Modelling Smartphone Usage: A Markov State Transition Model

Vassilis Kostakos, Denzil Ferreira, Jorge Goncalves, Simo Hosio
Center for Ubiquitous Computing, University of Oulu
Oulu, Finland
{vassilis; denzil.ferreira; jorge.goncalves; simo.hosio}@oulu.fi

ABSTRACT
We develop a Markov state transition model of smartphone screen use. We collected use traces from real-world users during a 3-month naturalistic deployment via an app-store. These traces were used to develop an analytical model which can be used to probabilistically model or predict, at runtime, how a user interacts with their mobile phone, and for how long. Unlike classification-driven machine learning approaches, our analytical model can be interrogated under unlimited conditions, making it suitable for a wide range of applications including more realistic automated testing and improving operating system management of resources.

Author Keywords
Markov chains; smartphone; model; prediction.

ACM Classification Keywords
E.1 [Data]: Data structures—Graphs and networks; **H.5.2 [Information Interfaces and Presentation]:** User Interfaces—Theory and methods

INTRODUCTION
We present a probabilistic model that captures statistical properties of smartphone usage, and can make runtime probabilistic estimations. Specifically, we have created a Markov model variant that describes the probability of observing different events associated with interacting with a smartphone's screen. Our model captures four important screen events that can be observed on smartphones: screen is turned on, turned off**,** locked or unlocked.

Our model estimates the probability of observing one of these events based on the event that was most recently observed and how much time has elapsed. In addition, our model can take into account context: the varying battery levels, time of day, day of week, and 3 distinct user profiles in calculating its estimations. To develop our model, we re-analysed data from a 3-month study whereby 218 users were observed using their smartphones on a daily basis in realistic settings [19].

Markov model variants, such as the one we present here, are typically used to describe systems that have a set of observable states or features that are mutually exclusive, and entail a notion of probability in how transitions between states take place. Such models consist of the "states" themselves (*i.e.*, the smartphone screen events that Android broadcasts), and a *transition matrix* that defines the probabilities of moving between these states. Broadly speaking, the analysis of such models can offer insights into emergent system behaviour, and can help develop strategies in managing the systems.

Due to the simplicity and power of this modelling approach, it has been very effective at modelling system behaviour across a variety of domains, and a very rich literature exists on using State Transition models. Such idealized models can capture many of the statistical regularities of complex systems. For example, the health of a population can be effectively modelled using Markov-based techniques, which can then be used in assessing health technology, developing clinical decision strategies, and conducting health-economic evaluation [40]. Most often, state-transition models are used in the evaluation of risk factor interventions, diagnostic procedures, and disease management strategies.

Markov models and their underlying matrix algebra have also been proposed as a means of evaluating usability at design-time [44,45]. These approaches typically model UI elements and forms as states, and using Markov modelling can measure usability by estimating the number of transitions necessary to complete tasks. However, while useful in developing evaluation strategies, not all models incorporate a strong notion of *elapsed time*. This is a key contribution we make, which enables us to model how the state of a smartphone's screen evolves at runtime.

Automated tools for smartphone testing already attempt to model interaction at runtime [9,28], but largely rely on deterministic test scripts rather than empirical data [19]. On the other hand, our model:

- Can be used to develop realistic engines for automated smartphone testing. Our model can generate "synthetic" use traces that are highly representative of real-world use,

UbiComp '16, September 12 - 16, 2016, Heidelberg, Germany
 ACM 978-1-4503-4461-6/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971669

and can take into account contextual variables such as available battery, time of day, and user typology.

- Can help optimize the system resources of a smartphone at runtime. The probabilistic nature of our model allows the system to make runtime decisions on screen-related resource optimisation.
- Provides a theoretical basis for extending the model to describe which applications individuals use and for how long.

Our model focuses on interaction with the screen, and not the specificities of applications, their content, the UI layout, and settings. While this is a drastic simplification of smartphones, it *does* capture important statistical properties about how the phone is used. It is within this broader use that specific applications are launched. Just as in the case of Markov models used to describe the statistical properties of large complex systems (health, ecosystems, economy), our model focuses on a particular aspect of smartphones: when, how, and for how long they are in use.

RELATED WORK

An increasing body of work has begun to look at traces of smartphone use in an attempt to construct models of human behaviour and performance. For example, previous work has looked at how much time people spend using applications on smartphones. This has revealed that users exhibit "micro-usage" behaviour, i.e. they tend to use individual applications in short bursts of roughly 15 seconds [18], suggesting a strong temporal nature in smartphone use. Additionally, research has attempted to model from use traces how people touch the screen on smartphones [7,25], use the on-screen keyboard [10], and even develop keystroke-level modelling of application use [26]. An important application of this kind of work is to support rapid prototyping and testing of mobile applications. For instance, Holleis & Schmidt [27] developed a keystroke level model to help in estimating task completion times when prototyping smartphone apps.

Such understanding and modelling of smartphone usage has allowed researchers to create intelligent launchers that reorder the applications not based on frequency of use, but by users' context, *e.g.*, location, calendar, time of day [39,41,49], and infer call-availability [37]. However, these models are fine tuned to a broad set of variables which are often challenging to obtain simultaneously for practical use and are therefore not trivial to evaluate or test.

The increasing availability of performance data is helping us understand better how smartphones are used, but so far a practical runtime model of smartphone use has not been developed. Such a model would allow us to improve resource allocation in smartphones at runtime [35], and also develop more realistic automated testing. While many automated testing tools do exist, they suffer from lack of interaction realism as we describe next.

Automated testing tools

As mobile phones begun to increase in popularity, testing of applications for phones was largely conducted in closed settings usually by manufacturers themselves [1]. Similarly, as application stores have become more popular recently, and so have their associated software development kits, new opportunities and challenges for mobile application developers emerged. For instance, mobile devices exhibit a rapid evolution and use of multiple standards, protocols and network technologies, while applications increasingly are deployed to a variety of platforms and hardware. Furthermore, documentation is often incomplete, and market trends change frequently.

To overcome these challenges, it is necessary to create models, tools and techniques to evaluate mobile applications in a more realistic and contextual manner. Previous work has proposed methods to collect the context of application usage directly from the users [3,30], which can then drive application development and testing. An alternative is the use of logging software on mobile devices to capture users' application and device usage [20]. Although useful, these techniques require time, resources and recruiting potential application users. This motivates the need for tools that support automated usability evaluations and testing [27].

The automated testing of user interfaces is not a novel concept. There exist a variety of suites for testing desktop usability [31], desktop content adaptation [15], mobile content adaptation [32], and web interfaces [4]. Beyond the desktop, for mobile form factors it has been suggested that automated tests are a cheaper and quicker alternative to field tests [29]. Some previous work has focused on facilitating UI testing on mobile platforms. For instance, Satoh's framework [38] emulated a mobile device at the application-level by implementing a mobile agent that simulates various network conditions. Kostiainen *et al.* [33] provide a framework for comparative usability testing, useful to test the optimal sequence to perform a certain task. Similarly, MobileTest [9] is a tool for automatic black box test of software on mobile devices. It adopts an event-based testing approach to simplify the design of test cases and provides reusability of tests using software agents.

These approaches, however, require source-code modification, and effort to manage the multitude of handsets required for testing. A recent popular approach has been the development of online services that offer to conduct automated testing on a variety of hardware devices [28]. While these services offer a very realistic hardware platform for these tests, they still mostly rely on automation scripts that are either:

- *deterministic*: they perform a predetermined sequence of "touches" on a device,
- *brute-force*: they try to "touch" every single component on any given screen until the software crashes, or

- *random*: they generate software events in a random fashion, or allow testers to define which type of events to generate (pseudo-random) [2].

Thus, we argue that while automated testing systems have greatly matured in terms of hardware realism, they are still lacking in terms of modelling realistic interaction and usage. An important contribution of our work is to begin to model such an interaction between a user and a device on a high-level: modelling how users turn on and off their phone, and especially modelling the temporal dynamics of this interaction, which previous work has shown to be temporally skewed [18]. This aspect of the interaction is crucial because it is within these boundaries that users actually use the applications on their phone – i.e. they do not use it when it is off.

Time-sensitive state transition models

In our attempt to provide an analytical model of smartphone screen use, we have adapted Markov chains. Since Hamilton's seminal application of Markov models to U.S. real Gross National Product growth and the well-known NBER business cycle classification [23], *time-sensitive* Markov approaches has been more widely adopted. Initially, Diebold *et al.* [14] and Filardo [21] showed that the assumption of a static transition matrix leads to very restrictive models for many empirical settings. They extended the basic Markov switching model by *allowing transition probabilities to vary over time*. This was achieved by incorporating observable covariates, including explanatory variables and lagged values of the dependent variable. While this approach can be very useful, it does not take into account continuous time [6], but rather implicitly allows for time to vary the transition probabilities.

Extending this work, continuous-time Markov chains were developed [34]. These make an attempt to replace the transition matrix (which entails static probabilities) with functions. Their aim is to model how much time a model spends in a particular state. This is very relevant to our own work, since we are interested in modelling how much time the phone screen will spend in a particular state, before changing states. However, continuous-time Markov models assume an exponential density distribution regarding the time spent at each state. We are interested in non-standard probability distributions, which we derive empirically.

Very recently there has there been work on developing time-varying transition probabilities for Markov regime switching models. Bazzi *et al.* [6] recently proposed a new, dynamic approach to model time variation in transition probabilities in Markov switching models. In their model, the transition probabilities vary over time as specific transformations of the lagged observations, and they are not limited to exponential probability distributions. Hence, they have extended the observation driven approach to time varying parameter models [11]. For the specification of the dynamics of the transition matrix, they adopted the generalized autoregressive score dynamics of Creal *et al.* [13]; similar dynamic score models have been proposed by Creal *et al.* [12] and Harvey [24]. In our case, the dynamics are derived empirically, by analysis of data logs from hundreds of users with more than 270,000 observed screen state transitions.

State transition models in literature

There has been work that directly uses Markov models to measure usability and improve design [44,45], although without incorporating a strong notion of elapsed time. Previous work models User Interfaces as a Markov state transition model by identifying the possible states that a system can be in (*e.g.*, a microwave cooker may be in one of 6 different states), and identifying how pressing various items of an interface cause a transition to another state.

To measure usability, Thimbleby *et al.* [45] define the notions of "Designer's" and "Knowledge-free" transition matrices. For a given task (e.g. transition from state 1 to state 2), the designer's matrix has 1's on the correct transitions and 0's on the incorrect transitions, thus indicating zero chance of making a wrong transition. A knowledge-free matrix assumes that all transitions are equiprobable, and thus describes a process where buttons are pressed randomly. It is proposed that a measure of task performance is to calculate the expected number of steps needed to complete a task, and a user's knowledge can be affected by providing hints at the interface (e.g. to avoid pressing some buttons), thus bringing their expected behaviour closer to the designer's matrix.

This previous work makes important practical contributions towards optimising user interface design. For example, the settings menu of a smartphone can be represented as a Markov model, and it is possible to calculate the estimated number of steps required to complete a variety of tasks. The menu can then be restructured, for example by moving options or making some buttons less prominent, until the expected number of steps is minimised.

In our work we apply many of the same techniques, but instead of optimising UI design and layout, we are interested in developing a realistic model of smartphone use that can make runtime estimations about what will happen next and when, and can be interrogated under a variety of conditions. The software running on today's phones is not monolithic anymore, but highly modular: applications can be added and removed at runtime, thus being in a constant state of flux that designers cannot a-priori analyse. As suggested by Thimbleby *et al.* [45], we have collected a large volume of usage data in order to realistically model smartphone usage.

However, due to the complexity and variety of modern smartphone interfaces, we have decided to make a simplification akin to those used when studying large complex systems like health. Specifically, we model interaction on a very high level by studying how people interact with the screen [46].

This simplification has allowed us to build a model that can make runtime probabilistic predictions that have a strong notion of elapsed time. In other words, while we model interaction on a very abstract level, we can be precise about *when* transitions are likely to occur, and also take into account contextual variables such as the battery level, day and time, and user profile.

METHOD

We have obtained and reanalysed a dataset which originates from a free and open application called Securacy [19]. The application allows users to monitor network activity on their mobile phones. Securacy does not affect the screen functionality and therefore its operational state. The dataset was collected from opt-in volunteers all over the world, and contains screen state events with precise timespans.

We analysed screen usage events that were broadcast by the Android operating system (and recorded): turning the screen on, turning the screen off, locking the screen, unlocking the screen. In addition, the dataset also contains application usage and battery logs. The application logs were useful to monitor the KeyguardManager process to detect instances of screen activity without explicit user actions – a notification or incoming call – which might turn on the screen automatically.

For example, when the user receives a phone call, the screen turns on, but the KeyguardManager remains locked. In Table 1 we show the four distinct events that were recorded (on, off, locked, and unlocked). Each event occurs at a distinct and non-overlapping point in time (thus satisfying Markov's model Mutually Exclusive & Collectively Exhaustive requirement). This is a crucial semantic detail: while conceptually a phone screen may be ON and LOCKED at the same time, the events triggered by the operating system are mutually exclusive and non-overlapping.

When the Securacy software detected one of these events, it logged the event along with the current clock reading of the phone, the unique ID of the device, and the current battery level. Because each event has an associated timestamp, the events can be sorted in terms of the time in which they occurred, and from this sequence of events we can infer *state transitions* on the phone.

States	Description
0: **Off**	Power to the screen has stopped
1: **On**	Power to the screen has been activated
2: **Lock**	Screen locked (to avoid accidental input)
3: **Unlock**	Screen unlocked (input is enabled)

Table 1. A list of the operating system states that was logged.

RESULTS

The dataset sample spans 90 days, between March and May 2014. In total 218 participants took part in the original study, but 21 were discarded because they had contributed incomplete data (*e.g.*, no application logs). All participants used Android phones, and therefore the state transitions we captured are consistent across our sample. In total, we retrieved 271,832 state transitions.

Basic Markov model

We first present a summary of all data. We calculate the probability of transitioning from one state to the next, as shown in Table 2. To calculate these probabilities, we classify our 278,832 state transitions into one of the 16 cells in this table, and then calculate the probabilities across each row. Thus, each row sums to 100%. The values on the diagonal are close to zero but not all zero, because these are events that may occur when abrupt events happen (*e.g.*, the phone reboots, battery depletes).

From \ To	0: Off	1: On	2: Lock	3: Unlock
0: Off	0.50	33.03	59.40	7.05
1: On	45.32	2.03	0	52.64
2: Lock	2.83	95.64	0	1.53
3: Unlock	80.50	13.58	0	5.92

Table 2. The Markov chain transition matrix for all states. Values are shown in percentages across each row.

To exemplify, in Figure 1 we provide a visual representation of the state transition table, and for each transition we provide a textual description of what it signifies, what may have triggered it, or what happened immediately before the transition (see Table 3).

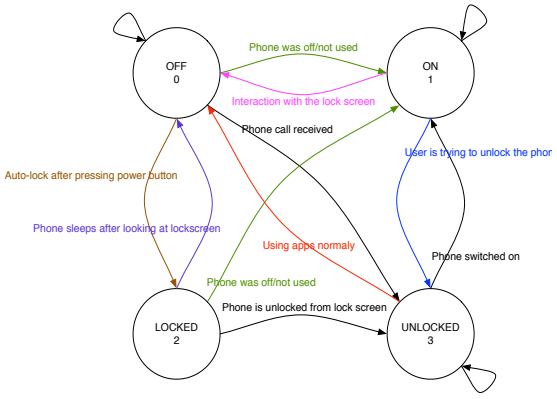

Figure 1. The four states of a smartphone's display, and what the transitions between them signal.

There are multiple transitions that our model takes into account. Some of them may be counter-intuitive, but the reader should keep in mind two important details. First, today's phones typically have a physical power button that turns off the screen. In addition, some may have an "auto-lock" function on their phone, which locks the screen whenever it turns off or after an idle timeout. Pressing the power button of a phone may not lock it but instead turns off the screen. A small number of users opt to disable the auto-lock option, thus turning off their phone screen without necessarily locking it.

From\To	0: Off	1: On	2: Locked	3: Unlocked
0: Off	n/a	The phone was off and the user took action to activate the screen.	If a phone is configured to auto-lock, then when it is turned off it will quickly and automatically lock itself.	This can happen when a phone-call is received.
1: On	The user was interacting with the lock-screen (e.g. checking time) and then turned off the phone. This transition helps us measure how long people glance at their lock-screen	n/a	n/a - to become locked, the screen needs to be turned off first.	The user was interacting with the lock-screen and then unlocked the phone. This transition helps us measure how quickly people unlock their phone
2: Locked	The user is interacting with the lock screen and then leaves the phone untouched until power-saving turns off the screen automatically.	The phone was not used (i.e. off and locked) and the user pressed the power button to turn the screen on.	n/a	Happens when the phone is locked (but the screen is on) and then unlocked again.
3: Unlocked	The user was using apps, and then pressed the power button to turn off the screen.	The screen turned off (but did not lock the phone) and then the user took action to turn on the screen.	n/a - to become locked, the screen needs to be turned off first.	n/a

Table 3. Explanation of what various transitions mean. For each transition we describe what actions happened before the transition was made.

It is important to note that our model is built to describe screen events. Since these are the tokens of our analysis, we have modelled them as "states" in a state transition model. Clearly, a smartphone may be in a variety of states, but for our purposes we are effectively treating a screen event as a state. This key shift in our perspective will allow us to incorporate more variables in our analysis, as we describe next.

Time-varying Markov model

For every sequential pair of screen events we recorded during the study we can calculate the actual time between them. This effectively tells us how much time was spent at each state in our model. In other words, we can define a time-varying Markov model to describe all transitions. To characterise the temporal distribution of all transitions, our first step is to generate a Gaussian kernel density estimation of the amount of time between subsequent events (Figure 2). Thus, across all devices in the dataset we consider how much time they spent in-between states, and subsequently we aggregate all temporal data across all devices. The distribution shown in Figure 2 is multimodal, and the most prominent local maxima are indicated on the x-axis.

There are multiple noticeable temporal signatures in Figure 2, which we attempt to further decompose. To achieve this, we plot the kernel density estimation for each unique transition (Figure 3), *i.e.*, for each pair of states. In other words, the kernel density distribution for transition ON-OFF is shown in the strip labelled "1-0". In Figure 3, we are able to see that the different transitions in our model are preceded by varying time periods.

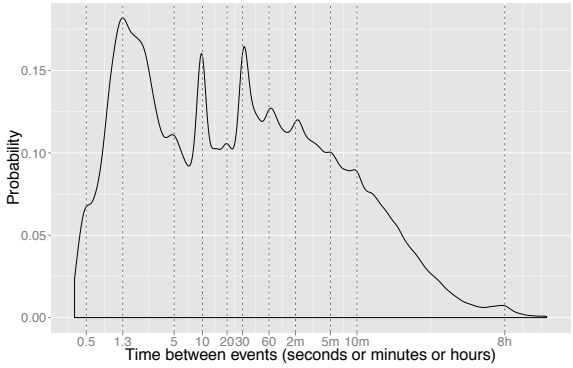

Figure 2. Gaussian kernel density estimation, indicating the probability (y-axis) that a subsequent screen event will take place after a certain amount of time (x-axis).

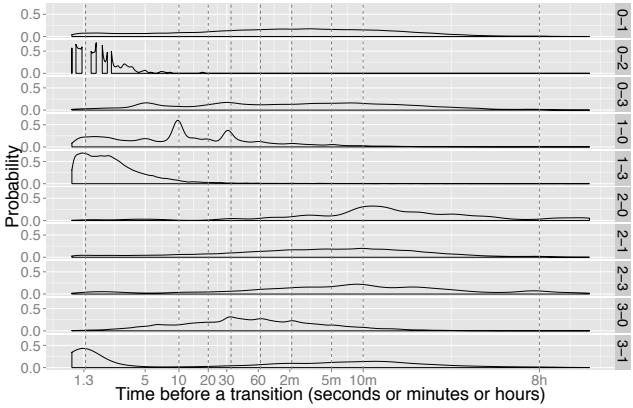

Figure 3. Gaussian kernel density estimation for each distinct transition (right y-axis strip). The left y-axis indicates the probability that the transition will take place after a certain amount of time (x-axis).

We interpret Figure 3 as follows. Assuming the phone is currently in state 1, we wish to investigate the probability of transitioning to another state. One possibility is that the phone may transition to state 0, and therefore we look at the strip "1-0" and determine that there is an increased chance this transition would happen 10 or 30 seconds after arriving to state 1. Another possibility is that the phone transitions to state 3, and looking at the strip "1-3" we determine that this transition is likely to happen within 5 seconds of arriving to state 1. There is 0 probability of transitioning to state 2. From this example analysis we obtain the intuitive understanding that once the phone arrives to state 1, the first few seconds are moments where there is high probability of transitioning to state 3, but after 5 seconds or so it becomes much more likely to transition to state 0.

Incorporating context into the model

Next, we proceed to further unpack each strip from Figure 3 in multiple ways. We first break it down by the hour of day in which this transition was observed (Figure 4 top), as well as the day of week in which this transition was observed (Figure 4 bottom). These two figures show a heat map visualisation of when a particular transition is most likely to occur (hour of day, day of week), and how much time is likely to precede it.

We interpret Figure 4 as follows. Assuming that we are interested in the transition from state 1 to state 0, we can look at Figure 4a to see at which times this transition is likely to happen. For instance, we observe that this transition is most likely to happen after 10 or 30 seconds, but we also see how this varies for different times of the day. Thus, we find that between 2am and 7am there is a reduced probability of this transition being observed. Similarly, we can look at strip "1-0" in Figure 4 to determine the day of the week where this transition is likely to occur. We observe that on Sunday this transition is slightly less probable.

Next, our analysis considers the battery level of the phone when a state transition takes place. In other words, at the moment a transition was recorded by our software, we also noted the battery level at that moment. In Figure 5 we see the Gaussian kernel estimation densities of time between events (similar to Figure 2), but now color-coded to reflect 4 possible battery levels. They represent four quartiles of battery level as follows:

- q1 indicates the battery level is between 100%-75%,
- q2 indicates 74%-50%,
- q3 indicates 49%-25%, and
- q4 indicates 24%-0%.

In addition, we break down each of the four quartile sets estimations into hour of day (Figure 6 top), and day of week (Figure 6 bottom). We interpret this figure just like Figure 4. Therefore, we observe that when the battery of the phone is above 75% (*i.e.* in q1) then screen transitions are most likely during work hours (Figure 6 top, strip 1), while when

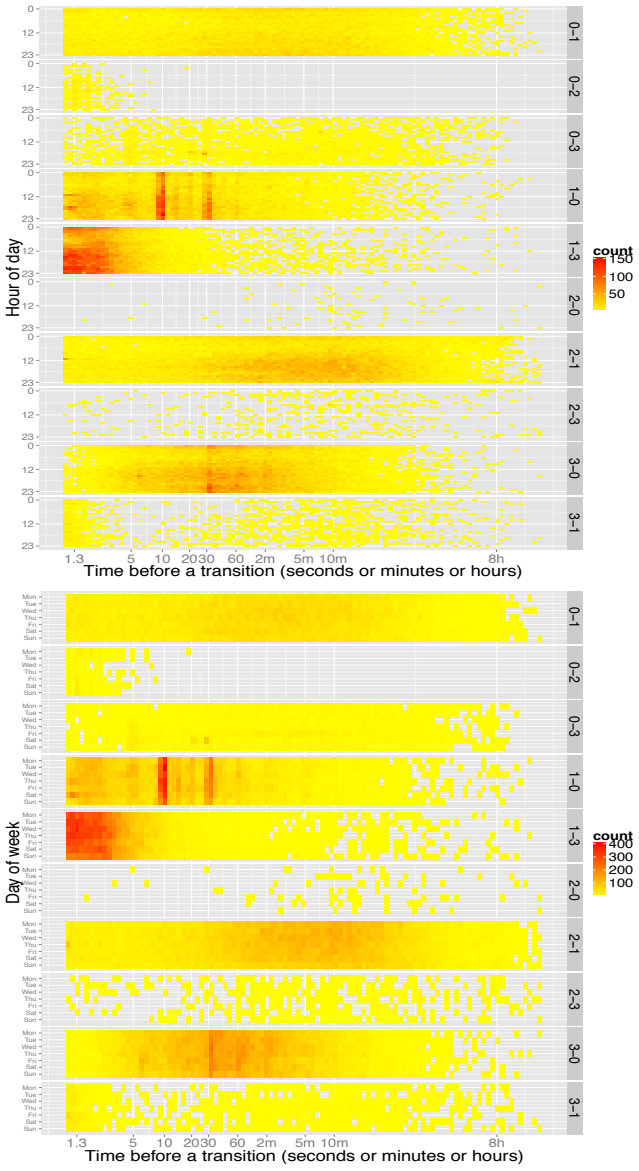

Figure 4. Heatmaps showing the frequency (colour) of observing state transitions (right y-axis strip) at certain hours (y-axis top) or days (y-axis bottom) and the amount of time that preceded them (x-axis).

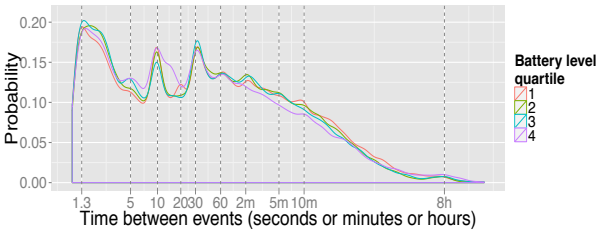

Figure 5. Individual quartiles of battery level (100%-75%-50%-25%) Gaussian kernel density estimation. Shows the probability (y-axis) that a subsequent state transition will occur after a certain amount of time (x-axis) given a battery level (colour).

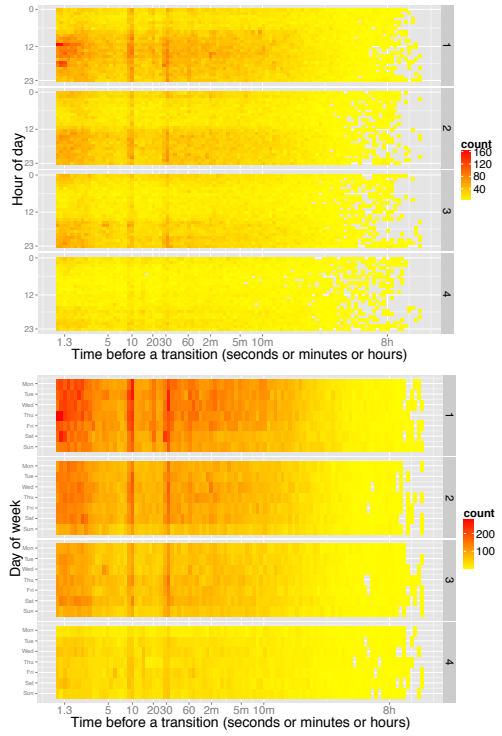

Figure 6. Heatmaps showing the frequency (colour) of observing a state transition after a certain amount of time (x-axis) for a given battery level (right y-axis strip), hour (y-axis top), and day (y-axis bottom).

the battery level is below 25% (i.e. in q4) then screen transitions are more likely outside work hours (Figure 6 top, strip 4).

Incorporating user typology into the model

Each user exhibited a distinct usage of their smartphone screen, with varying temporal signatures in their transitions. For instance, in Figure 7 we show the Gaussian kernel density estimation for three different devices in our dataset. We observe that their use varies considerably.

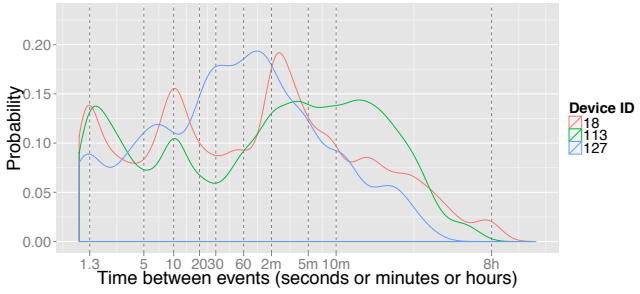

Figure 7. For 3 devices (colour) we show the Gaussian kernel density estimation. This indicates the probability (y-axis) that a subsequent state transition will take place after a certain amount of time (x-axis).

Given this diversity, we investigated whether user typologies can be identified through clustering. Thus, we next proceed to determine whether different users exhibit varying behaviour in how they transition between screen states. We first visualise the unique "footprint" of each participant in terms of transitioning between states, using a scatterplot heatmap (Figure 8).

Figure 8. The heatmap visualises the behaviour of each device. Each point represents for each particular device (y-axis) its propensity (colour) to wait a certain amount of time (x-axis) between screen state transitions.

From Figure 8 we extract an array of frequency values for each device (one row corresponds to one device), which are effectively a histogram. Each device is one row in this figure, and for each row we have the same bins, and a frequency for each bin. Using these frequency values we can apply k-means clustering iteratively for varying cluster numbers, and use the elbow method to generate Figure 9. Applying heuristics to this graph, we deduced that the optimum number of clusters is 3, each containing 39, 87 and 70 devices respectively.

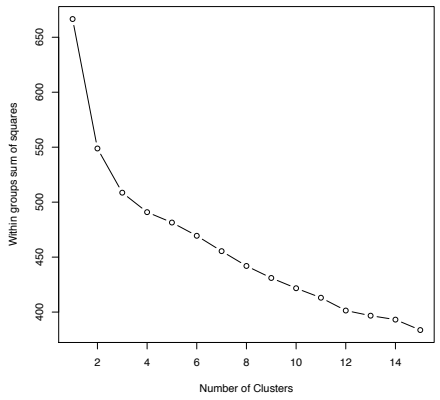

Figure 9. An elbow graph showing how varying the numbers of device clusters affects the within groups sum of squares.

Next, for each cluster we plot the respective Gaussian kernel density estimation (Figure 10). This figure provides a summary description of the transition behaviour for users in each cluster. As a result, we are also able to visualise the differences between each cluster. For instance, we observe that the red line is below other lines when time is less than 2 mines, and above other lines when time is above 2 minutes. This means that users in cluster 1 are less likely to have short transition (e.g., at 5, 30 or 60 seconds), and more likely to have longer transitions (e.g. above 2 minutes).

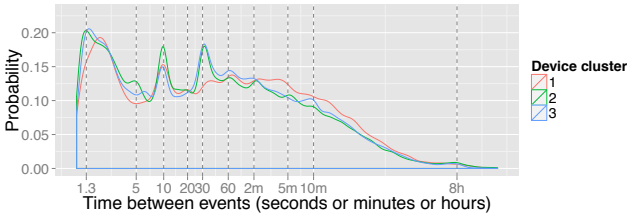

Figure 10. For each of the 3 device clusters we show the Gaussian kernel density estimation. This indicates the probability (y-axis) that a subsequent event will take place after a certain amount of time (x-axis) depending on the device cluster (colour).

In addition, we are able to break down the density estimation for each device cluster depending on the hour of day (Figure 11 top) and day of week (Figure 11 bottom). Just like in the previous day/week heatmaps, we are able to visually observe the behaviour of each cluster of devices at different times of day / week. We observe that devices in cluster 2 are more likely to make screen transitions during work hours (Figure 11 top, strip 2), and workdays (Figure 11 bottom, strip 2).

DISCUSSION

We first present a worked example of how the analytical model we have described so far can be interrogated to make runtime probabilistic estimations about a smartphone's screen state. We then reflect on our findings in light of prior work.

Interrogating the model at runtime

We have presented an extensive analysis that has enabled us to define a time-variant transition matrix of screen use. Through a variety of visualisations and graphs we have shown how empirical data can be used to augment our transition model to take into account: granular definition of time, a notion of calendar time (hour of day and day of week), battery level, and user archetype. Underlying all the visualisations and charts we have presented is a model that entails the notion of time between transitions. We now demonstrate how our model can be used for making runtime predictions for multiple purposes. Our examples are not exhaustive, but should be sufficient to bootstrap future application development.

Our model allows mobile operating systems, or interaction simulators, to make runtime probabilistic estimations about

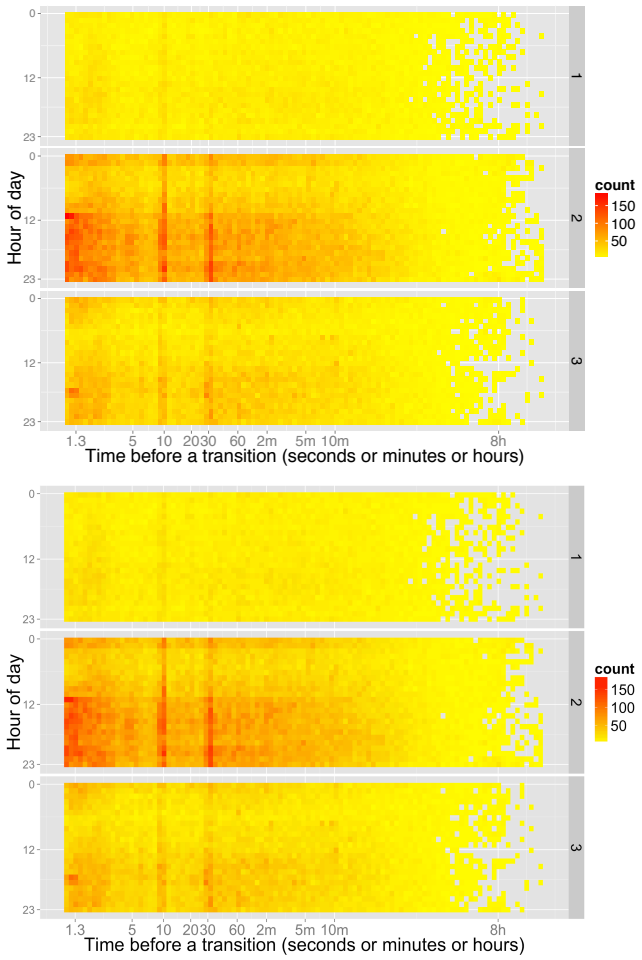

Figure 11. Heatmaps with the frequency (colour) of observing a state transition after a certain amount of time (x-axis) for a given device cluster (right y-axis strip), hour (y-axis top), and day (y-axis bottom).

the near future, and in doing so make decisions that can help optimise its resources. For instance, an operating system can interrogate our model to answer the following arbitrary questions:

- How much time, on average, do we spend at each state?
- Starting in state 3, how much time (on average) does it take to reach either state 2 or 1?
- If the user turns on the phone (state 1), what is the probability that the phone remains in that state for 20 seconds? For 120 seconds? How does this change with context?

We now illustrate how our model can answer the latter question. The solutions we describe here can be verified and replicated using the R code included with this paper.

We begin by answering: If we arrive to state 1 (screen ON), what is the probability that we are still in state 1 after 20 seconds? To calculate the answer, we need to calculate the probabilities that a transition to any other state will be made

in less than 20 seconds. Thus, the solution to the question is:

= 1 - (
probability we move to state 0 in less than 20 seconds +
probability we move to state 2 in less than 20 seconds +
probability we move to state 3 in less than 20 seconds)

First, we rewrite the equation in a more useful form:

= 1 - (
prob. we move to state 0 * (integral 0<t<20 for 1➔0) +
prob. we move to state 2 * (integral 0<t<20 for 1➔2) +
prob. we move to state 3 * (integral 0<t<20 for 1➔3))

Then, using Table 2 to obtain transition probabilities, we have:

= 1 - (0.4532 * (integral 0<t<20 for 1➔0) +
(0 * (integral 0<t<20 for 1➔2) +
(0.5264 * (integral 0<t<20 for 1➔3))

Finally, calculating the integrals from Figure 3 to obtain probabilities regarding time we obtain

= 1 - (0.4532 * 0.6659034 + 0 + 0.5264 * 0.9705692)
= 0.187

Thus, the probability of staying in state 1 for at least 20 seconds is 18.7%. Following the same process, we can make estimations for other arbitrary cases. For example, we calculate that the probability of staying in state 1 for at least 120 seconds drops to 6.9%. Furthermore, we can take additional variables into consideration. The probability of staying in State 1 for 20 seconds – when the battery is above 75% – is 19%, and drops to 17.7% if the battery is below 25%. Finally, the probability of staying in that state for 20 seconds when it is 2pm is 17.5%, while it goes up to 18.4% at 4am. The R code included with this paper works through these examples to arrive at our reported values.

We evaluated the precision of our model's estimations against a different dataset that we obtained and re-analysed [46]. This independent dataset contained 34,169 screen transitions, application and battery logs from a different population (n=17) collected during 2015. We validate the estimations of our model by manually analysing the transitions observed in this independent dataset. The RMSE is approximately 7.8%.

Question	Model	Observed
State 1 for 20 s.	18.7%	10.5%
State 1 for 120 s.	6.9%	4.5%
State 1 for 20 s. (battery > 75%)	19.0%	10.9%
State 1 for 20 s. (battery < 25%)	17.7%	10.8%
State 1 for 20 s. (2pm)	17.5%	10.8%
State 1 for 20 s. (4am)	18.4%	6.6%

Table 4. Comparison of the estimations made by our model vs. observed transitions in an independent dataset.

In addition, using solely the data from Table 2 it is possible to apply traditional Markov modelling analysis, for instance to study how the system state evolves over time [44,45]. An important contribution of our work is that instead of using a static transition matrix, it can effectively generate a dynamic transition matrix based on the contextual variables we take into account.

Finally, the analysis can be enriched by incorporating a virtual clock (day and hour), battery status (4 distinct quartiles of battery charge level), and one of 3 user behaviour strategies. Incorporating these additional variables effectively means using a different probability distribution to determine the amount of time to wait before making a certain transition. To incorporate these variables simultaneously, the multiple probability distributions for each possible combination of the variables are aggregated to derive a single probability distribution.

Towards context-rich interaction models

Our analysis demonstrates 3 ways in which analytical models of interaction can be enriched by incorporating context. We show that day and time, battery level, and user type, have a substantial effect on the transition probabilities of certain events, but not all.

Incorporating battery into the model showed that as the battery depletes, the overall usage begins to decline, as expected. We also note a jump in the frequency of events between 10-20 seconds when the battery level is in the lowest quartile (less than 25% charge). This suggests that users are more likely to use their phone in such short bursts when they are running out of battery, most likely to preserve energy. Previous work has shown that mobile users are battery-conscious both about use [17] and charging [16]. The fact that our model incorporates this behaviour is additional evidence about its fidelity.

Additionally, our analysis identified 3 types of usage. Type 2 exhibits strong scheduling in terms of hour and day. Type 3 exhibits strong scheduling in terms of time, but not in terms of day. Finally, Type 1 does not exhibit a substantial scheduling. The cluster sizes (39, 87, 70) indicate the relative popularity of each archetype.

Previous work has used the Experience Sampling Method to collect rich in-situ data on why and how people use their mobile devices [18], and our model implicitly quantifies the different strategies that users exhibit. This classification can be used as a basis for establishing optimisation profiles for state switching.

For instance, a device manufacturer that is able to longitudinally determine the archetype of different users can incorporate state switch predictions already into the operating system. This can be used to shorten the idle times when a phone is just waiting for the auto shut-off and locking of the display, leading into enhanced battery performance. Finally, it is possible to treat each user as their own cluster, making it possible to design even more

efficient, runtime optimisations by using our model driven by each user's own data [50].

Markov chains & mobile devices

In the context of mobile computing, Markov chains have been used to improve the reliability of cloud computing environments [36], or optimizing menu structures [43]. Recently, Markov chains have also been utilised in combating sophisticated malware: Suarez-Tangil *et al.* generated malware trigger-conditions for individual mobile users, based on their unique phone use patterns [42]. Finally, Markov models have been used to predict the next used application [22].

This latter approach can readily be incorporated into our model: each application can represent a phone state, and therefore analysis of state transitions could become richer. However, it is challenging to account for applications that may be removed (thus affecting transition probabilities), or newly installed applications for which the model may not have enough data to incorporate. In addition to modelling interaction, such an approach could be used to identify flaws in the system. A recent review on the topic [8] suggests that use of Markov chains in software testing has already resulted in uncovering critical flaws in complex industrial systems and can significantly improve testing processes in general.

In our case, understanding and optimizing the use of mobile devices is an increasingly relevant challenge due to the widespread popularity and increasing fragmentation of these devices. There have already been some studies that attempt to systematically collect real-world usage data from such devices. For example, the *Device Analyzer* project collected in-depth contextual data from over 16,000 mobile devices in 175 countries and made the data set publicly available for the research community to reuse [47]. Another framework, AWARE, is openly available for researchers, developers and individuals alike, and is used to capture hardware-, software-, and human-based data from Android powered smart phones [5]. The rich data collected by the framework can then be freely used in e.g. optimising the battery life of mobile devices [16] and understanding application use [18].

Limitations

We should point out a number of limitations of our work. First, our study collected data from Android devices only and it is very likely that for different operating systems the results may differ. In addition, it is questionable whether the Google Play app store can reach a representative sample of the smartphone users, since there are smartphone users who are not active users of app stores. As a result, the model we have presented may prove to be inaccurate for a segment of the population. As we discussed, it is possible to treat each user as a cluster of their own, therefore attempt to create a model solely from their behaviour. This, however, could require a substantial amount of time, and may not accurately capture rare events. Finally, users may change the patterns how they use their mobile devices rapidly [47], and therefore the model may become outdated.

CONCLUSION

Gartner predicts that in 2016 alone closer to 2 billion mobile devices will be shipped [48], even optimisations that intuitively appear insignificant can yield highly positive network effects. Our work improves our understanding of mobile phone use by modelling the wider context of screen state transitions. The model we present does not consider details of individual applications – it is app-agnostic and considers the entire "session." Since different applications have different usage patterns [18], we point out that these patterns take place within a single usage session that lasts from the moment the phone is turned on until it is turned off and auto-locked.

While application use has a key role in phone state transitions, empirical knowledge about the typical sessions enveloping the application usage can help, *inter alia*, develop simulators for testing purposes or optimizing mobile operating systems. The work we present allows for modelling of screen state transitions based on empirical data. Our simplest model can be used to guide a basic simulation of phone use, generate realistic "use traces", or runtime prediction. However, we argue that it also offers a solid and practical foundation for further developing rigorous modelling of interaction with mobile devices.

ACKNOWLEDGMENTS

This work is partially funded by the Academy of Finland (Grants 276786-AWARE, 285062-iCYCLE, 286386-CPDSS, 285459-iSCIENCE), and the European Commission (Grants PCIG11-GA-2012-322138, 645706-GRAGE, and 6AIKA-A71143-AKAI).

REFERENCES

1. Pekka Abrahamsson, Antti Hanhineva, Hanna Hulkko, Tuomas Ihme, Juho Jäälinoja, Mikko Korkala, Juha Koskela, Pekka Kyllönen and Outi Salo. 2004. Mobile-D: An Agile Approach for Mobile Application Development. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications*, ACM, 174-175. http://doi.acm.org/10.1145/1028664.1028736
2. Application Exerciser Monkey. Retrieved 17/12/2014 from http://developer.android.com/tools/help/monkey.html
3. Leena Arhippainen and Marika Tähti. 2003. Empirical evaluation of user experience in two adaptive mobile application prototypes. In *International Conference on Mobile and Ubiquitous Multimedia*, 27-34.
4. Richard Atterer and Albrecht Schmidt. 2007. Tracking the Interaction of Users with AJAX Applications for Usability Testing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1347-1350. http://doi.acm.org/10.1145/1240624.1240828

5. AWARE: Android Mobile Context Instrumentation Framework. Retrieved 29/01/2014 from http://www.awareframework.com
6. Marco Bazzi, Francisco Blasques, Siem S. J. Koopman and Andre Lucas. 2014. *Time Varying Transition Probabilities for Markov Regime Switching Models.*
7. Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. 2014. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 1991-2000. http://doi.acm.org/10.1145/2556288.2557354
8. Divya Bindal. 2013. A review of markov model for estimating software reliability. *International journal of advanced research in computer science and software engineering* 3, 6: 426-433.
9. Jiang Bo, Long Xiang and Gao Xiaopeng. 2007. MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices. In *Proceedings of the Second International Workshop on Automation of Software Test*, IEEE Computer Society, 8-8. http://dx.doi.org/10.1109/AST.2007.9
10. Ulrich Burgbacher and Klaus Hinrichs. 2014. An Implicit Author Verification System for Text Messages Based on Gesture Typing Biometrics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2951-2954. http://doi.acm.org/10.1145/2556288.2557346
11. David R. Cox. 1981. Statistical analysis of time series: Some recent developments [with discussion and reply]. *Scandinavian Journal of Statistics* 8: 93-115.
12. Drew Creal, Siem S. J. Koopman and André Lucas. 2011. A dynamic multivariate heavy-tailed model for time-varying volatilities and correlations. *Journal of Business & Economic Statistics* 29, 4: 552-563.
13. Drew Creal, Siem J. Koopman and André Lucas. 2012. Generalized autoregressive score models with applications. *Journal of Applied Econometrics* 28, 5: 777-795.
14. Francis X. Diebold, Joon-Haeng Lee and Gretchen G. C. Weinbach. 1994. Regime switching with time-varying transition probabilities. In *Nonstationary time series analysis and cointegration* C Hargreaves (eds.). Oxford University Press, 283-302.
15. Jacob Eisenstein and Angel Puerta. 2000. Adaptation in Automated User-interface Design. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, ACM, 74-81. http://doi.acm.org/10.1145/325737.325787
16. Denzil Ferreira, Anind K. Dey and Vassilis Kostakos. 2011. Understanding human-smartphone concerns: a study of battery life. In *International Conference on Pervasive Computing*, Springer-Verlag, 19-33. http://ubicomp.oulu.fi/files/pervasive11.pdf
17. Denzil Ferreira, Eija Ferreira, Jorge Goncalves, Vassilis Kostakos and Anind K. Dey. 2013. Revisiting Human-Battery Interaction with an Interactive Battery Interface. In *International Joint Conference on Pervasive and Ubiquitous Computing* , ACM, 563-572. http://ubicomp.oulu.fi/files/ubicomp13b.pdf
18. Denzil Ferreira, Jorge Goncalves, Vassilis Kostakos, Louise Barkhuus and Anind K. Dey. 2014. Contextual Experience Sampling of Mobile Application Micro-Usage. In *International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM, 91-100. http://ubicomp.oulu.fi/files/mobilehci14.pdf
19. Denzil Ferreira, Vassilis Kostakos, Alastair R. Beresford, Janne Lindqvist and Anind K. Dey. 2015. Securacy: An Empirical Investigation of Android Applications' Network Usage, Privacy and Security. In *Conference on Security and Privacy in Wireless and Mobile Networks*, ACM, 11:1-11:11. http://ubicomp.oulu.fi/files/wisec15.pdf
20. Denzil Ferreira, Vassilis Kostakos and Anind K. Dey. 2015. AWARE: mobile context instrumentation framework. *Frontiers in ICT* 2, 6: 1-9.
21. Andrew J. Filardo. 1994. Business-cycle phases and their transitional dynamics. *Journal of Business & Economic Statistics* 12, 3: 299-308.
22. Charles Gouin-Vallerand and Neila Mezghani. 2014. An Analysis of the Transitions Between Mobile Application Usages Based on Markov Chains. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ACM, 373-378. http://doi.acm.org/10.1145/2638728.2641700
23. James D. Hamilton. 1989. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society* 57, 2: 357-384.
24. Andrew C. Harvey. 2013. *Dynamic models for volatility and heavy tails : with applications to financial and economic time series.* Cambridge University Press.
25. Niels Henze, Enrico Rukzio and Susanne Boll. 2011. 100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, ACM, 133-142. http://doi.acm.org/10.1145/2037373.2037395
26. Paul Holleis, Friederike Otto, Heinrich Hussmann and Albrecht Schmidt. 2007. Keystroke-level Model for Advanced Mobile Phone Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1505-1514. http://doi.acm.org/10.1145/1240624.1240851
27. Paul Holleis and Albrecht Schmidt. 2008. MakeIt: Integrate User Interaction Times in the Design Process of Mobile Applications. In *Pervasive Computing*, Springer Berlin Heidelberg, 56-74.
28. Jouko Kaasila, Denzil Ferreira, Vassilis Kostakos and Timo Ojala. 2012. Testdroid: automated remote UI testing on Android. In *International Conference on Mobile and Ubiquitous Multimedia*, ACM, 28:1-28:4. http://ubicomp.oulu.fi/files/mum12a.pdf
29. Titti Kallio and Anne Kaikkonen. 2005. Usability testing of mobile applications: A comparison between

laboratory and field testing. *Journal of Usability studies* 1, 4-16: 23-28.

30. Eeva Kangas and Timo Kinnunen. 2005. Applying User-centered Design to Mobile Application Development. *Commun. ACM* 48, 7: 55-59.
31. David J. Kasik and Harry G. George. 1996. Toward Automatic Generation of Novice User Test Scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 244-251. http://doi.acm.org/10.1145/238386.238519
32. Panu Korpipää, Jonna Häkkilä, Juha Kela, Sami Ronkainen and Ilkka Känsälä. 2004. Utilising context ontology in mobile device application personalisation. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, ACM, 133-140. http://doi.acm.org/10.1145/1052380.1052399
33. Kari Kostiainen, Ersin Uzun, N Asokan and Philip Ginzboorg. 2007. Framework for comparative usability testing of distributed applications. In *Security User Studies: Methodologies and Best Practices Workshop*.
34. Norris. 1998. *Markov chains.* Cambridge University Press.
35. Abhinav Parate, Matthias Böhmer, David Chu, Deepak Ganesan and Benjamin B. M. Marlin. 2013. Practical Prediction and Prefetch for Faster Access to Applications on Mobile Phones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 275-284. http://doi.acm.org/10.1145/2493432.2493490
36. JiSu Park, HeonChang Yu, KwangSik Chung and Eunyoung Lee. 2011. Markov Chain Based Monitoring Service for Fault Tolerance in Mobile Cloud Computing. In *Advanced Information Networking and Applications (WAINA)*, IEEE, 520-525.
37. Martin Pielot. 2014. Large-scale Evaluation of Call-availability Prediction. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 933-937. http://doi.acm.org/10.1145/2632048.2632060
38. I Satoh. 2003. A testing framework for mobile computing software. *Software Engineering, IEEE Transactions on* 29, 12: 1112-1121.
39. Choonsung Shin, Jin-Hyuk Hong and Anind K. Dey. 2012. Understanding and Prediction of Mobile Application Usage for Smart Phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 173-182. http://doi.acm.org/10.1145/2370216.2370243
40. Uwe Siebert, Oguzhan Alagoz, Ahmed M. Bayoumi, Beate Jahn, Douglas K. Owens, David J. Cohen and Karen M. Kuntz. 2012. State-Transition Modeling: A Report of the ISPOR-SMDM Modeling Good Research Practices Task Force–3. *Medical Decision Making* 32, 5: 690-700.
41. Vijay Srinivasan, Saeed Moghaddam, Abhishek Mukherji, Kiran K. Rachuri, Chenren Xu and Emmanuel E. M. Tapia. 2014. Mobileminer: Mining your frequent patterns on your phone. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 389-400.
42. Guillermo Suarez-Tangil, Mauro Conti, Juan E. Tapiador and Pedro Peris-Lopez. 2014. Detecting Targeted Smartphone Malware with Behavior-Triggering Stochastic Models. In *Computer Security - ESORICS 2014* (eds.). Springer International Publishing, 183-201.
43. Lee Suk-Won and Myung Ro-Hae. 2007. Prediction of Mobile Phone Menu Selection with Markov Chains. *Journal of Korean Institute of Industrial Engineers* 33, 4.
44. Harold Thimbleby. 2004. User Interface Design with Matrix Algebra. *ACM Trans. Comput.-Hum. Interact.* 11, 2: 181-236.
45. Harold Thimbleby, Paul Cairns and Matt Jones. 2001. Usability Analysis with Markov Models. *ACM Trans. Comput.-Hum. Interact.* 8, 2: 99-132.
46. Niels van Berkel, Chu Luo, Theodoros Anagnostopoulos, Denzil Ferreira, Jorge Goncalves, Simo Hosio and Vassilis Kostakos. 2016. A Systematic Assessment of Smartphone Usage Gaps. In *Conference on Human Factors in Computing Systems*, ACM. http://ubicomp.oulu.fi/files/chi16a.pdf
47. Daniel T. Wagner, Andrew Rice and Alastair R. Beresford. 2014. Device Analyzer: Large-scale Mobile Data Collection. *SIGMETRICS Perform. Eval. Rev.* 41, 4: 53-56.
48. Worldwide Device Shipments to Grow 1.9 Percent in 2016, While End-User Spending to Decline for the First Time. Retrieved 16/03/2016 from http://www.gartner.com/newsroom/id/3187134
49. Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell and Tanzeem Choudhury. 2013. Preference, Context and Communities: A Multi-faceted Approach to Predicting Smartphone App Usage Patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers*, ACM, 69-76. http://doi.acm.org/10.1145/2493988.2494333
50. Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal and Jie Liu. 2012. Fast app launching for mobile devices using predictive user context. In *MobiSys*, 113-126. http://dl.acm.org/citation.cfm?id=2307636.2307648