from the user's perspective. They can be understood as event logs created by two independent parties, the user and the service provider, both controlling their share of actions. We propose *multi-party event logs* as an extension of event logs with information on the parties, allowing user journeys to be analysed as weighted games between two players. To reduce the size of games for complex user journeys, we identify *decision boundaries* 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 decision boundary analysis from multi-party event logs has been implemented and evaluated on the BPI challenge 2017 event log with promising results, and can be connected to existing process mining pipelines.

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

1 Introduction

In a competitive market, a good user experience is crucial for the survival of service providers [1]. User journeys model the interaction of a user (or customer) with a company's services (service provider) from the user's perspective. One of the earliest works to map user journeys was proposed by Bitner *et al.* in the form of service blueprinting [2]. Current tools can model and analyse individual journeys with the aim to improve services from the customers' point of view [3,4].

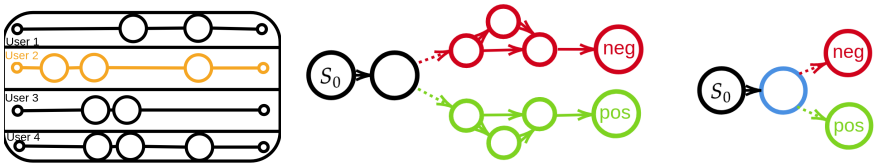
User journey analysis methods based on event data exploit events recording the user interactions with a service and its underlying information systems. Due to the sequential nature of user-service interactions, process mining techniques that assume grouped sequences of events as input, have been used to analyse

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

user journeys [5,6,7,8]. For example, Bernard *et al.* explore and discover journeys from events [7] and Terragni and Hassani use event logs of user journeys to give recommendations [5]. Input events are treated in the same way as for the analysis of business processes: each journey is an instance of a process (case) recorded in a sequence of events (trace) where each event represents an activity occurrence.

In contrast to a 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 one or more service providers. This invites a specific view on the source event log, where some events are controlled by the user and others by service providers. At the end of the journey, some events represent desirable outcomes for the service provider (positive events) whereas others represent undesirable outcomes (negative events). Such partition of the event log into desired and undesired cases or process outcomes has been explored before. *Deviance mining* classifies cases to investigate deviations from expected behaviour [9]. A binary partition of the event log into positive and negative cases was used in, e.g., logic-based process mining [10,11] and error detection [12]. Outcome prediction aims to predict the outcome of a process case based on a partial trace [13,14]. However, these works do not consider the interactions between user and service providers in user journeys as interactions between independent parties. Results of game theory have previously been used by Saraeian and Shirazi for anomaly detection on mined process models [16] and by Galanti *et al.* for explanations in predictive process mining [17]; in contrast to our work, these works do not use game theory to account for multiple independent parties in the process model.

In this paper, we propose a *multi-party* view for user journeys event logs and present a model reduction based on game theory. We have recently shown how to model and analyse a user journey as a two-player weighted game, in a small event log (33 sequences) from a real scenario that could be manually analysed [15]. However, in scenarios with a large number of complex user journeys, the resulting game can be challenging for manual analysis. This paper introduces a k-sequence transition system extension on the directly follows graph of the multi-party game approach presented in [15], and proposes a novel method to automatically detect *decision boundaries* for user journeys. The method can be useful for the analysis of the journeys since it identifies the parts from where the game becomes deterministic with respect to the outcome of the journey, i.e., the service provider has no further influence on the outcome afterwards. We apply our method to the BPIC'17 dataset [18] as an example of complex user-service interactions, which is available in a public dataset. BPIC'17 does not include information on which activities are controlled by the user (a customer is applying for a loan) and which are controlled by the service provider (a bank). We add this information based on domain knowledge and define *multi-party event logs* as an extension of event logs with party information for user journeys. The application on BPIC'17 demonstrates the feasibility and usefulness of our approach. Our results show that we can automatically detect the most critical parts of the game that guarantees successful and, respectively unsuccessful, user journeys.



(a) Log

(b) User Journey Game with Implying Subgraphs.

(c) Decision Boundary with Merged Clusters.

Fig. 1: Construction of the Decision Boundary Reduction.

This analysis could be extended with automated methods targeting predictive and prescriptive analysis, e.g., recommendations for process improvement.

The outline of our paper is illustrated in Fig. 1. Section 2 introduces necessary definitions and summarizes *user journey games*. These are extended with a novel game theoretical reduction method in Sects. 3 and 4. Section 5 illustrates our reduction method and the results on BPIC'17 and Sect. 6 concludes the paper.

2 User Journey Games

This section provides background on our previous work on user journey games [15]. The input to the user journey analysis is an *event log* [19] storing records of observations of interactions between a user and one or more service providers. An *event log* L is a multiset of observed *traces* over a set of actions [19]. Given a universe $\mathscr{A}$ of actions, traces $\tau \in L$ are finite, ordered sequences $\langle a_0, \ldots, a_n \rangle$ with $a_i \in \mathscr{A}$, i.e., $L \in \mathcal{B}(\mathscr{A}^*)$. Given an event log L, we introduce the concept of a *multi-party event log* $\mathcal{L} = \langle L, P, I \rangle$ in which each event belongs to a *party*, where P is the set of parties and the function I extends the traces $\tau \in L$ with information for each event about the initiating party from P.

Transition systems $S = \langle \Gamma, A, E, s_0, T \rangle$ have a set Γ of states, a set A of actions (or labels), a transition relation $E \subseteq \Gamma \times A \times \Gamma$, an initial state $s_0 \in \Gamma$ and a set $T \subseteq \Gamma$ of final states. A *weighted transition system* $\mathcal{S}$ extends a transition system S, with a weight function w indicating the impact of every event [20]. *Weighted games* partition the events and consider them as actions in a weighted transition system, *controllable* actions A_c and *uncontrollable* actions A_u [21]. Only actions in A_c can be controlled. Actions in A_u are decided by an adversarial environment. When analysing games, we look for a *strategy* that guarantees a desired property, i.e. winning the game by reaching a certain state. A strategy is a partial function $\Gamma \to A_c \cup \{\lambda\}$ deciding the actions of the controller in a given state (here, λ denotes the "wait" action, letting the adversary move).

User journeys capture how a user moves through a service by engaging in so-called *touchpoints*, which are either actions performed by the user or a communication event between the user and a service provider [3]. User journeys are inherently goal-oriented. Users engage in a service to reach a goal, e.g. receiving a loan or visiting a doctor. If they reach the goal, the journey is *successful*, otherwise *unsuccessful*. This can be modelled by a transition system with final

states T, and successful goal states from a subset $T_s \subseteq T$: every journey ending in $t \in T_s$ is successful. A journey's success does not only depend on the actions 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. We define *user journey games* as weighted transition systems with goals and self-interested parties [15]:

Definition 1 (User journey games). *A* user journey game *is a weighted game* $G = \langle \Gamma, A_c, A_u, E, s_0, T, T_s, w \rangle$*, where*

- Γ *are states that represent the touchpoints of the user journey,*
- A_c *and* A_u *are disjoint sets of actions respectively initiated by the service provider and the user,*
- $E \subseteq \Gamma \times A_c \cup A_u \times \Gamma$ *are the possible transitions between touchpoints,*
- $s_0 \in \Gamma$ *is an initial state,*
- $T \subseteq \Gamma$ *are the final states of the game,*
- $T_s \subseteq T$ *are the final states in which the game is successful, and*
- $w : E \to \mathbb{R}$ *specifies the weight associated with the different transitions.*

The analysis of services with a large number of users requires a notion of user feedback [3]: Questionnaires provide a viable solution for services with a limited number of users, but not for complex services with many users. In a user journey game, the weight function w denotes the impact that an interaction has on the journey. A user journey game construction is described in [15]. When building user journey games from event logs, we used Shannon entropy [22] together with majority voting to estimate user feedback without human intervention. The more certain the outcome of a journey becomes after an interaction, the higher the weight of the corresponding edge. *Gas* extends weights to (partial) journeys so they can be compared. Given an event log L and its corresponding weighted transition system $\mathcal{S}$, the gas $\mathcal{G}$ of a journey $\tau \in L$ accumulates the weights when replaying τ along the transitions in $\mathcal{S}$, $\mathcal{G}(\tau) \coloneqq \sum_{a_i \in \tau} w(a_i)$.

Formal statements about user journey games can be analysed using a model checker such as Uppaal Stratego [23]. Uppaal Stratego extends the Uppaal system [24] by games and stochastic model checking, allowing properties to be verified up to a confidence level by simulations (avoiding the full state space exploration). If a statement holds, an enforcing strategy is computed. To strengthen the user-focused analysis of user journeys, we assume that an adversarial environment exposes the worst-case behaviour 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 always reaching a successful final state. Define two state properties `positive` for a successful and `negative` for an unsuccessful final state. The keyword `control` indicates a game with an adversarial environment and `A<>` searches for a strategy where the flag `positive` eventually holds at some state in all possible paths of the game:

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

If the strategy `pos` exists, it can be further analysed and refined to, e.g., minimize the number of steps or gas to reach a final state within an upper bound time `T`:

Algorithm 1 Decision Boundary Detection

Input: User journey game $G = \langle \Gamma, A_c, A_u, E, s_0, T, T_s, w \rangle$, unrolling constant n
Output: Decision Boundary $M \subset \Gamma$

1: Assert Termination of Model Checker
2: Initialize mapping $R : \Gamma \to \{\mathbf{True}, \mathbf{False}\}$
3: **for** State $s \in \Gamma$ **do**
4: Game $G' \leftarrow \textsc{Descendants}(s)$
5: Game $G'' \leftarrow \textsc{Acyclic}(G', n)$
6: Update $R(s) \leftarrow \textsc{Query}(G'')$
7: Set $\Gamma_P \leftarrow \{s \in \Gamma \mid \bigwedge R(s') \quad \forall s' \in \textsc{Descendants}(s)\}$
8: Set $\Gamma_N \leftarrow \{s \in \Gamma \mid \bigwedge \neg R(s') \quad \forall s' \in \textsc{Descendants}(s)\}$
9: Add State s_{pos} and s_{neg} to G ▷ States implying outcome
10: **for** State $s \in \Gamma$ **do**
11: **if** $s \in \Gamma_P$ **then** $\textsc{Merge}(G, s_{pos}, s)$
12: **else if** $s \in \Gamma_N$ **then** $\textsc{Merge}(G, s_{neg}, s)$
13: $M \leftarrow \emptyset$
14: **for** State $s \in \Gamma$ **do** ▷ Build decision boundary
15: **if** $\{s_{pos}, s_{neg}\} = \{t \mid t \in (s, t) \in E\}$ **then** $M \leftarrow M \cup \{s\}$
16: **return** M

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

Strategies can be stochastically evaluated using a number of runs `X`, e.g., evaluate the minimal gas of the refined strategy within an upper bound time `T`:

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

3 Decision Boundaries

A decision boundary abstracts a game to focus on crucial parts from where the future outcome is decided. Finding the decision boundary in a complex game can be useful; e.g., there might be no guarantee to find a successful game strategy `pos` (see Sect. 2). Such a strategy can only be found for certain states in the game, which may be scattered around and therefore hard to analyse when using non-automatic methods. Moreover, detecting the decision boundary that lead to outcomes in the game from where there is no possibility of recovery can be used to propose further recommendations for service improvement. Figure 1b and 1c illustrate the game abstraction using decision boundaries. The red and green marked parts of the game in Fig. 1b display guaranteeing areas. Once the journey reached a state within those areas, the outcome becomes deterministic. Since all reachable states from a red or green state share the same outcome, they can be abstracted away (Fig.1c).

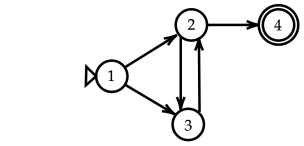
Algorithm 1 computes the decision boundary for a game G. The mapping R, from states s to Boolean, stores whether there exists a successful strategy `pos` that starts from each state $s \in \Gamma$ (Lines 2–6). The algorithm computes a reachable sub-game G' for every state s using the function $\textsc{Descendants}(s)$, which

computes the parts of G which are reachable from s by path exploration. Function $\textsc{Acyclic}(G', n)$ unrolls n times all loops in G', e.g. by a breadth-first search strategy. An example of loop-unrolling in games is displayed in Fig. 2. The resulting acyclic game G'' is then model checked with $\textsc{Query}(G'')$ to look for a successful strategy `pos`. The result is stored in $R(s)$.

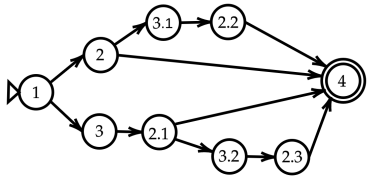
Furthermore, some states are segregated into two sets, Γ_P and Γ_N, based on the results from the previous computation (Lines 7–8). States from which it is only possible to reach positive, respectively negative, results are assigned to Γ_P, respectively Γ_N. States in these sets guarantee the outcome of the game. The game is simplified by abstracting all states in Γ_P, respectively Γ_N, into one state s_{pos}, respectively s_{neg}, using the function Merge (Lines 9–12). Once one of these states is reached, the journey becomes deterministic; the service provider has no further influence on the final outcome. *The states which point to s_{pos} and s_{neg} form the decision boundary* (Lines 13–15).

4 Mining Decision Boundaries

Event logs obtained from user journeys record actions performed by several parties. A user can send messages to a service offered by a service provider and a service provider can send messages to a user currently using the service. It is common practice that artefacts of these actions are recorded in the service provider's event logs, particularly the order of actions. However, knowledge about which party has triggered which action is commonly ignored while collecting such logs.



(a) Cyclic Model.



(b) Acyclic Model.

Fig. 2: Unrolling Example.

In this paper, we approximate multi-party event logs $\mathcal{L}$ by pre-defining a party function I mapping actions a in event log L to a party in $P = \{C, U\}$, where C denotes the service provider and U the user. For simplicity, we assume that different service provider parties are captured by the same party C.

We can now build a user journey game from a multi-party event log following [15] (see Fig. 1 for an overview). We then extend the obtained directly follows graph to a k-sequence transition system $S_L^k = \langle \Gamma, A, E, s_0, T \rangle$ [25], which considers states that record the last k actions happening in the traces of L and stores them in a single state; e.g., the 2-action states of trace $\langle a, b, c \rangle$ are $\{\langle a \rangle, \langle a, b \rangle, \langle b, c \rangle\}$. This abstraction captures more information in the game, improving the precision of the game and the alignment between game and log.

We insert an initial state s_0 at the beginning of each trace $\tau \in L$, and a final state $t \in T$ at the end of each trace $\tau \in L$. Let H denote the set of states corresponding to the k-sequence abstraction for all traces in L, then the states are defined by $\Gamma \subseteq \{s_0\} \cup T \cup H$. The transition relation E is constructed over

adjacent actions in all traces $\tau \in L$. An edge (s_i, a_{i+1}, s_{i+1}) is in E if there is a trace $\tau \in L$ where the last action in state s_i is followed by the last action a_{i+1} in s_{i+1}. A transition with action a_{i+1} in S_L^k, means that the corresponding action has also been performed in τ.

The constructed transition system S_L^k is transformed into a user journey game by computing the weights on the transitions (see Sect. 2), and applying function I to compute the set $A_c = \{a \mid I(a) = C\}$ of actions controlled by the service provider and the set $A_u = \{a \mid I(a) = U\}$ of actions controlled by the user. The user journey game is used to compute its decision boundary (Sect. 3). States behind the decision boundary are merged into *successful* and *unsuccessful* states (Fig. 1c). The result is a strongly reduced game preserving all information on the decision structure.

5 Evaluation on BPIC'17

The BPI Challenge 2017 (BPIC'17) [18] provides an event log recording actions in loan applications from a Dutch financial institute. Since this event log has records of interactions between users and a service provider, including calls, it is a suitable event log for user journey analysis. However, we needed to make assumptions to complete the missing information for our scenario, e.g., which journeys are successful or unsuccessful and infer the party function I with knowledge about which actions are triggered by which party.

The event log contains activities from the following groups: Application (A), Offer (O) and Workflow (W) [26]. 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 cancelled. We define a party function I, based on domain knowledge and official information given in the BPIC'17 forum.[1] We assume that only users can *cancel*, *submit* or *complete* an application, and that users decide whether *calls* take place. We further assume that accepted offers are successful journeys, cancellations are unsuccessful journeys: both parties would prefer a different outcome 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 exclude declined application journeys from the analysis, given their ambiguity.

BPIC'17 is known to include a substantial change in the service provider's process, called a *concept drift* [27], 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.

[1] `https://www.win.tue.nl/promforum/categories/-bpi-challenge-2017`

5.1 User Journey Game Generation

We now report on the generation of the user journey game for the BPIC'17 event log, with focus on the preprocessing of the data. The full implementation is given in the accompanying artefact.[2] We pre-processed BPIC'17 by discretising 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 to hours and several call interactions in one journey consist of repeated adjacent occurrences of events associated to one call. To discretise the duration, we first aggregate repeated and adjacent calls. After the aggregation, we consider calls with 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 cancelled. To simplify journeys, every event associated to an offer or cancellation is ignored after one of the offers is accepted. Offers are automatically cancelled if there is no response after 20 days. We differentiate between actively cancelled 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., *W_Call after offers* is always followed by *A_Complete*, therefore one of them can be removed systematically. We removed outliers and only kept journeys whose variance 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 of Sect. 4. We first generated the transition system S_L^3, with sequence history 3. The party function I and weight w transformed S_L^3 into a user journey game.

5.2 Simulation

Stochastic simulations can help a service provider to evaluate strategies, to guide their users along their services, before implementing them. We evaluated different strategies on the user journey game for BPIC'17 until July, using Uppaal Stratego to learn and compare the outcomes. In the experiments reported in this section, we consider three strategies. In the strategy `max`, the service provider can guide the user through the service by maximizing the final gas, while in the strategy `step` the service provider is minimising the expected number of

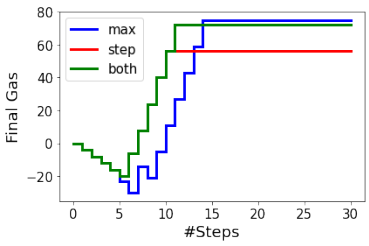
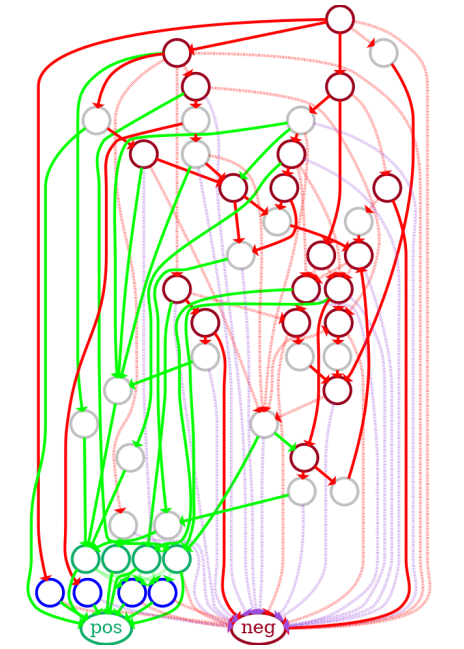


Fig. 3: Simulations under Different Strategies.

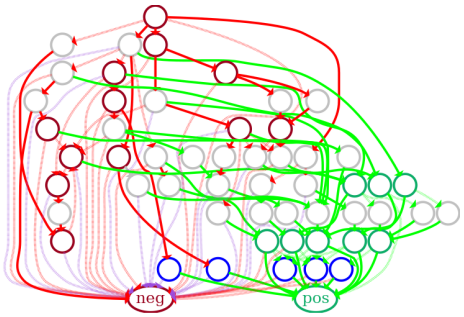
[2] `https://github.com/smartjourneymining/bpi_games/releases/tag/EdbA22`

steps. We also consider a strategy with the combination of both, `both`. Furthermore, we treat the user as controllable to allow a comparative analysis between the strategies, so that all the strategies reach a positive final state.

The simulations in Fig. 3 show the developments of the gas value under 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 customer's perspective, it is not optimal to have negative experiences (negative gas) to complete the service successfully. 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 gas minimum by 33%, but it also reduces the final gas by 25%. The combined strategy `both` maximizes the final gas while minimising 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.



(a) Before July 2016



(b) From August 2016

Fig. 4: Decision Boundaries (Blue) for both BPIC'17 Event Logs.

5.3 Decision Boundary

An exhaustive search over all states revealed the decision boundaries for both BPIC'17 event logs, using the algorithm in Alg. 1. Figure 4a shows the decision boundary for the first part, i.e. until July, and Fig. 4b for the second part. The states *positive* and *negative* incorporate all states with a certain outcome. Blue states mark the decision boundary. Time-out cancellation edges are violet, edges with a positive weight are green and edges with a negative weight are red.

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.

Analysing the decision boundary reveals that most states are negatively biased and have a direct connection to *negative*. With uncontrollable user actions, the service provider has no means to guide the user to a successful outcome, except for a small positive cluster around *positive*. Most positive states require long journeys. A detailed analysis reveals that three out of four states in the decision boundary are related to calls. The action *"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.

The figure reveals many time-out cancellations from various states during the journey, even for paths that are very far into the application process. Such cancellations are unsatisfactory for both the service provider and user, since both parties invested time and resources into the journey and preferred a different outcome. The service provider can draw two action recommendations: reaching the positive outcome should be simplified, thus the decision boundary could be extended, and well-progressed journeys should be increasingly prevented from time-outs, thus reducing the number of time-outs of progressed journeys.

Figure 4b shows the process model for the later data set. The figure shows that the process model changed significantly after the concept drift in Juli 2016. The new decision boundary inherits all states except one and contains one new state. The positioning of the new boundary has improved. The decision boundary improved in two parts: it reaches further into the negative part of the game and increased in size. While the previous decision boundary contained only four states, the updated decision boundary contains five states. The updated decision boundary includes three out of four states of the previous decision boundary and the fourth state lies now before the decision boundary. Additionally, it contains two new states: one was previously in the positive cluster and one new state.

Besides the total number of nodes also the reachability of the decision boundary improved. The number of nodes reaching the decision boundary increased by 22%. The amount of timeouts within advanced journeys is reduced. Customers that continue far into the journey are more prone to finish successfully or to cancel by themselves. The average number of actions from start to time-out reduced from 5.4375 to 5, thus the user journey improved generally.

The service provider can now start to investigate the actions related to states in the decision boundary.

6 Conclusion

This paper proposes a novel view on user journey event logs by introducing multi-party event logs that differentiate between the actors of actions leading to events. To promote a user-centric view, the service provider is modelled as controllable and the user as uncontrollable. Based on such a multi-party event log, we show how to use user journey games to model the interaction between user and service provider, and use model checking to find strategies with guaranteed outcomes.

We introduce an analysis to identify *decision boundaries*; these constitute a crucial part of a game at which the outcome of the user journey is determined.

Decision boundaries are useful since strategies that guarantee a positive outcome for all paths are unlikely in complex user journeys. The decision boundary additionally serves to reduce the size of the game. This enables us to apply the user journey game approach to the BPIC'17 dataset, which is a real-life event log of a complex user journey that can be transformed to a multi-party event log. The decision boundary gives clear insights into determining factors for the BPIC'17 user journey before and after a concept drift. Our analysis reveals the changes done in the workflow and demonstrates the support and applicability for further analysis through our method, assuming a transformation of the BPIC'17 event log into a multi-party event log, and assuming that users actually have an influence on their journey through their active decisions.

User journey games and decision boundary analysis open many interesting directions for future work. We plan to combine user journey games with well-established process mining tools to discover process models for behaviour leading to determining states. Furthermore, we would like to automate recommendations for improvement, based on the decision boundary. While the decision boundary is helpful for analysing the interaction between a user and service providers, the analysis is still hand-made. We also plan to generalise the approach to cyclic models to make it agnostic to the current unrolling bound n in each cycle. Furthermore, we would like to investigate probabilistic games to capture ambiguities within user actions. Finally, we would like to implement a multi-party event log in cooperation with companies to study real interactions between user and service provider.

References

1. 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)
2. Bitner, M.J., Ostrom, A.L., Morgan, F.N.: Service blueprinting: A practical technique for service innovation. California Management Review **50**(3), 66–94 (2008)
3. Rosenbaum, M.S., Otalora, M.L., Ramírez, G.C.: How to create a realistic customer journey map. Business Horizons **60**(1), pp. 143–150 (2017)
4. 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)
5. Terragni, A., Hassani, M.: Optimizing customer journey using process mining and sequence-aware recommendation. In: Proc. 34th Symposium on Applied Computing (SAC 2019), ACM Press, pp. 57–65 (Apr 2019)
6. Bernard, G., Andritsos, P.: Contextual and behavioral customer journey discovery using a genetic approach. In: Proc. 23rd Eur. Conf. on Advances in Databases and Information Systems (ADBIS 2019). LNCS 11695, pp. 251–266. Springer (2019)
7. Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. In: 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),
8. Hassani, M., Habets, S.: Predicting next touch point in a customer journey: A use case in telecommunication. In: Eur. Conf. on Modeling and Simulation (2021)

9. Nguyen, H., Dumas, M., Rosa, M.L., Maggi, F.M., Suriadi, S.: Mining business process deviance: a quest for accuracy. In: OTM Confederated Intl. Conferences, LNCS 8841, Springer, pp. 436–445 (2014)
10. Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. In: Intl. Conf. on Inductive Logic Programming, Springer, pp. 132–146 (2007)
11. Bellodi, E., Riguzzi, F., Lamma, E.: Probabilistic declarative process mining. In: Intl. Conf. on Knowledge Science, Engineering and Management, Springer (2010)
12. Rubin, V.A., Mitsyuk, A.A., Lomazova, I.A., van der Aalst, W.M.: Process mining can be applied to software too! In: Proceedings of the 8th ACM/IEEE Intl. Symposium on Empirical Software Engineering and Measurement, pp. 1–8 (2014)
13. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), pp. 17:1–17:57 (2019)
14. 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) (2021)
15. Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: Weighted games for user journeys. In: Proc. SEFM 2022. LNCS 13550, pp. 253–270. Springer (2022).
16. Saraeian, S., Shirazi, B.: Process mining-based anomaly detection of additive manufacturing process activities using a game theory modeling approach. Computers & Industrial Engineering **146** (2020) 106584
17. Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N.: Explainable predictive process monitoring. In: Proc. ICPM, IEEE, 1–8 (2020)
18. van Dongen, B.: BPI Challenge 2017 (2017) https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
19. van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer (2016)
20. Thrane, C., Fahrenberg, U., Larsen, K.G.: Quantitative analysis of weighted transition systems. J. Logic and Algebraic Programming **79**(7), pp. 689–703 (Oct 2010)
21. Bouyer, P., Cassez, F., Fleury, E., Larsen, K.G.: Optimal strategies in priced timed game automata. In: Proc. FSTTCS. LNCS 3328, pp. 148–160. Springer (2004)
22. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), pp. 379–423 (July 1948)
23. David, A., Jensen, P.G., Larsen, K.G., Mikučionis, M., Taankvist, J.H.: Uppaal Stratego. In: Proc. TACAS 2015. LNCS 9035, pp. 206–211. Springer (2015)
24. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. Intl. Journal on Software Tools for Technology Transfer **1**(1-2), pp. 134–152 (Dec 1997)
25. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: A two-step approach to balance between underfitting and overfitting. Software & Sys. Modeling **9**(1), pp. 87–111 (Jan 2010)
26. 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 `https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2017:bpi2017_winner_academic.pdf`
27. Adams, J.N., Zelst, S.J.v., Quack, L., Hausmann, K., van der Aalst, W.M., Rose, T.: A framework for explainable concept drift detection in process mining. In: Intl. Conf. on Business Process Management. Springer, pp. 400–416 (2021)
28. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson (2020)