Stochastic Games for User Journeys [⋆]

Paul Kobialka (✉)[1], Andrea Pferscher[1], Gunnar R. Bergersen[1,2], Einar Broch Johnsen[1], and S. Lizeth Tapia Tarifa[1]

[1] University of Oslo, Oslo, Norway
`{paulkob,andreapf,gunnab,einarj,sltarifa}@ifi.uio.no`
[2] GrepS B.V., Utrecht, the Netherlands
`gunnar.bergersen@greps.com`

Abstract. Industry is shifting towards service-based business models, for which user satisfaction is crucial. User satisfaction can be analyzed with user journeys, which model services from the user's perspective. Today, these models are created manually and lack both formalization and tool-supported analysis. This limits their applicability to complex services with many users. Our goal is to overcome these limitations by automated model generation and formal analyses, enabling the analysis of user journeys for complex services and thousands of users. In this paper, we use stochastic games to model and analyze user journeys. Stochastic games can be automatically constructed from event logs and model checked to, e.g., identify interactions that most effectively help users reach their goal. Since the learned models may get large, we use property-preserving model reduction to visualize users' pain points to convey information to business stakeholders. The applicability of the proposed method is here demonstrated on two complementary case studies.

Keywords: User journeys · Data-driven model construction · Automata learning · Model checking · Stochastic games · PRISM

1 Introduction

The *servitization of business* describes a shift towards offering products as services [44]. This shift makes companies more dependent on user satisfaction; e.g., it has become much easier to change service providers. Investment in user satisfaction pays off [17], which raises the following question: How can we formally model and analyze the way users experience their interaction with a service?

User journeys model services from the users' perspective [41]. They describe how users employ a service to achieve a goal. User journeys may include many paths, capturing different sequences of actions between a service and its users. These models enable the analysis of user experience along different (intended or unintended) paths through a service. Although most user journeys today are created manually by domain experts and the associated user experience is captured through interviews [22, 41], the method has been successful at providing

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

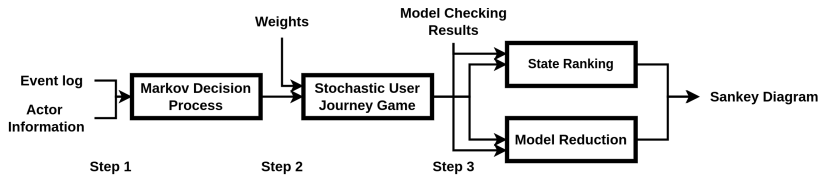


Fig. 1: Steps to create Sankey diagrams from the event logs of the case studies.

feedback to improve services. However, tool support for the modeling and analysis of user journeys is sparse [23], which makes the method difficult to apply in complex domains and to services with numerous and diverse users.

A recent line of work aims to automatically mine user journeys and analyze them using formal methods [26,28,30,31]. This significantly reduces the manual effort needed to create models and enables a different scale of complexity in the analyzed services and number of users. Starting from event logs, which are widely available for software services, *process mining* [1] and *automata learning* [18] can automatically generate behavioral models of user journeys from these logs, such as finite state automata. These can then be analyzed by *model checking* [4].

This paper goes beyond previous work by modeling user journeys as *stochastic games* [11]. We exploit the underlying distribution of events in the event log, which was ignored in previous work. Stochastic games allow complex user behavior to be captured, yet the resulting games can still be model checked. Figure 1 summarizes the steps applied to event logs to analyze user experience. These steps elegantly combine and extend several known techniques. *Step 1* generates stochastic automata from event logs by means of automata learning. *Step 2* converts the learned automata into stochastic weighted games. The resulting games are analyzed using probabilistic model checking to derive optimal strategies. *Step 3* ranks critical actions after which users tend to abandon their journey and visualizes the outcome of these novel analyses via a property-preserving visualization technique, to improve the interpretability of the stochastic game results.

We apply these steps to two case studies: an industrial case study [30,31] and a benchmark [15] from the literature. The case studies are complementary in complexity and differ in the number of users. In both cases, we identified potential service improvements and automatically uncovered caveats. The case studies suggest that our method is able to address two pressing industrial challenges: (1) the automated construction of stochastic user journey models for complex services from event logs, and (2) identification of service bottlenecks by automated analysis of models that reflect user experience. In short, the contributions of this paper are: (1) a formalization of user journeys as stochastic weighted games exploiting the underlying distribution of events in the logs; (2) a tool chain combining automata learning and model-checking techniques to automatically analyze stochastic user journey games; (3) a method for property-preserving model reduction to visualize the stochastic games results; and (4) the automated stochastic modeling and analysis of two case studies to showcase the usefulness and applicability of the proposed combination of techniques and their extensions.

2 Preliminaries

In the following, we write $\mathcal{D}(X)$ for the set of probability distributions over a set X, where a distribution $\mu\colon X \to [0,1]$ is such that $\sum_{x\in X}\mu(x) = 1$.

Event Logs. An event log records so-called touchpoints (or events) between users and a service provider. A *trace* $\tau = (a_0, \ldots, a_n) \in \mathscr{A}^*$ is a finite, ordered sequence over an alphabet $\mathscr{A}$ of events. An *event log* L is a multi-set of such traces [1]. A *multi-actor event log* $\mathcal{L} = \langle L, \Pi, \alpha\rangle$ assigns an initiating actor to each event in an event log L [26]; the set Π contains a set of actors, and the actor-mapping function $\alpha\colon \mathscr{A} \to \Pi$ assigns events $a \in \mathscr{A}$ to an actor $\pi \in \Pi$.

Automata Learning. To learn stochastic automata from event logs, we use the passive automata learning algorithm IOAlergia [36]. IOAlergia learns stochastic automata for reactive systems defined by MDPs [36], based on Alergia [10]. State merging exploits the underlying probabilities of events in the log. An MDP is a tuple $\langle \Gamma, A_{\text{in}}, A_{\text{out}}, \delta, s_0, \lambda\rangle$ with finite sets of states Γ, input actions A_{in} and output actions A_{out}, a stochastic transition function $\delta\colon \Gamma \times A_{\text{in}} \to \mathcal{D}(\Gamma)$, an initial state $s_0 \in \Gamma$, and a labeling function $\lambda\colon \Gamma \to A_{\text{out}}$. We let $E_\delta \subseteq \Gamma \times A_{\text{in}} \times \Gamma$ denote the finite set of transitions such that $\delta(s,a)(s') > 0$ for all triples $(s,a,s') \in E_\delta$. We assume MDPs to be deterministic; i.e., $s' = s''$ holds for all transitions $\delta(s,a)(s'), \delta(s,a)(s'')$ such that $\delta(s,a)(s') > 0$, $\delta(s,a)(s'') > 0$ and $\lambda(s') = \lambda(s'')$.

Let an input/output log L_{io} consist of traces $\tau_{\text{io}} = (\lambda(s_0), (i_0, o_0), \ldots, (i_n, o_n))$ in which input and output actions alternate, starting with an initial output $\lambda(s_0)$, which is only observed in the initial state. Given L_{io}, IOAlergia creates an input/output frequency prefix tree acceptor (IOFPTA), where states are labeled with output actions and transitions with input actions and frequencies. In the IOFPTA, every path in the tree represents a prefix of a trace in $\tau_{\text{io}} \in L_{\text{io}}$, and the frequency denotes the number of traces sharing this path. After creating the IOFPTA, IOAlergia merges states. Two states are merged if they (1) have the same output label, (2) are locally compatible, and (3) all their successor states with the same output labels are compatible. Local compatibility is based on the Hoeffding bound [25]: two states s, s' are compatible if, for all inputs $i \in A_{\text{in}}$,

$$\left|\frac{f(s,i,o)}{n(s,i)} - \frac{f(s',i,o)}{n(s',i)}\right| \leq \sqrt{\frac{1}{2}\log\frac{2}{\epsilon}}\left(\frac{1}{\sqrt{n(s,i)}} + \frac{1}{\sqrt{n(s',i)}}\right),$$

where $f(s,i,o)$ is the frequency of the transition to state o and $n(s,i)$ the sum of frequencies, for input i in state s. The parameter $\epsilon \in (0,2]$ steers the algorithm's eagerness for state merging; e.g., $\epsilon = 2$ leads to no state merges. Therefore, the MDP might contain several states representing the same event. When no states can be merged, the transition frequencies are normalized to create an MDP.

User Journey Games. A *user journey game* [30, 31] is a weighted two-player game $\langle \Gamma, A_C, A_U, E, s_0, T, T_s, w\rangle$, where Γ is a finite set of states, A_C and A_U are disjoint sets of actions, $E \subseteq \Gamma \times A_c \cup A_U \times \Gamma$ is a transition relation, $s_0 \in \Gamma$ an initial state, $T \subseteq \Gamma$ a set of final states, $T_s \subseteq T$ successful final states, and $w : E \to \mathbb{R}$ a weight function. Actions are separated into two disjoint sets: *controllable actions* A_C are taken by the service provider and *uncontrollable actions* A_U by the user. User journey games are *deterministic* if $s' = s''$ for $(s, a, s'), (s, a, s'') \in E$. Uncontrollable actions have higher precedence than controllable actions: hence, the user chooses actions first but might do nothing.

A *stochastic multi-player game* (SMG) [11] is a tuple $\langle \Pi, \Gamma, A, (\Gamma_i)_{i \in \Pi}, s_0, \delta \rangle$, where Π is a set of players, Γ a set of states, A a finite set of actions, $(\Gamma_i)_{i \in \Pi}$ a partition of states among players, $s_0 \in \Gamma$ an initial state, and $\delta : \Gamma \times A \to \mathcal{D}(\Gamma)$ a stochastic transition function. SMGs partition the states among the players; players can take enabled actions if the current state is in their partition. An action $a \in A$ is *enabled* in a state s if there is a transition to another state with non-zero probability, i.e., $\exists s' \in \Gamma : \delta(s, a)(s') > 0$. The set of transitions E_δ defined by δ includes all triples $(s, a, s') \in \Gamma \times A \times \Gamma$ with $\delta(s, a)(s') > 0$. Games can include a reward structure $r : E_\delta \to \mathbb{Q}_{\geq 0}$ mapping transitions to positive rewards (modeling weighted transitions). Rewards accumulate during the game.

Analyzing Stochastic Multiplayer Games. We are interested in analyzing a player's *strategy*, which determines the player's actions in each state. For simplicity, we focus on *memory-less* strategies, where the choice of action is determined by the current state. A *strategy* [11] for player $i \in \Pi$ in an SMG is a partial function $\Gamma_i \to \mathcal{D}(A)$ that maps states to distributions over actions.

PRISM-games [11, 32] extends the probabilistic model checker PRISM [34] to games. While PRISM can resolve non-determinism to establish strategies for a single player, PRISM-games can resolve nondeterminism for multiple, possibly competing players. The logic *Probabilistic Alternating-time Temporal Logic with Rewards* (rPATL) allows reasoning about SMGs by expressing temporal properties [11]. The syntax of rPATL is given by:

$$\phi \coloneqq \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle \Xi \rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle \Xi \rangle\rangle R^r_{\bowtie \chi}[\mathbf{F}^* \phi] \mid \langle\langle \Xi \rangle\rangle P_{\blacktriangleright\!\blacktriangleleft}[\psi] \mid \langle\langle \Xi \rangle\rangle R^r_{\blacktriangleright\!\blacktriangleleft}[\mathbf{F}^* \phi]$$
$$\psi \coloneqq \mathbf{X}\phi \mid \phi \mathbf{U}^{\leq k} \phi \mid \phi \mathbf{U} \phi$$

rPATL is a CTL-style branching-time temporal logic that extends state properties ϕ to path formula ψ with probabilistic and reward constraints. Here, p is an atomic proposition. The *coalition* operator $\langle\langle \Xi \rangle\rangle$ denotes the subset $\Xi \subseteq \Pi$ of players that collaborate in a query; these players share a common goal against the remaining adversarial players. The *probabilistic* operator $P_{\bowtie q}$, where $\bowtie \in \{<, \leq, \geq, >\}$ is a comparison operator and $q \in \mathbb{Q} \cap [0, 1]$ is a probability bound, indicates a probabilistic query under bound $\bowtie q$. The *expected cumulative reward* operator $R^r_{\bowtie \chi}$ evaluates the reward structure r for eventually reaching ϕ under bound $\bowtie \chi$, where $\chi \in \mathbb{Q}_{\geq 0}$ is a reward bound and r is a reward structure. The *quantitative operators* $P_{\blacktriangleright\!\blacktriangleleft}$ and $R^r_{\blacktriangleright\!\blacktriangleleft}$, with $\blacktriangleright\!\blacktriangleleft \in \{\min{=}?, \max{=}?\}$, return the smallest, respectively largest, value that the given coalition of players Ξ can enforce. The superscript $*$ of the *eventually* operator $\mathbf{F}$ expresses the cost for paths when ϕ is not reached, it may be infinity (∞), zero ($\mathbf{0}$), or accumulated along the path ($\mathbf{c}$). Further temporal logic operators can be constructed from the next operator $\mathbf{X}$, the until operator $\mathbf{U}$, and the bounded until operator $\mathbf{U}^{\leq k}$; for example, the *globally* operator $\mathbf{G}\phi$ is defined via $\mathbf{U}$: $\neg(\top \mathbf{U} \neg\phi)$ [11].

3 Case Study Overview

We conduct two complementary case studies: an industrial application (*GrepS*) and a research benchmark (*BPIC'17*). We explain the steps of our method on GrepS. BPIC'17 includes thousands of journeys and demonstrates scalability.

GrepS. The company GrepS offers programming skill evaluations for Java [6]. The customers of GrepS are organizations that use the service in the hiring process to identify proficient applicants. *Users* of the service, the assessed trainees, usually complete the assessment within 1–2 weeks. The service comprises three phases: (1) sign up, (2) solve all programming tasks, and (3) review and share the skill report with the customer. In a *successful* journey, the user completes all tasks and shares the results with the organization. Otherwise, the journey is *unsuccessful*. The event log contains *anonymized user logs* as tabular data [29]. To construct multi-actor event logs, the actor-mapping function α was detailed by combining domain knowledge and interaction with a GrepS developer.

BPIC'17. The BPI Challenge 2017 captures a loan application process from a bank. Users can *cancel*, *submit* or *complete* applications, and accept phone *calls* from the bank. The process can have three different outcomes: (1) an offer can be accepted by the user, (2) the application can be declined by the bank, or (3) the application can be canceled by the user. We exclude declined applications as they occur due to external factors, e.g., indebtedness. Thus, user journeys are successful if the user accepts one of the provided loan offers; cancellations are unsuccessful. The event log contains anonymized user logs as tabular data [15]. To construct multi-actor event logs, the actor-mapping function α was detailed by combining domain knowledge with information given in the BPIC'17 forum.[3]

Interestingly, BPIC'17 contains a substantial change in the service provider's underlying process, a *concept drift* [2]. To investigate the impact of the concept drift on the user journey, we split the log: The first part (*BPIC'17-1*) contains traces until the change occurred in July 2016, and the second part (*BPIC'17-2*) contains the traces after the change.

The BPIC'17 event log is preprocessed to clear inconsistencies [26,40]. Specifically, we discretized call durations: A trace might contain several events associated with one call, and calls ranging from seconds to hours. Thus, we aggregate repeated calls and classify them by their duration into "short", "long", or "super long". We exclude calls with an aggregated speaking time of less than 60 seconds. We also distinguish different offers within the same trace. The service provider cancels offers if there is no response after 20 days. We distinguish actively canceled offers and cancellations by the service provider due to timeout. We also found some redundant events; e.g., the event *W_ Call after offers* was always followed by *A_ Complete*, so we merged these events. To remove outliers we kept only traces that appear more than once in the log; in the end, both logs still contain more than 5000 journeys.

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

4 From Logs to Stochastic Games

We explain how stochastic user journey games are constructed from multi-actor event logs $\mathcal{L} = \langle L, \Pi, \alpha \rangle$, i.e., the first two steps in Fig. 1. Step 1 generates an MDP M from the multi-actor event log $\mathcal{L}$. Step 2 constructs a weighted stochastic game, extending M with weights and actor information. These *stochastic user journey games* combine user journey games and SMGs (see Sect. 2).

In a multi-actor event log $\mathcal{L}$, the set of actors Π is assumed to include the *service provider* C, who initiates all actions controlled by the offering company, and the *user* U, who initiates all remaining actions. We assume that users engage in only one action at a time; hence, our focus here will be on turn-based games as models for user journeys, and not on models with parallelism.

Step 1. We first learn an MDP $M = \langle \Gamma, A_{\text{in}}, A_{\text{out}}, \delta, s_0, \lambda \rangle$ with IOAlergia. For the construction of M, we make sure that the traces $\tau \in L$ are in the required format of input/output pairs by extending each trace $\tau = (a_0, \dots, a_n)$ to an input/output trace $\tau_{\text{IO}} = (\lambda(s_0), (\mathsf{env}, \lambda(s_0)^{\alpha(a_0)}), (act(a_0), a_0), \dots, (\mathsf{env}, a_{n-1}^{\alpha(a_n)}),$ $(act(a_n), a_n), (\mathsf{env}, a_n^{\alpha(res)}), (act(res), res))$. Each $a_i \in \tau$ is encoded by a pair $(\mathsf{env}, a_{i-1}^{\alpha(a_i)})$ where env is a generic input action indicating the next player, followed by an output action $a_{i-1}^{\alpha(a_i)}$ that indicates the player who initiates event a_i from a_{i-1} according to the actor-mapping function α. This pair is followed by a pair $(act(a_i), a_i)$, which uses a function $act\colon \mathscr{A} \to A_{\text{in}}$ to map events to input actions, where the output action corresponds to the event itself. A naive mapping could be $act(a_i) = a_i$, relating each event to a deterministic action. However, it is often useful to introduce a mapping that abstracts slightly from the events to better reflect the problem domain in the actions. Each τ_{IO} starts with an initial output $\lambda(s_0)$ and ends with a final output res, which is $\mathsf{successful}$ if τ records a successful user journey and $\mathsf{unsuccessful}$ otherwise. This resulting set of input/output traces is given to IOAlergia (see Sect. 2). By including input/output pairs $(\mathsf{env}, a_{i-1}^{\alpha(a_i)})$ in the traces, the learned MDP provides the probability distribution for the actions of the next player.

Step 2. The MDP M obtained in Step 1 is extended to a stochastic user journey game by means of a weight function $w : E_\delta \to \mathbb{R}$, labeling transitions with weights, and partitioning the states Γ into service provider states Γ_C and user states Γ_U. For the automatic construction of the weight function w, we exploit the distinction between successful and unsuccessful user journeys in the event log to compute a numerical value that represents the impact of an action on the outcome of the user journey. The calculation of w is based on previous work [30, 31]. For every transition $e \in E_\delta$, we let $w(e) = (1 - H(e, L)) \cdot majority(e, L)$, where H is the entropy of successful and unsuccessful journeys. The weight is positive if the majority of traversals are successful journeys, otherwise negative. The weight is maximal, respectively minimal, for transitions occurring exclusively in successful, respectively unsuccessful, journeys. The accumulated weight along a path in a user journey game, called *gas*, then represents the user's "motivation" to continue the journey [30, 31].

Table 1: Model checking queries for SUJGs.

Name	Query	Description
Q1	$\langle\langle C \rangle\rangle P_{\max=?}[\mathbf{F}\,\mathsf{successful}]$	Probability of a successful journey
Q2	$\langle\langle C, U \rangle\rangle R^{\text{NEG}}_{\min=?}[\mathbf{F}\,\mathsf{successful} \mid \mathsf{unsuccessful}]$	Boundaries for accumulated positive and negative rewards
Q3	$\langle\langle C \rangle\rangle R^{\text{NEG}}_{\min=?}[\mathbf{F}\,\mathsf{successful} \mid \mathsf{unsuccessful}]$	
Q4	$\langle\langle C \rangle\rangle R^{\text{POS}}_{\max=?}[\mathbf{F}\,\mathsf{successful} \mid \mathsf{unsuccessful}]$	
Q5	$\langle\langle C \rangle\rangle R^{\text{NEG}}_{\min=?}[\mathbf{C}_{\leq S}]$	Step bounded reward
Q6	$\langle\langle C \rangle\rangle R^{\text{POS}}_{\max=?}[\mathbf{C}_{\leq S}]$	
Q7	$\langle\langle C \rangle\rangle R^{r}_{\max=?}[\mathbf{F^c}\,\mathsf{successful}]\ r \in \{\text{NEG}, \text{POS}, \text{STEPS}\}$	Expectation of reward structures
Q8	$\langle\langle C \rangle\rangle P_{\max=?}[(\mathbf{F}\,\mathsf{successful}\ \&\ \text{gas} \geq G_0\ \&\ \text{steps} \leq S)\ \&\ (\mathbf{G}\,\text{gas} \geq G1)]$	Constrained success probability

The controllable and uncontrollable states are identified using the actor-mapping function α to map states to the actors C (service provider) and U (user); e.g., the set of states in Γ_C corresponds to the copies of output actions where C controls the next action: $a_{i-1}^{\alpha(a_i)}$, where $\alpha(a_i) = C$. Then $\Gamma_C = \{s \in \Gamma \mid \exists a \in A_{\text{out}} : \lambda(s) = a^C\}$, and $\Gamma_U = \{s \in \Gamma \mid \exists a \in A_{\text{out}} : \lambda(s) = a^U \vee \lambda(s) = a\}$.

The weight function w and the state partitioning allows the MDP to be transformed into a weighted, two-player SMG, hereafter called a *stochastic user journey game (SUJG)*, i.e., a tuple $G = \langle \{C, U\}, \Gamma, A_{\text{in}}, (\Gamma_i)_{i \in \{C,U\}}, s_0, \delta, T, T_s, w \rangle$, where final states $T = \{s \in \Gamma \mid \lambda(s) = \mathsf{successful} \vee \lambda(s) = \mathsf{unsuccessful}\}$, successful final states $T_s = \{s \in \Gamma \mid \lambda(s) = \mathsf{successful}\}$, and w the weight function. Note that every user journey game can be transformed into an equivalent SUJG.

5 Queries for Stochastic User Journey Games

We here assume that users do not interact infinitely with a service provider but eventually stop. Therefore, we consider SUJGs to be *stopping* games, in which we reach almost surely terminal states with reward zero [33].

Step 3. We now consider the probabilistic model checking of properties that are crucial for the success of user journeys. The violation of these properties allows us to locate problematic states where the user journey may be improved. The constructed SUJG may contain loops with a positive or negative sum of weights. For this reason, we distinguish queries applicable to games with *reward structures* and with *bounded integer encodings*. Table 1 lists properties that we analyzed for the case studies, and that we discuss below. The queries are specified in rPATL, where C denotes the service provider and U denotes the user.

Let us first analyze the probability of completing a user journey successfully; i.e., to what extent can service provider C guarantee the successful outcome of the game? Query Q1 quantifies the service provider's ability to guide an independent user. Searching for states that return a small probability of reaching any $s \in T_s$ uncovers states from which the service provider has little or no probability of successfully guiding the user. Thus, the journey is likely to fail. Here, $\mathsf{successful}$ is a predicate that only holds in the successful final state T_s, and $\mathsf{unsuccessful}$ is a predicate that holds in the final states $T \setminus T_s$.

Reward Structures decouple accumulated rewards from the state space in PRISM-games and allow efficient computation of accumulated rewards. In turn-based SMGs, PRISM-games only supports positive rewards. Thus, we use two reward structures: POS for positive and NEG for negative gas (see Sect. 4). The weight of a transition in the SUJG contributes to the corresponding structure, i.e., positive weights add to POS, and negative weights add to NEG. Many services contain transitions with negative weights, e.g., reflecting actions that may be unintuitive for the user. To analyze the effect of these transitions, we consider queries concerning the user experience. Query Q2 determines the lower bound for the negative reward that the user must accumulate to achieve any outcome, by assuming that both actors cooperate. Queries Q3 and Q4 determine the minimum NEG and maximum POS reward that the service provider can guarantee, independent of the user, over successful and unsuccessful journeys, respectively. Rewards can also be used to relate gas to the number of steps taken so far: Queries Q5 and Q6 return the minimum negative or maximum positive accumulated reward (denoted $\mathbf{C}$) within the first S steps that C can guarantee.

Bounded Integer Encodings combine positive and negative weights in one variable, enabling queries on their difference. Every transition changes the value of this variable by the corresponding positive or negative weight, reflecting the gas along the paths in the game (see Sect. 4). We also consider a step counter that is updated for each transition. To restrict the size of the search space, we give this variable a bound (i.e., $\text{steps} \coloneqq \min(\text{steps}+1, X)$ for some X). We then use concentration inequalities such as *Markov's inequality* and cumulative reward structures to calculate the expected values of POS, NEG, and STEPS in Q7, and derive upper and lower bounds that include at least a minimum part of the distribution. Note that this construction is only needed in the presence of loops and that the *expected total rewards*, used to bound the model, are finite as we assume stopping games. Query Q8 determines the service provider's probability for a successful journey with a minimum amount of gas along the path, a maximum amount of steps, and an overall lower bound for the gas. This multi-objective query searches for a successful final state where gas $\geq G_0$ and steps $\leq S$, while ensuring that gas never decreases below G_1, for constants G_0, S, G_1.

Experiments. PRISM-games supports *experiments* on queries that instantiate a variable, e.g., the maximum number of steps, with all values in a given integer interval. We use experiments to compare different values of player activity by modifying the probabilities for the service provider or user to take their actions first. Additionally, we vary the allowed number of steps to investigate how the probabilities of a successful outcome change with a limited number of steps.

6 Model Reduction for Visualization

Model checking may reveal weaknesses in the service design and unsatisfiable queries may suggest a need for changes. However, an unsatisfiable query does not by itself identify the actions that negatively affect the largest number of users.

To help prioritize options during service redesign, we rank actions based on their expected influence on the user journey outcome, to identify the most critical actions for the largest number of users (cf. Step 3, Fig. 1). We synthesize strategies maximizing the probability of a successful outcome by returning a maximizing strategy for the service provider and a minimizing strategy for the user, based on the queries in Sect. 5. These strategies resolve the players' choice of action in the SUJG via an induced Markov chain $M' = \langle \Gamma', \delta', s_0 \rangle$; the states Γ' of M' form a, possibly smaller, subset of the states Γ of the original SUJG, i.e., $\Gamma' \subseteq \Gamma$. (The construction of the induced Markov chain M' from an SMG is detailed in [12].)

We say that users are *guidable* if the probability that they can successfully complete the journey is greater than zero. Let the function $\mathscr{R} : \Gamma' \to [0, 1]$ map states $s \in \Gamma'$ to the (intermediate) results of the probabilistic query Q1, expressing the probability of reaching the successful outcome from s. The difference in guidable users between two neighboring states s and s' is the absolute difference between $\mathscr{R}(s)$ and $\mathscr{R}(s')$, multiplied by the users traversing between these states. Formally, the *difference* $\mathrm{diff} \colon \Gamma' \to \mathbb{R}$ in state $s \in \Gamma'$ is the absolute difference in guidable users between s and all neighboring states s':

$$\mathrm{diff}(s) = \sum_{s' \in \Gamma'} |\mathscr{R}(s) - \mathscr{R}(s')| \cdot \#_{\mathcal{L}}^{\Gamma'}(s, s') \ . \tag{1}$$

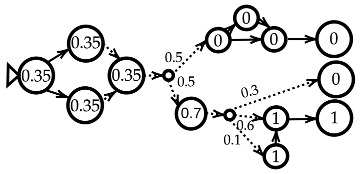
Here, $\#_{\mathcal{L}}^{\Gamma'}(s, s')$ denotes the number of users traversing from s to s' as recorded in the log $\mathcal{L}$, where $s' \in \Gamma'$ and $\delta'(s, s') > 0$. For non-neighboring states, let $\#_{\mathcal{L}}^{\Gamma'}(s, s') = 0$. States can then be ranked in descending order by their difference.

Visualizations of Results. Real-world processes with complex structures and many users result in models that might be hard for humans to interpret correctly. We discuss a model visualization method based on the model-checking results that allows model reduction while preserving the ranking order.

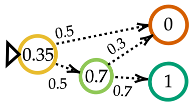
The state space of M' can be abstracted into clusters of states with an equal probability of success as defined by $\mathscr{R}$. Neighboring states with the same results can be merged. States $\{s' \in \Gamma' \mid (s, s') \in E_{\delta'} \wedge \mathscr{R}(s) = \mathscr{R}(s')\}$ can be merged into a state s. We also merge successful final states $T_s \cap \Gamma'$ and unsuccessful final states $(T \setminus T_s) \cap \Gamma'$. Note that the reduced model preserves all transitions to states that negatively impact the user journey, and that the merge operation is commutative.

To visualize fluctuations in guidable users along the user journey, we transform the reduced model into a *Sankey* diagram [39]. We opted for Sankey diagrams since they seem accessible to a wide range of stakeholders with some previous insights into the user behavior [19]. Each bar in the diagram illustrates changes in guidable users, divided into flows of lost and gained guidable users. The largest bars indicate states that are promising candidates for improvement. Note that the bars are not monotonic as they do not visualize the absolute number of users in a state, but the weighted difference in guidable users.

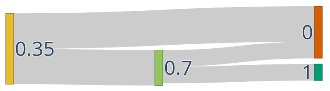
A heat map visualizes the result mapping $\mathscr{R}$ in the reduced Markov chain. By clustering similar states, we can keep diagrams fairly small without compromising the analysis. Fig. 2a shows a SUJG with three necessary user actions to reach a successful outcome. States are annotated with the probability of reaching the



(a) SUJG annotated with model checking results.



(b) Reduced Markov chain.



(c) Sankey diagram generated from the reduced Markov chain.

Fig. 2: We visualize the model checking results in a Sankey diagram that is generated from the learned stochastic user journey games (SUJGs).

successful final state, dotted lines represent uncontrollable user actions, annotated with their probabilities. Fig. 2b shows the *reduced Markov chain*, where two actions divide the states into four clusters with 35%, 70%, 100%, and 0% probability of success, respectively. The insights gained from the induced Markov chain are then visualized as a Sankey diagram in Fig. 2c. The example illustrates flow capacities through the distribution of 100 users.

7 Case Study Results

We present results for the GrepS and BPIC'17 case studies from Sect. 3. The steps described in Sects. 4–6 are assembled in a tool chain, implemented in Python 3.10.12, and available online [27]. For automata learning, we use the IOAlergia implementation of AALpy [37] (v. 1.4) and, for model checking, PRISM-games [11,32] (v. 3.2.1). All experiments ran on a laptop with 32 GB memory and an i7-1165G7 @ 2.8 GHz Intel processor within few hours.

Table 2: Model checking results for GrepS and BPIC'17.

Name	**GrepS**	**BPIC'17-1**	**BPIC'17-2**
Q2	16.49	33.11	33.87
Q3	50.55	37.35	36.07
Q4	44.98	67.79	68.07

GrepS. Figure 3 shows the generated cyclic game, where touchpoints are represented as states, identified by T and a number. It encodes a heat map, ranging from yellow states to green states; the darker a state's green, the greater its probability for success (orange is the unsuccessful state). Transitions with negative weights are orange, and those with positive weights green. The figure highlights the three phases of GrepS' user journey. Phase 1 consists of touchpoints T0–T4, Phase 2 of T5–T20 and Phase 3 of T21–T26. Users receive a new task in T9, T11, T13, T15, and T17. Feedback to users is given after every task. Users share their results with the client company in T26. For readability, we merged the service-provider controlled and user controlled states, which we introduced due to the input/output format of the traces, see Step 1 in Sect. 4, with their preceding touchpoint-labeled states (the full model is available at [27]). For GrepS, we assume that users, when it is their turn, can transition according to the recorded events, or do nothing, i.e., transition to a service-provider state, if available.

We investigate the limits for the positive and negative weights that the service provider can guarantee during the journey, with the user and on its own. Table 2 presents results for model checking the queries Q2–Q4 (see Table 1) for both case studies. For GrepS, the user must endure a significant number of negatively weighted transitions, since the maximum accumulated POS (Q4) is smaller than the minimum accumulated NEG (Q3). Cooperation (Q2) results in a 67.37% reduction in accumulated NEG.

We analyze the impact of the users' and service provider's activity on the user journey by varying the probability in the game's transitions, to change how eager a player is in taking action. Figure 4a shows the results for these changes: on the horizontal axis, $q = 0$ means that the player takes action according to the frequencies of the original game, $-1 \leq q < 0$ means that the service provider gradually increases the probability of taking action (the service provider always takes an action, if available, with $q = -1$). Similarly, for $1 \geq q > 0$, the user gradually increases the probability of taking action (until always taking an action, if available, with $q = 1$). The vertical axis shows the probability of a successful journey (Q1); interestingly, GrepS has a linear gain from being more active and a non-linear loss from being more passive. Figure 4b shows the results for queries Q5 and Q6 by comparing the maximal accumulated positive and the minimum accumulated negative weights for the first S steps of the journey, revealing that negative weights surpass positive weights, especially at the beginning of a journey.

To evaluate whether the service provider can guide users to a successful outcome with limited steps and lower bounds for the gas, we consider the model with bounds derived from query Q7 (see Sect. 5). We bound the integer encodings by 10 times their expected value, which includes at least 90% of the traces. Figure 4c shows the development in guiding the user under Q8. The plot's labels are pairs (G_0, G_1), where G_0 is the minimum gas in the final state and G_1 the lower bound for gas along the journey. For pairs with the same results, we only plot pairs with the maximum final gas and the maximum gas along the journey. The plot shows that experiencing a journey with high minimal gas and reaching a successful outcome are conflicting goals; maximizing minimal gas clearly affects the probability of success for the user journey. For the best probability of success (51%), GrepS needs to guide the users through the negatively weighted transitions, which reach a minimum gas of -64. Actually, the user never fully recovers positive gas in this journey, which ends with a negative gas of -4.

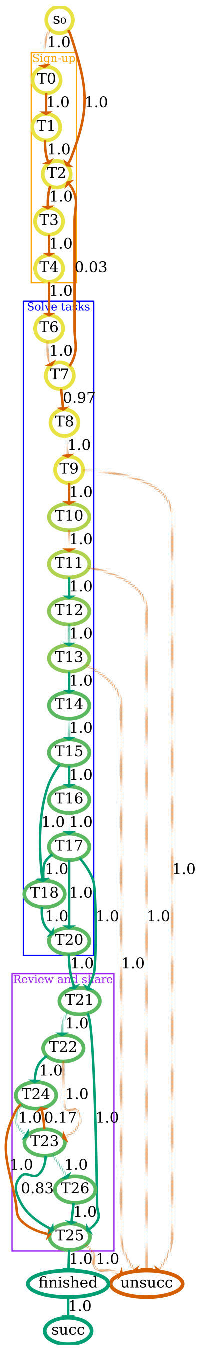
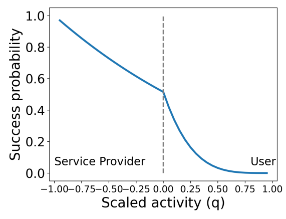
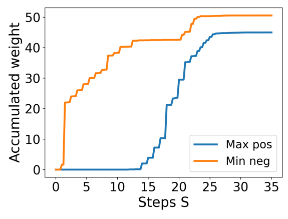


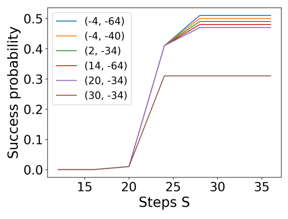
Fig. 3: Simplified model of GrepS' user journey.



(a) Parametric eagerness of the players (Query Q1)



(b) Gas by steps (Queries Q5 & Q6)



(c) Bounded experiment (Query Q8) over (G_0, G_1)

Fig. 4: Experiment results for the GrepS case study.

The analysis has shown that users face negative experiences and that the service provider can offer guidance. We now consider where the journey can be improved to help users reach a successful outcome. Figure 5 shows the derived Sankey diagram with observed users as flow capacities, as described in Sect. 6. The reduced model contains only 6 states, while the mined one has 65 states. Based on the state ranking function (Eq. 1), state T25, where users accept or reject their test results, appears as the most critical state for a successful journey; it determines whether the user will (or not) reach a successful final state; in fact, 25% percent of the users recorded in the log fail their journey immediately after this state. The second most critical state is the first task T9 (where 37.5% of all users are lost), followed by the other tasks. However, at these points in the journey, several user-controlled actions are required for a successful journey, which makes GrepS dependent on the user's cooperation in these states.

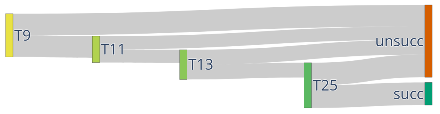


Fig. 5: Sankey diagram of Greps' user journey for guidable users.

Thus, the SUJGs allow us to identify specific states for enhancing the journey: T9 and T25. Our analysis clearly shows that GrepS needs to be active to achieve a successful user journey (Fig. 4). We note that most negatively weighted transitions are user-controlled, suggesting that GrepS can prevent users from "derailing" from a successful journey by being more active within the user journey. If GrepS provides less guidance, users tend to abandon their journeys more easily.

Stakeholder validation of GrepS results. We presented the results obtained for GrepS to a company stakeholder[4] to obtain feedback on our results and their presentation format. The stakeholder was not involved in performing the case study; the other authors only had access to the event log from GrepS, provided in 2021. This validation was done after the analysis results were available.

He was familiar with Sankey diagrams and immediately observed that our analysis makes non-trivial insights accessible to key-stakeholders, varying from concrete recommendations to non-trivial prescriptions on company behavior. From the company's perspective, prioritizing limited resources to improve the users' success rate and experience is challenging. Our case study substantiates

[4] The third author of this paper is a long-term stakeholder of GrepS.

that automated analyses based on event logs are a viable alternative to current best-practices based on heuristics, and promise to reduce assessment efforts.

The identification of T25 as a candidate for improvement (Fig. 5) had actually been discovered independently by GrepS, confirming our analysis. This step is currently supplemented by a manual follow-up step, since completing the user journey successfully is crucial to provide a good user experience. The second suggested task, T9, is not obvious to GrepS and introduces options they have not yet considered, namely to spend resources on guiding the user rather than further optimizing the negative weighted sign-up phase (see Fig. 4b).

The analysis of actor eagerness related to the probability of success (Fig. 4a) is novel and implies that revenue from resources invested in guiding users can be computed. This allows GrepS to evaluate whether to spend more resources on guiding users, given the linear scaling of success probability, or to cut costs through less guidance, reducing manual work while increasing service adversity.

Figures 4b and 4c can be used to relate user profiles and user journeys. A user's motivation to complete tests and share results despite negatively weighted actions, is initially unknown. If the company had some prior knowledge about the initial motivation of a user or a group of users, it would be possible to model different journeys through the service. In particular, Fig. 4c can support such endeavors, because different bounds can be identified for different planned journeys with corresponding probabilities for success.

BPIC'17. Applying Steps 1 and 2 to BPIC'17 yields models with 95 states for BPIC'17-1 and 131 for BPIC'17-2. Step 3 reduces the models to 32 and 47 states, respectively (i.e., +60% reduction). When filtering on reachable states, using the generated strategy, the models shrink to 15 and 19 states, respectively. Figure 7 shows the Sankey diagrams for the two event logs. For readability, we omit the names of states with the least difference in guidable users and use a heat map as in Fig. 3.

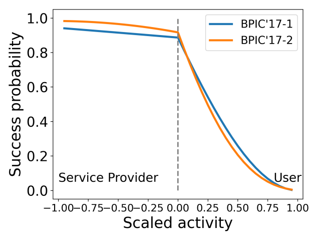
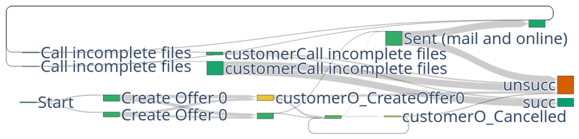


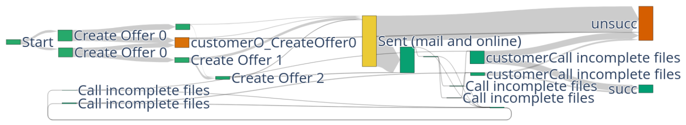
Fig. 6: Parametric eagerness for Q1 in BPIC'17.

The comparison of model checking results between the two models with queries Q2–Q4 (see Table 2) shows some small improvements from BPIC'17-1 to BPIC'17-2. Figure 6 compares different levels of player eagerness for both SUJGs, model checking Q1. It reveals improvements in the service. BPIC'17-2 outperforms BPIC'17-1 starting from $q = 0.06$ when increasing the service provider's probability to take an action. (Plots showing results for the remaining queries, similar to the queries for the GrepS case study, are available online [27].)

Figure 7 shows the positive impact for BPIC'17-2 after the concept drift. In BPIC'17-1, the number of guidable users remains constant through the user journey, with the most critical state causing only 27% of the total user difference. In BPIC'17-2, the main critical state causes a total of 50% difference of guidable users. We also observe a change in loan offers: the 2nd and 3rd offers are prominent in the reduced BPIC'17-2 model (while they were merged with other states or omitted in BPIC'17-1), each with decreasing flow capacity. Furthermore, the



(a) BPIC'17-1



(b) BPIC'17-2

Fig. 7: Sankey diagrams generated from the reduced BPIC'17 models.

probability of guiding users from "*customer Create Offer 0*" reduced; this state is marked yellow in BPIC'17-1 and orange in BPIC'17-2, indicating a decrease in user experience. In both journeys, the second most critical state, a short call due to incomplete files, is user-controlled, but its fraction of the total guidable user's difference decreased from 26.6% to 12.5%. This can be interpreted as evidence that the service provider improved this call state after the concept drift. However, we observe that BPIC'17-2 still lacks proper guidance for the effect of the call, based on the direct transition to the unsuccessful final state.

Threats to Validity. For model learning with IOAlergia, we set the parameter ϵ (which regulates state merging) according to the size of the underlying event log and the assumed complexity of the service. For GrepS, we set $\epsilon = 0.1$ due to a small number of possible journeys, while for BPIC'17, we set $\epsilon = 0.8$ to capture different decisions and possible executions. Insights from GrepS highly depend on ϵ, where a larger ϵ restricts state merging. For BPIC'17, we observe that the eagerness experiment (Fig. 6) replicates for various ϵ values, though with variations for either small or large ϵ values. Further investigations are needed to draw rigorous conclusions about this relation. The model-checking analysis in Step 3, which generate Sankey diagrams, do not require a minimal flow of users. Strategies might exploit rarely observed behavior, they do not consider a minimum bound for the coverage of users. Table 1 presented queries that target Pareto optimization problems to optimize multiple conflicting objectives, e.g., limited steps and minimal gas in positive states. We explored solutions to these problems with PRISM-games experiments, but one could also search for all solutions. The efficiency of our technique depends on automata learning and model checking; all presented results are reproducible within ~ 9 h.

8 Related Work

Related work primarily focuses on designing domain-specific modeling languages that allow modeling from the user's perspective. The methods developed [5, 9, 14, 20, 22, 23, 35, 38, 41] concentrate on manually constructing user journeys based on expert knowledge [9], user questionnaires [21,41], or given event logs [5]. The analysis of the resulting models is typically also performed manually. However, Lammel *et al.* [35] propose an ontology-based technique that allows the automatic generation of visualizations to provide further insights.

Process discovery [1] is a technique to automatically generate models from event logs and has been applied to generate different types of user journey models such as customer journey maps (CJM) [7, 8, 24] or transition systems [26,28,30,31,42]. CJMs represent grouped traces in the event logging, unlike our work where we mine a general model. Existing approaches [26, 28, 31, 43] that use process discovery techniques to mine transition systems ignore the underlying distribution of events. By capturing the probabilities in the model, we can perform a finer analysis and visualization, and provide guidelines to the service provider in case of changing behavior. In our previous work [31], we also generated weighted deterministic user journey games and applied model checking to find bottlenecks in the service. By applying automata learning instead of process discovery techniques, we enhance this approach to generate probabilistic games.

Automata learning techniques [3, 13, 16, 45] have been used to mine process models, e.g., transition systems or Petri nets, from given event logs. However, our proposed approach incorporates the users' perspective. While existing techniques may also consider the underlying probability distribution of the event log constructing the model, they neglect it for later analysis. Wieman *et al.* [45] derive improvements for industrial case studies manually from the learned model.

9 Conclusion

This paper presents two complementary case studies for the automated modeling and analysis of user journeys from event logs. Our analysis tool chain combines automata learning and model-checking techniques, based on a formalization of user journeys as stochastic weighted games that exploits the underlying distribution of events in the log. Model-checking results are used in property-preserving model reduction, which allows us to automatically identify and rank actions that are critical to the outcome of the user journey and visualize their effect. To the best of our knowledge, this is the first work using stochastic games in an automated method to analyze and improve user journeys.

The investigated case studies demonstrate the applicability of our approach to real-world services, varying in size and complexity. The results of the case studies lead us to three main observations: (1) model visualization creates compact Sankey diagrams for complex services that facilitate the interpretation of formal analyses; (2) the model reduction preserves changes in the underlying journeys, e.g., the concept drift for BPIC'17; and (3) the state ranking method effectively identifies candidate states for service redesign, based on user experience. Compared to previous work, our exploitation of the underlying probabilistic distribution of events enabled a more targeted analysis of the user journeys. For future work, automatically capturing the actor information in the event logs would make our approach less dependent on domain knowledge.

Data Availability Statement. The artifact to replicate the presented results is publicly available on Zenodo at https://doi.org/10.5281/zenodo.12529995.

References

1. 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
2. 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: International Conference on Business Process Management. vol. 12875, pp. 400–416. Springer (2021), https://doi.org/10.1007/978-3-030-85469-0_25
3. Agostinelli, S., Chiariello, F., Maggi, F.M., Marrella, A., Patrizi, F.: Process mining meets model learning: Discovering deterministic finite state automata from event logs for business process analysis. Information Systems **114**, 102180 (2023), https://doi.org/10.1016/J.IS.2023.102180
4. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
5. 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. International Conference on Information Systems (ICIS 2018). Association for Information Systems (2018), https://aisel.aisnet.org/icis2018/modeling/Presentations/4
6. Bergersen, G.R., Sjøberg, D.I.K., Dybå, T.: Construction and Validation of an Instrument for Measuring Programming Skill. IEEE Transactions on Software Engineering **40**(12), 1163–1184 (Dec 2014), https://doi.org/10.1109/TSE.2014.2348997
7. 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 - Proceedings 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
8. 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
9. 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
10. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Carrasco, R.C., Oncina, J. (eds.) Proc. Second International Colloquium on Grammatical Inference and Applications (ICGI-94). Lecture Notes in Computer Science, vol. 862, pp. 139–152. Springer (1994), https://doi.org/10.1007/3-540-58473-0_144
11. Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. Formal Methods in System Design **43**(1), 61–92 (2013), https://doi.org/10.1007/S10703-013-0183-7
12. Chen, T., Forejt, V., Kwiatkowska, M.Z., Simaitis, A., Trivedi, A., Ummels, M.: Playing stochastic games precisely. In: Koutny, M., Ulidowski, I. (eds.) Proc. 23rd International Conference on Concurrency Theory (CONCUR 2012). Lecture Notes

in Computer Science, vol. 7454, pp. 348–363. Springer (2012), https://doi.org/10.1007/978-3-642-32940-1_25

13. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Transactions on Software Engineering and Methodology (TOSEM) **7**(3), 215–249 (1998), https://doi.org/10.1145/287000.287001
14. 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
15. van Dongen, B.: BPI Challenge 2017 (2017), https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
16. Esparza, J., Leucker, M., Schlund, M.: Learning workflow Petri nets. Fundamenta Informaticae **113**(3-4), 205–228 (2011), https://doi.org/10.3233/FI-2011-607
17. 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
18. Gold, E.M.: Language identification in the limit. Information and Control **10**(5), 447–474 (1967), https://doi.org/10.1016/S0019-9958(67)91165-5
19. Gutwin, C., Mairena, A., Bandi, V.: Showing flow: Comparing usability of Chord and Sankey diagrams. In: Schmidt, A., Väänänen, K., Goyal, T., Kristensson, P.O., Peters, A., Mueller, S., Williamson, J.R., Wilson, M.L. (eds.) Proc. 2023 Conference on Human Factors in Computing Systems (CHI 2023). pp. 825:1–825:10. ACM (2023), https://doi.org/10.1145/3544548.3581119
20. Halvorsrud, R., Boletsis, C., Garcia-Ceja, E.: Designing a modeling language for customer journeys: Lessons learned from user involvement. In: Proc. 24th International Conference on Model Driven Engineering Languages and Systems (MODELS 2021). pp. 239–249. IEEE (2021), https://doi.org/10.1109/MODELS50736.2021.00032
21. 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
22. 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
23. 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. International Conference on Services Computing (SCC 2021). pp. 367–369. IEEE (2021), https://doi.org/10.1109/SCC53864.2021.00051
24. Harbich, M., Bernard, G., Berkes, P., Garbinato, B., Andritsos, P.: Discovering customer journey maps using a mixture of Markov models. In: Ceravolo, P., van Keulen, M., Stoffel, K. (eds.) Proc. 7th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2017). CEUR Workshop Proceedings, vol. 2016, pp. 3–7. CEUR-WS.org (2017), http://ceur-ws.org/Vol-2016/paper1.pdf
25. Hoeffding, W.: Probability inequalities for sums of bounded random variables. In: Fisher, N.I., Sen, P.K. (eds.) The Collected Works of Wassily Hoeffding, pp. 409–426. Springer (1994), https://doi.org/10.1007/978-1-4612-0865-5_26

26. Kobialka, P., Mannhardt, F., Tapia Tarifa, S.L., Johnsen, E.B.: Building user journey games from multi-party event logs. In: Proc. 3rd International Workshop on Event Data and Behavioral Analytics (EdbA 2022). Lecture Notes in Business Information Processing, vol. 468. Springer (2022), https://doi.org/10.1007/978-3-031-27815-0_6
27. Kobialka, P., Pferscher, A., Johnsen, E.B., Tapia Tarifa, S.L.: Supplementary material: Stochastic games for user journeys. https://github.com/smartjourneymining/probabilistic_games/releases/tag/FM2024 (2024)
28. Kobialka, P., Schlatte, R., Bergersen, G.R., Johnsen, E.B., Tapia Tarifa, S.L.: Simulating user journeys with active objects. In: de Boer, F.S., Damiani, F., Hähnle, R., Johnsen, E.B., Kamburjan, E. (eds.) 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
29. Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: Weighted games for user journeys (data set). https://doi.org/10.5281/zenodo.6962413 (2022), accessed: 2024-04-01
30. Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: Weighted games for user journeys. In: Schlingloff, B., Chai, M. (eds.) Proc. 20th International Conference on 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
31. Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: User journey games: Automating user-centric analysis. Software and Systems Modeling **23**(3), 605–624 (2024), https://doi.org/10.1007/s10270-024-01148-2
32. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In: Lahiri, S.K., Wang, C. (eds.) Proc. 32nd International Conference on Computer Aided Verification (CAV 2020). Lecture Notes in Computer Science, vol. 12225, pp. 475–487. Springer (2020), https://doi.org/10.1007/978-3-030-53291-8_25
33. Kwiatkowska, M., Parker, D., Wiltsche, C.: PRISM-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives. Int. J. Softw. Tools Technol. Transf. **20**(2), 195–210 (2018), https://doi.org/10.1007/S10009-017-0476-Z
34. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd International Conference on Computer Aided Verification (CAV 2011). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011), https://doi.org/10.1007/978-3-642-22110-1_47
35. Lammel, B., Korkut, S., Hinkelmann, K.: Customer experience modelling and analysis framework - a semantic lifting approach for analyzing customer experience. In: Proc. 6th International Conference on Innovation and Entrepreneurship (IE 2016). GSTF (Dec 2016), http://hdl.handle.net/11654/24293
36. Mao, H., Chen, Y., Jaeger, M., Nielsen, T.D., Larsen, K.G., Nielsen, B.: Learning deterministic probabilistic automata from a model checking perspective. Machine Learning **105**(2), 255–299 (2016), https://doi.org/10.1007/S10994-016-5565-9
37. 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
38. 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

39. Riehmann, P., Hanfler, M., Froehlich, B.: Interactive Sankey diagrams. In: Stasko, J.T., Ward, M.O. (eds.) IEEE Symposium on Information Visualization (InfoVis 2005). pp. 233–240. IEEE Computer Society (2005), https://doi.org/10.1109/INFVIS.2005.1532152
40. Rodrigues, A.M.B., Almeida, C.F.P., Saraiva, D.D., Felipe, B., Moreira, Spyrides, G.M., Varela, G., Krieger, G., Igor, T., Peres, Dantas, L.F., Lana, M., Alves, O.E., França, R., Ricardo, Neira, A., Gonzalez, S.F., Fernandes, W., Barbosa, S.D.J., Poggi, M., Lopes, H.C.V.: Stairway to value: mining a loan application process (2017), https://www.win.tue.nl/bpi/2017/bpi2017_winner_academic.pdf
41. 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
42. 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
43. 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
44. Vandermerwe, S., Rada, J.: Servitization of business: Adding value by adding services. European Management Journal **6**(4), 314–324 (1988), https://doi.org/10.1016/0263-2373(88)90033-3
45. Wieman, R., Aniche, M.F., Lobbezoo, W., Verwer, S., van Deursen, A.: An experience report on applying passive learning in a large-scale payment company. In: Proc. International Conference on Software Maintenance and Evolution (ICSME 2017). pp. 564–573. IEEE Computer Society (2017), https://doi.org/10.1109/ICSME.2017.71