Nudging Strategies for User Journeys: Take a Path on the Wild Side*

Einar Broch Johnsen, Paul Kobialka, Andrea Pferscher, and Silvia Lizeth Tapia Tarifa

Department of Informatics, University of Oslo, Oslo, Norway
{einarj,paulkob,andreapf,sltarifa}@ifi.uio.no

Abstract. With the emergence of service orientation as a major business driver, companies crucially depend on understanding the flow of their services from the user's perspective. Models of these *user journeys* help to create a common understanding, but in practice their availability is limited. Process mining addresses the challenge of creating models that enable processes to be analyzed. Our goal is to mine user journey models. In this paper, we use automata learning algorithms to create behavioral models of stochastic user behavior from a given data set. The initially learned automaton is annotated with time and cost variables to capture aspects of the user experience. In a game scenario, we can model check properties of these enriched automata regarding the user behavior. Using UPPAAL, we can synthesize strategies for nudging users into a different behavior. The approach is illustrated in a case study with a large dataset describing user behavior for a well-known music streaming application. Can we synthesize a strategy that nudges a computer science professor to take a path on the wild side of the usual listening habits?

Keywords: User journeys · Nudging · Process discovery · Passive automata learning · UPPAAL

1 Introduction

The servitization of business [46] results in business models shaped by user demand: user satisfaction is critical to the success of a business, and directly impacts financial rewards [21]. Consequently, businesses invest in improving the user experience they offer to their users. User journeys describe the actual communication and interaction between service provider and user, when engaging in a service, from the user's point of view. Traditionally, user journeys are modeled and analyzed manually, based on, e.g., questionnaires addressing selected users to identify the experienced user journey and the associated user satisfaction (e.g. [23, 42]). Although successful, this approach to the modeling and analysis

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

of user journeys comes with inherent restrictions [24]: it lacks tool support for automated analysis and does not scale beyond a very limited number of users.

The analysis of user journeys can also be achieved by data-driven formal methods. In a series of papers [29–33], we have explored how formal models can be generated from the event logs of digital services and automatically analyzed using existing tool-supported analysis methods. In this line of work, we have used process mining techniques [1] to generate automata from event logs. The extracted automata can be extended into *weighted automata* [11]; the weights can reflect a notion of user experience as aggregated values computed from numerous actual user journeys recorded in the event logs. The generated automata can be analyzed using model-checking tools such as Uppaal [35]. The automata can be further transformed into *weighted games*, modeling the interactions between the users and the service provider. To capture the underlying stochasticity of these interactions, the weighted games can be further extended into *stochastic weighted games* [30]. Using tools such as Uppaal Stratego [19] and PRISM-games [16], these *user journey games* can be analyzed, e.g., to derive optimal strategies, thus providing strategic recommendations for service providers to guide their users through the service. These strategies can again form the basis for actor-based simulations in tools such as ABS [10, 27, 28].

In this paper, we review the ideas underlying data-driven formal methods for user journey analysis, starting from a large data set: the event logs of the music streaming service Spotify [12]. We are interested in whether model-checking techniques can be used to derive nudging strategies [45] for selected user groups; for example, can we identify a strategy for the music streaming service that optimizes for changing the musical taste of a computer science professor? The crucial events to analyze are the *skip* actions of a user when streaming music, which suggests dissatisfaction or impatience with the track proposed by the algorithm (e.g., [37]). In this paper, we explore techniques for *passive automata learning* [22] to generate *probabilistic, timed games* from event logs. To this aim, we extend our previous work on stochastic user journey games [30] to account for timing properties. We further extend the resulting user journey games with *user profiles* that identify the initial states of the games we consider. Our objective is to use the resulting probabilistic and timed games to find the optimal strategy for the music streaming service, that nudges the professor to be adventurous and take a path on the wild side through the user journey game.

2 Preliminaries

Let X^* denote the finite, ordered sequences (or traces) $x_0 \cdot x_1 \ldots x_{n-1}$ over a set X, with $x_i \in X$ for $0 \leq i < n$ and $n \in \mathbb{N}$ is the length of the sequence. Let ϵ denote the empty sequence, $s \cdot s'$ the concatenation of sequences s, s', and x the singleton sequence for any $x \in X$. We write $\mathcal{B}(X)$ for the set of multisets over X.

2.1 Event Logs

An *event log* [1] records the behavior of a system as sequences of observed events. The set of observable events A of an event log is called its *alphabet*.

Definition 1 (Event log [1]). *An* event log $\mathcal{T}$ *over an alphabet* A *is a multiset of traces over* A, $\mathcal{T} \in \mathcal{B}(A^*)$.

2.2 Automata Learning

Automata learning is a technique to automatically generate behavioral models from a sample, which is a set of system traces. The goal of automata learning algorithms is to generate an automaton that models the unknown language of the system under learning (SUL). Depending on the way the sample is generated, we distinguish between two learning paradigms: active and passive learning. Active learning algorithms interact with the SUL to generate the sample, instead passive learning algorithms use a given sample.

We here focus on passive learning from a given event log. Many passive learning algorithms assume that the sample contains *positive* traces that are part of the language of the SUL, as well as *negative* traces that are not part of the language. Angluin [7] showed that by considering specific attributes of the sample, e.g., the distribution of the actions, positive samples are sufficient to learn an accurate model. Alergia [13] has shown that considering certain properties of the sample, e.g. the distribution of events, makes it possible to learn an accurate model only from positive samples. To learn Markov chains, we consider a variant of Alergia that is available in the state-of-the-art machine learning library AALpy [38]. This variant of Alergia starts with the creation of a *frequency prefix tree acceptor* from an event log $\mathcal{T}$:

Definition 2 (Frequency prefix tree acceptor (FPTA)). *An* FPTA *is a tuple* $\mathcal{P} = \langle Q, q_0, A, E, F, L \rangle$, *where*

- Q *is the finite set of states,*
- $q_0 \in Q$ *is the initial state (or root),*
- A *is the observable universe of events,*
- $E \subseteq Q \times Q$ *is the set of directed transitions between states,*
- $F \colon E \to \mathbb{N}$ *is a labeling function assigning frequencies to transitions, and*
- $L \colon Q \to A$ *is a labeling function for states with events from the alphabet* A.

A *path* is a sequence $q_0 \cdot q_1 \dots q_m$ of states such that there are transitions $q_{i-1} \to q_i$ for all $0 < i \leq m$. Traces $t \in A^*$ can be obtained by applying the event-labeling function L to each element of a path $q_0 \cdot q_1 \dots q_m$; i.e., $t = L(q_0) \cdot L(q_1) \dots L(q_m)$.

The FPTA created by Alergia is a concise representation of the underlying event log $\mathcal{T}$, where each path in the FPTA represents a prefix of a trace $t \in \mathcal{T}$ in the event log. To ensure that there is a single initial state, we let all traces start with an identical event. If two traces have equal prefixes, they share the

same path for the prefix. A frequency assigned to the transition indicates the cumulative number of traces that share this transition in their path. After the tree construction, Alergia merges the states of the tree to create a Markov chain. The state merging is done from the root (the initial state) to the leaves, coloring the states to indicate possible merge candidates, with suitable merge candidates having the same event labels. Let $q \to q'$ be a transition from state q to state q', with $(q, q') \in E$. The auxiliary function $out(q)$ defines the number of outgoing transitions from a state $q \in Q$, i.e., the number of elements of the set $\{q' | q \to q' \in E\}$. The compatibility check between two states $q, q' \in Q$ is based on the Hoeffding bound [26], which defines that the two states are equal if

$$\left| \frac{F(q \to q_x)}{out(q)} - \frac{F(q' \to q_y)}{out(q')} \right| \leq \sqrt{\frac{1}{2} \log \frac{2}{\alpha}} \left(\frac{1}{\sqrt{out(q)}} + \frac{1}{\sqrt{out(q')}} \right), \tag{1}$$

where $L(q_x) = L(q_y)$. The parameter $\alpha \in (0, 1]$ represents a *confidence parameter* that allows the probability of the merging state to be adjusted based on the assumptions of the underlying event log. Thus, if we assume that the event log adequately represents the underlying distribution of the SUL, we should set α low, otherwise, α should be set higher to account for likely larger differences between the compared states due to missing observations. If two states can be merged, we accumulate the frequencies at outgoing transitions to the same states. If no further states can be merged, the Alergia variant for learning Markov chains updates the accumulated frequencies for the transitions to probabilities such that they fulfill the requirements of the probabilistic state transition function (see Def. 3). Alergia then returns the Markov chain that was finally generated.

2.3 Markov Chains

A *Markov chain* [40] defines the observable random behavior of a system by a finite state machine. On a semantic level, the behavior defined by the Markov chain implements a Markov process, i.e., the future behavior only depends on the current state independent from the history of events. Let $Dist(X)$ be a set of probability distributions over a finite set X, where for each $\mu \in Dist(X)$, $\mu \colon X \to [0, 1]$ and $\sum_{x \in X} \mu(x) = 1$ holds.

Definition 3 (Markov chain). *A* Markov chain *is a tuple* $\mathcal{M} = \langle Q, q_0, A, \delta, L \rangle$, *where*

- Q *is a finite set of states,*
- $q_0 \in Q$ *is an initial state,*
- A *is an observable universe of events,*
- $\delta \colon Q \to Dist(Q)$ *is a partial stochastic state transition function, and*
- $L \colon Q \to A$ *is a state labeling function.*

Note that δ *is a partial function, hence we do not require for all* $q \in Q$ *to be defined in* δ. *Let* $Q' \subseteq Q$ *be the set for which* δ *is defined.* $\mathcal{M}$ *is* deterministic *iff* $\forall q \in Q', \forall q', q'' \in Q \colon \delta(q)(q') > 0 \wedge \delta(q)(q'') > 0 \implies q' = q'' \vee L(q') \neq L(q'')$.

2.4 Priced Timed Systems under Stochastic Behavior

Markov chains can be extended by (1) considering the passage of time, (2) extending the state transition function for dedicated actions to determine the distribution of successor states, (3) accounting for journey experience features as costs/weights, and (4) separating controllable from uncontrollable actions.

Stochastic timed automata (STAs) [39] extend Markov chains with timed behavior. STAs account for timed behavior via real-valued clock variables that define invariants *inv* on states and constraints *enab* that enable transitions, and allow for multiple actions each leading to a distribution of successor states (thereby implementing the Markov chain extensions (1) and (2)). Clocks are non-negative, real-valued variables. Let $\mathcal{C}$ denote a (finite) set of *clocks*. The set $\mathcal{G}(\mathcal{C})$ of *clock constraints* consists of Boolean constraints $c \sim x$ over clock variables $c \in \mathcal{C}$, a comparison parameter $\sim \in \{\leq, <, >, \geq\}$ and a constant $x \in \mathbb{N}$. The *clock valuation* function $\nu \colon \mathcal{C} \to \mathbb{R}_{\geq 0}$ maps clocks to their current value. We write $\nu \models g$ if the clock valuation ν fulfills the clock constraint g Let $\nu + x$ express that the value of every clock $c \in \mathcal{C}$ is increased by a constant $x \in \mathbb{R}_{\geq 0}$ and let $\nu[r]$ express that all clocks $c \in r$ are reset to zero, where $r \subseteq \mathcal{C}$.

Definition 4 (Stochastic timed automata (STAs) [39]). *A* stochastic timed automaton (STA) *is a tuple* $\mathcal{A} = (Q, q_0, A, \mathcal{C}, \Sigma, inv, enab, prob, L)$*, where*

- Q, q_0, A, L *are defined as for Markov chains,*
- $\mathcal{C}$ *is a finite set of clocks,*
- Σ *is a finite set of actions,*
- $inv \colon Q \to \mathcal{G}(\mathcal{C})$ *maps states to their invariants,*
- $enab \colon Q \times \Sigma \to \mathcal{G}(\mathcal{C})$ *is a transition-enabling condition function, and*
- $prob \colon Q \times \Sigma \to Dist(2^{\mathcal{C}} \times Q)$ *is a (partial) stochastic transition function.*

We briefly outline the semantics of STAs (for further details, see Norman *et al.* [39]): States of STAs are pairs $(q, \nu) \in Q \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$ such that all invariants are satisfied, $\nu \models inv(q)$. In each state (q, ν), a time elapse $t \in \mathbb{R}_{\geq 0}$ is possible if the invariants $inv(q)$ are continuously satisfied, resulting in state $(q, \nu + t)$, or an enabled action $a \in \Sigma$ is taken to decide on a stochastic transition from the current state. An action $a \in \Sigma$ is *enabled* in state (q, ν) if $\nu \models enab(q, a)$. A state (q, ν) is *urgent* if no time can elapse in q.

Stochastic priced timed automata (SPTAs). An SPTA is an STA extended with *price variables*. We consider functions $P \colon Q \to \mathbb{R}_{\geq 0}$ that assign a rate for the accumulation of prices while time passes in a state. Price variables are monotonic, real-valued variables that account for resources spent while halting in states. States may utilize an exponential rate of increase for price variables, incurred constantly within a state. For formal definitions of priced timed automata on a semantic level, we refer to Behrman *et al.* [9]. Figure 3 depicts an SPTA including the price variables: `acousticness_c` and `duration_c`.

Stochastic priced timed games (SPTGs). An SPTG divides the set Σ of actions of an SPTA into controllable actions Σ_c and uncontrollable actions Σ_u. In, e.g., the synthesis of a strategy, the controller can only select actions in Σ_c and the analysis is conducted under a worst-case assumption about the actions Σ_u, i.e., without insights into which actions would be chosen by the environment.

Model checking of games. UPPAAL [35] is a tool that allows the modeling, simulation, and verification of timed systems. The tool can be used to synthesize controllers [8] for systems considering timed and stochastic behavior [19]. UPPAAL can also synthesize cost-optimal strategies by considering real-valued cost variables and cost rates [34]. The UPPAAL STRATEGO extension allows game settings, which are by now integrated into the latest UPPAAL version. *Strategies* assign a set of actions to each state that guarantee or optimize a certain outcome. Further details on the computation of strategies can be found in [14,18].

User journey games (UJGs) [33] can be modeled as SPTGs by assigning the controllable actions to the service provider and the uncontrollable actions to the user. Analysis for a UJG is user-centric since it does not constrain the user in its possible actions and it requires the service provider to guide the user through the system. In our previous work, we explored the use of strategies for user journey games [29,31–33], for among others, highlighting changes in the UJG over time, and game theoretic reductions to discover outcome determining interactions. In this paper, we are going to use strategies to nudge users toward exploring other alternative behaviors in the UJG.

3 Method

We now explain our method to generate a UJG for nudging users in terms of a concrete case study. The method allows us to automatically create behavioral models from event logs by learning Markov chains and enriching them to an STA with meta-data in the dataset from which the event log is generated. First, we explain the process of converting the dataset into an event log and extracting features from the dataset to create an STA. We then lift the created STA to an SPTG by separating actions into controllable and uncontrollable, where the controller takes deterministic actions, and the environment performs stochastic actions. All resources and code are published online in the project's repository.[1]

3.1 Case Study: The Music-Streaming User Journey

We consider a case study in which users interact with a music streaming service provider to listen to music. Users listen to tracks in listening sessions. Depending on the context, the service provider suggests or selects tracks for the user. The user responds by deciding whether to listen to or skip each particular track. Thus,

[1] https://github.com/smartjourneymining/spotify_journey/releases/tag/wang2024

the user journey consists of back-and-forth interaction between user and service provider. This scenario can be understood as a user journey in which both the service provider and user are interested in maximizing the listening time while minimizing the number of skips. The goal of the case study is to synthesize *nudging strategies*, concretely, to use the inferred user journey game to nudge users toward a different musical taste. Our case study uses the *Music-Streaming Sessions Dataset* (MSSD), provided by Brost *et al.* [12].

Brost *et al.* [12] present six *challenges* targeting techniques to predict the skipping of tracks given a session prefix. Due to a lack of datasets recording rich user interactions within a system, the accompanying MSSD that records user interactions with the listened tracks, has been made available online.[2] The proposed challenges are mainly aimed at research on recommender systems, e.g., recommendations for long-term user satisfaction or proactive recommendations and interventions. However, *Challenge 4* targets the evaluation of "user journeys" under the consideration of "user moods", i.e. the analysis of profiles for musical taste. The method described in our paper is concerned with using the MSSD to nudge users towards a different musical taste profile.

The full dataset contains 160 million listening sessions with 3.7 million unique tracks recorded by the music streaming platform Spotify. Each session contains 10–20 different tracks; shorter sessions are excluded and longer sessions are truncated to 20 tracks to protect user's privacy. Sessions record the listened tracks and additional user interactions, the listening context (e.g., a pre-generated playlist or a personal playlist), whether the track was skipped, and whether the user took a short or long pause before playing the track. Sessions are recorded individually and are not linked to other sessions by the same user. The track dataset is accompanied by metadata about the tracks containing characteristics such as acoustics, danceability, energy, and many more. MSSD further includes a "sample" version, recording 10 000 listening sessions, which we use in the remainder of the paper to demonstrate our method. An excerpt of the dataset is shown in Table 1, the session information is displayed in Table 1a, and the track information in Table 1b.

3.2 Disentangling the Music-Streaming User Journey

To model the user journey, we utilize several aspects of the information recorded in the MSSD. As a first step, we parse the tabular records of the MSSD by grouping tracks by their sessions in the order they were played. Each recorded session is then processed to include a designated *start state* (start session) and a *start profile state* (startprofile) that is computed by averaging track metadata of the first five listened tracks. From there on, the service provider and user engage in a loop of entering a listening context, proposing tracks and reacting to tracks. If the current listening context is controllable, i.e., the *context state* (select context_i) corresponds to "radio", "editorial playlists", "charts", or "personalized playlists", the service provider chooses the next track by selecting a *song state*

[2] https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge

Table 1: Excerpt from the Music-Streaming Sessions Dataset (MSSD) [12]. The dataset can split into two tables, where Table 1a provides streaming session information and Table 1b includes metadata about the individual tracks.

SessionID	Session Position	*TrackID*	···	Not skipped
2d8ead1e	1	a19d63a	···	True
2d8ead1e	2	s8d42lo	···	False
a9082maw	1	ao8r7n4	···	True
633e57da	1	a19d63a	···	True
⋮	⋮	⋮	⋱	⋮

(a) Excerpt of music session data.

TrackID	Duration	Acousticness	Danceability	···	Energy
a19d63a	103.74	0.37	0.39	···	0.82
s8d42lo	185.28	0.82	0.48	···	0.17
ao8r7n4	159.59	0.93	0.48	···	0.36
l92bz48	362.63	0.47	0.57	···	0.49
⋮	⋮	⋮	⋮	⋱	⋮

(b) Excerpt of track metadata.

(song_i); otherwise the *context state* corresponds to "catalog" or "user collection", and some track is played by randomly selecting a *played state* (played_i). (We use *track* when we do not want to differentiate user selection from service selection; in the sequel, a *track state* is either a song state or a played state.) The user responds to the track by either skipping it or listening to it completely. MSSD contains different binary skipping information, called **Skip_1**, **Skip_2**, and **Skip_3**. The different flags indicate when the track has been skipped. For example, **Skip_1** is true if a track is skipped shortly after the start, whereas **Skip_3** would also include very late skips. As suggested in the MSSD challenge, we use the field **Skip_2** from the dataset as ground truth for skipping tracks, thereby ignoring short skips towards the end of a track. Further, the user may take a break from a listening session and return afterward to a context in which a new track is chosen. The *end state* (end) marks the end of a recorded listening session. We constrain sessions to include exactly 20 tracks, the maximum length of sessions in the MSSD, though any other number of tracks could have been chosen. Sessions that are shorter than 20 tracks are not part of the event log. Figure 1 summarizes the overall structure of the considered music-streaming user journey. Solid transitions represent controllable and dashed transitions uncontrollable actions. Circles (○) represent stochastic branching points.

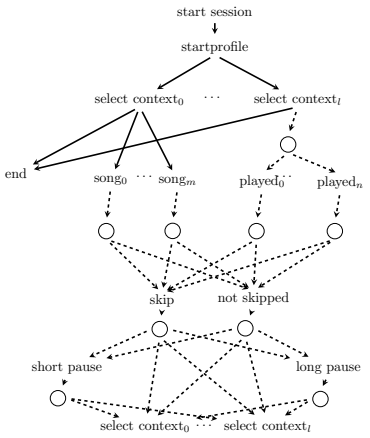
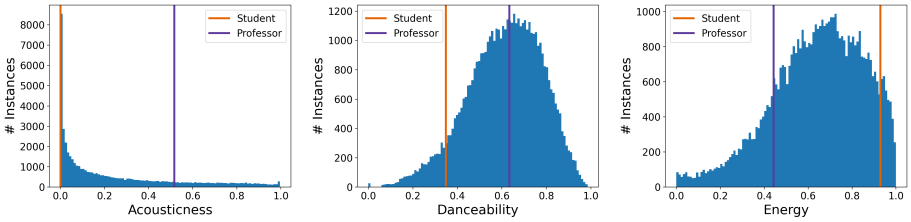


Fig. 1: The basic layout of a music-streaming user journey.

3.3 User Profiles for Musical Taste

A user profile for musical taste can be represented by a vector in which the components reflect the musical user preferences in terms of the features described in the metadata of the tracks, e.g., danceability, energy, etc.



(a) Acousticness histogram (b) Danceability histogram (c) Energy histogram

Fig. 2: Histograms over track features in the MSSD. We also indicate the user profiles of the professor and student in the histograms.

For our case study, we collected two real user profiles: one representing the musical preferences of a professor[3] and the other representing those of a Ph.D. student, both in computer science. The profiles were constructed by querying the Spotify API for the long-term favorite tracks of the subjects in question by a Python script using the *spotipy*[4] library, and averaging quantitative track features. For simplicity, we concentrated on a subset of the available metadata features, including the three features that maximized the difference between the two profiles: *acousticness*, *danceability*, and *energy*. Each track gets an associated a feature vector $\mathbf{f} \in \mathbb{N}^3_{\geq 0}$, constructed by the following structure: $\mathbf{f} := \langle \textit{acousticness}, \textit{danceability}, \textit{energy} \rangle$. Figure 2 shows the two profiles over the histograms of track features in the MSSD. Observe that the user profile of the student tends towards the "*wild side*" with respect to acousticness, danceability and energy (i.e., a musical preference in the direction of heavy metal). Consequently, we use the professor profile as the start profile $\mathbf{s}$ to be nudged, targeting the student's profile $\mathbf{t}$. To unify the representation of all track features, we multiply the track features that are originally in the range $[0, 1]$, by 10 and round down, to lift them to the range $[0, 10] \cap \mathbb{N}_{\geq 0}$. The start profile then becomes $\mathbf{s} := \langle 5, 6, 4 \rangle$, and the target profile $\mathbf{t} := \langle 0, 3, 9 \rangle$.

3.4 Learning the Structure of the Music-Streaming User Journeys

We now describe how session and track information are parsed into sequences of observed events to create an event log. We then explain how this event log is used to learn the stochastic structure of the process, following Sect. 2.2.

From Dataset to Event Log. We preprocess the MSSD according to the structure depicted in Fig. 1 for the creation of an event log. After identifying the individual sessions, we generate the user profile for each session. The user profile is the averaged track metadata of the first five listened tracks.

[3] The particular individual is not an author of this paper.
[4] https://github.com/spotipy-dev/spotipy

Applying Abstractions to the Event Log. To reduce the number of included sessions and thereby reduce the learning time, we filtered the sessions in the event log by the start and target user profiles $\mathbf{s}$ and $\mathbf{t}$. Only sessions with a profile $\mathbf{p}$ with a distance $\leq k$ from $\mathbf{s}$ and $\mathbf{t}$ in each profile dimension are retained. Thus, remaining user journeys have a start profile $\mathbf{p}$ satisfying:

$$\forall_{i \in [0,\ldots,|\mathbf{p}|-1]} \quad \mathbf{p}_i \geq \min(\mathbf{s}_i, \mathbf{t}_i) - k \wedge \mathbf{p}_i \leq \max(\mathbf{s}_i, \mathbf{t}_i) + k. \tag{2}$$

In our experiments, we set $k \coloneqq 1$, resulting in a total of 8 600 included sessions.

We encode tracks according to the year the track was released, e.g. 1990 or 2010, its key (related to scale, either "major" or "minor"), and the three discretized track features. With this encoding, similar tracks are grouped into the same event. Additionally, different states are introduced for tracks proposed by the service provider, to model the later controllable actions in the UJG, and tracks selected by the user, the later uncontrollable actions, resulting in multiple states for the same track. An example of an event sequence with two tracks is:

$$\begin{aligned}&\mathsf{start} \cdot \mathsf{startprofile}\langle 5,6,4\rangle \cdot \mathsf{select\_context}(\mathsf{radio}) \cdot \mathsf{song\_2010\_minor\_}\langle 2,7,8\rangle \\ &\quad \cdot \mathsf{not\_skipped} \cdot \mathsf{select\_context}(\mathsf{user\_playlist}) \cdot \mathsf{played\_1990\_major}\langle 4,7,3\rangle \ldots\end{aligned}$$

Discovering the Underlying Behavioral Model. We use the Alergia automata learning algorithm for Markov chains (see Sect. 2.2) to learn the stochastic structure of the underlying user journey. Automata learning allows us to automatically generate stochastic models from large datasets in a reasonable amount of time. In the following, we extend the learned Markov chain to a UJG used to nudge users from their start profile to a target profile.

3.5 Generating Music-Streaming User Journeys

The learned Markov chain describes the stochastic structure of the user journey. We now extend the Markov chain with timed behavior, costs, and controllable and uncontrollable actions to create a UJG, following Sect. 2.4.

When streaming music from a platform, the passage of time is an important characteristic. In our work, we model a user getting used to different track profiles by aggregating the track features over the actual listening times; i.e., the longer a user listens to an energetic track, the more energy is accumulated. Therefore, we extend the learned Markov chain with real-valued clock variables that capture the time spent while listening to tracks, resulting in an STA (see Def. 4). Here, we use the track durations as logged in the MSSD.

Figure 3 depicts a simplified UJG, including two user profiles and three tracks. The UJG defines one clock variable c that is used to model listening and pausing times. The clock progresses only in states where the user actively spends time by either listening to a track or taking a pause; all other states are marked as urgent, meaning that they progress to the next state immediately. For simplicity, we define constants for short and long pauses that determine the duration during which a user can dwell in the paused states. The states where a

track is selected are also urgent and lead immediately to the skip or not skipped states while setting a minimum and maximum listening time. The *maximum listening time* is set to be the maximum track duration in minutes over all tracks aggregated in the targeted track state. The *minimum listening time* is set to be the minimum track duration over all aggregated tracks in the targeted track state, if the track is not skipped. Otherwise, when the track is skipped, it is set to be the aggregated minimum minus one, capped to be at least one. For example, the state not_skipped_1 of the UJG in Fig. 3, defines an invariant limiting the dwell time to the maximum listening time, where the earliest possible time to leave is regulated by the minimum listening time.

We extend the STA to an SPTA (Sect. 2.4) by adding *price* variables for duration, acousticness, danceability, and energy. Transitions from song and played states update variables storing update ratios for each price variable, which are set as invariant in the skipped and not skipped states. We use discretized values for the continuous features as ratios; e.g., listening to a long track with high energy accumulates more energy than listening to a short track with high energy or a long track with low energy. Skipping sets each ratio to a constant of one; a basic increase is accumulated but insignificant. Duration is accumulated at the same rate as time passes. For readability, the music-streaming UJG shown in Fig. 3 only considers one feature, acousticness. In the example, the played track played_X has high acousticness, which is considered if the track is not skipped.

In the final step of our transformation, we build a UJG by partitioning the set of actions into *controllable* and *uncontrollable* actions, see Fig. 1. We resolve the stochastic observations in the learned Markov chain that are controlled by the service provider by neglecting the learned probabilities and defining these actions to be deterministic. For these actions, the service provider can deterministically propose a track song_i from the controllable contexts. In Fig. 3 uncontrollable actions are dashed, whereas controllable ones are solid. For example, the state editorial_playlist defines a listening context that is controlled by the service provider, therefore song_Y and song_Z can be selected by the controller. However, skipping a track is, e.g., controlled by the user. To account for the user behavior in the event log, we annotate the uncontrollable transitions with the probabilities in the learned Markov chain. For example, if the controller chooses song_Y, then the environment skips the track with a probability of 0.1.

By controlling the confidence bound in Alergia, we control the sensitivity by which states are considered to be equal and thus merged. Therefore, the confidence parameter α, see Sect. 2.2, allows to influence the structure of resulting music-streaming user journeys. We illustrate the effect of α by a simplified UJG shown in Fig. 4, which includes two tracks song_X and song_Y. In these user journeys, the probabilities of skipping song_X change, if the user previously listened to song_X. Choosing a higher α results in journeys as depicted in Fig. 4, where we can see two states representing song_X.

In the future, we plan to extend the model generation to Markov decision processes, which will allow us to have a precise differentiation between actions that are controllable by the service provider and by the user.

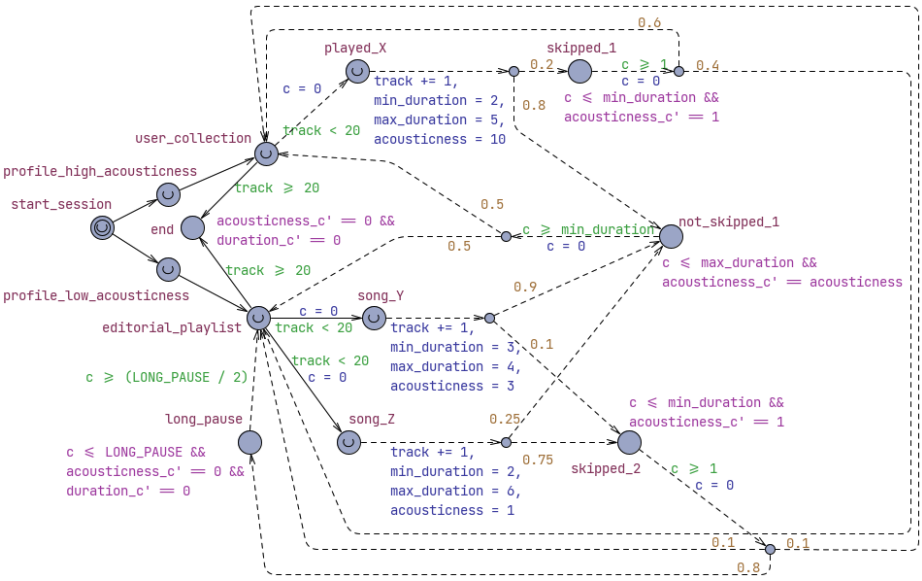


Fig. 3: User journey game modeling music-streaming user journeys that contain some elements of the basic structure shown in Fig. 1. The model considers two user profiles (`profile_high_acousticness`, `profile_low_acousticness`) and three different track categories (`played_X`, `song_Y`, `song_Z`).

3.6 Synthesizing Nudging Strategies

We use Uppaal to generate service provider strategies to nudge a user from her user profile towards a target profile by synthesizing a strategy that optimizes track features towards the target profile. According to the differences between the concrete start (the professor) and target (the student) profiles (Sect. 3.3), we minimize acousticness and danceability, and maximize energy and duration; visiting the start profile is encoded as a boolean flag. To evaluate the achieved nudging, we compare it to a random strategy, which selects tracks randomly.

Uppaal synthesizes strategies by optimizing values, including clocks, represented by variables that can be used to define verification queries. For example, to maximize acousticness in the UJG shown in Fig. 4, we could propose song_X from the start, risking many skips. However, by first proposing song_Y and then—unless this track is skipped—proposing song_X, the chances of the user not skipping this track increase, thereby increasing the accumulated acousticness. The strategy synthesized by Uppaal should comply with this observation. The strategy for nudging a user towards a different user profile is computed by:

```
strategy nudge = maxE(-danceability_c - 3*acousticness_c + energy
  +duration_c) [<= 1000]{MusicJourney.location, visited_start}
    ->{danceability_c, acousticness_c, duration_c, energy_c}:
      <> MusicJourney.end && visited.
```

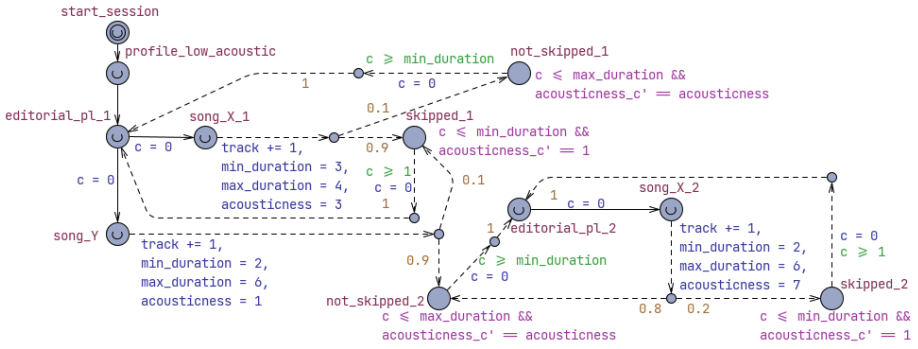


Fig. 4: Concatenated UJG.

We introduce the flag `visited`, which is set to TRUE when the state startprofile for the given start profile **s** was visited, and a weight of 3 for the acousticness as its expected maximum under a random strategy is only a third of danceability and energy. To facilitate more efficient learning, we consider only `MusicJourney.location`, `visited_start`, and the four price clocks to be observable. After generating the nudging strategy, we evaluate the expected achieved maximum for feature

$$\texttt{f} \in \{\texttt{duration\_c}, \texttt{danceability\_c}, \texttt{acousticness\_c}, \texttt{energy\_c}\}$$

under strategy `nudge` for 10 000 runs:

```
E[<=1000; 10000] (max: f) under nudge
```

and compare it to the achieved maximum by a random strategy:

```
E[<=1000; 10000] (max: f).
```

The exported strategy stores a mapping from states to actions and the reward for each action in each state. We parse the file to gain insights into which actions are recommended and how the nudging traverses the UJG.

4 Evaluation

We generate a UJG from the event log based on the MSSD. We then synthesize a strategy to nudge a person from a given start user profile towards a target user profile. The quantified achieved nudging is measured and collected when analyzing the different UJGs resulting from the different learned automata, when using Alergia with different confidence bounds. As described in Sect. 3.6, we then compare the expected feature values of our synthesized strategy with the expected feature values of a random strategy. Additionally, we investigate the

Table 2: Comparison of generated nudging strategies and random track selection. The aim is to decrease acousticness and danceability, and increase duration and energy. The presented numbers show improvement of nudging in percentage, averaged over 10 synthesized strategies, for different confidence values α.

α	Feature	random	nudge
0.1	Acousticness	90.35	-42.83%
0.1	Danceability	232.15	-2.77%
0.1	Duration	50.09	+14.82%
0.1	Energy	235.13	+37.34%
0.6	Acousticness	90.54	-37.29%
0.6	Danceability	232.28	-3.63%
0.6	Duration	50.10	+13.84%
0.6	Energy	235.49	+39.62%
0.9	Acousticness	90.12	-36.46%
0.9	Danceability	232.31	+1.92%
0.9	Duration	50.09	+17.35%
0.9	Energy	235.33	+45.43%

α	Feature	random	nudge
0.999	Acousticness	89.70	-40.49%
0.999	Danceability	232.03	-6.98%
0.999	Duration	50.15	+9.84%
0.999	Energy	235.11	+33.56%
0.9999	Acousticness	90.09	-40.0%
0.9999	Danceability	232.00	-2.4%
0.9999	Duration	50.07	+9.5%
0.9999	Energy	235.52	+29.73%
0.9999999999999	Acousticness	90.26	-39.5%
0.9999999999999	Danceability	232.38	-4.33%
0.9999999999999	Duration	50.08	+13.33%
0.9999999999999	Energy	234.93	+39.26%

path traversed by executing the nudging strategy and highlight the best tracks for nudging a professor to *"take a path on the wild side"* of musical taste.

All experiments were performed on a laptop with 32 GB memory and an i7-1165G7 Intel processor, and were implemented in Jupyter Notebooks using Python 3.10.6. The resources are published in an online repository.[5] For automata learning, we used AALpy [38], version 1.3.3, and for the synthesis of the strategies, we used Uppaal 5.0.0.[6]

In Table 2, we collect estimates for all four features under the nudging strategy against the random strategy, entries for nudging are given in percentage of the measured change in comparison to random. Note that our nudging strategy aims to minimize acousticness and danceability while maximizing duration and energy. An experiment (one entry in the table) consists of synthesizing 10 strategies with the default parameters for Uppaal's Q-learning implementation [47], we then average the calculated estimates. The synthesis of multiple strategies aims to compensate for imprecisions due to the limited exploration budget of the Q-learning setup for the given state space of the underlying UJG. We repeat these experiments for the different confidence values α that were used for learning the behavioral model in Alergia. As explained in Sect. 2.2, the confidence parameter α influences the state merging when learning the behavioral model. For the MSSD and its separation into different user profiles, we assume due to the variety of different listening behaviors that the dataset lacks multiple samples for equal user profiles. Therefore, we aim for higher α values considering that we have less confidence that the MSSD represents each user profile adequately. The number of states of the learned Markov chains varies between 6 734 and 6 851 states. In future work, it would be interesting to evaluate the influence of α by comparing the different learned user journeys from the same event log.

[5] https://github.com/smartjourneymining/spotify_journey/releases/tag/wang2024
[6] https://uppaal.org/

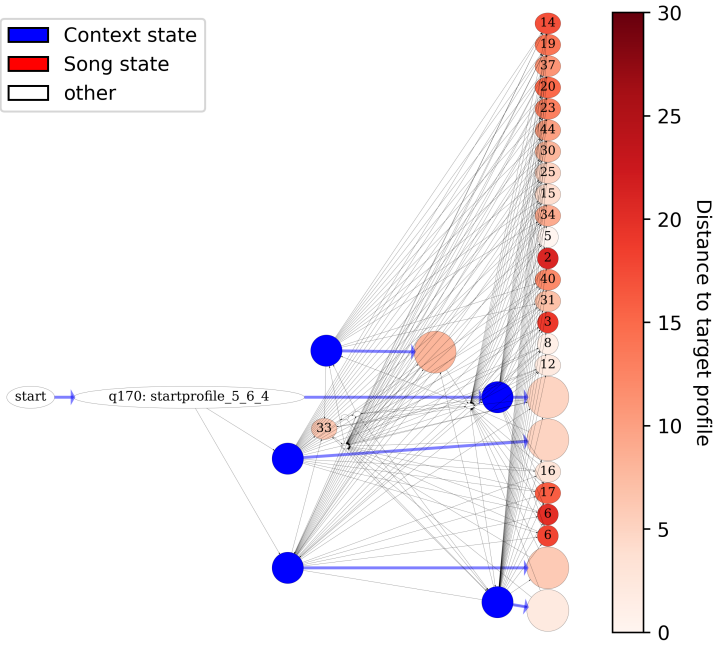


Fig. 5: Partial illustration showing the path in a user journey from the start profile of a professor towards a profile with a *wilder* taste in music.

The results presented in Table 2 show that the nudging strategy has a major influence on the user profile, often outperforming the random strategy, however danceability cannot always be reduced. The nudging strategy effectively influences the user towards the target profile by significantly changing all features except danceability; e.g., energy is increased by up to 45.43%. The results also show that the confidence parameter α has less influence on the synthesis of the nudging strategy, which suggests that we can develop a successful nudging strategy independently of the learned Markov chain.

Figure 5 shows parts of a learned UJG generated with the maximal confidence value used in the experiments (0.9999999999999). The depicted part of the UJG only shows the states explored by the synthesized strategy. For visibility, we removed any transition or state labels and applied a color-encoding for state types. Therefore, Fig. 5 abstracts from the time constraints of the original generated UJG. We color song states in a range of red tones, context states in blue, and other states in white. To calculate the red tones of states, we first encode the distance between track features $\mathbf{f}$ of a track and the target profile $\mathbf{t}$ using the sum of the element-wise distances, i.e. $\sum_i |\mathbf{t}_i - \mathbf{f}_i|$, we then map these results into a range of red tones, where the larger the distance to the target, the darker the color with red tonalities. We highlight the best selection of controllable actions from explored contexts with blue transitions and their corresponding song states

are enlarged. Figure 5 also shows transitions to darker red states that could be traversed earlier in the nudging strategy to reach context states with blue transitions. The depicted UJG merges other song states into abstracted song states, aggregating song states with the same track feature distance. The aggregated states are labeled with the number of clustered song states. For example, the uppermost red state aggregates 14 concrete song states, all of which have the same distance to the target profile. The figure shows that most of the explored tracks are closer to the target profile (states with lighter red tonalities), and only a few states with darker red tonalities have a high count number.

Finally, we investigate the best actions in each state by parsing the path generated by the nudging strategy and selecting the actions leading to states with a maximal associated value. When guiding the user from the start profile, the nudging strategy selects, e.g., a track with the feature vector $\langle 1, 3, 5\rangle$, that lies in between the starting profile $\mathbf{s} := \langle 5, 6, 4\rangle$ and the target profile $\mathbf{t} := \langle 0, 3, 9\rangle$. The average feature vector of the five selected songs with the smallest distance to the target is $\langle 1.2, 4.6, 7\rangle$. One of the five tracks has an aggregated distance of 1 to the target, three tracks have an acousticness of one or smaller, two tracks have a danceability of seven, and four tracks have an energy of seven or above.

5 Related Work

The Music Streaming Sessions Dataset used was proposed by Brost *et al.* [12] in the context of skip prediction, to spur progress for counterfactual analysis. The dataset triggered a wide range of work on skip prediction using, e.g., recurrent neural networks [15,25,49] as well as on recommender systems [41,43]. Brost *et al.* consider several research directions that could be enabled by the published dataset, among others, research challenges related to user journeys. Going in this direction, our work uses the dataset to automatically build a user-centric model of listening behavior to nudge users towards different musical taste.

Spotify. Zhang *et al.* [48] investigate user behavior from Sweden, the United Kingdom, and Spain on a Spotify dataset from 2010–2011. They found statistical patterns in several aspects of the user journey: (1) daily patterns in starting a session, playback arrival, and session length, (2) switching behavior between different devices, (3) the favorite time for using Spotify, and (4) correlations between successive sessions and downtimes. Compared to our work, the authors had access to non-disclosed user information, allowing them to identify consecutive user sessions, and to provide statistical insights also on a global level. In contrast, we learned a model formalizing a music-streaming user journey based only on publicly available data describing finite streaming sessions, which were non-relatable to concrete users. Furthermore, we use model-checking tools to generate strategies optimizing specific aspects of the user journey, e.g., nudging a user towards a different music genre. Anderson *et al.* [6] use a non-public, massive Spotify dataset recorded over a year of sessions, with more than 100 million distinct users and 70 billion listened tracks, to investigate the diversity of listening behavior and churn, and to advocate diversity in recommender systems.

Similar to our work, they distinguish between tracks that are selected by the user and by an algorithm. Meggetto *et al.* investigate skip behavior in music recommendations [36], using clustering to propose four different categories of skipping behavior, and propose a deep reinforcement learning model using explainable AI techniques to discover the most relevant features for skip prediction [37].

Automata learning and model checking. Automata learning has previously been used to extract finite state machines (FSM) from event logs. Cook and Wolf [17] propose a tool to learn FSMs of software processes. Similar to our work, they also consider the Markov property to construct FSMs from event logs. However, in contrast to our technique, they neglect the learned probabilities in their created models and require manual post-processing of the learned model. Agostinelli *et al.* [2] studied different automata learning algorithms for learning FSMs from event logs, assuming the availability of traces that are not part of SUL. Esparza *et al.* [20] present an algorithm for learning Petri nets that actively interacts with the SUL to complete the sample. Aichernig *et al.* [3,5] extend existing models by timed values to create STAs. They create the time variables by actively sampling the system using property-based testing techniques. Similar to our work, they then use the generated STAs for statistical model checking. Automata learning has also been used to learn timed behavior directly from a set of timed traces by applying a meta-heuristic search [4, 44]. Most closely related to this work is our previous work on learning stochastic multi-player games from event logs [30]. In contrast to [30], we here use automata learning to create Markov chains instead of Markov decision processes, as we assume users to be fully stochastic in their actions and the streaming service to be deterministic.

6 Conclusion

Summary. In this paper, we show how automata learning and statistical model checking can be used to generate nudging strategies for user behavior. The proposed method is demonstrated on a case study of user behavior for a music-streaming service. We show that the method can be used to answer questions such as whether the service can nudge a professor to be adventurous and take a path on the "wild side" through the user journey game. In the case study, We learned Markov chains from the Spotify Music-Streaming Sessions Dataset (MSSD) [12] and mined user profiles from a professor in computer science and a PhD student. Afterward, we enriched the Markov chain to a user journey game by adding timed behavior, cost variables and (un)controllable actions. The generated nudging strategies noticeably influence the listening behavior when compared to a random strategy on a set of musical features.

Discussion. The automatic generation of user journey games in this paper is based on Markov chains. We justify this approach by assuming that, for the considered case study, user behavior follows a stochastic process that can be described by a finite state structure fulfilling the Markov property. For the generation of the user journey game, we define some actions as controllable, ignoring

their probabilities, and make them deterministic. Uncontrollable actions reflect learned stochastic behavior from the underlying dataset. Therefore, the generated user journey games do not have any uncontrollable deterministic actions. Other case studies might require an adaption regarding this modeling technique. The construction of user journey games in this paper approximated time boundaries and constants using averaged values. Other techniques might capture these values more precisely, but would require more in-depth expertise on the given data. Furthermore, we did not study differences between Markov chains learned with different parameters that reflect the confidence in the underlying distribution of the dataset. Our results show that a successful nudging strategy can be synthesized regardless of the parameters. For other analysis techniques, the influence of the confidence parameter might be more critical.

Our work derives user profiles from the MSSD and, based on these profiles, creates a user journey model that defines the listening behavior of users. By constructing UJGs, we also synthesize strategies to nudge a user towards a different musical taste. However, we have not explored how the synthesized strategy actually works when applied to the user. There could be a risk that actual user behavior might not align as planned when a user is actively nudged by the underlying service or application. Indeed, validating the synthesized strategy would be very interesting, but would require advanced access to the music-streaming service to steer the music recommendations based on the features we derived as well as the logging mechanism for the experienced user journey.

Future Work. This paper opens a range of possibilities for future work. In our experiments, the strategies generated with the default Q-learning parameters in Uppaal explore only a small fraction of the state space leading to possible fluctuations in quality. We plan to investigate techniques to improve the synthesis of strategies by adapting the Q-learning technique. This work only considered the small version of the MSSD. In the future, we will examine the feasibility of automata learning for massive datasets as presented in the challenge.

Another interesting extension of our work would be lifting the synthesized strategies to more easily understandable actions. or our nudging strategy, we could, e.g., suggest specific genres that should be suggested to the users with a certain profile for adapting their taste. In previous work [30], we already worked towards making UJGs easier to understand by illustrating model-checking results using Sankey diagrams that enable the service bottleneck analysis.

Acknowledgments We thank the providers of user profiles, and Florian Lorber for his help with designing the Uppaal model.

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. 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. Inf. Syst. **114**, 102180 (2023), https://doi.org/10.1016/j.is.2023.102180
3. Aichernig, B.K., Bauerstätter, P., Jöbstl, E., Kann, S., Korosec, R., Krenn, W., Mateis, C., Schlick, R., Schumi, R.: Learning and statistical model checking of system response times. Softw. Qual. J. **27**(2), 757–795 (2019), https://doi.org/10.1007/s11219-018-9432-8
4. Aichernig, B.K., Pferscher, A., Tappler, M.: From passive to active: Learning timed automata efficiently. In: Lee, R., Jha, S., Mavridou, A. (eds.) Proc. 12th International NASA Formal Methods Symposium (NFM 2020). Lecture Notes in Computer Science, vol. 12229, pp. 1–19. Springer (2020), https://doi.org/10.1007/978-3-030-55754-6_1
5. Aichernig, B.K., Schumi, R.: How fast is MQTT? - Statistical model checking and testing of IoT protocols. In: McIver, A., Horváth, A. (eds.) Proc. 15th International Conference on Quantitative Evaluation of Systems (QEST 2018). Lecture Notes in Computer Science, vol. 11024, pp. 36–52. Springer (2018), https://doi.org/10.1007/978-3-319-99154-2_3
6. Anderson, A., Maystre, L., Anderson, I., Mehrotra, R., Lalmas, M.: Algorithmic effects on the diversity of consumption on Spotify. In: Huang, Y., King, I., Liu, T., van Steen, M. (eds.) Proc. The Web Conference 2020 (WWW'20). pp. 2155–2165. ACM / IW3C2 (2020), https://doi.org/10.1145/3366423.3380281
7. Angluin, D.: Identifying languages from stochastic examples. Tech. rep., Yale University (1988), https://cpsc.yale.edu/sites/default/files/files/tr614.pdf
8. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., Lime, D.: UPPAAL-Tiga: Time for playing games! In: Damm, W., Hermanns, H. (eds.) Proc. 19th International Conference on Computer Aided Verification (CAV 2007). Lecture Notes in Computer Science, vol. 4590, pp. 121–125. Springer (2007), https://doi.org/10.1007/978-3-540-73368-3_14
9. Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Priced timed automata: Algorithms and applications. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.P. (eds.) Proc. Third International Symposium, on Formal Methods for Components and Objects (FMCO 2004). Lecture Notes in Computer Science, vol. 3657, pp. 162–182. Springer (2005), https://doi.org/10.1007/11561163_8
10. Bjørk, J., de Boer, F.S., Johnsen, E.B., Schlatte, R., Tapia Tarifa, S.L.: User-defined schedulers for real-time concurrent objects. Innov. Syst. Softw. Eng. **9**(1), 29–43 (2013), https://doi.org/10.1007/S11334-012-0184-5
11. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N.: Quantitative analysis of real-time systems using priced timed automata. Commun. ACM **54**(9), 78–87 (2011), https://doi.org/10.1145/1995376.1995396
12. Brost, B., Mehrotra, R., Jehan, T.: The music streaming sessions dataset. In: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L. (eds.) Proc. World Wide Web Conference (WWW 2019). pp. 2594–2600. ACM (2019), https://doi.org/10.1145/3308558.3313641
13. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Carrasco, R.C., Oncina, J. (eds.) Proc. 2nd International Colloquium 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

14. Cassez, F., David, A., Fleury, E., Larsen, K.G., Lime, D.: Efficient on-the-fly algorithms for the analysis of timed games. In: Abadi, M., de Alfaro, L. (eds.) Proc. 16th International Conference on Concurrency Theory (CONCUR 2005), Lecture Notes in Computer Science, vol. 3653, pp. 66–80. Springer (2005), https://doi.org/10.1007/11539452_9
15. Chang, S., Lee, S., Lee, K.: Sequential skip prediction with few-shot in streamed music contents. CoRR **abs/1901.08203** (2019), http://arxiv.org/abs/1901.08203
16. Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: PRISM-games: A model checker for stochastic multi-player games. In: Piterman, N., Smolka, S.A. (eds.) Proc. 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2013). Lecture Notes in Computer Science, vol. 7795, pp. 185–191. Springer (2013), https://doi.org/10.1007/978-3-642-36742-7_13
17. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Methodol. **7**(3), 215–249 (1998), https://doi.org/10.1145/287000.287001
18. David, A., Jensen, P.G., Larsen, K.G., Legay, A., Lime, D., Sørensen, M.G., Taankvist, J.H.: On time with minimal expected cost! In: Cassez, F., Raskin, J. (eds.) Proc. 12th International Symposium pn Automated Technology for Verification and Analysis (ATVA 2014), Lecture Notes in Computer Science, vol. 8837, pp. 129–145. Springer (2014), https://doi.org/10.1007/978-3-319-11936-6_10
19. David, A., Jensen, P.G., Larsen, K.G., Mikucionis, M., Taankvist, J.H.: UPPAAL Stratego. In: Baier, C., Tinelli, C. (eds.) Proc. 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015). Lecture Notes in Computer Science, vol. 9035, pp. 206–211. Springer (2015), https://doi.org/10.1007/978-3-662-46681-0_16
20. Esparza, J., Leucker, M., Schlund, M.: Learning workflow Petri nets. Fundam. Informaticae **113**(3-4), 205–228 (2011), https://doi.org/10.3233/FI-2011-607
21. 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 (2006), https://doi.org/10.1509/jmkg.70.1.003.qxd
22. Gold, E.M.: Language identification in the limit. Inf. Control. **10**(5), 447–474 (1967), https://doi.org/10.1016/S0019-9958(67)91165-5
23. 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
24. 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
25. Hansen, C., Hansen, C., Alstrup, S., Simonsen, J.G., Lioma, C.: Modelling sequential music track skips using a Multi-RNN approach. CoRR **abs/1903.08408** (2019), http://arxiv.org/abs/1903.08408
26. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58**(301), 13–30 (1963), https://doi.org/10.2307/2282952
27. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: A core language for abstract behavioral specification. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) Proc. 9th International Symposium on Formal

Methods for Components and Objects FMCO 2010. Lecture Notes in Computer Science, vol. 6957, pp. 142–164. Springer (2010), https://doi.org/10.1007/978-3-642-25271-6_8
28. Johnsen, E.B., Schlatte, R., Tapia Tarifa, S.L.: Integrating deployment architectures and resource consumption in timed object-oriented models. J. Log. Algebraic Methods Program. **84**(1), 67–91 (2015), https://doi.org/10.1016/J.JLAMP.2014.07.001
29. Kobialka, P., Mannhardt, F., Tapia Tarifa, S.L., Johnsen, E.B.: Building user journey games from multi-party event logs. In: Proc. 3rd Intl. Workshop on Event Data and Behavioral Analytics (EdbA 2022). LNBIP, vol. 468. Springer (2022), https://doi.org/10.1007/978-3-031-27815-0_6
30. Kobialka, P., Pferscher, A., Bergersen, G.R., Johnsen, E.B., Tapia Tarifa, S.L.: Stochastic games for user journeys. In: Platzer, A., Pradella, M., Rossi, M., Rozier, K.Y. (eds.) Proc. 26th International Symposium on Formal Methods (FM 2024). Lecture Notes in Computer Science, Springer (2024), to appear
31. 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
32. Kobialka, P., Tapia Tarifa, S.L., Bergersen, G.R., Johnsen, E.B.: Weighted games for user journeys. In: Proc. 20th International Conference Software Engineering and Formal Methods (SEFM 2022). Lecture Notes in Computer Science, vol. 13550, pp. 253–270. Springer (2022), https://doi.org/10.1007/978-3-031-17108-6_16
33. 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
34. Larsen, K.G., Behrmann, G., Brinksma, E., Fehnker, A., Hune, T., Pettersson, P., Romijn, J.: As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In: Berry, G., Comon, H., Finkel, A. (eds.) Proc. 13th International Conference on Computer Aided Verification (CAV 2001). Lecture Notes in Computer Science, vol. 2102, pp. 493–505. Springer (2001), https://doi.org/10.1007/3-540-44585-4_47
35. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. Int. J. Softw. Tools Technol. Transf. **1**(1-2), 134–152 (1997), https://doi.org/10.1007/S100090050010
36. Meggetto, F., Revie, C., Levine, J., Moshfeghi, Y.: On skipping behaviour types in music streaming sessions. In: Demartini, G., Zuccon, G., Culpepper, J.S., Huang, Z., Tong, H. (eds.) Proc. 30th ACM International Conference on Information and Knowledge Management (CIKM'21). pp. 3333–3337. ACM (2021), https://doi.org/10.1145/3459637.3482123
37. Meggetto, F., Revie, C., Levine, J., Moshfeghi, Y.: Why people skip music? On predicting music skips using deep reinforcement learning. In: Gwizdka, J., Rieh, S.Y. (eds.) Proc. Conference on Human Information Interaction and Retrieval (CHIIR 2023). pp. 95–106. ACM (2023), https://doi.org/10.1145/3576840.3578312
38. Muškardin, E., Aichernig, B.K., Pill, I., Pferscher, A., Tappler, M.: AALpy: an active automata learning library. Innov. Syst. Softw. Eng. **18**(3), 417–426 (2022), https://doi.org/10.1007/S11334-022-00449-3
39. Norman, G., Parker, D., Sproston, J.: Model checking for probabilistic timed automata. Formal Methods Syst. Des. **43**(2), 164–190 (2013), https://doi.org/10.1007/S10703-012-0177-X

40. Norris, J.R.: Markov chains. Cambridge series in statistical and probabilistic mathematics, Cambridge University Press (1998)
41. Ricci, F., Rokach, L., Shapira, B. (eds.): Recommender Systems Handbook. Springer (2015), https://doi.org/10.1007/978-1-4899-7637-6
42. 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
43. Schedl, M., Knees, P., McFee, B., Bogdanov, D.: Music recommendation systems: Techniques, use cases, and challenges. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 927–971. Springer (2022), https://doi.org/10.1007/978-1-0716-2197-4_24
44. Tappler, M., Aichernig, B.K., Larsen, K.G., Lorber, F.: Time to learn - learning timed automata from tests. In: André, É., Stoelinga, M. (eds.) Proc. 17th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2019). Lecture Notes in Computer Science, vol. 11750, pp. 216–235. Springer (2019), https://doi.org/10.1007/978-3-030-29662-9_13
45. Thaler, R.H., Sunstein, C.R.: Nudge: Improving decisions about health, wealth, and happiness. Penguin (2009)
46. Vandermerwe, S., Rada, J.: Servitization of business: Adding value by adding services. European Management Journal **6**(4), 314–324 (Dec 1988), https://doi.org/10.1016/0263-2373(88)90033-3
47. Watkins, C.J.C.H., Dayan, P.: Technical note Q-learning. Mach. Learn. **8**, 279–292 (1992), https://doi.org/10.1007/BF00992698
48. Zhang, B., Kreitz, G., Isaksson, M., Ubillos, J., Urdaneta, G., Pouwelse, J.A., Epema, D.H.J.: Understanding user behavior in Spotify. In: Proc. INFOCOM 2013. pp. 220–224. IEEE (2013), https://doi.org/10.1109/INFCOM.2013.6566767
49. Zhu, L., Chen, Y.: Session-based sequential skip prediction via recurrent neural networks. CoRR **abs/1902.04743** (2019), http://arxiv.org/abs/1902.04743