Probabilistic Formal Analysis of App Usage to Inform Redesign

Oana Andrei, Muffy Calder, Matthew Chalmers, Alistair Morrison, and Mattias Rost

School of Computing Science, University of Glasgow, UK

Abstract. Evaluation and redesign of user-intensive mobile applications is challenging because users are often heterogeneous, adopting different patterns of activity, at different times. We set out a process of integrating *statistical*, longitudinal analysis of actual logged behaviours, *formal, probabilistic discrete state models* of activity patterns, and hypotheses over those models expressed as *probabilistic temporal logic properties* to inform redesign. We employ formal methods not to the design of the mobile application, but to characterise the different probabilistic patterns of actual *use* over various time cuts within a population of users. We define the whole process from identifying questions that give us insight into application usage, to event logging, data abstraction from logs, model inference, temporal logic property formulation, visualisation of results, and interpretation in the context of redesign. We illustrate the process through a real-life case study, which results in a new and principled way for selecting content for an extension to the mobile application.

1 Introduction

Evaluation and redesign of deployed user-intensive mobile applications (apps) is challenging because users are often heterogeneous and adopt different patterns of activity, at different times. Good redesign must support users' different styles of use, and should not be based solely on static attributes of users, but on those styles, which may be dynamic. This raises many questions, including: what characterises the usage of a user, how should we identify the different styles of use, how does that characterisation evolve, e.g. over an individual user trace, and/or over days and months, and how do *properties* of usage inform evaluation and redesign? This paper attempts to answer these questions, setting out a novel process of integrating statistical analysis of logged behaviours, probabilistic formal methods and probabilistic temporal logic with rewards.

Our approach is based on integrating three powerful ingredients: (1) *inference* of *admixture* probabilistic Markov models (called *activity patterns*) from automatically logged data on user sessions, (2) *characterisation* of the activity patterns by probabilistic *temporal logic* properties using model checking techniques, and (3) *longitudinal* analysis of usage data drawn from different time cuts (e.g. the first day, first month, second month, etc.). Our contribution is defining the whole process from identifying questions that give us insight into

an app usage, to event and attribute logging, data pre-processing and abstraction from logs, model inference, temporal logic property formulation using the probabilistic temporal logic PCTL with rewards [1], visualisation of results and interpretation in the context of redesign. Our work provides new insights into app usage and affords new redesign ideas that are solidly grounded in observed activity patterns. We apply *scientific* and *formal* methods in a novel way to the observed *use* of artefacts that have been *engineered*. We illustrate throughout with a case study of AppTracker [2], a freely available mobile, personal productivity app that allows users to collect quantitative statistics about the usage of all apps installed on their iOS devices, i.e., iPhones, iPads, or iPods.

Initially, we *instrument* the app of interest to log usage behaviours and process them into sets of user traces expressed in terms of higher level actions. These actions are carefully selected, jointly by analysts and developers, to relate to the intended analysis: they determine the scope of properties and the dimensions of the state space underlying the model. We segment the sets of traces into different time cuts so that we can determine how activity patterns evolve over time.

For each time cut of user traces we infer *admixture bigram models of activity patterns*, where activity patterns are discrete-time Markov chains. We use admixture models because we are not classifying users into a single prototypical behavioural trait (or usage style), but we have complex behavioural traits where individuals move between patterns during an observed user trace. Bigrams, which provide the conditional probability of an action given the preceding action, are one of the most successful models for language analysis (i.e. streams of symbols) and are good representations for populations of dynamic, heterogeneous users [3]. We characterise each user trace as an admixture of K activity patterns shared within the population of users. K is an important exploratory tool, and rather than assuming or finding an optimal value for K, we use it to explore the variety of usage styles that are meaningful to redesign. We typically start with low K values, but the choice may be dependent on factors intrinsic to the app. For a given K value, the parameters of the inferred model are the probabilities of a given action (from a preceding action), for each activity pattern, as well as the probabilities of transitioning *between* activity patterns. It is important to note that we are not inferring the underlying system topology, which is determined by the functionality of the app. We are investigating an artefact that has been engineered, but there may be differing generating processes of use. We employ a standard local non-linear optimisation algorithm for parameter estimation – the Expectation-Maximisation (EM) algorithm [4]. We use EM, as opposed to say MCMC, because it is fast and computationally efficient for our kind of data. EM converges provably to a local optimum of the criterion, in this case the likelihood function, and as such validation is not an issue. To the best of our knowledge, inferring such temporal structures has not been described outside our group.

We then *hypothesise* temporal probabilistic properties, expressed in PCTL extended with rewards [1], [5] to explore the activity patterns, considering various admixture models, values for K, and time cuts. We compare the distribution of patterns in the population of users longitudinally and structurally drawing on all

the formal analysis to provide new, grounded insights into possible redesign. In our case study, our analysis mitigates against a simple partitioning of different versions, specific to each activity, but rather offers a new and principled way of selecting glanceable information as an extension of the app.

Three AppTracker designers were involved in this paper, guiding the integration of formal analysis with hypotheses about user behaviours. This is our second application of formal analysis to models of inferred user behaviour: in [6] we defined activity patterns for an individual user as a user metamodel with respect to a population of users, and analysed a mobile game app. This work differs substantially in that here our goal is redesign in the context of a different app, we use the parameter K as an exploratory tool, employ completely different temporal properties (e.g. using rewards) and longitudinal analysis, and analyse the whole population of users, comparing distributions of activity patterns across the user population longitudinally for a fixed and different values for K.

2 Technical Background

We assume familiarity with Markov models, PCTL, PRISM probabilistic model checking, bigram models and Expectation-Maximisation algorithms.

A **discrete-time Markov chain** (DTMC) is a tuple $\mathcal{D} = (S, \bar{s}, \mathtt{P}, l)$ where: S is a set of states; $\bar{s} \in S$ is the initial state; $\mathtt{P} : S \times S \to [0,1]$ is the transition probability function (or matrix) such that for all states $s \in S$ we have $\sum_{s' \in S} \mathtt{P}(s, s') = 1$; and $l : S \to 2^{\mathcal{A}}$ is a labelling function associating to each state s in S a set of valid atomic propositions from a 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 $\mathtt{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. A transition is also called a *time-step*.

Probabilistic Computation Tree Logic (PCTL) [1] allows expression of a probability measure of the satisfaction of a temporal property. The syntax is:

$$\begin{array}{l} \textit{State formulae } \phi ::= \textit{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathtt{P}_{\bowtie p}[\psi] \\ \textit{Path formulae } \psi ::= \mathsf{X}\,\phi \mid \phi\,\mathsf{U}^{\leq n}\,\phi \mid \mathsf{F}^{\leq n}\,\phi \end{array}$$

where a ranges over a set of atomic propositions $\mathcal{A}$, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, and $n \in \mathbb{N} \cup \{\infty\}$. State formulae are also called *temporal properties*. The usual semantics apply, with U and F standing for *until* and *eventually*, respectively.

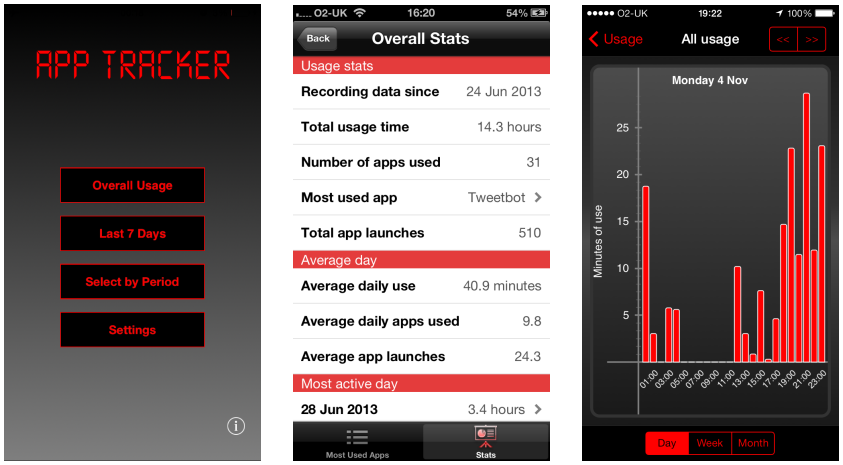
PRISM [7] computes a satisfaction probability, e.g. $\mathtt{P}_{=?}[\,\psi\,]$, allowing also for *experimentation* when the range and step size of the variable(s) are specified. PRISM supports a *reward*-based extension of PCTL, called *rPCTL*, that assigns non-negative real values to states and/or transitions. $\mathtt{R}\{x\}_{=?}\left[\,\mathsf{C}^{\leq N}\right]$ computes the reward named x accumulated along *all* paths within N time-steps, $\mathtt{R}\{x\}_{=?}[\,\mathsf{F}\,\phi\,]$ computes the reward named x accumulated along *all* paths until ϕ is satisfied. Filtered probabilities check for properties that hold from sets of states satisfying given propositions. Here we use `state` as the filter operator: e.g., $\mathtt{filter}(\mathtt{state}, \phi, \textit{condition})$ where ϕ is a state formula and *condition* a Boolean proposition uniquely identifying a state in the DTMC.

Inference of admixture bigram models. Given a vocabulary V of size n, a trace over V is a finite non-empty sequence of symbols from V. Let $S = \{0, 1, \ldots, n\}$ be the set of states and consider a bijective mapping $V \mapsto S$. Let x be a data sample of M traces over V, $x = \{x_1, \ldots, x_M\}$. Each trace x_m can be represented as a DTMC: the set of states $S = \{0, 1, \ldots, n\}$, the initial state is 0, the transition probability matrix is the $n \times n$ *transition-occurrence matrix* such that x_{mij} on position (i, j) gives the number of times the pair (x_{mi}, x_{mj}) occurs in the trace x_m. Consider K $n \times n$ transition matrices denoted Φ_k over the states in S, for $k = 1, \ldots, K$, such that Φ_{kij} denotes the probability of moving from state i to state j. Also consider a $M \times K$ matrix Θ such that at any point in time Φ_k is used by the trace x_m with probability Θ_{mk}. Let $\lambda = \{\Phi_k, \Theta_{mk} \mid k = 1, \ldots, K;\ m = 1, \ldots, M\}$ be the parameters of the statistical model. We use the EM algorithm [4] to find maximum likelihood parameters λ of observing each trace x_m, restarting the algorithm whenever the log-likelihood has multiple-local maxima. The result is an *admixture bigram model*: a Θ-weighted mixture of the K DTMCs Φ_k. The model is bigram because only dependencies between adjacent symbols in the trace are considered.

3 Case Study: AppTracker

AppTracker is an iOS application that provides a user with information on the usage of their device. It operates on iPhones/iPads/iPods, running in the background and monitoring the opening and closing of apps as well as the locking and unlocking of the device. It was released in August 2013 and downloaded over 35,000 times. The interface displays a series of charts and statistics to give insight into how long one is spending on their device, the most used apps, how these stats fluctuate over time, etc.. Figure 1 shows three views from the app. The main menu screen offers four main options (Fig. 1(a)). The first menu item, *Overall Usage*, contains quick summaries of all the data recorded since AppTracker was installed opening the views TopApps and Stats (Fig. 1(b)). The second menu item, *Last 7 Days*, displays a chart limited to the activity recorded over the last 7 days. The third menu item, *Select by Period*, shows statistics for a selected period of time. For example, one could investigate which apps one used the most last Saturday, see how the time one spent on Facebook varied each day across last month, or examine patterns of use over a particular day (Fig. 1(c)). The final menu option, *Settings*, allows a user to start and stop the tracker, or to reset their recorded data. A *Terms and Conditions* screen is shown to a user on first launch that describes all the data that will be recorded during its use and provides contact details to allow the user to opt out at any time.

Preparing raw logged data. Data is collected within the SGLog framework [8]. Each log, stored in a MySQL database, contains information about the user, the device, and the event that took place. For our analysis, we are interested in the events resulting in a switch between views within the app. The raw data is extracted and processed using JavaScript to obtain user traces of views (for each user). A special view denotes when the user leaves the app (UseStop) and we



(a) Main menu (b) Overall stats (c) Device usage for one day

Fig. 1. Screenshots from AppTracker

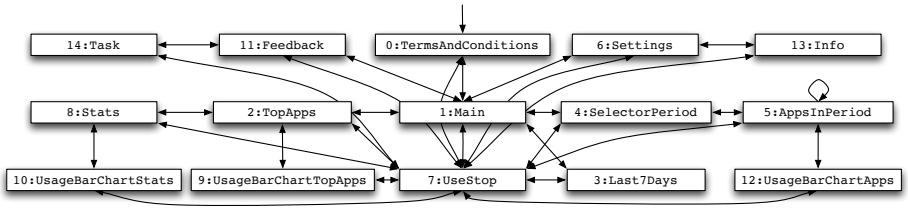


Fig. 2. AppTracker state diagram

define a *session* as the event sequence delimited by two UseStop states, except the initial session which starts from the TermsAndConditions. This results in a total of 15 unique views, with transitions, illustrated in Fig. 2 with the following meaning: (0) TermsAndConditions is the terms and conditions page; (1) Main is the main menu screen; (2) TopApps shows the summary of all recorded data; (3) Last7Days shows the last 7 days of top 5 apps used; (4) SelectPeriod shows app usage stats for a selected time period; (5) AppsInPeriod shows apps used for a selected period; (6) Settings shows the settings options; (7) UseStop stands for closing/sending to background the AppTracker; (8) Stats shows statistics of app usage; (9) UsageBarChartTopApps shows app usage when picked from TopApps; (10) UsageBarChartStats shows app usage when picked from Stats; (11) Feedback shows a screen for giving feedback; (12) UsageBarChartApps shows app usage when picked from AppsInPeriod; (13) Info shows information about the app; (14) Task shows a feedback question chosen from the Feedback view. The 15 views relate directly to the underlying atomic propositions used later in the DTMCs and we map user traces to 15×15 transition-occurrence matrices.

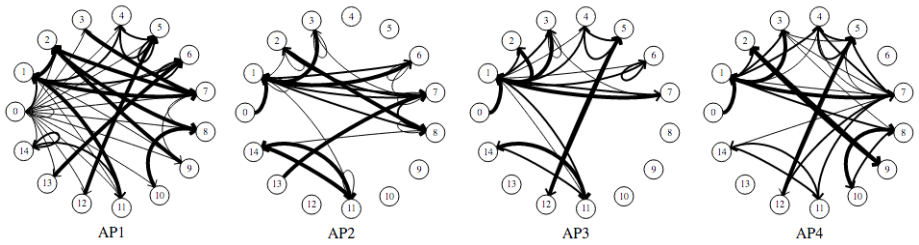


Fig. 3. The DTMCs of the activity patterns for $K = 4$ and first month of usage.

Data for this study. All data was gathered between August 2013 and May 2014, from 489 users. The average time spent within the app per user is 626s (median 293s), the average number times going into the app is 10.7 (median 7), the average user trace length is 73.6 view transitions (median 46). We segment the user traces into time cuts of the interval form $[d_1, d_2)$, which includes the user traces from the d_1-th up to the d_2-th day of usage.

4 Inferring Admixture Bigram Models

For each chosen value of K and time cut of the logged data we obtain K DTMCs with 15×15 transition matrices called *activity patterns* and an $M \times K$ matrix Θ, where M is the number of user traces, and with each row a distribution over the K activity patterns. For each activity pattern APk, for $k = 1, \ldots, K$, we generate automatically a PRISM model with one variable x for the views of the app with values ranging from 0 to 14. For each state value of x we have a PRISM command defining all possible 15 probabilistic transitions where Φ_{kij} is the transition probability from state $x = i$ to the updated state $x' = j$ in activity pattern APk, for all $i, j = 0, \ldots, 14$. For each state value we associate the label corresponding to a higher level state in AppTracker (see the mapping in Fig. 2) as well as a reward structure which assigns a reward of 1 to that state. The PRISM file for each activity pattern also includes a reward structure assigning a rewards of 1 to each transition (or time step) in the DTMC. All our PRISM models have at most 15 states and at most 51 transitions.

We implemented the EM algorithm in Java, applying the algorithm to data sets with 100 iterations maximum and 200 restarts maximum. Running the EM algorithm takes about 119s for $K = 2$, 162s for $K = 3$, and 206s for $K = 4$ on a 2.8GHz Intel Xeon. Timings are obtained by running the algorithm 90 times. The algorithm is single threaded and runs on one core.

As example, Figure 3 illustrates state-transition diagrams of all the $K = 4$ activity patterns, the thickness of the transitions corresponding to ranges of probability: the thicker the line, the higher the probability of that transition. Note this illustration does not include the distribution over the activity patterns.

Table 1. rPCTL properties PROP1–PROP5

ID	Formula and informal description
PROP1	$\mathsf{P}_{=?}[\,!\,\ell\,\mathsf{U}^{\leq N}\,\ell\,]$: Probability of visiting a ℓ-labelled state for the first time from the initial state within N time steps
PROP2	$\mathsf{R}\{\text{"}r\_\ell\text{"}\}_{=?}[\mathsf{C}^{\leq N}]$: Expected number of visits to a ℓ-labelled state from the initial state within N time steps
PROP3	$\mathsf{R}\{\text{"}r\_Steps\text{"}\}_{=?}[\mathsf{F}\ \ell]$: Expected number of time steps to reach a ℓ-labelled state from the initial state
PROP4	$\texttt{filter}(\texttt{state}, \mathsf{R}\{\text{"}r\_Steps\text{"}\}_{=?}[\mathsf{F}\ \ell_1],\ \ell_2)$: Expected number of time steps to reach a ℓ_1-labelled state labelled from a ℓ_2-labelled state
PROP5	$\texttt{filter}(\texttt{state}, \mathsf{P}_{=?}[((!\,\ell_1)\&(!\,\text{"}\mathsf{UseStop}\text{"}))\ U^{\leq N}\ \ell_1], \ell_2)$: Probability of reaching for the first time a ℓ_1-labelled state from a ℓ_2-labelled state during a session

5 Analysing rPCTL Properties

We have found that most patterns for logic properties (e.g. probabilistic response, probabilistic precedence, etc.) relate to the design of reactive systems and are not generally helpful for evaluation of user-intensive apps. However, a study of which patterns would be useful is beyond the scope of this paper. Table 1 lists the rPCTL properties we used, with state labels ℓ, ℓ_1, ℓ_2. PROP1, PROP2, and PROP3 are the properties we investigated initially; PROP4 and PROP5 were identified later, prompted by designers' hypotheses and inconclusive initial results. PROP4 generalises PROP3 by analysing traces starting with a chosen state, not necessarily the initial one.

We inferred models for $K \in \{2, 3, 4\}$ for various time cuts and performed analysis of rPCTL properties PROP1 – PROP5 on all activity patterns. For brevity, here we show only properties concerning the states: TopApps, Stats, SelectPeriod, Last7Days, UseStop. These five states showed significant results and differences across time cuts and temporal properties and the designers showed particular interest in them when formulating hypotheses about the actual app usage.

We adopt the following interpretations of model checking results for PROP1, PROP2, and PROP3 in our case study for the same value of N. We say that a pair of state and activity pattern (ℓ, APi) scores a better (resp. worse) result than (ℓ', APj), for all $1 \leq i \neq j \leq K$, where either $\ell \neq \ell'$ or $i \neq j$, if: PROP1 returns a higher (resp. lower) value, PROP2 a higher (resp. lower) value, and PROP3 a positive lower (resp. higher) value for (ℓ, APi) than for (ℓ', APj).

Analysing rPCTL properties for $K = 2$. We verify PROP1, PROP2, and PROP3 on the two activity patterns AP1 and AP2 for six time cuts: first day $[0, 1)$, first week minus the first day $[1, 7)$, the first month minus the first week $[7, 30)$, the first month $[0, 30)$, the second month $[30, 60)$ and the third month $[60, 90)$, and for N ranging from 10 to 150 with step-size 10. Here we only show the results for $N = 50$ in Table 2. The best results with respect to the property

Table 2. PROP1 (the probability of reaching a given state for the first time within N steps), PROP2 (the expected number of visits to a given state within N steps), and PROP3 (the expected number of time steps to reach a given state) checked for different states and time cuts, and for $N = 50$ steps. The best results with respect to the property checked across the two patterns are in bold font.

Property	Time	TopApps		Stats		SelectPeriod		Last7Days		UseStop	
	cut	AP1	AP2	AP1	AP2	AP1	AP2	AP1	AP2	AP1	AP2
PROP1	$[0, 1)$	**0.99**	**0.99**	**0.99**	0.83	0.47	**0.79**	0.49	**0.96**	**0.99**	**0.99**
	$[1, 7)$	**0.99**	**0.99**	**0.98**	0.80	0	**0.93**	0	**0.98**	**0.99**	**0.99**
	$[7, 30)$	**0.99**	**0.99**	**0.99**	0.64	0.01	**0.94**	0.84	**0.96**	**0.99**	**0.99**
	$[0, 30)$	**0.99**	**0.99**	**0.99**	0.75	0.21	**0.92**	0.44	**0.98**	**0.99**	**0.99**
	$[30, 60)$	**0.99**	**0.99**	0	**0.90**	0.73	**0.83**	0.56	**0.98**	**1**	0.99
	$[60, 90)$	**1**	0.95	**0.96**	0.72	0	**0.94**	0	**0.97**	**1**	0.99
PROP2	$[0, 1)$	**13.94**	7.44	**7.63**	2.15	0.79	**1.82**	0.70	**3.13**	**11.41**	6.17
	$[1, 7)$	**17.22**	5.77	**4.00**	2.31	0	**3.97**	0	**4.03**	**12.91**	6.30
	$[7, 30)$	**14.93**	7.15	**5.43**	1.47	0.01	**4.61**	1.78	**3.41**	**12.86**	5.74
	$[0, 30)$	**14.67**	6.48	**5.08**	1.90	0.24	**3.58**	0.58	**3.99**	**11.00**	6.51
	$[30, 60)$	**13.40**	6.83	0	**3.76**	**4.41**	2.04	0.85	**4.54**	**12.46**	5.61
	$[60, 90)$	**17.30**	5.83	**2.94**	2.60	0	**3.26**	0	**4.43**	**13.96**	5.63
PROP3	$[0, 1)$	**3.31**	8.41	**8.18**	28.67	79.32	**32.46**	74.87	**15.56**	**4.86**	7.88
	$[1, 7)$	**2.05**	10.70	**12.44**	31.90	∞	**19.12**	∞	**12.38**	**3.85**	7.55
	$[7, 30)$	**2.52**	9.68	**9.70**	48.61	∞	**17.78**	26.61	**14.58**	**3.88**	8.44
	$[0, 30)$	**3.05**	9.73	**11.01**	36.03	209.68	**19.94**	87.54	**12.19**	**4.67**	7.43
	$[30, 60)$	**4.04**	10.34	∞	**22.33**	38.21	**28.28**	61.74	**11.08**	**1**	8.82
	$[60, 90)$	**2.02**	15.28	**16.53**	39.68	∞	**17.41**	∞	**11.56**	**3.57**	8.90

checked across the two patterns are in bold font: we can easily see that the two patterns correspond to different behaviours (results) with respect to the five states to be more likely, more often, and more quickly reached. Note that looking at the analysis results for UseStop, on average we see twice as many sessions under AP1 than under AP2 and the average session length in terms of time steps under AP2 is double the average session length under AP1.

If we overlook (for now) the results for the time cut $[30, 60)$, we conclude that there are two distinct activity patterns:

AP1: Overall Viewing pattern corresponds to more likely, more often, and more quickly to reach TopApps and Stats, thus more higher level stats visualisations and shorter sessions.

AP2: In-depth Viewing pattern corresponds to more likely, more often, and more quickly to reach Last7Days and SelectPeriod, thus more in-depth stats visualisations and longer sessions.

Now considering the time cut $[30, 60)$, on Table 2 we note slightly different results for this time cut compared to the more consistent results for the other five time cuts: a high number of visits to and a relative low number of time steps to reach Stats no longer belongs to AP1, but to AP2; PROP1 and PROP2 for SelectPeriod no longer discriminate clearly between AP1 and AP2 due to very close results. As a consequence we analyse the additional rPCTL properties (see Table 3)

Table 3. Properties PROP4 (expected number of time steps to reach a row state from a column state for each pattern) and PROP5 (probability of reaching for the first time a row state from a column state during a session for each pattern) verified for $K = 2$, time cut $[30, 60)$. **Blue-coloured, bold font results** are characteristic to the **Overall Viewing pattern**, while red-coloured font results to the In-depth Viewing pattern; default text colour means inconclusive.

PROP4	Pattern	TopApps	Last7Days	SelectPeriod	Main
TopApps	AP1	–	**3.62**	**11.09**	**2.22**
	AP2	–	10.56	14.89	9.347
Last7Days	AP1	**61.83**	–	**68.68**	**59.56**
	AP2	14.01	–	15.61	10.08
SelectPeriod	AP1	**38.30**	**36.69**	–	**36.03**
	AP2	31.21	28.77	–	27.28
UseStop	AP1	**1.49**	**3.61**	9.23	**3.33**
	AP2	11.74	6.24	11.51	7.82

PROP5	Pattern	TopApps	Last7Days	SelectPeriod
TopApps	AP1	–	**0.66**	0.26
	AP2	–	0.26	0.30
Last7Days	AP1	**0.006**	–	**0.02**
	AP2	0.49	–	0.38
SelectPeriod	AP1	**0.008**	**0.08**	–
	AP2	0.23	0.15	–

for the time cut $[30, 60)$. We colour-code the results to correspond to the Overall Viewing pattern in **blue-coloured, bold font** and to the In-depth Viewing in red-coloured font. Except for two inconclusive pairs of results (default text colour), Table 3 tells us that the two activity patterns learned from the time cut $[30, 60)$ are respectively similar to the two activity patterns identified for the rest of time cuts analysed previously. The difference in the behaviour around Stats could be explained by a new usage behaviour of the AppTracker around the 30^{th} day of usage due to approximately a full month worth of new statistics, leading to a spurt of more exploratory usage of AppTracker. We note that for the time cut $[60, 90)$ the results listed in Table 2 make again a clear distinction between the two activity patterns with respect to the states SelectPeriod and Last7Days. We might say that in the third month the exploratory usage of AppTracker settles down and users know exactly what to look for and where. A finer-grained longitudinal analysis based on one-week time cuts could reveal additional insight into the behaviour involving Stats around the 30^{th} day of usage.

Our conclusion concerning the two types of activity patterns meets the developers' hypothesis about two distinct usages of the apps. However they expected also to see one pattern revolving around TopApps and Stats, one around SelectPeriod and another one around Last7Days. Since we analysed the admixture model for $K = 2$, we only got two distinct patterns, the last two patterns conjectured by developers being aggregated into a single one. As a consequence, we investigate higher values for K.

Analysing rPCTL properties for $K = 3$. We analyse PROP1, PROP2, and PROP3 on the admixture model inferred for $K = 3$, time cut $[0, 30)$ and $N = 50$. For brevity, we omit the details, and based on the results we characterise the three patterns as follows:

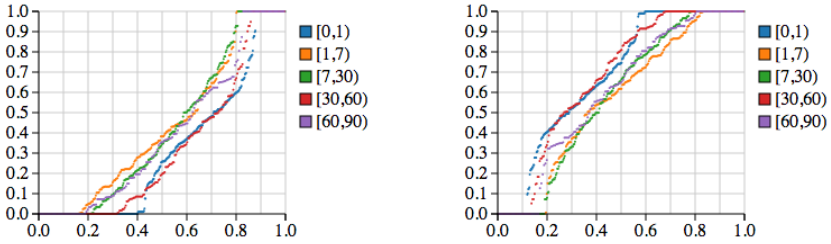
- AP1 is an Overall Viewing pattern because TopApps and Stats have best results for all three properties. SelectPeriod and Last7Days are absent. The sessions are twice as short and twice more frequent than for the In-depth Viewing pattern.
- AP2 is a 'weaker' Overall Viewing pattern than AP1 because TopApps has worse results, and better results than Stats and Last7Days; SelectPeriod is absent.
- AP3 is an In-depth Viewing pattern because SelectPeriod has the best results, followed closely by TopApps and Last7Days.

Analysing rPCTL properties for $K = 4$. We analyse PROP1, PROP2, and PROP3 on the admixture model inferred for $K = 4$, time cut $[0, 30)$ and for $N = 50$. Again, details are omitted and we characterise the patterns as follows:

- AP1 is mainly a TopApps Viewing activity pattern because it has the best results for TopApps, compared to Stats, SelectPeriod, and Last7Days which score very low results.
- AP2 is a Stats – TopApps Viewing activity pattern, with very low results from Last7Days; SelectPeriod is absent.
- AP3 is an In-depth Viewing pattern with dominant Last7Days followed closely by TopApps and SelectPeriod; Stats is absent.
- AP4 is mainly a TopApps Viewing pattern because TopApps has the best results, while all other states need on average an infinite number of time steps to be reached. The fact that it takes on average an infinite number of time steps to reach the end of a session (i.e., the state UseStop) motivated us to analyse this pattern with other temporal properties and for other states. As a consequence we saw that UsageBarChartTopApps has similar properties as TopApps, meaning that this pattern corresponds to repeatedly switching between TopApps and UsageBarChartTopApps.

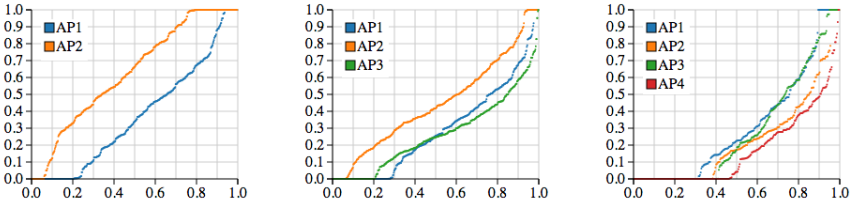
Based on the results obtained for UseStop we observe: twice as many sessions for AP1 than for AP2 and AP3, only a couple of sessions on average for AP4, fewer views per session (i.e., shorter sessions) for AP1 than for AP2 and AP3.

Longitudinal Θ-based comparison. In addition to analysing rPCTL properties, we also compare how the distribution Θ of the two activity patterns for the entire population of users changes in time. For each time cut considered for the rPCTL analysis above and activity pattern AP2, we order non-decreasingly the second column of Θ and re-scale its size to the interval $[0, 1]$ to represent the horizontal axis, while the ordered Θ values are projected on the vertical axis. Figure 4 shows the Θ values for AP1 and AP2 for the population of users across the first three months of usage. We conclude that during the first day of usage, up to 40% of users exhibit exclusive In-depth Viewing behaviour (probability close to 1 on the y-axis) corresponding to an initial exploration of the app with significant number of visits to TopApps, Stats, SelectPeriod, and Last7Days. Also, at most 10% of the users exhibit exclusive Overall Viewing behaviour maybe because they feel less adventurous in exploring the app, preferring mostly the first menu option of looking at TopApps and subsequently at Stats. We note that the



(a) Overall Viewing AP (b) In-depth Viewing AP

Fig. 4. Longitudinal comparison of the activity pattern distributions Θ over the population of users for $K = 2$ and time cuts $[0, 1)$, $[1, 7)$, $[7, 30)$, $[30, 60)$, $[60, 90)$



(a) $K = 2$ (b) $K = 3$ (c) $K = 4$

Fig. 5. Pattern distributions for $K = 2$ (Overall Viewing, In-depth Viewing), $K = 3$ (Overall Viewing, weak Overall Viewing, In-depth Viewing,) and $K = 4$ (mainly TopApps Viewing, equally Stats and TopApps Viewing, In-depth Viewing with no Stats, exclusive TopApps and UsageBarChartTopApps), time cut $[0, 30)$.

distributions of the two activity patterns in the population of users are similar for the time cuts $[0, 1)$ and $[30, 60)$ – probably because more users exhibit a more exploratory behaviour during these times (new types of usage statistics become available after one month of usage). At the same time, the plots for the time cuts $[1, 7)$, $[7, 30)$, and $[60, 90)$ are also similar, and we think that they correspond to a settled (or routine) usage behaviour.

Structural Θ-based comparison. In Fig. 5 we plot the weightings of all users for each activity patterns for $K \in \{2, 3, 4\}$ and the time cut $[0, 30)$. Figure 5(a) tells us that for $K = 2$ the In-depth Viewing has higher weightings across the user population with almost 25% of the users using the app exclusively like this, hence either exploring the app or genuinely interested in in-depth usage statistics. Figure 5(b) tells us that almost 10% of the users are exclusively interested in TopApps, Stats and Last7Days but not SelectPeriod; this behaviour is the most popular among users. From Fig. 5(c) we see that almost 50% of the users do not behave according to AP4 – switching repeatedly between TopApps and UsageBarChartTopApps. Note that for $K = 3$ and $K = 4$ no pattern stands out as very different from the others.

6 Formal Analysis Informs Redesign

We now consider how our results provide insights for redesign, in the context of the case study. Our initial analysis uncovered two activity patterns, characterised by the type of usage stats the user is examining: Overall Viewing – more high-level usage statistics for the entire recorded period, or In-depth Viewing – more in-depth usage statistics for specific periods of interest. Neither is significantly dominant over the other: for the majority of users, usage is fairly evenly distributed between the two patterns. This suggests that a revised version of AppTracker should continue to support both patterns.

We note the two patterns identified for $K = 2$ correspond closely to options presented on AppTracker's main menu (see Fig. 1(a)), as follows. Overall Viewing indicates a greater likelihood of using TopApps and Stats, which are interface screens accessed through the *Overall Usage* menu item. In-depth Viewing indicates a greater likelihood of reaching SelectPeriod and Last7Days, which are accessed through *Select by Period* and *Last 7 Days*, but also some usage of TopApps and Stats. Our results indicate that sessions corresponding to Overall Viewing are generally shorter: meaning that users are performing fewer actions between launching AppTracker and exiting back to the device's home screen. These two different patterns suggest that, in a future version of AppTracker, if developers want to keep the two major styles of usage separated between different screens, they could design explicitly for the *glancing*-like short interactions in *Overall Usage* and longer interactions in a new *Select by Period* screen along with the initial *Last 7 Days* screen. Also more filtering and querying tools could be added to *Select by Period*.

We wondered if users are simply following the paths suggested by the main menu (Fig. 1(a)). We therefore probed further, considering $K \in \{3, 4, 5\}$ (details are omitted for $K = 5$). For $K = 3$, if the analysis was merely mirroring the menu structure, we might expect to see one pattern centred around each of the first three main menu items. Although we see the pattern AP2 centred around TopApps, Stats, and Last7Days but no SelectPeriod, we do not see a pattern centred around SelectPeriod and not including Last7Days. For $K = 4$ we find Last7Days and SelectPeriod together in a pattern, with the former view slightly more popular than latter one; this combination also occurred for $K = 2$ and $K = 5$. For $K = 4$ we see a distinct new pattern showing users repeatedly switching between TopApps and UsageBarChartTopApps. TopApps is an ordered list of the user's most used apps; selecting an item from this list opens UsageBarChart-TopApps, a bar chart showing daily minutes of use of this app. This persistent switching suggests a more investigatory behaviour, which is more likely to be associated with the In-depth Viewing. Yet this behaviour is occurring under the *Overall Usage* menu item, which we hypothesised and then identified as being associated with more glancing-like behaviour. This suggests that our results are providing more nuanced findings than simple uncovering of existing menu structure. We therefore suggest that if developers want to separate the two types of usage between different menu items even more, they could move the TopApps – UsageBarChartTopApps loop from *Overall Usage* to *Select by Period*.

Glancing activity patterns. Discovering a *glancing* activity pattern provides significant benefits for app redesign. Since the release of the iOS 8 SDK in 2014, Apple has allowed the development of *'Today widgets'* – extensions to apps comprising small visual displays and limited functionality appearing in the Notification Centre. Beyond the advice from Apple's Human Interface Guidelines[1], developers struggle to decide which pieces of their app's contents would best suit inclusion in a Today widget: few conventions have built since the release. Developers have to rely on their own judgement to select appropriate content from their app to populate this view. In our analysis, we have uncovered explicitly the specific screens that people look at when they are undertaking short sessions of glancing-type behaviour, i.e. the typical glancing patterns for AppTracker – the Overall Viewing pattern and the TopApps-centred patterns. In identifying such activity patterns, our approach provides a more principled method of selecting content appropriate for an app extension such as a Today widget.

7 Discussion

Related Work. Logging software is frequently used to understand program behaviours, and typically to aid program comprehension – building an understanding of how the program executes [9]. There are various techniques that use logs of running software, such as visualising logs (e.g. [10]) and capture and replay (e.g. [11]), with the aims of failure analysis, evaluating performance, and to better understand the system behaviour (as it executes). In contrast, we are interested in ways users interact with software, and we do so by analysing logs captured during actual use. The difference is important. In the case of program comprehension, log analysis is used to understand better what is going on within the code and how the artefact is engineered (in order to be better prepared for improvements and maintenance). In our case, log analysis provides insights about distinct styles of use and informs improvements of the high-level design. For example, in [12] the authors infer FSMs for modelling a system's behaviour, while we infer DTMC of different actual usage behaviours, and admixtures thereof, to model populations of users. There is complementarity with the approach of [13], which employs usage logs and applies temporal logical analysis, but a key difference is their models are based on static user attributes (e.g. city location of user) rather than on inferred behaviours. Their approach assumes within-class use to be homogeneous, whereas our research demonstrates within-class variation.

Methodological issues. Our approach is a collaboration between developers familiar with app development and instrumentation, and analysts familiar with statistics and formal modelling. We note some methodological issues.

First, our approach gains from having significant volumes of log data to work on, for reliable application of statistical methods. However, neither the volume of log data nor number of users is relevant for the probabilistic model checking,

[1] `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/AppExtensions.html`

only the number of higher level states for analysis selected from the raw data determines the complexity. Therefore *our approach scales* because it does not depend on the number of activity patterns or the number of users/data size, but on the state space of the abstract model of the app. The AppTracker case study involved a vocabulary of 15 views/states, which was comfortably manageable, but our approach would have difficulty with very large vocabularies.

Second, the issue of *what to log* is not trivial. The collection of log entries can include anything from a button press, to the change of WiFi signal of the device, and so on. In the case of AppTracker, we decided to use states corresponding to individual screens possible to transition to within the app. This highlights the rather simple nature of AppTracker – it is essentially a browser of information. In contrast, a game such as Angry Birds allows the user to perform a much more complex set of actions. Even after pruning the logs to include only user actions (rather than lower level device events), one still needs to decide how to model these actions as a state space. The chosen state space will ultimately influence what activity patterns become prominent. We therefore suggest that discussion and preliminary analysis be done early in the development process, so that the decisions about what to log and what the state space should be are made by developers and analysts jointly in a well-informed way.

Third, the activity patterns and their number (i.e. K) are key to analysis. The patterns are inferred by various standard statistical methods based on non-linear optimisation. We do not model for predictability, there is no *true model* of the generating process, but one that is posited based on known characteristics such as the sequential nature of issuing app events. We study the time-series behaviour that has been logged from a probabilistic perspective. The admixture model is important because we are defining a complex behavioural trait where the individual moves between patterns during an observed user trace. The number of patterns K is an important exploratory tool, there is no optimal value for K.

8 Conclusions and Future Work

We have outlined an approach to exploring and gaining insight into usage patterns that informs redesign based on probabilistic formal analysis of actual app usage. Our approach is a combination of bottom up statistical inference from user traces, and top down probabilistic temporal logic analysis of inferred models. We have illustrated this via the mobile app AppTracker, and discussed how the results of this analysis inform redesign that is grounded in existing patterns of usage. A notable conclusion of our work is that, while our analysis of AppTracker's use identifies several clearly distinct activity patterns, it also reveals the distribution of activity patterns over the population of users and over time. For AppTracker, this mitigates against a simple partitioning of the app into two different versions, each specific to one activity pattern. In addition, our analysis offers a more principled way of selecting glanceable information.

AppTracker developers are currently implementing a redesign based on our analysis, and we eagerly await new data sets of logged behaviours for further

analysis. Future work will involve that analysis, as well as patterns for logic properties and generalisation of our approach to a principled way of providing software redesign guidelines as part of a user-centered design process.

Acknowledgements. This research is supported by the EPSRC Programme Grant *A Population Approach to Ubicomp System Design* (EP/J007617/1).

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
2. Bell, M., Chalmers, M., Fontaine, L., Higgs, M., Morrison, A., Rooksby, J., Rost, M., Sherwood, S.: Experiences in Logging Everyday App Use. In: Proc. of Digital Economy'13, ACM (2013)
3. Girolami, M., Kabán, A.: Simplicial Mixtures of Markov Chains: Distributed Modelling of Dynamic User Profiles. In Thrun, S., Saul, L.K., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16 (NIPS'03), MIT Press (2004) 9–16
4. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological) **39**(1) (1977) 1–38
5. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic Model Checking. In Bernardo, M., Hillston, J., eds.: SFM. Volume 4486 of LNCS., Springer (2007) 220–270
6. Andrei, O., Calder, M., Higgs, M., Girolami, M.: Probabilistic Model Checking of DTMC Models of User Activity Patterns. In: Proc. of QEST'14. Volume 8657 of LNCS., Springer (2014) 138–153
7. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In Gopalakrishnan, G., Qadeer, S., eds.: Proc. of CAV'11. Volume 6806 of LNCS., Springer (2011) 585–591
8. Hall, M., Bell, M., Morrison, A., Reeves, S., Sherwood, S., Chalmers, M.: Adapting ubicomp software and its evaluation. In Graham, T.C.N., Calvary, G., Gray, P.D., eds.: Proc. of EICS'09, ACM (2009) 143–148
9. von Mayrhauser, A., Vans, A.M.: Program comprehension during software maintenance and evolution. IEEE Computer **28**(8) (1995) 44–55
10. Fittkau, F., Waller, J., Wulf, C., Hasselbring, W.: Live trace visualization for comprehending large software landscapes: The ExplorViz approach. In Telea, A., Kerren, A., Marcus, A., eds.: Proc. of VISSOFT'13. (2013) 1–4
11. Gomez, L., Neamtiu, I., Azim, T., Millstein, T.D.: RERAN: Timing- and Touch-sensitive Record and Replay for Android. In Notkin, D., Cheng, B.H.C., Pohl, K., eds.: Proc. of ICSE'13, IEEE / ACM (2013) 72–81
12. Beschastnikh, I., Brun, Y., Ernst, M.D., Krishnamurthy, A.: Inferring models of concurrent systems from logs of their behavior with CSight. In Jalote, P., Briand, L.C., van der Hoek, A., eds.: Proc. of ICSE '14, Hyderabad, India, ACM (2014) 468–479
13. Ghezzi, C., Pezzè, M., Sama, M., Tamburrelli, G.: Mining Behavior Models from User-Intensive Web Applications. In: Proc. of ICSE'14, Hyderabad, India, ACM (2014) 277–287