Data-driven modelling and probabilistic analysis of interactive software usage

Oana Andrei*, Muffy Calder

School of Computing Science, University of Glasgow, G12 8RZ, UK

Abstract

This paper answers the research question: how can we model and understand the ways in which users *actually* interact with software, given that usage styles vary from user to user, and even from use to use for an individual user. Our first contribution is to introduce two new probabilistic, admixture models, inferred from sets of logged user traces, which include observed and latent states. The models encapsulate the temporal and stochastic aspects of usage, the heterogeneous and dynamic nature of users, and the temporal aspects of the time interval over which the data was collected (e.g. one day, one month, etc.). A key concept is *activity patterns*, which encapsulate common observed temporal behaviours shared across a set of logged user traces. Each activity pattern is a discrete-time Markov chain in which observed variables label the states; latent states specify the activity patterns. The second contribution is how we use parametrised, probabilistic, temporal logic properties to reason about hypothesised behaviours within an activity pattern, and between activity patterns. Different combinations of inferred model and hypothesised property afford a rich set of techniques for understanding software usage. The third contribution is a demonstration of the models and temporal logic properties by application to user traces from a software application that has been used by tens of thousands of users worldwide.

Keywords: Interactive Software, Log Analysis, Usage Behaviour, Admixture Models, Latent Variables, Probabilistic Model Checking
62F15, 60J10, 68Q85, 68Q60

1. Introduction

Software developers cannot always anticipate how users will use their software: actual usage styles may be quite different from what the system designers envisaged, and/or they may change over time. Moreover users are heterogeneous: they may adopt different usage styles for the same software, furthermore, each individual user may move between different styles, from one interaction session to another, or even during a session. This

*Corresponding author
Email addresses: Oana.Andrei@glasgow.ac.uk (Oana Andrei), Muffy.Calder@glasgow.ac.uk (Muffy Calder)

may be for a variety of contextual reasons, for example, time of day, time since last session, length of session, purpose of engagement, environment (e.g. on a train or at a desk) or device (PC, tablet, wearable). The research question we address is how can we model and understand the ways in which users *actually* interact with a software system, over time?

When the software system is instrumented and we have time series data about sequences of user-initiated events from logged user traces (i.e., observed variables), the question can be refined to: how can we abstract data-driven models and then analyse them with the purpose of discovering, evaluating, and communicating meaningful patterns of usage (i.e., hidden or latent variables)? The aim of this paper is to show that new types of probabilistic models, inferred from logged user traces, and analysis using probabilistic temporal logic properties, provides a novel and effective solution.

First, we consider models and we define two new probabilistic models that encapsulate temporal and dynamic aspects of software usage. Existing models such as first-order hidden Markov models are not suitable because in addition to the relationships between latent and observed variables we also have relationships between observed states from each user trace. A key new concept is *activity patterns*, which encapsulate observed temporal behaviours. Each activity pattern is a discrete-time Markov chain in which observed variables label the states. We define two new parametrised, admixture discrete-time Markov models that are based on a finite number K of activity patterns (the mixture components). The new models include both observed and latent states: the former are within the activity patterns and the latter specify the activity patterns. The models are data-driven and we segment logged data into time series data encapsulating different usage time intervals, e.g. interactions over first month, second month, third month. We use maximum-likelihood parameter estimation techniques for inferring model parameters from different time series data.

Second, we consider model analysis using probabilistic temporal logics. We define classes of temporal properties (over observed and latent states) in the probabilistic temporal logics PCTL and PCTL* [1]. These properties encapsulate hypotheses about indicative temporal behaviours within an activity pattern, and between activity patterns. We encode the models and properties in the PRISM model checker [2] for automated reasoning. Combinations of different model and property, over different time intervals, afford a rich set of techniques for discovering, evaluating, and communicating patterns of usage. Additionally, we analyse by inspection of the distributions over activity patterns.

Figure 1 summarises our approach. It is important to note we are not inferring the underlying software structure, which is determined by the designed functionality of the software. We are investigating the differing generating processes of software *usage* over a dynamic, heterogeneous population of a users. The logged data are user-initiated events. We identify basic common traits of usage emerging within a population of users and classify users as *admixtures* of basic traits. In statistical modelling, in a mixture model each user trace is mapped to one single behavioural trait, whereas in admixture models each user trace has a distribution over all behavioural traits. To the best of our knowledge, inferring admixture temporal structures has not been described in prior work outside our group.

In summary, the contributions of the paper are:

- two new admixture latent state Markov models: population admixture model

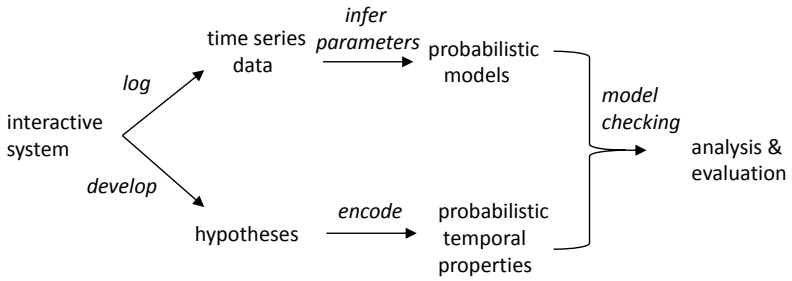


Figure 1: Overall approach for data-driven modelling and analysis of software usage.

(PAM) and generalised population admixture model (GPAM),

- flattened form of the GPAM model as discrete-time Markov chain and encoding in the probabilistic model checker PRISM,
- probabilistic temporal logic properties that can be applied to either or both an activity pattern or/and a generalised population admixture model,
- guidance on interpretation and comparison of analysis results for combinations of PAM and GPAM models and the temporal logic properties,
- a demonstration of the approach through application to data sets from a user-intensive mobile software application that has been used by tens of thousands of users worldwide.

In the next section we describe how logged data is segmented into time series data. In Sect. 3 we discuss the requirements for models of software usage and in Sect. 4 we define the PAM and GPAM models and parameter inference. In Sect. 5 we define two classes of temporal logic properties: single activity pattern and multiple activity pattern properties, and in Sect. 6 we discuss how to interpret and compare results across different models, noting that the inference process can generate the activity patterns in different orders, for each model. In Sect. 7 we introduce our example interactive software, the mobile application *AppTracker* [3], which has been downloaded more than 35,000 times. Section 8 gives an overview of our analysis results, using PAM and GPAM models, single and multiple activity pattern properties, and distributions over activity patterns in PAM models. We reflect on our results, our contribution and related work in Sect. 9, and conclude in Sect. 10.

2. From raw logged data to user traces

We assume that the users' interactions with software are logged via features embedded in the software. For instance we used the SGLog framework [4] to collect records in a MySQL database about the user, the device, the event that took place and the timestamp. We are interested only in the events leading to a switch between views within the software, hence these events are determined by the software functionality and they

give an appropriate level of abstraction. We processed the logs to obtain discrete time series data of timestamped user-initiated events for each device unique identifier. We call such discrete time series *user traces* with the caveat that each user trace corresponds to a unique device identifier since a user may be running the same software on several devices. In the remainder of this paper, we conflate devices with users.

Assume we have a population of M user traces $\{\alpha^1, \ldots, \alpha^M\}$, and n pairwise distinct event labels $\{l_1, \ldots, l_n\}$ occurring in these traces. While it is not fundamental to our approach, we assume here that logged interactions include two distinguished events `UseStart` and `UseStop` that denote the beginning and end of each user interaction *session* respectively. A session begins with the application being launched or being brought to the foreground, and ends when the application is closed or sent to the background. Assuming there are no logging errors (or at least such errors are fixed during a data cleaning process), each user trace is a concatenation of sessions. We expect to discard trivial user traces such as traces containing only session start and end `UseStart` and `UseStop` event labels, or containing only fewer than five sessions.

In summary, each user trace is a temporal ordering of events that contains a variable number of sessions, and each session is a variable-length sequence of timestamped events.

Because we have timestamps for session start and end, we can segment a set of traces according to intervals of the form $[d_1, d_2]$, which includes consecutive sessions from the d_1^{th}-th up to the d_2^{th} day of usage. For example, logged data for the first month of usage is the set of user traces in the interval [0,30]. Timestamps are discarded after segmentation.

Segmented data sets of user traces form the basis of our data-driven modelling and analysis approach; we now consider models of usage behaviour that we infer from the data sets.

3. On data-driven models of software usage

Our experience indicates that the most useful usage models depend not (only) on static attributes such as the location, gender, age of users, but on the *dynamic* behavioural patterns we observe through logged interactions. This means we are interested in more than a "bag of words" approach [5] to the representation of user behaviours: we require models that expose the temporal relationships between events and behavioural patterns, for sets of users.

Motivated by [6, 7], we start from the assumption of the existence of a common set of behavioural patterns that can be estimated from all observed user traces, and that by allowing them to interleave, we can account for more complex individual and population behaviour. This means we require statistical models that allow us to focus on *activity patterns* of software usage, and their occurrence within heterogeneous *populations of users*, rather than simply within individuals. We expect activity patterns to be dynamic in the sense that the observed pattern changes over time, both for an individual user and for a population. For example, each individual user may move between different patterns, from one interactive session to another, or even during a session.

The key questions when defining and selecting usage models are: how do we model each activity pattern and how do we model the multiple generating processes, i.e. the interleavings, within a population? In the following we discuss these questions informally; in Section 4 we give formal definitions of discrete-time Markov models, hidden Markov models, and the new admixture Markov models we use in our analysis.

3.1. Activity patterns

The simplest and most common statistical models for time dependent sequences of words (such as our user traces) are first order Markov chains, and we will use these to model activity patterns. Since we consider events generated sequentially in discrete time as opposed to continuous time, activity patterns are formally modelled as discrete-time Markov chains defined as follows.

Let $\mathcal{A}$ be a finite set of atomic propositions. A **first-order discrete-time Markov chain (DTMC)**, is a tuple $(S, \bar{s}, \mathbf{P}, \mathcal{L})$ where: S is the set of states; $\bar{s} \in S$ is the initial state, $\mathbf{P} : S \times S \to [0, 1]$ is the transition probability function (or matrix) such that for all states $s \in S$ we have $\sum_{s' \in S} \mathbf{P}(s, s') = 1$; $\mathcal{L} : S \to 2^{\mathcal{A}}$ is a labelling function associating to each state s in S a set of valid atomic propositions from the set $\mathcal{A}$. A *path* (or execution) of a DTMC is a non-empty sequence $s_0 s_1 s_2 \ldots$ where $s_i \in S$ and $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. For DTMC $\mathcal{D}$ the set of paths starting from state s is $Path^{\mathcal{D}}(s)$. A transition is also called a *time-step*.

The Markovian property means we consider only (memoryless) relationships between states, and first-order means modelling relationships between pairs of adjacent symbols (events) in the user traces. Markovian models are computationally tractable. We note that first-order Markov models have been used for modelling in many related problems such as: human navigation on the Web [8, 9, 10, 11, 12] where states correspond to visited webpages, usability analysis [13] where states correspond to button presses in general, mobile applications [14, 15] where states correspond to device screen events, human interactions with a search engine [16], human-human interactions in social group meetings [17].

While higher-order Markov models could deliver higher modelling accuracy, they could also increase complexity because they encode tuples of states; note that any higher-order model can be encoded as a first-order Markov models. We employ first-order models for activity patterns because the words (event labels) in the user traces correspond directly to states in the Markov chain, we do not know a priori which tuples of states are likely to be most significant or which higher order would give most benefit, and the temporal logic properties will allow us to reason over paths within (and between) activity patterns.

3.2. Multiple Generating Processes

We have more than a clustering problem: each user trace is not modelled by one activity pattern, but at different points in time it may be modelled by a different activity pattern. We therefore take an *admixture* approach. The admixture concept derives from genetics and the analysis of populations where individuals have mixed ancestry: in mixture models each individual has a probability of inheriting a trait, while in admixture models each individual inherits a fraction of his/her traits from ancestors. Here we use the more general terminology of component, rather than trait.

We define two new admixture models, each of a fixed number, K, of components. The components are activity patterns and each of them is generated by a different process. We can also say that for each admixture model we have a K-way categorical variable (whose values range from 1 to K) with each value k, $1 \leq k \leq K$, denoting an activity pattern. Each activity pattern has the same set of underlying states, but the transition probabilities between states are different in each activity pattern.

3.3. Latent Variable Models

The standard latent variable model is the hidden Markov model [18], a Markov model distinguish between observed states and unobserved (or latent) states; the observed states are generated by the latent states.

A **first-order hidden Markov model (HMM)** is a tuple $(\mathcal{X}, \mathcal{Y}, \pi, A, B)$ where: $\mathcal{X}$ is the set of hidden (or latent) states $\mathcal{X} = \{1, \ldots, K\}$; $\mathcal{Y}$ is the set of observed states generated by hidden states; $\pi : \mathcal{X} \to [0, 1]$ is an initial distribution, where $\sum_{x \in \mathcal{X}} \pi(x) = 1$; $A : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is the transition probability matrix, such that for all $x \in \mathcal{X}$ we have $\sum_{x' \in \mathcal{X}} A(x, x') = 1$; $B : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ is the observation probability matrix, such that for all $x \in \mathcal{X}$ we have $\sum_{y \in \mathcal{Y}} B(x, y) = 1$.

One can regard HMMs as a generalisation of mixture models in which the hidden variables define the mixture components. However, in our context, this would mean observed states (i.e., the user-generated events) would be in the components, i.e. the activity patterns, and HMMs express relationships only between latent states, and not between observed states. Thus HMMs are not suitable for our purposes. But the distinction between latent and observed is useful: we just require a more nuanced approach to components and their combination, building on the distinction between observed states for activity patterns and unobserved variables for mixing. We will define two new models that meet this requirement.

In summary, to encapsulate the behaviours of dynamic and heterogeneous users, we define two new models that are:

- *probabilistic* – statistical models of the different generating processes of heterogeneous, dynamic users,
- *Markovian* – behaviour is determined by current state, not process history,
- *admixture* – usage styles are complex and drawn from a probability distribution, representing individuals that move between different patterns during an observed trace.

The two models are called Population admixture model (PAM), which is an *admixture of discrete-time Markov chains*, and Generalised population admixture model (GPAM), which is an *auto-regressive hidden Markov model*.

4. Population Admixture Models (PAMs) and Generalised PAMs

A pictorial representation of the PAM and GPAM models is given in Fig. 2. Both models involve K latent states, and the same set of observed states in the component activity patterns. The main difference between the two models is the way in which activity patterns are combined: in PAM models, there is a set of distributions $\Theta = \{\theta_m\}_{m=1,M}$, with θ_m the distribution over the activity patterns for the m-th user trace, whereas in GPAM models there is a matrix A that indicates the likelihood of transition between the latent states (and thus between activity patterns).

The number of mixture components/activity patterns, K, is an exploratory tool: we do not try to find the "correct" or optimal value for K, instead we explore the variety of usage styles that are meaningful to software evaluation (e.g. we might develop distinct software versions for three or four different usage styles). We now define the models formally.

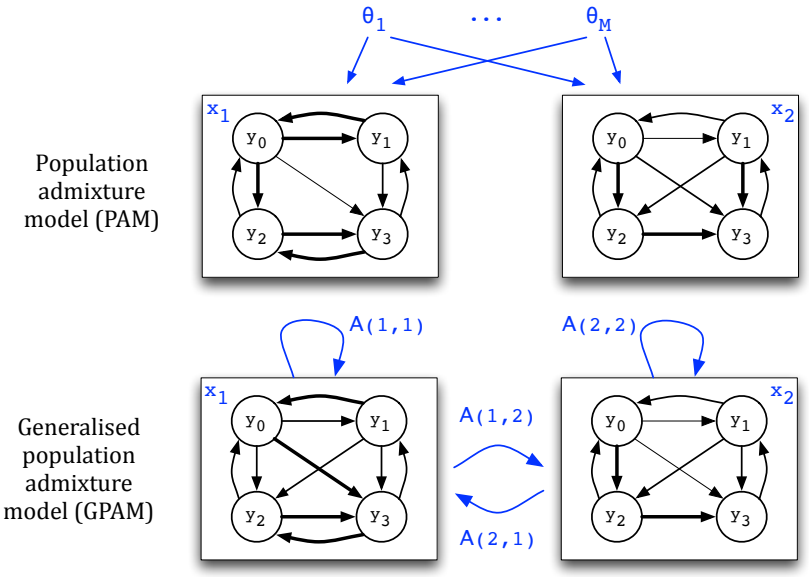


Figure 2: Pictorial representation of PAM and GPAM for $K = 2$, two latent states x_1 and x_2, and four observed states $y_0 - y_3$. The two activity patterns for each of the models (either PAM or GPAM) are the DTMCs in boxes. Transition probabilities in each DTMC are indicated by the thickness of transitions. There are M user traces with θ_m the probabilistic distribution of the m-th user trace over the K activity patters, for $1 \leq m \leq M$.

4.1. Definitions

Hidden Markov models do not include transitions between observed states, but only between hidden states. However the less common auto-regressive hidden Markov model (AR-HMM) [18] includes both transitions between observed states and transitions between latent states. Formally, a **first-order auto-regressive hidden Markov model** is a tuple $(\mathcal{X}, \mathcal{Y}, \pi, A, B)$ where: $\mathcal{X}, \mathcal{Y}, \pi, A$ are as defined for HMMs; $B : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ is the observation probability matrix, such that for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ we have $\sum_{y' \in \mathcal{Y}} B(x, y, y') = 1$.

Let $\mathcal{P}$ be a population of user traces as defined in Section 2 and $\mathcal{A}$ the set of the labels of all events occurring in $\mathcal{P}$.

Definition 1 (PAM). *For a given positive integer K, a* **population admixture model** *for the user trace population $\mathcal{P}$ is a tuple $(\mathcal{X}, \mathcal{Y}, B, \mathcal{L}, \Theta)$ where:*

- *$\mathcal{X}$ is the set of latent states, $\mathcal{X} = \{1, \ldots, K\}$,*
- *$\mathcal{Y}$ is the set of observed states, $\mathcal{Y} = \{0, \ldots, n-1\}$,*
- *B is the observation probability matrix with $B : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ such that for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ we have $\sum_{y' \in \mathcal{Y}} B(x, y, y') = 1$,*
- *$\mathcal{L} : \mathcal{Y} \to \mathcal{A}$ is the labelling function,*

- $\Theta = \{\theta_1, \ldots, \theta_M\}$ *is the set of distributions over all latent states for each user trace in the population, such that for all* m, $1 \leq m \leq M$, $\theta_m : \mathcal{X} \to [0, 1]$ *with* $\sum_{x \in \mathcal{X}} \theta_m(x) = 1$.

Definition 2 (GPAM). *For a given positive integer* K*, a* **generalised population admixture model** *(GPAM) for the user trace population* $\mathcal{P}$ *is a tuple* $(\mathcal{X}, \mathcal{Y}, \pi, A, B, \mathcal{L})$ *where* $(\mathcal{X}, \mathcal{Y}, \pi, A, B)$ *is an AR-HMM and* $\mathcal{L} : \mathcal{Y} \to \mathcal{A}$ *is the labelling function.*

Definition 3 (AP). *For any PAM* $(\mathcal{X}, \mathcal{Y}, B, \mathcal{L}, \Theta)$ *or GPAM* $(\mathcal{X}, \mathcal{Y}, \pi, A, B, \mathcal{L})$*, and latent state* $x \in \mathcal{X}$*, the tuple* $(\mathcal{Y}, \mathcal{L}^{-1}(\texttt{UseStart}), B(x), \mathcal{L})$ *is a discrete-time Markov chain we call* **activity pattern**.

Figure 3 illustrates structural differences between four Markov models, using graphical representation known as dynamic Bayesian networks (DBNs) [18, 19]. Shaded nodes are observed states; clear nodes are latent states. $\mathcal{X}_t$ and $\mathcal{Y}_t$ denote the latent state and the observed state at time step t respectively, where $\mathcal{X}_t \in \mathcal{X}$, $\mathcal{Y}_t \in \mathcal{Y}$, and $t \geq 1$. Vertical edges indicate dependencies between latent and observed states at one time instant (i.e. the latent state generates the observed state); horizontal edges indicate the time dependencies within each set of states. These simple examples do not exhibit the branching between observed states that is a strong feature of our models, but they serve to illustrate the following key differences. Traditional HMMs express relationships only between latent states, but not between observed states, so they are not suitable for our approach to modelling behavioural patterns. PAM models are admixtures of first-order Markov chains (DTMCs) that express relationships between observed states, with Θ defining distributions over the latent states. In PAM models we can study the values for Θ across all users or subpopulations thereof. However, it is not possible to relate changes to latent states to changes in the observed states. GPAM models are AR-HMMs, which express explicit relationships between both observed and latent states.

For clarity we note that our approach has two temporal aspects: data logging intervals and probabilistic transitions between observed states in the generated models, they should not be confused as they are at different scales.

4.2. Parameter inference

The number of activity patterns K, the sets $\mathcal{X}$ and $\mathcal{Y}$, and the function $\mathcal{L}$ are predetermined for both PAM and GPAM. The parameters we need to learn are: the observation matrix B and set of distributions Θ for PAM, and the transition matrix A, observation matrix B, and initial distribution π for GPAM. We infer values for these parameters from the data set consisting of the population of user traces using statistical methods for maximum likelihoods, as follows.

Given a set $\mathcal{P}$ of user traces, we compute $n \times n$ transition-occurrence matrices for each trace α such that matrix position (i, j) is the number of times the subsequence $\alpha_i \alpha_j$ (i.e. two adjacent event labels) occurs in α. This is simple to compute and is done once for each data set. This is the input data for parameter inference algorithm.

For PAM we employ the local non-linear optimisation Expectation–Maximisation (EM) algorithm [20] for finding maximum likelihood parameters of observing each trace, restarting the algorithm whenever the log-likelihood has multiple-local maxima. EM converges provably to a local optimum of the criterion, in this case the likelihood function.

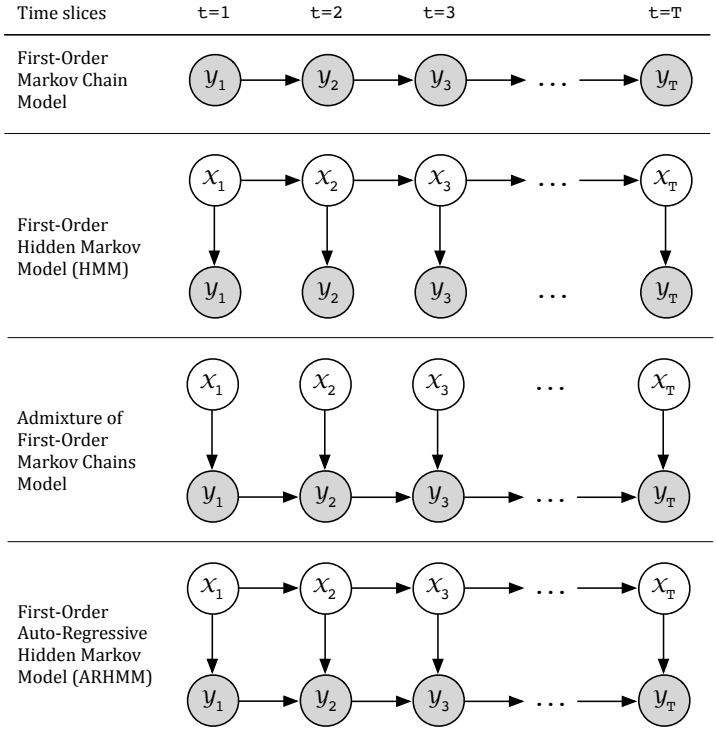


Figure 3: Dynamical Bayesian Network representation of some Markov models showing the dependencies between the observed states (clear nodes) and the hidden states (shaded nodes). $\mathcal{X}_t$ ($\mathcal{Y}_t$) is the hidden (resp. observed) state at time step t.

For GPAM (AR-HMM) we employ the Baum–Welch algorithm [21], which uses also the EM algorithm. We use EM, as opposed to say Markov chains Monte Carlo methods, because it is fast and computationally efficient for our kind of data.

4.3. From admixture models to PRISM *models*

PRISM is a probabilistic model checker [2] used for formal modelling and analysis of systems that exhibit random or probabilistic behaviour. Its high-level state-based modelling language supports a variety of probabilistic models, including DTMCs. PRISM also allows models to be augmented with rewards, which are positive real values assigned to states and transitions for the purpose of reasoning over expected values of these rewards.

In both admixture models PAM and GPAM, each activity pattern is a DTMC, therefore straightforwardly specifiable as a PRISM model or given as an input to PRISM in command line in a specific matrix format.

We can also encode a GPAM as a DTMC by *flattening* it as follows. We pair each observed state with a hidden state. The transition probability between two any such pairs of states (x, y) and (x', y') is the probability of moving from hidden state x to hidden state x' times the probability of moving between the observed states y to y' while

in the current hidden state x', i.e., $\mathbf{P}((x,y),(x',y')) = A(x,x') \cdot B(x',y,y')$. The initial state $(\mathcal{L}^{-1}(\texttt{UseStart}),0)$ is a dummy that encodes the global initial distribution. This flattening operation shares similarities with the composition between an HMM and a deterministic finite state machine (see [22]), however we are composing an HMM with a DTMC.

Definition 4. *Given a GPAM $(\mathcal{X},\mathcal{Y},\pi,A,B,\mathcal{L})$, we flatten it into the discrete-time Markov chain $(S,(\bar{y},0),\mathbf{P},\mathcal{L})$ where:*

- $S = (\{0\} \cup \mathcal{X}) \times \mathcal{Y}$, *where* $\mathcal{X} = \{1,\ldots,K\}$,
- $\bar{y} = \mathcal{L}^{-1}(\texttt{UseStart})$ *is the initial observed state with the label* `UseStart`,
- $\mathbf{P}((x,y),(x',y')) = A(x,x') \cdot B(x',y,y')$, *for* $x > 0$,
- $\mathbf{P}((0,\bar{y}),(x',y')) = \begin{cases} \pi(x') & \text{if } y' = \bar{y} \text{ and } x' > 0 \\ 0 & \text{otherwise,} \end{cases}$
- $\mathcal{L}(x,y) = \{x\} \times \mathcal{L}(y)$.

For clarification, DTMCs are used at two levels of abstraction: to represent individual activity patterns and to represent the flattened GPAMs in which the activity patterns are embedded.

Assuming that the observed state with the label `UseStart` is 0, $1 \leq i \leq K$ and $0 \leq j \leq (n-1)$, then the PRISM model specifying a GPAM has the following template:

module GPAM
$x : [0\,..\,K]$ **init** 0;
$y : [0\,..\,n-1]$ **init** 0;

$[]\ (x=0)\,\&\,(y=0) \longrightarrow \pi(1) : (y'=0)\,\&\,(x'=1) + \ldots + \pi(K) : (y'=0)\,\&\,(x'=K);$

$$\begin{aligned}[]\ (x=i)\,\&\,(y=j) \longrightarrow\ & A(i,1) * B(1,j,0) : (x'=1)\,\&\,(y'=0) + \ldots + \\ & A(i,1) * B(1,j,n) : (x'=1)\,\&\,(y'=n) + \\ & A(i,2) * B(2,j,0) : (x'=2)\,\&\,(y'=0) + \ldots + \\ & A(i,K) * B(K,j,n) : (x'=K)\,\&\,(y'=n);\end{aligned}$$

endmodule

5. Probabilistic temporal properties for analysis

5.1. Probabilistic logics and model checking: brief overview

We assume familiarity with the probabilistic logics PCTL and PCTL*, rewards, and model checking [2, 1]; basic definitions are below.

Probabilistic Computation Tree Logic (PCTL) and its extension PCTL* allow one to express a probability measure of the satisfaction of a temporal property by a DTMC. Their syntax is the following:

$$\begin{aligned} \textit{State formulae} \quad & \Phi ::= \textit{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathtt{P}_{\bowtie p}[\Psi] \mid \mathtt{S}_{\bowtie p}[\Phi] \\ \textit{PCTL Path formulae} \quad & \Psi ::= \mathtt{X}\,\Phi \mid \Phi\,\mathtt{U}^{\leq N}\,\Phi \\ \textit{PCTL* Path formulae} \quad & \Psi ::= \Phi \mid \Psi \wedge \Psi \mid \neg\Psi \mid \mathtt{X}\,\Psi \mid \Psi\,\mathtt{U}^{\leq N}\,\Psi \end{aligned}$$

where a ranges over a set of atomic propositions, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, and $N \in \mathbb{N} \cup \{\infty\}$.

This is a minimal set of operators; the propositional operators *false*, disjunction and implication can be derived. Two common derived path operators are: the *eventually* operator F where $\mathsf{F}^{\leq n}\,\Phi \equiv \mathit{true}\,\mathsf{U}^{\leq n}\,\Phi$ and the *always* operator G where $\mathsf{G}\,\Psi \equiv \neg(\mathsf{F}\,\neg\,\Psi)$. If $n = \infty$ then superscripts are omitted.

PCTL and PCTL* formulae are interpreted over states of a DTMC: state formulae Φ are evaluated over states and path formulae Ψ over paths. We say that a DTMC satisfies a state formulae Φ if the initial state of the DTMC satisfies Φ. We denote by $s \models \Phi$ that state s satisfies Φ (or Φ is evaluated to true in state s). Then $s \models \mathit{true}$ is always true; $s \models a$ iff $a \in \mathcal{L}(s)$; $s \models \neg\Phi$ iff $s \models \Phi$ is false; $s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$; $s \models \mathtt{P}_{\bowtie p}[\Psi]$ iff the probability that Ψ is satisfied by the paths starting from state s meets the bound $\bowtie p$; $s \models \mathtt{S}_{\bowtie p}[\Phi]$ iff the steady-state (long-run) probability of being a state that satisfies Ψ meets the bound $\bowtie p$. Informally, the path formulae $\mathsf{X}\,\Phi$ is true on a path starting in s iff Φ is satisfied in the next state following s in the path, whereas $\Phi_1\,\mathsf{U}^{\leq N}\,\Phi_2$ is true in on a path ω iff Φ_2 is satisfied within N time-steps and Φ_1 is true up until that point.

In PRISM we can replace the bounds $\bowtie p$ with =? and obtain a numerical value that makes the property true. PRISM supports a *reward*-based extension of PCTL called *rPCTL* that assigns non-negative real values to states and/or transitions. $\mathtt{R}_{rwd=?}\left[\,\mathsf{C}^{\leq N}\right]$ computes the expected value of the reward named *rwd* accumulated along *all* paths within N time-steps, $\mathtt{R}_{rwd=?}\left[\,\mathsf{F}\,\Phi\,\right]$ computes the expected value of the reward named *rwd* accumulated along *all* paths until Φ is satisfied. Filters check for properties that hold when starting from sets of states satisfying given propositions. In this paper we use **state** as the filter operator: e.g., **filter**(**state**, Φ, *condition*) where Φ is a state formula and *condition* a Boolean proposition uniquely identifying a state in the DTMC. In the following, for convenience, we refer to properties as rPCTL/PCTL/PCTL* properties, though strictly they also include PRISM operators.

5.2. Classes of temporal properties

We categorise probabilistic temporal logic properties as either *single activity pattern* properties (SAP properties), which means they refer to behaviour within an activity pattern, or *multiple activity pattern* properties (MAP properties), which means they refer to behaviour that may involve several activity patterns in a flattened GPAM. The former reveal characteristics of each activity pattern, while the latter may reveal how or when users change activity patterns. It is important to note that we cannot check properties from the latter set on PAM models because PAMs include a distribution over activity patterns and not transition probabilities between activity patterns, as in GPAMs.

In order to distinguish between SAP and MAP properties, we define the following state formulae and rewards:

$$\begin{array}{rrcl}
\textit{Observed state formulae} & \varphi & ::= & \mathtt{true} \mid \ell \mid y = j \mid \neg\,\varphi \mid \varphi \wedge \varphi \\
\textit{Hidden state formulae} & \gamma & ::= & \mathtt{true} \mid x = i \mid \neg\,\gamma \mid \gamma \wedge \gamma \\
\textit{Non-probabilistic state formulae} & \phi & ::= & \varphi \mid \gamma \mid \neg\phi \mid \phi \wedge \phi
\end{array}$$

where ℓ ranges over the set of observed state labels $\mathcal{A}$, j over the set of observed states identifiers $\{0, \ldots, n-1\}$ and i over the set of hidden state identifiers $\{0, 1, \ldots, K\}$ (with 0 denoting a dummy hidden state in the DTMC flattening of a GPAM).

We define the following templates of reward structures for all values of $j \in \{0, \ldots, n-1\}$ and $i \in \{1, \ldots, K\}$:

- `rLabelj` and `rLabeljAPi` for computing the expected number of visits to an observed state j and to a state (i, j) respectively, where `Labelj` has to be instantiated with the actual label of the observed state j, and
- `rSteps` and `rStepsAPi` for computing the number of all time steps in the model and the number of all time steps in activity pattern i within a GPAM respectively.

```
rewards "rLabelj"
 (y=j):  1;
endrewards
```

```
rewards "rLabeljAPi"
 (y=j) & (x=i) :  1;
endrewards
```

```
rewards "rSteps"
 [] true :  1;
endrewards
```

```
rewards "rStepsAPi"
 [] (x=i) :  1;
endrewards
```

SAP properties involve only observed state formulae φ; *MAP properties* can also involve hidden state formulae γ and so involve the more general formulae ϕ. This means that SAP properties can involve the observed variable y whereas MAP properties can also involve the hidden variable x. In the following, recall, for a GPAM state (i, j), i refers to the activity pattern and j to the state within that pattern. There is a similar distinction for rewards: `rLabeljAPi` and `rStepsAPi` are multiple activity pattern rewards as their definitions also include the hidden variable i, whereas `rLabelj` and `rSteps` are single activity pattern rewards.

Table 1 lists three basic classes of probabilistic temporal properties defined over state formulae: VisitProb, VisitCount, and StepCount, where the reward name *stateRwd* can be either `rLabelj` or `rLabeljAPi`, while *stepRwd* can be either `rSteps` or `rStepsAPi`. VisitProb computes the probability of visiting a state satisfying ϕ_2 within N steps when starting from the unique state satisfying ϕ_0 while ϕ_1 holds in all states visited. VisitCount computes the average number of visits to the state specified in the reward structure *stateRwd* when starting from the unique state satisfying ϕ_0, taking at most N steps. StepCount computes the average number of steps taken from the unique state satisfying ϕ_0 until reaching a state satisfying ϕ_1 (if such a state is not reachable, then the cumulated reward is infinity).

5.3. Examples of SAP and MAP properties

Table 2 illustrates some instances of the property classes in Table 1. $\mathsf{VPinit}_{\mathsf{AP}}$ and $\mathsf{VPinit}_{\mathsf{GPAM}}$ are simple reachability properties for a particular state in either an activity pattern (AP) or a GPAM respectively. $\mathsf{VCinit}_{\mathsf{AP}}$ computes the expected number of visits to a state j in an AP, while $\mathsf{VCinit}_{\mathsf{GPAM}}$ computes only the number of visits to state j while in the pattern i in a GPAM. $\mathsf{SCinit}_{\mathsf{AP}}$ and $\mathsf{SCinit}_{\mathsf{GPAM}}$ compute the expected number of steps needed to reach a particular state in either an AP or a GPAM.

Two additional SAP properties instances of VisitProb and StepCount for reasoning over an activity pattern are:

- $\mathsf{VPsession}_{\mathsf{AP}}$: $\mathbf{filter}(\mathbf{state}, \mathtt{P}_{=?}[\,(\neg\,\mathtt{UseStop})\ \mathsf{U}^{\leq N}(y = j_1)], (y = j_0))$ for computing the probability of reaching observed state j_1 from j_0 within the same session;
- $\mathsf{SCstate2state}_{\mathsf{AP}}$: $\mathbf{filter}(\mathbf{state}, \mathtt{R}_{\mathtt{rSteps}=?}[\mathsf{F}\,(y = j_1)], (y = j_0))$ for computing the average number of steps from state j_0 to state j_1.

The following PCTL/PCTL* MAP properties apply to a flattened GPAM:

Table 1: Classes of rPCTL properties.

Name	**Description**	**rPCTL property**
VisitProb	Probability that starting from state satisfying ϕ_0, ϕ_1 holds until reaching a state in which ϕ_2 holds, within N time-steps.	$\textbf{filter}(\textbf{state}, \mathtt{P}_{=?}[\phi_1 \mathsf{U}^{\leq N} \phi_2], \phi_0)$
VisitCount	Starting from state satisfying ϕ_0, expected reward cumulated over N time-steps.	$\textbf{filter}(\textbf{state}, \mathtt{R}_{stateRwd=?}[\mathsf{C}^{\leq N}], \phi_0)$
StepCount	Starting from state satisfying ϕ_0, expected number of steps cumulated before reaching a state satisfying ϕ_1.	$\textbf{filter}(\textbf{state}, \mathtt{R}_{stepRwd=?}[\mathsf{F}\, \phi_1], \phi_0)$

Table 2: SAP and MAP instances of VisitProb, VisitCount, and StepCount parametrised by observed state j, activity pattern i, and N time-steps.

ID	**Activity Pattern (AP)**	**GPAM**
VPinit	$\mathtt{P}_{=?}[\mathtt{true}\, \mathsf{U}^{\leq N} (y = j)]$	$\mathtt{P}_{=?}[\mathtt{true}\, \mathsf{U}^{\leq N} (x = i \wedge y = j)]$
VCinit	$\mathtt{R}_{\mathtt{rStatej}=?}[\mathsf{C}^{\leq N}]$	$\mathtt{R}_{\mathtt{rStatejAPi}=?}[\mathsf{C}^{\leq N}]$
SCinit	$\mathtt{R}_{\mathtt{rSteps}=?}[\mathsf{F}\, (y = j)]$	$\mathtt{R}_{\mathtt{rSteps}=?}[\mathsf{F}(x = i \wedge y = j)]$

- $\mathsf{State2end}_{\mathsf{GPAM}}$:
  $\mathtt{P}_{\geq 1}[\, \mathsf{F}\, (x = i \wedge y = j)] \wedge \mathtt{P}_{\geq 1}[\, \mathsf{G}\, ((x = i \wedge y = j) \Rightarrow \mathtt{P}_{\geq p}[(x = i)\, \mathsf{U}\, \mathtt{UseStop}\,])]$
  identifies observed states and patterns that lead to the end of the session with a high probability p. This is useful for studying user engagement analysis across different time intervals.

- $\mathsf{Response}_{\mathsf{GPAM}}$:
  $\mathtt{P}_{\geq 1}[\, \mathsf{F}\, \phi_1] \wedge \mathtt{P}_{\geq 1}[\, \mathsf{F}\, \phi_2] \wedge \mathtt{P}_{\geq 1}[\, \mathsf{F}\, ((\neg\phi_1 \wedge \neg\phi_2)\, \mathsf{U}\, \mathtt{P}_{\geq 1}[(\phi_1 \wedge \neg\phi_2)\, \mathsf{U}\, \mathtt{P}_{\geq p}[\, \mathsf{X}\, \phi_2]])]$
  checks for correlations between two properties that involve observable states and activity patterns: if eventually one property holds (for a period of time), then immediately afterwards, the second property holds.

- $\mathsf{LongRunPattern}_{\mathsf{GPAM}}$: $\mathtt{S}_{=?}[\, x = i\,]$ computes the long-run probability of being in each of the activity patterns. This can give us more insight into the popularity of the patterns.

The parametrised properties presented in this section are not exhaustive, they are indicative of the kinds of hypotheses we can pose and are more than sufficient for our application. In particular, variants of the $\mathsf{Response}_{\mathsf{GPAM}}$ property would vary the number of steps until ϕ_2 holds with probability greater or equal to p, e.g., after one, or two, or five

steps; we found the use of the next operator sufficiently discriminatory for the example software application considered in this paper (note that PRISM does not allow bounded until operators at that depth, e.g., $\mathtt{P}_{\geq p}[\mathsf{F}^{\leq N}\phi_2]$). In the next section we describe how to interpret and compare temporal property results across the K activity patterns, and also longitudinal comparisons of results for models over different time intervals.

6. Interpreting and comparing property results

For a given data set, a value for K, and inferred model (either PAM of GPAM), we model check properties for each of the activity patterns. To aid interpretation of the quantitative results, we order them from "best" to "worst", which is greatest to least value for VisitProb and VisitCount and the least to greatest value for StepCount. This ordering reflects the judgements: a greater probability to visit a state (VisitProb), a greater number of state visits (VisitCount), and fewer steps to reach a state (StepCount) are all indicators of greater (user) interest in a state.

We encode the ordering using colours for maximum three activity patterns thus: blue, purple, orange, ranging from "best" results to "worse" results respectively.

We illustrate interpretation through a simple example based on the results we will present in Sect. 8. For the PAM model with $K = 3$, inferred from the the data set corresponding to the second month of usage, assume we obtain these results for property $\mathsf{VCinit}_{\mathsf{AP}}$ and visit counts to an observed state labelled by `OverallUsage` within $N = 50$ time-steps: 13.03 for the first activity pattern, which we call AP1, 5.68 for the second activity pattern, which we call AP2, and 5.06 for the third activity pattern, which we call AP3. It is important to observe that AP1, AP2, and AP3 are simply labels that are *local* to that model. Using our colour coding, we have 13.03 for AP1, 5.68 for AP2, and 5.06 for AP3. These results are given in the first row, left hand side of Table 3.

Table 3: Example results for property $\mathsf{VCinit}_{\mathsf{AP}}$ instantiated for the observed state labelled by `OverallUsage` in PAM models ($K = 3$) inferred from the data sets corresponding to the first and second month of usage.

Usage	**Original ordering**			**Reordered**		
month	**AP1**	**AP2**	**AP3**	**Label1**	**Label2**	**Label3**
2^{nd} month	13.03	5.68	5.06	13.03	5.68	5.06
3^{rd} month	0.83	12.37	8.46	12.37	8.46	0.83

Now consider longitudinal analysis – comparing property results for models inferred from data sets over *different* time intervals. We cannot simply compare the results for the i^{th} activity pattern in one model with the results for the i^{th} activity pattern in another model, because the *order* in which the activity patterns are inferred is non-deterministic (recall, for each data set we infer the observation probability matrix B). Specifically, the types of activity patterns may be generated in different orders, i.e. AP1, AP2, and AP3 are local, not global labels. Moreover, even when comparing property results for corresponding activity patterns, we should not expect results to be identical, but rather they should have similar relationships to the results for the other activity patterns (for that model).

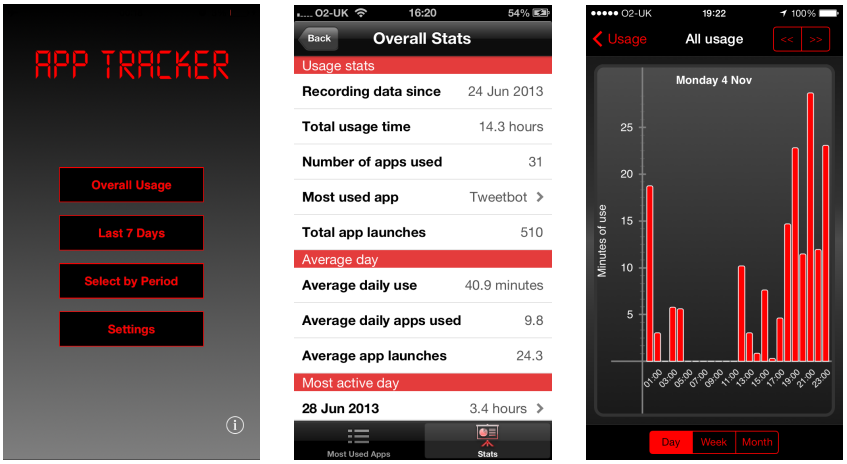
Now consider the model PAM with $K = 3$ inferred from the third month of usage and the same property $\mathsf{VCinit}_{\mathsf{AP}}$ as above. The results are given in second row, left hand side of Table 3: 0.83 for the first activity pattern AP1, 12.37 for the second activity pattern AP2, and 8.46 for the third activity pattern AP3. (Note, these are actual results.) Using our colour coding, we have 0.83 for AP1, 12.37 for AP2, and 8.46 for AP3. Again, the labels AP1, AP2, and AP3 are *local*.

It is easy to see from the colour coding that AP1 in the model from the first month is similar to AP2 in the model from the third month. On the right hand side of Table 3 we reorder the results using a global mapping to names Label1, Label2, Label3. It is usually helpful, after reviewing results for several properties, to introduce meaningful names for global labels such as GLANCING or INDEPTH, for activity patterns. In this example it was straightforward to swap over results manually, however, this may not be straightforward for bulk, automated analysis, and when considering multiple properties. In the remainder of this paper we will report results in the order they were generated, unless it is simple and pertinent to reorder.

We now turn our attention to the example software application.

7. Example: AppTracker mobile app

AppTracker [3] is a personal productivity iOS mobile application that runs in the background, monitoring the opening and closing of (other) apps. It displays a series of charts and statistics, offering insight to users into the time spent on their device, the most used apps, how these stats fluctuate over time, etc.. The main menu screen offers four main options (Fig. 4(a)). The first menu item, *Overall Usage*, contains summaries of all the data recorded since AppTracker was installed and opens the views `OverallUsage` and `Stats` (Fig. 4(b)). The second menu item, *Last 7 Days*, opens the view `StackedBars` and displays a chart indicating activity recorded over the last 7 days. The third menu item, *Select by Period*, opens the view `PeriodSelector` and shows statistics for a selected period of time. For example, this could be which apps were used most since last Saturday, or how time spent on Facebook varied each day last month, or hourly device usage for a particular day (Fig. 4(c)). The final menu option, *Settings*, allows a user to start and stop the tracker, or to reset their recorded data. We are interested in the 16 user-initiated events that switch between views and start and stop a session, as illustrated by the state diagram in Fig. 5. These events determine the atomic propositions used in our models as described the following definition of the state labelling function: $\mathcal{L}(0) =$ `UseStart`, $\mathcal{L}(1) =$ `T&C` is the terms and conditions page, $\mathcal{L}(2) =$ `Main` is the main menu screen, $\mathcal{L}(3) =$ `OverallUsage` shows the summary of all recorded data, $\mathcal{L}(4) =$ `StackedBars` shows the last seven days of top five apps used, $\mathcal{L}(5) =$ `PeriodSelector` shows app usage stats for a selected time period, $\mathcal{L}(6) =$ `AppsInPeriod` shows apps used for a selected period, $\mathcal{L}(7) =$ `Settings` shows the settings options, $\mathcal{L}(8) =$ `UseStop` stands for closing/sending to background the AppTracker, $\mathcal{L}(9) =$ `Stats` shows statistics of app usage, $\mathcal{L}(10) =$ `UBCOverallUsage` shows app usage when picked from `OverallUsage`, $\mathcal{L}(11) =$ `Feedback` shows a screen for giving feedback, $\mathcal{L}(12) =$ `UBCStats` shows app usage when picked from `Stats`, $\mathcal{L}(13) =$ `Info` shows information about the app, $\mathcal{L}(14) =$ `UBCApps` shows app usage when picked from `AppsInPeriod`, $\mathcal{L}(15) =$ `Task` shows a feedback question chosen from the `Feedback` view. `UseStart` and `UseStop` distinguish the beginning and end (resp.) of a session.



(a) Main menu (b) Overall stats (c) Daily device usage

Figure 4: Screenshots from AppTracker.

The log data are stored in a MySQL database by the SGLog framework. Raw data is extracted from the database and processed using JavaScript to obtain user traces in the following JSON format: information about the user's device, start and end data of AppTracker usage, and list of sessions. For example, the start of a user trace may look like the following:

```
[{"deviceid":"x","firstSeen":"2013-08-20 09:10:59","lastSeen":"2014-03-24
09:57:32","sessions":[[{"timestamp":"2013-08-20 09:11:01","data":
"UseStart"},{"timestamp":"2013-08-2009:11:02","data":"T&C"},{"timestamp":
"2013-08-20 09:11:23", "data":"Main"},{"timestamp":"2013-08-2009:11:46",
"data":"OverallUsage"},...]}]
```

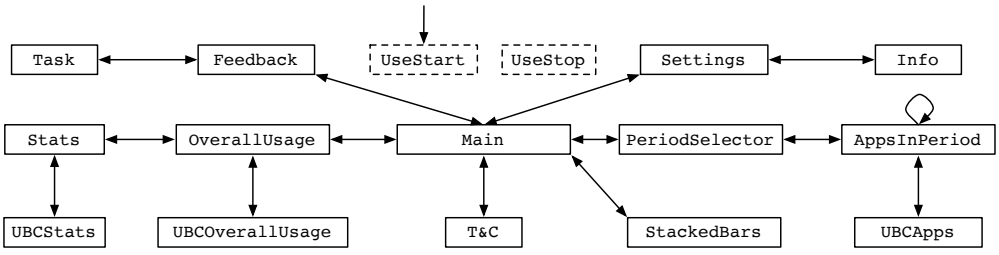


Figure 5: AppTracker state diagram. `UseStart` labels the initial state; all (15) outgoing and incoming arrows are omitted for `UseStart` and `UseStop`.

AppTracker was first released in August 2013 and downloaded over 35,000 times; our data sets are taken from a sample of 322 users traces during 2013 and 2014. The maximum session count over all the traces is 129, the minimum was limited to 5.

For learning parameters of the PAM and GPAM models for different time intervals (first month, second month, third month) and values for K (2, 3, 4, and 5), the EM algorithm was restarted 200 times with 100 maximum number of iterations for each restart. As example performance, we implemented the algorithm for GPAM parameters in Java and ran it on a 2.8GHz Intel Xeon (single thread, one core); for the data set consisting of the first month of usage (332 user traces), the algorithm takes 3.2 min for $K = 2$, 4.1 min for $K = 3$, and 5.3 min for $K = 4$. Figure 6 illustrates state-transition diagrams of all $K = 3$ activity patterns in a GPAM, where the thickness of the transitions corresponding to ranges of probability: the thicker the line, the higher the probability of that transition. Note that this illustration does not include the probabilistic transitions between the hidden states.

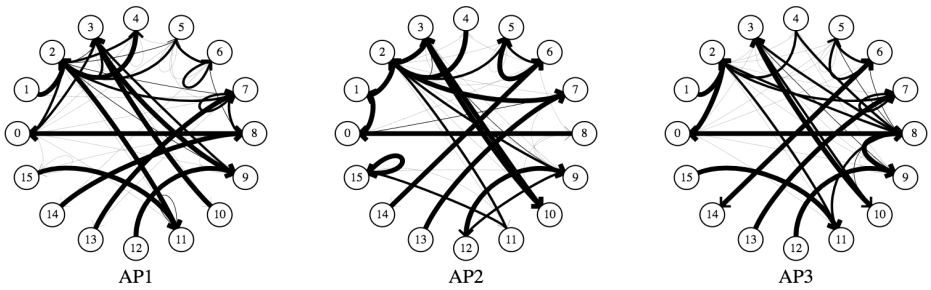


Figure 6: Graphical representation of the DTMCs underlying the three activity patterns of the GPAM for the first month of usage: the nodes are the observed states from 0 to 15 and the thickness of the transition arrow is proportional.

8. Example analysis

Analysis is hypothesis driven, drawing on the properties defined in Sect. 5. We have found a useful approach is: consider the single activity pattern properties initially in PAM and GPAM models, then multiple activity pattern properties in GPAM models, followed by the distribution of activity patterns in PAM models. For illustration, we consider data sets for three different time intervals (1^{st} month, 2^{nd} month, 3^{rd} month of usage), and compare the results for PAM and GPAM models. For simplicity, assume $N = 50$ for all time-bounded properties. Previous PAM analyses for AppTracker [23] include values for N ranging from 10 to 150 with step-size 10; from those results we concluded that, for the AppTracker data, $N = 50$ is sufficiently discriminatory. If the result of model checking a reward-based property does not produce a number (due to state unreachability, or filter satisfying no states, or the iterative method not converging within 100,000 iterations) this is denoted by "—".

The goal of this section is to give a flavour of the types of analysis possible and an indication of how to iterate through combinations of model and property, rather than a comprehensive analysis of the AppTracker app usage. For this reason, some interpretations are given here without full detail, and some results are consigned to the Appendix and on the website at `http://www.dcs.gla.ac.uk/research/S4/tasum`.

8.1. Single activity pattern properties for PAM and GPAM

Table 4 gives results for $\mathsf{VPinit}_{\mathsf{AP}}$, $\mathsf{VCinit}_{\mathsf{AP}}$, and $\mathsf{SCinit}_{\mathsf{AP}}$ where observed state j is instantiated with the state labels `OverallUsage`, `StackedBars`, `PeriodSelector`, `Stats`, on PAM with $K = 2$. The results for the properties $\mathsf{VCinit}_{\mathsf{AP}}$ and $\mathsf{SCinit}_{\mathsf{AP}}$ instantiated with `UseStop` in Table 5 are indicative of the average session count and the average session length respectively.

Table 4: Results for $\mathsf{VPinit}_{\mathsf{AP}}$, $\mathsf{VCinit}_{\mathsf{AP}}$, and $\mathsf{SCinit}_{\mathsf{AP}}$ on PAM with $K = 2$.

Prop.	Usage	`OverallUsage`		`StackedBars`		`PeriodSelector`		`Stats`	
	month	AP1	AP2	AP1	AP2	AP1	AP2	AP1	AP2
$\mathsf{VPinit}_{\mathsf{AP}}$	1^{st}	0.98	0.99	0.97	0.32	0.90	0.09	0.65	0.98
	2^{nd}	0.99	0.96	0.54	0.97	0.74	0.76	0.19	0.84
	3^{rd}	0.99	0.91	0.00	0.98	0.15	0.93	0.86	0.65
$\mathsf{VCinit}_{\mathsf{AP}}$	1^{st}	5.35	12.02	3.58	0.38	3.50	0.10	1.49	4.13
	2^{nd}	10.44	5.69	0.81	4.18	4.52	1.55	0.20	3.15
	3^{rd}	12.95	4.76	0.00	4.55	0.45	3.38	1.86	2.06
$\mathsf{SCinit}_{\mathsf{AP}}$	1^{st}	12.34	2.91	13.33	131.99	21.55	519.32	48.15	12.02
	2^{nd}	3.53	12.72	65.08	11.44	36.78	35.29	240.77	27.52
	3^{rd}	2.30	19.73	—	10.28	300.63	17.67	25.80	48.13

Table 5: Results for $\mathsf{VCinit}_{\mathsf{AP}}$ and $\mathsf{SCinit}_{\mathsf{AP}}$ instantiated with `UseStop` on PAM with $K = 2$: $\mathsf{VCinit}_{\mathsf{AP}}$ computes the average number of sessions within 50 time steps, $\mathsf{SCinit}_{\mathsf{AP}}$ computes the average session length.

Usage	$\mathsf{VCinit}_{\mathsf{AP}}$		$\mathsf{SCinit}_{\mathsf{AP}}$	
month	AP1	AP2	AP1	AP2
1^{st}	5.65	8.99	7.64	4.39
2^{nd}	8.82	5.37	4.58	8.42
3^{rd}	10.29	5.59	3.75	7.94

8.1.1. Global labels for activity patterns in PAM with $K = 2$

Based on the results listed in Tables 4 and 5 we identify two distinct activity pattern labels as follows:

BROWSING corresponds to more likely, more often, and more quickly reaching the states `OverallUsage` and `Stats`, thus more *high-level stats visualisations* with shorter and more frequent sessions (than the INDEPTH pattern).

INDEPTH corresponds to more likely, more often, and more quickly reaching `StackedBars` and `PeriodSelector`, thus more *in-depth stats visualisation* with longer and less frequent sessions (than the BROWSING pattern).

Under this interpretation, for the first month of usage data AP1 is an INDEPTH pattern and AP2 a BROWSING pattern, while for the second month and the third month of usage AP1 is a BROWSING pattern and AP2 an INDEPTH pattern.

8.1.2. Further investigations for PAM with $K = 2$

Observe that state `Stats` does not fall clearly into one pattern or the other. We discussed this with the designers of the AppTracker app and found out that `Stats` does not show much information until after the first month, i.e. there are no statistics to inspect during the first month. We investigated further the behaviour involving `Stats` and the states specific to each of patterns by analysing the properties $\mathsf{VPsession}_{\mathsf{AP}}$ (Table 6) and $\mathsf{SCstate2state}_{\mathsf{AP}}$ (Table 7) for PAM models with $K = 2$ and second month of usage. The results of the $\mathsf{VPsession}_{\mathsf{AP}}$ properties involving `Stats` score better in InDepth (AP2) than in Browsing (AP1). The average step counts as computed by $\mathsf{SCstate2state}_{\mathsf{AP}}$ shows that `Stats` is reached in fewer steps in the InDepth pattern and also the session is longer when `Stats` is visited (10.87 steps until the end of the session in InDepth than 1.55 steps in `OverallUsage`). We conclude that in the second month of usage, the state `Stats` is more characteristic of the InDepth activity pattern, while in the first and the third month of usage `Stats` is characteristic of the Browsing pattern.

Table 6: Results for $\mathsf{VPsession}_{\mathsf{AP}}$ on PAM with $K = 2$ and second month of usage: probability to reach one state from another within the same session.

Target state	`OverallUsage`		`StackedBars`		`PeriodSelector`		`Stats`	
	AP1	**AP2**	**AP1**	**AP2**	**AP1**	**AP2**	**AP1**	**AP2**
`OverallUsage`	1.00	1.00	0.71	0.23	0.25	0.28	0.33	0.52
`StackedBars`	0.01	0.48	1.00	1.00	0.02	0.38	0.00	0.49
`PeriodSelector`	0.01	0.19	0.11	0.12	1.00	1.00	0.00	0.20
`Stats`	0.02	0.59	0.01	0.13	0.00	0.17	1.00	1.00

Table 7: Results for $\mathsf{SCstate2state}_{\mathsf{AP}}$ on PAM with $K = 2$ and second month of usage: average step counts from one state to another.

Target state	`OverallUsage`		`StackedBars`		`PeriodSelector`		`Stats`	
	AP1	**AP2**	**AP1**	**AP2**	**AP1**	**AP2**	**AP1**	**AP2**
`OverallUsage`	0.00	0.00	4.18	13.60	12.34	18.48	4.03	11.44
`StackedBars`	67.03	14.81	0.00	0.00	73.79	17.35	67.40	14.42
`PeriodSelector`	38.73	37.90	36.67	35.80	0.00	0.00	39.09	37.50
`Stats`	237.24	16.23	241.43	28.73	249.58	33.69	0.00	0.00
`UseStop`	1.69	11.12	4.10	5.63	9.39	11.98	1.55	10.87

8.1.3. Global labels for activity patterns in PAM with $K = 3$

We now consider PAM models for $K = 3$ and first, second and third month of usage; results are in the Appendix in Table A.13. From these results we identify a new type of pattern that we label as Glancing. This pattern is centred around a particular state which scores very good results while the other states score poorly; usually the Glancing pattern in PAM is also characterised by the shortest and most frequent sessions.

When we examine the results over different time intervals, we find some interesting results. In the first month of usage the three activity patterns are mapped (respectively)

to InDepth, Browsing, Glancing (around `OverallUsage`); in the second month of usage they are mapped to Glancing (around `OverallUsage`), InDepth, and Browsing; and in the third month of usage they are mapped to Glancing (around `StackedBars`), Glancing (around `OverallUsage`), and InDepth.

It is is beyond the scope of this paper, but we note briefly that these results prompted further investigations into the different relationships between states in each Glancing like pattern, using properties $\mathsf{VPsession}_{\mathsf{AP}}$ and $\mathsf{SCstate2state}_{\mathsf{AP}}$, as well as extending the analysis to other states (that are one level down in the hierarchical menu) such as `UBCOverallUsage`, `UBCStats`, `AppsInPeriod`, and `UBCApps`.

8.1.4. Activity pattern analysis in GPAM

We analyse the individual patterns of GPAM in a similar way to PAM. We consider results for $\mathsf{VPinit}_{\mathsf{AP}}$, $\mathsf{VCinit}_{\mathsf{AP}}$, $\mathsf{SCinit}_{\mathsf{AP}}$ for states `OverallUsage`, `StackedBars`, `PeriodSelector`, `Stats`, `UseStop`, on GPAM with $K = 2$. Tables 8 and 9 give results. For the first month of usage, AP1 is labelled as a Browsing pattern and AP2 an InDepth pattern. For the second and the third months of usage, AP1 is an InDepth pattern and AP2 a Browsing pattern.

Table 8: Results for $\mathsf{VPinit}_{\mathsf{AP}}$, $\mathsf{VCinit}_{\mathsf{AP}}$, and $\mathsf{SCinit}_{\mathsf{AP}}$ on GPAM with $K = 2$.

Prop.	**Usage**	`OverallUsage`		`StackedBars`		`PeriodSelector`		`Stats`	
	month	**AP1**	**AP2**	**AP1**	**AP2**	**AP1**	**AP2**	**AP1**	**AP2**
$\mathsf{VPinit}_{\mathsf{AP}}$	1^{st}	0.99	0.59	0.92	0.91	0.56	0.90	0.98	0.46
	2^{nd}	0.87	0.99	0.98	0.31	0.93	0.00	0.45	0.96
	3^{rd}	0.91	0.99	0.97	0.02	0.96	0.10	0.56	0.91
$\mathsf{VCinit}_{\mathsf{AP}}$	1^{st}	15.55	0.89	2.52	2.39	1.19	2.62	4.75	0.65
	2^{nd}	2.28	14.27	5.06	0.40	4.29	0.01	0.80	4.04
	3^{rd}	3.00	14.73	4.48	0.02	4.61	0.10	1.28	3.63
$\mathsf{SCinit}_{\mathsf{AP}}$	1^{st}	3.44	56.55	19.28	20.07	59.94	21.53	13.68	81.13
	2^{nd}	23.41	2.15	9.02	137.09	18.90	5483.99	85.45	15.97
	3^{rd}	19.55	2.23	10.46	2269.78	15.49	483.09	61.19	21.01

Table 9: Results for $\mathsf{VCinit}_{\mathsf{AP}}$ (average number of sessions within 50 time steps) and $\mathsf{SCinit}_{\mathsf{AP}}$ (average session length) instantiated with `UseStop` on GPAM with $K = 2$.

Usage	$\mathsf{VCinit}_{\mathsf{AP}}$		$\mathsf{SCinit}_{\mathsf{AP}}$	
month	**AP1**	**AP2**	**AP1**	**AP2**
1^{st}	0.47	10.82	102.07	3.51
2^{nd}	6.17	7.55	7.09	5.36
3^{rd}	5.43	7.35	8.28	5.56

When we analyse the activity patterns in GPAM with $K = 3$, see Table A.14 in the appendix, we uncover a new type of pattern, which we call Loop, for AP2 in the first month of usage. The fact that $\mathsf{VCinit}_{\mathsf{AP}}$ and $\mathsf{SCinit}_{\mathsf{AP}}$ score poorly for this pattern (as

opposed to a BROWSING pattern) led us to investigate more observed states and properties. We found that `UBCOverallUsage` has similar results to `OverallUsage`, leading us to conclude that AP2 in the first month of usage corresponds to repeatedly switching between `OverallUsage` and `UBCOverallUsage`; this pattern is less glancing-like and more in-depth stats visualisation. The global labels of the patterns for GPAM $K = 3$ are mapped (respectively) as follows: for the first month of usage, they are mapped to INDEPTH, LOOP, and BROWSING; for the second month of usage, they are mapped to GLANCING (around `OverallUsage`), INDEPTH, BROWSING; for the third month, they are mapped to BROWSING, GLANCING (around `StackedBars`), and INDEPTH.

The two glancing patterns we identified (centred around the states `OverallUsage` and `StackedBars`) might suggest that we are merely uncovering the top level menu structure where *Overall Usage* (mapped to the observed state `OverallUsage`) is the top-level button and *Last 7 Days* (mapped to `StackedBars`) is the second button from the top. However, when analysing admixture models for $K = 3$ and higher, we did not find a pattern solely centred around the state `PeriodSelector` (abstracting the events generated by pressing the third button from the top, *Select by Period*) without including `StackedBars`. And so we conclude that activity patterns do *not* correspond directly to the top-level menu structure.

We now turn our attention to the additional properties we can investigate in GPAM.

8.2. Multiple activity pattern properties for GPAM

Recall that MAP properties can also refer to hidden state variables. In Section 5 we defined the property classes VisitProb, VisitCount, and StepCount which can be instantiated into MAP properties $\mathsf{VPinit}_{\mathsf{GPAM}}$, $\mathsf{VCinit}_{\mathsf{GPAM}}$, and $\mathsf{SCinit}_{\mathsf{GPAM}}$ (see Table 2), as well as the MAP properties $\mathsf{State2end}_{\mathsf{GPAM}}$, $\mathsf{Response}_{\mathsf{GPAM}}$, and $\mathsf{LongRunPattern}_{\mathsf{GPAM}}$. We already showed how properties of the former class can be analysed and interpreted, now we focus on the latter set of MAP properties.

8.2.1. Likelihood of observed state to lead to end of session

The $\mathsf{State2end}_{\mathsf{GPAM}}$ property computes the probability p that *always from a screen view j in a pattern i, eventually, without changing the pattern, the session ends*, therefore helps us to identify most likely screen views (observed states) and patterns that diminish user engagement (i.e. lead to end of the session). Table 10 lists the results for $\mathsf{State2end}_{\mathsf{GPAM}}$ on GPAM with $K = 2$ for the first and second months of usage, where $-$ denotes that the property is evaluated to false.

For the first month of usage (and this also holds for all time intervals starting from first day of usage), the states `Stats`, `UBCOverallUsage` and `UBCApps` are the most likely to disengage users in BROWSING patterns where the probabilities p to do so are the highest over all states (see in bold), while in the INDEPTH pattern the property either does not hold, or it holds for close to zero values of p. While for the screen views corresponding to the states `UBCOverallUsage` and `UBCApps` this effect was expected because of their lowest level in the hierarchical menu, we were intrigued by the results for `Stats`. We discussed these results with the designers and again concluded they were a consequence of few statistics being available during the first month. The designers subsequently suggested changing the monthly `Stats` view to show something engaging (e.g. default illustrations of monthly stats and/or quick tutorial) up until the first 30 days of usage, so that users

do not disengage and end the session. During the second month of usage, for $K = 2$ the probabilities p are similar for both patterns, except for the states `UBCOverallUsage` and `UBCApps` for which the property does not hold because `UBCOverallUsage` does not occur in the InDepth pattern nor does `UBCApps` in the Browsing pattern.

Table 10: Results for $\mathsf{State2end}_{\mathsf{GPAM}}$ on GPAM with $K = 2$ and the first two months of usage.

State	InDepth		Browsing	
	1^{st} **month**	2^{nd} **month**	1^{st} **month**	2^{nd} **month**
`OverallUsage`	—	0.72	0.41	0.67
`StackedBars`	0.08	0.80	0.41	0.65
`PeriodSelector`	—	0.67	0.26	0.62
`Stats`	—	0.70	**0.70**	0.66
`AppsInPeriod`	0.08	0.62	0.14	0.35
`UBCOverallUsage`	—	—	**0.84**	0.59
`UBCStats`	—	0.66	0.13	0.60
`UBCApps`	—	0.64	**0.83**	—

8.2.2. Response property

The property $\mathsf{Response}_{\mathsf{GPAM}}$ checks for correlations between two state formulae that involve observable states and/or activity patterns: if eventually one property holds (for a period of time), then immediately afterwards, the second property holds. An example of such state formulae are $\phi_1 =$ `StackedBars` $\wedge (x = i)$ and $\phi_2 =$ `PeriodSelector` $\wedge (x = i)$; in this case $\mathsf{Response}_{\mathsf{GPAM}}$ computes the probability p that once `StackedBars` is reached in pattern i then `PeriodSelector` is also reached in pattern i in the next state. We also consider the reverse property where $\phi_1 =$ `PeriodSelector` $\wedge$ $(x = i)$ and $\phi_2 =$ `StackedBars` $\wedge$ $(x = i)$. Table 11 lists the results of analysing $\mathsf{Response}_{\mathsf{GPAM}}$ on GPAMs for $K = 2$ and $K = 3$ both ways: `PeriodSelector` as response for `StackedBars` and `StackedBars` as response for `PeriodSelector`.

Table 11: Results for $\mathsf{Response}_{\mathsf{GPAM}}$ on GPAMs: probabilities that `PeriodSelector` is a response to `StackedBars` (`StackedBars` $\Rightarrow$ `PeriodSelector`) and probabilities that `StackedBars` is a response to `PeriodSelector` (`PeriodSelector` $\Rightarrow$ `StackedBars`).

Usage	`StackedBars` $\Rightarrow$ `PeriodSelector`					`PeriodSelector` $\Rightarrow$ `StackedBars`				
month	**K = 2**		**K = 3**			**K = 2**		**K = 3**		
	AP1	**AP2**	**AP1**	**AP2**	**AP3**	**AP1**	**AP2**	**AP1**	**AP2**	**AP3**
1^{st}	0.62	0.25	0.11	0.91	0.22	0.13	0.14	0.21	0.06	0.04
2^{nd}	0.46	0.06	0.01	0.10	0.74	0.29	0.06	0.08	0.55	0.18
3^{rd}	0.38	0.02	0.86	0.02	0.31	0.27	0.13	0.04	0.39	0.14

We investigated whether `StackedBars` and `PeriodSelector` are always associated with InDepth, or if they are characteristic of different patterns (in which case our analysis would simply uncover the high level menu structure of the app). Based on the global

labels we assigned to the patterns, we observe that for $K = 2$ in the second and the third month of usage `StackedBars` is more likely to be followed by `PeriodSelector` in InDepth pattern (AP2) than it is in the Browsing pattern (AP3); the same holds for `StackedBars` as a response to `PeriodSelector` which is more likely in the InDepth pattern than in the Browsing pattern. For $K = 3$ we see a high probability of `StackedBars` to be a response to `PeriodSelector` in the InDepth patterns AP1/1st month and AP2/2nd month and the Glancing (around `StackedBars`) pattern AP2/3rd month; however the probability that `PeriodSelector` is a response to `StackedBars` is lower in InDepth patterns compared to Overall patterns. These results support a conclusion we formulated earlier: that while we found patterns centred around `StackedBars`, we did not found patterns centred around `PeriodSelector` without involving `StackedBars`. Therefore our admixture model for $K = 3$ did not uncover the high-level menu structure of the app, i.e., a pattern for each of the buttons *Overall Usage* (state `OverallUsage`), *Last 7 Days* (state `StackedBars`), and *Select by Period* (state `PeriodSelector`).

8.2.3. Steady state property

The last MAP property we consider is $\mathsf{LongRunPattern}_{\mathsf{GPAM}}$, which computes the steady-state probability for each of the patterns. The results are listed in Table 12. For GPAM with $K = 2$ we see a higher prevalence for an exploratory behaviour (AP1 is Browsing) in the first usage month, than during the subsequent usage months (AP2 is a Browsing pattern in both second and third month). For GPAM with 3 components, it is only in the third month (when the usage behaviour settles down) that the InDepth pattern prevails (see AP3); for the first month of usage, the Loop pattern around `OverallUsage` (AP2) and the Browsing pattern (AP3) score slightly better as these patterns correspond to an expected initial exploratory behaviour; for the second month of usage, the Glancing pattern around `OverallUsage` (AP1) prevails (this could be explained by the richer information offered by the `Stats` screen view after the first month).

Table 12: Results for $\mathsf{LongRunPattern}_{\mathsf{GPAM}}$ on GPAM with $K = 2$ and $K = 3$.

Usage	**K=2**		**K=3**		
month	**AP1**	**AP2**	**AP1**	**AP2**	**AP3**
1st	0.55	0.44	0.27	0.36	0.35
2nd	0.65	0.34	0.43	0.18	0.38
3rd	0.53	0.46	0.25	0.25	0.49

Finally, we return to the PAM model and consider the distribution of activity patterns.

8.3. Pattern distribution analysis for PAM

PAM models include a distribution over all activity patterns, for each user trace. We compare the overall distributions across the first, second, and third usage month, for $K = 2$ in Figure 7. For each time interval I and i^{th} activity pattern, we order non-decreasingly the vector of values $[\theta_m(i)]_{m=1,M(I)}$ where $M(I)$ is the number of user traces in the data set determined by the time interval I and obtain the vector $[a_{m'}]_{m'=1,M(I)}$;

then on the x-axis we project the user ordinal in the $[a_{m'}]_{m'=1,M(I)}$ vector (i.e., m' ranging from 1 to $M(I)$) and on the y-axis the ordered values of the $[a_{m'}]_{m'=1,M(I)}$ vector. For example, assume we have $M(I) = 5$ user traces in the time interval I, $K = 2$, and $[\theta_m]_{m=1,M(I)} = [(0.25, 0.75), (0.88, 0.22), (0.45, 0.55), (0, 1), (0.63, 0.27)]$. Considering the first pattern, i.e., $i = 1$, then $[\theta_m(1)]_{m=1,M(I)} = [0.25, 0.88, 0.45, 0, 0.63]$ and after ordering it we obtain the vector $[0, 0.25, 0.45, 0.63, 0.88]$. The x-axis is labelled by the user trace ranking, ordinals 1 to 5, and the y-axis is labelled by probabilities from 0 to 1. We then plot the non-decreasingly ordered probabilities 0, 0.25, 0.45, 0.63, 0.88 against the five (ranked) user traces.

We note in Fig. 7 that the θ distributions for the activity pattern AP1 in the first month of usage as well as for the activity pattern AP2 in the second and third month of usage have a concave appearance. Previous analyses showed that they correspond to InDepth activity patterns, while their counterparts (with a convex appearance) correspond to Browsing activity patterns. In addition this tells us that the InDepth activity pattern is more likely than the Browsing activity pattern, especially in the second month of usage because there are more user traces with the θ value very close or equal to 1 than user traces with the θ value very close or equal to 0. This corroborates the result from the $\mathsf{LongRunPattern}_{\mathsf{GPAM}}$ analysis above in the second and the third month of usage. We can also compare longitudinally the number of user traces exhibiting mostly only one activity pattern (probability very close or equal to 1). For instance, in the second month of usage almost 23% user traces are exclusively of type InDepth while in the first month the proportion is 11% and the third month 13%. The higher percentage of almost exclusive InDepth activity pattern in the second month of usage could be explained by high user interest in particular content available only after one month of usage (e.g. in the information provided in the `Stats` view).

8.4. Summary of analysis

It is easy to get lost in the plethora of detailed results. We summarise them as follows.

When $K = 2$, the two types of activity patterns are InDepth and Browsing. In the first month of usage, likelihoods of the two patterns are very close, for both PAM and GPAM models, however in second and third months the InDepth pattern was more prevalent for PAM. Within Browsing, the states most likely to lead to disengagement during the first month of usage are `Stats`, `UBCOverallUsage`, and `UBCApps`.

We had not anticipated the effect of the new content available in `Stats` after the first month – indeed, we were unaware there was any new content until we studied the analysis results! The new content (or absence) appears to have a profound effect on engagement (when in Browsing) and the focus of attention (when in InDepth in the second month).

When $K = 3$, we identified other types of activity patterns in addition to the InDepth and Browsing patterns. One type of new pattern is Glancing for short-burst and high frequency behaviour centred usually around one or two particular states (such as AP1 in the second month of usage centred around `OverallUsage` and `Stats` and AP2 in the third month of usage centred around `StackedBars` all in GPAM). Another new pattern is the Loop pattern (AP2 in GPAM first month of usage) which is in fact an (unexpected) in-depth visualisation type of pattern centred around the states `OverallUsage` and `UBCOverallUsage`.

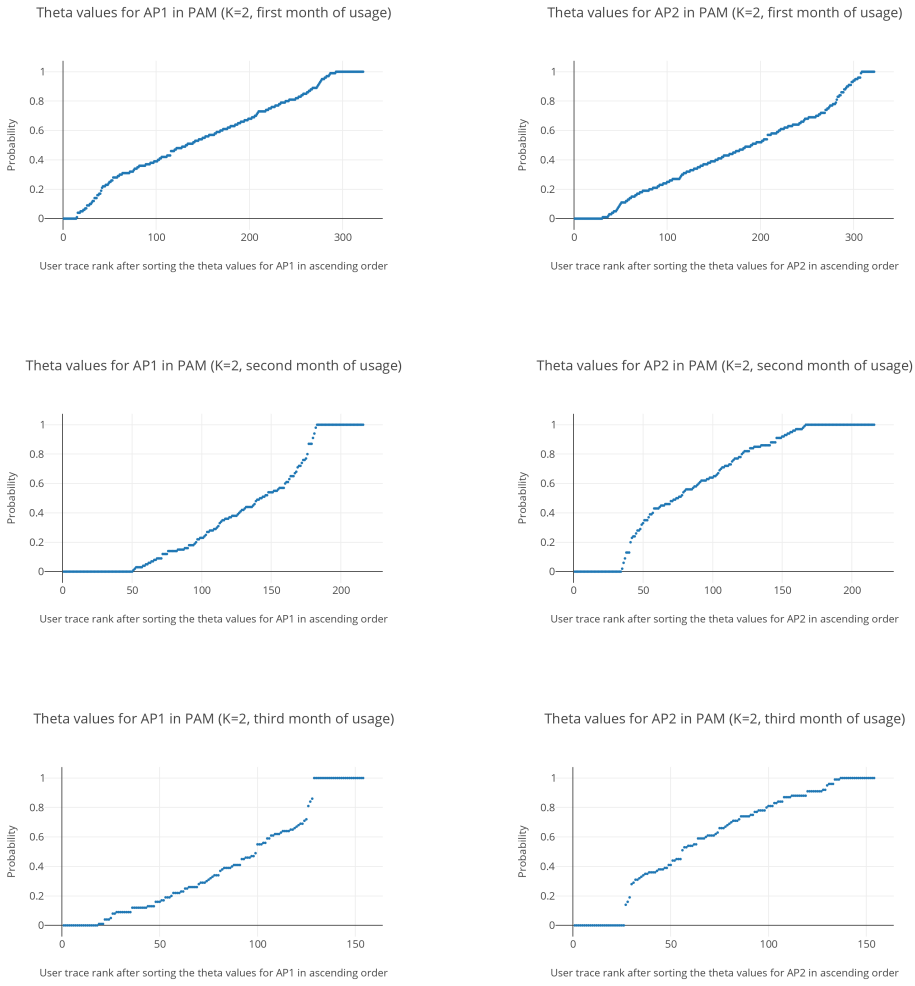


Figure 7: Activity pattern distribution Θ per activity pattern and usage month for PAM with $K = 2$: the plotted values are ordered non-decreasingly, thus the x-axis corresponds to the user ordinal in the $\Theta(i)$ vector after ordering, while the y-axis corresponds to the values in the $\Theta(i)$ vector after ordering, where i is either 1 for AP1 or 2 for AP2. For instance, in the top-left plot, the 100^{th} user trace in the $\Theta(1)$ vector after ordering is mapped to the value $a_{100} = 0.39$.

An interesting finding for this software application is that the activity patterns we uncovered did *not* follow the top level menu structure. This has prompted discussions and further research within the app design team about the role and impact of hierarchical menus on user interactions (e.g. *should* the top-level menu structure support the activity patterns?). We also uncovered a loop between the states `OverallUsage` and `UBCOverallUsage` (see AP2 in GPAM with $K = 3$ first month of usage), which is characteristic of an in-depth visualisation behaviour and not so much to glancing; therefore

this type of visualisation could be moved from the `OverallUsage`-submenu to one of the in-depth type submenu items in order to make a clearer separations of app usage in the menu. Another finding concerns the way we segment the data set: the first month of usage does not provide insight as clear as the second and the third usage month; this is due to many users still exploring the app in the first month, while the users using the app for longer periods of time are more engaged and usually have a more settled/consistent behaviour. Therefore either breaking down the first month into smaller data set, e.g. by first day and by each of the four weeks, and/or selecting only those user traces continuing up to the third month of usage, could be more meaningful,

Finally we note that by identifying typical glancing patterns and the specific screens that users look at when they are undertaking short sessions of glancing-type behaviour, our approach may offer a principled way of selecting content appropriate for widget extensions.

All property analysis results discussed or mentioned in this paper, including the parameters for the models, the PRISM models and the PRISM properties, are listed on the website `http://www.dcs.gla.ac.uk/research/S4/tasum`.

9. Discussion

Which models, which properties. We have found it useful to begin analysis with SAP properties, followed by MAP properties and then pattern distributions. In our experience, starting with SAP properties gives us insight into the semantics of the patterns, which then motivates the selection of further SAP and the MAP properties. As with all hypothesis driven discovery, the results of one property (i.e. experiment) often motivates the design of the next property to be studied (i.e. another experiment). For example, the results in Table 4 for PAM models with $K = 2$ concerning to the state `Stats` were not clear as to which type of pattern it is more characteristic in the second month of usage. This led us to investigate two more properties: likelihood of going from `Stats` to any of the states highly characteristic of each pattern within the same session (see $\mathsf{VPsession}_{\mathsf{AP}}$ in Table 6) and the average number of steps from `Stats` to other states and vice versa (see $\mathsf{SCstate2state}_{\mathsf{AP}}$ in Table 7).

We note that while GPAM models are a generalisation of PAM models, the distinctive contribution of PAM is the user trace distribution, which gives us more detailed information than (GPAM) steady-state properties. For example it can reveal which user traces are "interesting" and should be investigated further. This is beyond the the scope of this paper, but such investigations involve checking properties of selected individual user traces, for example those that have engaged for a long/short time period, and/or defining sub-populations according to ranges of θ and checking properties on models inferred from that sub-population.

Choice of number of activity patterns.. The patterns are inferred by standard statistical methods based on non-linear optimisation. We do not model for predictability, for two reasons. First, because there is no true model of the generating process, but one that is posited based on known characteristics such as the sequential nature of user-initiated application events. This requires us to study the time-series behaviour that has been logged from a probabilistic perspective. Second, our goal is not to predict future behaviour, but indeed to influence future behaviour through (possible) redesign of application that, for

example, either makes desirable/common behaviour easier to attain, or undesirable/rare behaviour harder. The number K of admixture components is an important exploratory tool. We do not try to define an optimal (statistical) value for K, but rather use it to reflect the variety of usage styles that are meaningful in the context of the software application we study. Typically, we have considered models with up to $K = 5$ activity patterns.

Longitudinal and other types of analysis. Our example indicates results may be sensitive to the chosen time interval for logged data. Common sense indicates this is to be expected, for example differences between the first day of use and use after several weeks. However, there may be software-specific temporal sensitivities, for example, our application analysis revealed an unanticipated sensitivity in the first month and the second months of usage.

We note there are other (static) categorisations such as device type, timezone, length of user trace in number of sessions, frequency of sessions, and average length of sessions, to name a few. These of course can be combined with our approach.

Complexity. We remark that the complexity of models does not depend on the data but on the underlying functionality of the software. Our example software involved 16 observed states and this was easily tractable.

Related work. Our work was initially motivated by an empirical study of simplicial mixtures modelling webpage browsing and telephone usage modelling [6] for capturing common behavioural patterns in event streams. Based on this work, a hidden Markov model for automatic classification of software behaviour was suggested in [7] as possible future work. To our knowledge, no one else investigated admixture models for modelling software usage behaviour. The closest related work is [12] where DTMC models of user behaviour are based on static user attributes rather than on inferred behaviours, assuming within-class use to be homogeneous, whereas we demonstrate within-class variation. Recently [15] studied a simple Markov state transition model for smartphone usage with four states (on, off, locked, unlocked), which was constructed based on computing the frequencies of each bigram in the event stream. Another strand of recent work is focused in inferring probabilistic behavioural models (as DTMCs) for software systems using k-means clustering [24] and MapReduce [25].

Zang et al. [26] extended probabilistic model checking to HMMs, this is not applicable as we are interested in properties involving both latent and observed states; moreover, we do not analyse properties directly on AR-HMMs, but on their flattened version as DTMCs. The two hidden Markov models we use, PAM and GPAM, are examples of dynamic Bayesian networks (DBNs) [18, 19]. DBNs can be analysed using Bayesian statistical model checking [27], however our models are easily encoded and analysed using probabilistic model checking. Extensive works on inferring parameters for first-order HMM models (as DBNs) from user traces for runtime verification purposes can be found in [28, 22], however we are looking at a different class of HMM, namely admixture models, for identifying and analysing usage models within a population of user traces.

Recent works on modelling human mobility [29, 30] use Markov models to generate spatio-temporal patterns including information on periodicity, duration, and number of visits as well as distribution of distances. Their data sets are somewhat similar to our

user traces of software interaction, with spatial location mapping to software interaction events. Our Markovian models are simpler as they are time-homogeneous, allowing us to use probabilistic temporal logic for analysing and characterising them.

The work presented in this paper builds on three previous studies: initial results for AppTracker analysis using PAM models [23], analysis of individual user models with application to a mobile game application [14], and formal definitions for PAM and GPAM, with some initial analysis results [31].

10. Conclusions

We have defined two new data-driven models and probabilistic temporal logic properties to answer the research question: *how can we model and understand the ways in which users interact with software*? Our models are Markovian admixtures that include both observed and latent states and are based on inferring a finite number of activity patterns from sequences of user interactions (called user traces). In the PAM models, the activity patterns are encoded directly as DTMCs and a distribution is defined over them for each user trace. In the GPAM models, the activity patterns are the observation probability matrices in an auto-regressive hidden Markov model. Given sets of user traces over different time intervals (e.g. first month of usage, second month of usage), model parameters are learnt using the Expectation-Maximisation and Baum-Welch algorithms. Single activity pattern properties encapsulate hypotheses about behaviours within an activity, for example the number of steps from one state to another. Multiple activity pattern properties encapsulate more complex hypotheses that may involve several activity patterns, for example the likelihood of changing activity pattern after visiting a particular state.

Through a real-life example software application and logged user traces from 322 users of *AppTracker*, we have demonstrated how different combinations of data set, model, and temporal property reveal insights into how users actually use software. The results have proved useful for the particular application and sometimes the results were unexpected, for example the interpretations of the different activity patterns and how their distribution varies over different months of usage, and when activity patterns do or do not relate to the functionality of top level user menus. We simply did not have the tools to perform this type of analysis previously.

We conclude that by modelling and analysing the different generating process of software usage over a dynamic, heterogeneous population of users, we have developed new and effective tools for understanding how software is actually used. Future work will include investigating the limits of tractability, predictability of behaviours, and application to other types of interactive system.

Acknowledgment. This research was supported by EPSRC Programme Grants *A Population Approach to Ubicomp System Design* (EP/J007617/1) and *Science of Sensor Systems Software* (EP/N007565).

References

[1] C. Baier, J.-P. Katoen, Principles of Model Checking, The MIT Press, 2008.

[2] M. Z. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of Probabilistic Real-Time Systems, in: G. Gopalakrishnan, S. Qadeer (Eds.), Proceedings of CAV'11, Vol. 6806 of Lecture Notes in Computer Science, Springer, 2011, pp. 585–591.
[3] M. Bell, M. Chalmers, L. Fontaine, M. Higgs, A. Morrison, J. Rooksby, M. Rost, S. Sherwood, Experiences in Logging Everyday App Use, ACM Proceedings of Digital Economy'13.
[4] M. Hall, M. Bell, A. Morrison, S. Reeves, S. Sherwood, M. Chalmers, Adapting ubicomp software and its evaluation, in: T. C. N. Graham, G. Calvary, P. D. Gray (Eds.), Proceedings of EICS'09, ACM, 2009, pp. 143–148.
[5] S. Rogers, M. Girolami, A First Course in Machine Learning, Chapman and Hall/CRC, 2015.
[6] M. Girolami, A. Kabán, Simplicial Mixtures of Markov Chains: Distributed Modelling of Dynamic User Profiles, in: S. Thrun, L. K. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16 (NIPS'03), MIT Press, 2004, pp. 9–16.
[7] J. F. Bowring, J. M. Rehg, M. J. Harrold, Active learning for automatic classification of software behavior, in: G. S. Avrunin, G. Rothermel (Eds.), Proceedings of the International Symposium on Software Testing and Analysis (ISSTA'04), ACM, 2004, pp. 195–205.
[8] J. Borges, M. Levene, Data Mining of User Navigation Patterns, in: B. Masand, M. Spiliopoulou (Eds.), Web Usage Analysis and User Profiling: International WEBKDD'99 Workshop, Springer, 2000, pp. 92–112.
[9] F. Chierichetti, R. Kumar, P. Raghavan, T. Sarlós, Are web users really Markovian?, in: A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, S. Staab (Eds.), Proceedings of the 21st World Wide Web Conference 2012 (WWW'12), ACM, 2012, pp. 609–618.
[10] P. Singer, D. Helic, B. Taraghi, M. Strohmaier, Detecting Memory and Structure in Human Navigation Patterns Using Markov Chain Models of Varying Order, PLOS ONE 9 (7) (2014) 1–21.
[11] P. Singer, D. Helic, A. Hotho, M. Strohmaier, HypTrails: A Bayesian Approach for Comparing Hypotheses About Human Trails on the Web, in: A. Gangemi, S. Leonardi, A. Panconesi (Eds.), Proceedings of the 24th International Conference on World Wide Web, WWW 2015, ACM, 2015, pp. 1003–1013.
[12] C. Ghezzi, M. Pezzè, M. Sama, G. Tamburrelli, Mining Behavior Models from User-Intensive Web Applications, in: Proceedings of ICSE'14, Hyderabad, India, ACM, 2014, pp. 277–287.
[13] H. W. Thimbleby, P. A. Cairns, M. Jones, Usability analysis with Markov models, ACM Trans. Comput.-Hum. Interact. 8 (2) (2001) 99–132.
[14] O. Andrei, M. Calder, M. Higgs, M. Girolami, Probabilistic Model Checking of DTMC Models of User Activity Patterns, in: Proceedings of QEST 2014, Vol. 8657 of Lecture Notes in Computer Science, Springer, 2014, pp. 138–153.
[15] V. Kostakos, D. Ferreira, J. Gonçalves, S. Hosio, Modelling smartphone usage: a Markov state transition model, in: P. Lukowicz, A. Krüger, A. Bulling, Y. Lim, S. N. Patel (Eds.), Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'16), 2016, pp. 486–497.
[16] V. Tran, D. Maxwell, N. Fuhr, L. Azzopardi, Personalised Search Time Prediction using Markov Chains, in: J. Kamps, E. Kanoulas, M. de Rijke, H. Fang, E. Yilmaz (Eds.), Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR'17), ACM, 2017, pp. 237–240.
[17] G. Murray, Markov reward models for analyzing group interaction, in: E. Lank, A. Vinciarelli, E. E. Hoggan, S. Subramanian, S. A. Brewster (Eds.), Proceedings of the 19th ACM International Conference on Multimodal Interaction (ICMI'17), ACM, 2017, pp. 336–340.
[18] K. P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[19] L. Sucar, Probabilistic Graphical Models: Principles and Applications, Advances in Computer Vision and Pattern Recognition, Springer London, 2015.
[20] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, Journal of the Royal Statistical Society. Series B (Methodological) 39 (1) (1977) 1–38.
[21] L. Welch, Hidden Markov Models and the Baum-Welch Algorithm, IEEE Information Theory Society Newsletter, Dec. 2003.
[22] E. Bartocci, R. Grosu, A. Karmarkar, S. A. Smolka, S. D. Stoller, E. Zadok, J. Seyster, Adaptive Runtime Verification, in: S. Qadeer, S. Tasiran (Eds.), Proceedings of RV'12, Vol. 7687 of Lecture Notes in Computer Science, Springer, 2012, pp. 168–182.
[23] O. Andrei, M. Calder, M. Chalmers, A. Morrison, M. Rost, Probabilistic Formal Analysis of App Usage to Inform Redesign, in: Proceedings of iFM 2016, Vol. 9681 of Lecture Notes in Computer Science, Springer, 2016, pp. 115–129.
[24] K. S. Luckow, C. S. Pasareanu, Log2model: inferring behavioral models from log data, in: Proceed-

ings of the 18th IEEE International High-Level Design Validation and Test Workshop (HLDVT'16), IEEE, 2016, pp. 25–29.
[25] C. Luo, F. He, C. Ghezzi, Inferring software behavioral models with MapReduce, Sci. Comput. Program. 145 (2017) 13–36.
[26] L. Zhang, H. Hermanns, D. N. Jansen, Logic and Model Checking for Hidden Markov Models, in: Farn Wang (Ed.), Proceedings of FORTE 2005, Vol. 3731 of Lecture Notes in Computer Science, Springer, 2005, pp. 98–112.
[27] C. J. Langmead, Generalized Queries and Bayesian Statistical Model Checking in Dynamic Bayesian Networks: Application to Personalized Medicine, in: Proceedings of CSB'09, 2009, pp. 201–212.
[28] S. D. Stoller, E. Bartocci, J. Seyster, R. Grosu, K. Havelund, S. A. Smolka, E. Zadok, Runtime Verification with State Estimation, in: Proceedings of RV'11, Vol. 7186 of Lecture Notes in Computer Science, Springer, 2011, pp. 193–207.
[29] L. Pappalardo, F. Simini, Data-driven generation of spatio-temporal routines in human mobility, Data Mining and Knowledge Discovery 32 (3) (2018) 787–829.
[30] S. Jiang, Y. Yang, S. Gupta, D. Veneziano, S. Athavale, M. C. González, The TimeGeo modeling framework for urban mobility without travel surveys, Proceedings of the National Academy of Sciences 113 (37) (2016) E5370–E5378.
[31] O. Andrei, M. Calder, Temporal Analytics for Software Usage Models, in: A. Cerone, M. Roveri (Eds.), Software Engineering and Formal Methods, Vol. 10729 of Lecture Notes in Computer Science, Springer International Publishing, 2017, pp. 9–24.

Appendix A. Analysis of SAP properties on a PAM and on a GPAM for AppTracker

Table A.13: Properties $\mathsf{VPinit}_{\mathsf{AP}}$, $\mathsf{VCinit}_{\mathsf{AP}}$, $\mathsf{SCinit}_{\mathsf{AP}}$ for PAM, $K = 3$, $N = 50$

Prop.	Usage	OverallUsage			StackedBars			PeriodSelector			Stats			UseStop		
	month	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3
$\mathsf{VPinit}_{\mathsf{AP}}$	1st	0.88	0.99	0.99	0.97	0.76	0.16	0.98	0.00	0.00	0.34	0.84	0.99	0.99	0.99	0.99
	2nd	1.00	0.84	0.95	0.00	0.98	0.76	0.00	0.90	0.75	0.00	0.04	0.92	1.00	0.99	0.96
	3rd	0.44	0.99	0.96	0.99	0.00	0.75	0.08	0.13	0.93	0.45	0.81	0.78	0.99	0.99	0.86
$\mathsf{VCinit}_{\mathsf{AP}}$	1st	4.79	6.11	12.27	4.11	1.41	0.17	7.44	0.00	0.00	0.79	2.16	4.37	5.74	6.73	10.08
	2nd	13.03	5.68	5.06	0.00	5.61	1.58	0.00	6.31	1.18	0.00	0.06	4.98	11.87	6.08	4.95
	3rd	0.83	12.37	8.46	9.49	0.00	1.59	0.38	0.27	3.56	0.82	1.59	4.25	10.73	10.39	2.24
$\mathsf{SCinit}_{\mathsf{AP}}$	1st	22.78	8.80	2.44	11.46	34.97	277.88	11.47	—	—	115.82	28.01	10.80	7.48	6.24	3.80
	2nd	1.99	26.09	12.59	—	8.11	41.90	—	20.82	36.22	—	1094.65	20.57	3.08	7.22	11.63
	3rd	85.53	2.31	13.29	4.10	—	35.19	543.47	321.42	14.76	84.53	30.34	33.99	3.54	3.74	22.93

Table A.14: Properties $\mathsf{VPinit}_{\mathsf{AP}}$, $\mathsf{VCinit}_{\mathsf{AP}}$, $\mathsf{SCinit}_{\mathsf{AP}}$ for GPAM, $K = 3$, $N = 50$

Prop.	Usage	OverallUsage			StackedBars			PeriodSelector			Stats			UseStop		
	month	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3
$\mathsf{VPinit}_{\mathsf{AP}}$	1st	0.99	0.99	0.59	0.99	0.23	0.60	0.98	0.07	0.65	0.80	0.16	0.36	0.97	0.30	0.99
	2nd	0.99	0.44	0.65	0.78	0.99	0.98	0.19	0.71	0.97	0.94	0.00	0.25	0.70	0.99	0.99
	3rd	0.99	0.90	0.76	0.40	0.99	0.74	0.11	0.37	0.99	0.95	0.55	0.40	0.99	0.99	0.96
$\mathsf{VCinit}_{\mathsf{AP}}$	1st	5.37	22.66	1.12	5.94	0.27	0.89	3.48	0.10	1.80	1.92	0.26	0.45	2.54	0.34	11.74
	2nd	16.75	0.76	1.12	1.57	9.38	3.76	0.23	1.35	3.49	4.11	0.00	0.35	1.22	8.83	8.58
	3rd	12.84	2.79	4.00	0.50	8.16	1.58	0.51	0.44	5.59	4.10	0.89	1.25	8.56	8.75	3.57
$\mathsf{SCinit}_{\mathsf{AP}}$	1st	8.59	5.01	55.98	8.68	247.66	55.63	13.44	1153.58	47.65	29.52	295.84	111.09	17.16	146.32	3.25
	2nd	4.05	88.28	47.26	44.94	4.09	12.66	223.72	39.81	14.48	21.93	—	172.74	57.66	4.71	4.68
	3rd	2.27	21.54	34.66	98.96	5.05	36.49	416.06	106.05	8.39	16.21	63.78	96.67	4.65	4.65	13.47