Interpreting Computational Models of Interactive Software Usage

Oana Andrei
Oana.Andrei@glasgow.ac.uk
University of Glasgow, UK

Muffy Calder
Muffy.Calder@glasgow.ac.uk
University of Glasgow, UK

Matthew Chalmers
Matthew.Chalmers@glasgow.ac.uk
University of Glasgow, UK

Alistair Morrison
Alistair.Morrison@glasgow.ac.uk
University of Glasgow, UK

ABSTRACT

Evaluation of how users actually interact with interactive software is challenging because users' behaviours can be highly heterogeneous and even unexpected. Probabilistic, computational models inferred from low-level logged events offer a higher-level representation from which we can gain insight. Automatic inference of such models is key when dealing with large sets of log data, however interpreting these models requires significant human effort. We propose new temporal analytics to model and analyse logged interactions, based on learning admixture Markov models and interpreting them using probabilistic temporal logic properties and model checking. Our purpose is to discover, interpret, and communicate meaningful patterns of usage in the context of redesign. We illustrate by application to logged data from a deployed personal productivity iOS application.

Computational Modeling in Human-Computer Interaction, May 5, 2019, Glasgow, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Mathematics of computing** → **Markov networks**; *Maximum likelihood estimation*; • **Human-centered computing** → **User models**; *Ubiquitous and mobile computing design and evaluation methods*; • **Theory of computation** → **Modal and temporal logics**; **Verification by model checking**.

KEYWORDS

log analysis, usage behaviour, admixture, probabilistic temporal logic, model checking, mobile apps

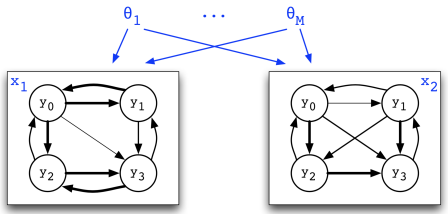
ACM Reference Format:
Oana Andrei, Muffy Calder, Matthew Chalmers, and Alistair Morrison. 2019. Interpreting Computational Models of Interactive Software Usage. In *Proceedings of Computational Modeling in Human-Computer Interaction.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

INTRODUCTION

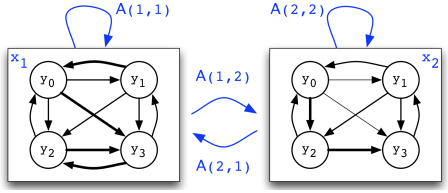
Interactive software users are heterogeneous in that they may adopt different usage styles for the same software. Furthermore, each individual user may move between different usage styles, from one interaction session to another, or even during a session, possibly due to a variety of contextual factors. To analyse these differing and dynamic usage styles, we defined a new temporal analytics approach [1–3] to model and analyse temporal data sets of logged interactions, with the purpose of discovering, evaluating, and communicating meaningful patterns of usage—and eventually informing the redesign of the interactive software.

Our temporal analytics approach is based on the following stages: (1) instrument the interactive software to log events, (2) create user traces from logs, (3) segment user traces based on selected time intervals of usage, (4) infer parameters for probabilistic models of behaviour for each set of user traces – admixture models with K activity patterns, (5) choose instances of temporal logic properties from predefined classes of properties, and define new properties specific to the software under evaluation, (6) run probabilistic model checking for combinations of probabilistic models and temporal logic properties, (7) interpret the results of probabilistic model checking, label activity patterns, and compare across models w.r.t. time intervals of usage and number of number of activity patterns (K), (8) recommend redesign ideas on the basis of these results and interpretations. This list only sketches a *cleaned* version of the analysis processes underlying our approach. In practice, the many different resources listed above are used together, and the members of the analytics team go back and forth between them often, combining and comparing the intermediate insights and results.

Here, we give an overview of our past and on-going research and illustrate it on an iOS app with over 40,000 downloads as of September 2017.



A population admixture model (PAM)



A generalised population admixture model (GPAM)

Figure 1: Pictorial representation of PAMs and GPAMs for $K = 2$ and two hidden states x_1 and x_2. The two activity patterns are the discrete-time Markov chains in each box, with y_0, y_1, y_2, y_3 the observed states. Transition probabilities are indicated by the thickness of transitions. θ_m is the probabilistic distribution over the K activity patterns for the m-th user trace in PAM, where $1 \leq m \leq M$ and M is the number of user traces. A is the transition matrix for hidden states in GPAM.

FROM LOGS TO USER TRACES

Raw log data is processed to obtain discrete time series data of timestamped user-initiated events for each unique device 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. The state labels `UseStart` and `UseStop` denote the beginning and end of each user interaction session respectively. 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. We segment a set of traces according to time intervals of usage.

POPULATION ADMIXTURE MODELS

Our experience of analysing interactive software indicates that usage models do not depend only on relatively static attributes such as location, gender or age of users, but also on dynamic attributes such as styles of use, that may change in time, shared in a population of users. Motivated by [4, 8], we assume the existence of a common set of activity patterns that can be estimated from all observed user traces in population of users. 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, and each individual user may move between different activity patterns.

We model activity patterns as first-order discrete-time Markov chains (DTMCs). First-order Markov models have been used for modelling in many related problems, such as human navigation on the Web [5, 7, 13], usability analysis [14], mobile applications [9], human interactions with search engines [16]. We defined two new computational models of usage behaviour [2] as admixtures of K activity patterns as follows:

- *Population admixture models* (PAMs) are admixtures of activity patterns with a distribution θ over hidden states (i.e., over activity patterns) for each user trace.
- *Generalised admixture models* (GPAMs) are first-order auto-regressive hidden Markov models (ARHMMs) [12] which express explicit relationships (probabilistic transitions) between observed states (within an activity pattern) and between hidden states (i.e., activity patterns).

Each of the models PAM and GPAM (illustrated in Fig. 1) offers a different perspective on usage behaviours, and consequently affords different analysis. Each admixture component has the same set of observed states, but the transition probabilities are different, as indicated by the thickness of transitions. The number of mixture components/activity patterns, $K \geq 2$, 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. We infer the parameters of the admixture models from the segmented user traces using statistical methods for maximum likelihoods, the Expectation-Maximisation algorithm [6] for PAMs and the Baum-Welch algorithm [17] for GPAMs.

Table 1: Syntax and semantics of PCTL formulae for M a DTMC, $init(M)$ the initial state of M, s a state, a an atomic proposition, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, $N \in \mathbb{N} \cup \{\infty\}$. If N is ∞ then it can be omitted.

State formulae

$\Phi ::= true \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{P}_{\bowtie p}[\Psi] \mid \mathsf{S}_{\bowtie p}[\Phi]$

Path formulae

$\Psi ::= \mathsf{X}\Phi \mid \Phi\,\mathsf{U}^{\leq N}\,\Phi$

Satisfaction relation

$M \models \Phi$ if $init(M) \models \Phi$

$s \models true$ is always true

$s \models false$ is always false

$s \models a$ iff a labels s

$s \models \neg\Phi$ iff $s \models \Phi$ is false

$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$

$s \models \Phi_1 \vee \Phi_2$ iff $s \models \Phi_1$ or $s \models \Phi_2$

$s \models \mathsf{P}_{\bowtie p}[\Psi]$ iff the probability that Ψ is satisfied by the paths starting from state s meets the bound $\bowtie p$

$s \models \mathsf{S}_{\bowtie p}[\Phi]$ iff the steady-state (long-run) probability of being in a state that satisfies Ψ meets the bound $\bowtie p$

$\mathsf{F}^{\leq n}\,\Phi \equiv true\,\mathsf{U}^{\leq n}\,\Phi$

INTERPRETING ADMIXTURE MODELS

Probabilistic Temporal Logic and Model Checking

We use probabilistic model checking with PRISM [10] for reasoning over DTMCs. Probabilistic computation tree logic (PCTL) allows expression of a probability measure of the satisfaction of a temporal property by a state of a DTMC, see Table 1. PRISM allows one to assign rewards/costs to states and/or transitions, and to compute expected values of cumulative rewards within a number of time-steps or until a state formula holds.

Probabilistic Temporal Logic Properties for Interpreting Admixture Models

We defined the following classes of temporal properties for analysing either individual activity patterns in PAM and GPAM as well as for analysing the DTMCs resulting from flattening GPAMs [2]:

- **VisitProb:** $\texttt{filter}(\texttt{state}, \mathsf{P}_{=?}[\phi_1 \mathsf{U}^{\leq N} \phi_2], \phi_0)$ computes the probability that ϕ_1 holds until reaching a state in which ϕ_2 holds, within N time-steps, when starting from a state satisfying ϕ_0.
- **VisitCount:** $\texttt{filter}(\texttt{state}, \mathsf{R}_{stateRwd=?}[\mathsf{C}^{\leq N}], \phi_0)$ computes the expected reward accumulated over N time-steps when starting from a state satisfying ϕ_0.
- **StepCount:** $\texttt{filter}(\texttt{state}, \mathsf{R}_{stepRwd=?}[\mathsf{F}\,\phi_1], \phi_0)$ computes the expected number of steps accumulated before reaching a state satisfying ϕ_1 when starting from a state satisfying ϕ_0.

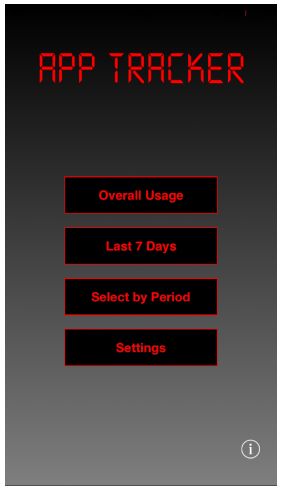
Some instances of these properties are: the probability to reach a particular screen view, the probability of reaching one state from another, the average number of visits to a state from another state, the average number of steps to each state, the average number of sessions – all within a fixed number of time-steps (e.g., button taps), the average session length. GPAMs allow us to reason over behaviours involving several activity patterns, for example the likelihood of changing activity pattern after visiting a particular state. We can also compute the long run probability to be in a particular activity pattern.

Evaluation of Model Checking Results

We adopt the general inductive approach for analysing qualitative evaluation data [15]: aggregating, categorising, and ordering the results; creating English language labels for categories based on multiple cross-readings of the tables alongside their textual (category) descriptions; and continually revising and refining the category sets. We require that the programmer(s), designer(s), analyst(s), and evaluator(s) collectively agree on the labels.

CASE STUDY: THE APPTRACKER MOBILE APP

AppTracker [11] is an app for jailbroken iOS devices consisting of a background logging framework that records information on device use, and a foreground UI that displays a rich set of charts and statistics on the user's app usage (see Fig. 2 for screenshots). Users can view lists of the most-used apps,



(a) Main view

(b) Stats view

Figure 2: AppTracker screenshots

filter by time period, or view detailed information on usage of an individual app. For the purposes of this paper, we consider data gathered from between August 2013 and May 2014 from 489 users. For analysis, we selected 16 user-initiated events that switch between views. These views determine the atomic propositions labelling the states in our probabilistic models.

We analysed PAMs and GPAMs for $K = 2, 3, 4, 5$, and the time intervals first day/week/month and second and third month of usage. For $K = 2$, the two types of activity patterns are InDepth – more in-depth usage statistics for specific periods of interest, and Browsing – more high-level usage statistics for the entire recorded period. For $K \geq 3$, we identified two additional types of activity patterns: one is Glancing for short-burst and high frequency behaviour centred usually around one or two particular states, and the other Loop for quick switches between two states (which is rather characteristic of an in-depth visualisation behaviour).

We had not anticipated the effect of the new content available in Stats (Fig. 2(b)) only after the first month of usage - 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).

The analysis showed the activity patterns we uncovered did not follow the top level menu structure. This prompted discussions within the design team, about the role and impact of hierarchical menus on user interactions, whether the top-level menu structure supports the activity patterns or not.

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.

Based on these results we recommended three changes for the redesign of AppTracker to achieve a clearer separation between the main menu options, one favorable to a glancing-like usage and the other to an in-depth usage, hence changing from three main option in version 1 to two main options in version 2: (i) move the states conducive to the Loop pattern uncovered in the Browsing part of the app to the InDepth part; (ii) move the Stats view to the InDepth part of the menu; (iii) add more content in the Glancing part of the app regarding today's device usage.

CONCLUSION AND FUTURE WORK

By modelling and analysing the different processes of software usage over a dynamic, heterogeneous population of users, we have developed new and effective tools for understanding how software is actually used. Our analytics, based on inferred admixture models and temporal probabilistic logic properties, offer insights into such populations via temporal and sequential patterns - i.e., forms of abstraction, structure and insight not afforded by traditional statistical, observational and visual methods. We are currently working on analysing and evaluating the newer version of the AppTracker app, redesigned on the basis of recommendations arising from our initial analysis. Future work will

involve application to other types of interactive systems, and generalisation of our approach to a principled way of providing software redesign recommendations as part of a user-centered design process.

ACKNOWLEDGMENTS

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] Oana Andrei and Muffy Calder. 2017. Temporal Analytics for Software Usage Models. In *Software Engineering and Formal Methods - SEFM 2017 Collocated Workshops (LNCS)*, A. Cerone and M. Roveri (Eds.), Vol. 10729. Springer, 9–24.

[2] Oana Andrei and Muffy Calder. 2018. Data-driven modelling and probabilistic analysis of interactive software usage. *Journal of Algebraic and Logical Methods in Programming (JLAMP)* 100 (2018), 195–214.

[3] Oana Andrei, Muffy Calder, Matthew Chalmers, Alistair Morrison, and Mattias Rost. 2016. Probabilistic Formal Analysis of App Usage to Inform Redesign. In *Proc. of iFM'16 (LNCS)*, Vol. 9681. Springer, 115–129.

[4] James F. Bowring, James M. Rehg, and Mary Jean Harrold. 2004. Active learning for automatic classification of software behavior. In *Proc. of ISSTA'04*, G. S. Avrunin and G. Rothermel (Eds.). ACM, 195–205.

[5] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamás Sarlós. 2012. Are web users really Markovian?. In *Proc. of WWW'12*, A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab (Eds.). ACM, 609–618.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1 (1977), 1–38.

[7] Carlo Ghezzi, Mauro Pezzè, Michele Sama, and Giordano Tamburrelli. 2014. Mining Behavior Models from User-Intensive Web Applications. In *Proc. of ICSE'14*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 277–287.

[8] Mark Girolami and Ata Kabán. 2004. Simplicial Mixtures of Markov Chains: Distributed Modelling of Dynamic User Profiles. In *NIPS'03*, S. Thrun, L. K. Saul, and B. Schölkopf (Eds.). MIT Press, 9–16.

[9] Vassilis Kostakos, Denzil Ferreira, Jorge Gonçalves, and Simo Hosio. 2016. Modelling smartphone usage: a Markov state transition model. In *Proc. UbiComp'16*, P. Lukowicz, A. Krüger, A. Bulling, Y.-K. Lim, and S. N. Patel (Eds.). ACM, 486–497.

[10] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Proc. of CAV'11 (LNCS)*, G. Gopalakrishnan and S. Qadeer (Eds.), Vol. 6806. Springer, 585–591.

[11] Alistair Morrison, Xiaoyu Xiong, Matthew Higgs, Marek Bell, and Matthew Chalmers. 2018. A Large-Scale Study of iPhone App Launch Behaviour. In *Proc. of CHI'18*. ACM, 344:1–344:13.

[12] Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press.

[13] Philipp Singer, Denis Helic, Andreas Hotho, and Markus Strohmaier. 2015. HypTrails: A Bayesian Approach for Comparing Hypotheses About Human Trails on the Web. In *Proc. of WWW'15*. ACM, 1003–1013.

[14] Harold W. Thimbleby, Paul A. Cairns, and Matt Jones. 2001. Usability analysis with Markov models. *ACM Trans. Comput.-Hum. Interact.* 8, 2 (2001), 99–132.

[15] David R. Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation* 27, 2 (2006), 237–246.

[16] Vu Tran, David Maxwell, Norbert Fuhr, and Leif Azzopardi. 2017. Personalised Search Time Prediction using Markov Chains. In *Proc. of ICTIR'17*, J. Kamps, E. Kanoulas, M. de Rijke, H. Fang, and E. Yilmaz (Eds.). ACM, 237–240.

[17] Lloyd Welch. 2003. Hidden Markov Models and the Baum-Welch Algorithm. *IEEE Inf. Th. Soc. Newsletter* (2003).