International Journal of Mobile Human Computer Interaction
Volume 8 • Issue 3 • July-September 2016

Investigating UI Displacements in an Adaptive Mobile Homescreen

Lauren Norrie, School of Computing Science, University of Glasgow, Glasgow, Scotland
Roderick Murray-Smith, School of Computing Science, University of Glasgow, Glasgow, Scotland

ABSTRACT

The authors present a system that adapts application shortcuts (apps) on the homescreen of an Android smartphone, and investigate the effect of UI displacements that are caused by the choice of adaptive model and the order of apps in the homescreen layout. They define UI displacements to be the distance that items move between adaptations, and they use this as a measure of stability. An experiment with 12 participants is performed to evaluate the impact of UI displacements on the homescreen. To make the distribution of apps in the experiment task less contrived, naturally generated data from a pilot study is used. The authors' results show that selection time is correlated to the magnitude of the previous UI displacement. Additionally, selection time and subjective rating improve significantly when the model is easy to understand and an alphabetical order is used, conditions that increase stability. However, rank order is preferred when the model updates frequently and is less easy to understand. The authors present their approach to adapting apps on the homescreen, and initial insights into UI displacements.

KEYWORDS

Adaptive Menus, Homescreen, Human Computer Interaction, Mobile Apps, Navigation, Smartphones, UI Displacement, Usability Experiment

INTRODUCTION

The homescreen is the main menu that is displayed on a mobile device. On Android, a small number of apps can be placed on the homescreen for fast access, and the app drawer can be opened to display the entire list of apps. As a user installs more apps, the time and effort required to locate ones that do not feature on the homescreen will increase. Though only a small number of installed apps are used frequently (Falaki et al., 2010), the set that are frequently used changes over time (Shin, 2012). Therefore, the homescreen needs to be organised regularly. However, arranging icons on the homescreen can be annoying and time consuming, and some users do not arrange their icons at all (Böhmer & Krüger, 2013b). Supporting the user with organising the homescreen will improve the usability of mobile devices.

DOI: 10.4018/ijmhci.2016070101.oa

With access to a user's app launch history, a system can predict the apps that are most likely to be launched next and adapt these on the homescreen automatically. The adaptive menu can replace a section of the homescreen to provide fast access to a selection of apps. The selection of apps is chosen by an adaptive model, which can be trained on a variety of features. Some features can be related to app use, such as launch frequency, and others may be contextual. Shin et. al. (2012) provide a detailed overview of features relating to mobile app use. In this work, we investigate the impact of stability on the homescreen by comparing adaptive models and the order of the layout.

Adaptive Menus

Menu design is an established field in HCI. Sears and Shneiderman (1994) created Split Menus for desktop applications, and demonstrated that selection time can be decreased by moving or copying the top 4 frequently used apps to the top of the menu. Findlater and McGrenere (2004) compared static, adaptive and adaptable split menus, with the naturally generated data of a single MS Office user. The majority of their participants wanted a personalised menu, and were better at customising the adaptable interface after they had used the adaptive one. As the homescreen is an adaptable interface, this suggests that an adaptive component could support users with organising their app icons. Fukazawa et al. (2009) note that adaptations can be more appropriate for novice users. Findlater and McGrenere (2008) compared the impact of screen size in adaptive user interfaces, and found that an 'adaptive interface is more beneficial when screen real estate is constrained' and that 'adaptive interfaces are low risk for small screens' (p. 1254). This provides motivation for exploring adaptive menus on a mobile device.

Accuracy and predictability are conflicting factors that affect the design of an adaptive menu. Gajos et al. (2008) define these as follows: 'a model's *accuracy* is the percentage of time that the necessary UI elements are contained in the adaptive area,' and 'a model is *predictable* if it follows a strategy users can easily model in their heads' (p. 1271). While accuracy is controlled by the model, predictability depends on how a user perceives the adaptations. Predictability can be influenced by *ease of understanding*, *frequency of adaptations*, and *stability*.

Accuracy and predictability have been explored by Gajos et al. (2008), with an adaptive split interface that copies MS Office functions into a designated adaptive toolbar. The content of the adaptive toolbar and the sequence of button presses were predetermined to ensure the desired level of accuracy (50% and 70%), and predictability was controlled by comparing a most recently used model with a random model. It was found that increased accuracy improved selection time more than increased predictability. However, the most recently used model significantly increased subjective ratings, including control, predictability and the extent to which participants knew an item was in the toolbar. Gajos et al. (2006) found that accuracy can also increase the utility of the adaptive interface. In our experiments, we consider models that have high accuracy (approximately 85%).

Predictability can also depend on the frequency of adaptations, i.e. the rate at which updates to the user interface are applied. The impact of the frequency of adaptations can be illustrated by the results of Sears and Shneiderman (1994) and Findlater and McGrenere (2004), which show that increasing updates to the user interface from a slow pace (once per session) to a faster pace (up to once per interaction) decreases performance with the same interface. Gajos et al. (2006) discuss this result and 'suspect that the cause stems from the fact that high frequency effectively reduces a mechanism's predictability'. Holding back on updates to an adaptive menu will preserve stability during periods of interaction. However, limiting updates to the UI will reduce accuracy, since the state of the model will only be reflected in the UI at certain points in time. In our experiment, we allow items to update after each interaction, and explore the impact of stability when updates are frequent.

Stability

In our experiment, we consider the impact of *predictability* and *stability* in an adaptive homescreen. As far as we are aware, stability has yet to be investigated on the homescreen. Zhang et al. (2013)

considered stability in an app drawer with their Nihao Launcher, and found that 'a larger UI difference does not necessarily require longer app lookup time.' However, the app drawer contains all apps installed on a smartphone, an average of 177 apps reported by Shin et al. (2012). In comparison, mobile users develop an accurate mental model of the selection of icons on the homescreen, making it possible to create a usable Imaginary Interface that relies on this memory alone (Gustafson et al., 2011). In the evaluation of the dynamic homescreen proposed by Shin et al. (2012), 'participants stated that apps sometimes appeared and disappeared unexpectedly,' and suggest that 'an effort can be made to minimize the movement of icons that persist between predictions, in order to reduce user distraction' (p. 181). Stability could help to improve the usability of an adaptive homescreen by reducing distraction and by helping users to maintain their mental model.

Stability can be enforced by the adaptive model, as demonstrated by AccessRank (Fitchett & Cockburn, 2012) and Hui et al. (2009). However, as stability acts as a feature within the model, it increases the model complexity. Another approach is to use visual highlighting or shrinking (Tsandilas & Schraefel, 2005), or to animate items gradually fading in (Findlater et al., 2009). To be spatially consistent, these techniques require all items to be displayed at once, which would make icons very small on a mobile device. Shin et al. (2012) explored highlighting on the adaptive homescreen, by indicating the app with the largest increase in probability. This was found to be confusing for participants, especially when the highlighting was inaccurate. Scarr et al. (2013) suggest a simpler approach to stability: 'in situations where content changes slowly, the user would gain the benefits of developing spatial memory; in situations where items change frequently, the user could switch to an alphabetic arrangement (or a list view)' (p. 3147). Ordering apps alphabetically is easy to understand, and helps to increase stability by updating the list only when an item is inserted or removed. We explore this technique in our experiments.

If items are inserted and removed very frequently, then alphabetical order could be a less predictable strategy than displaying the most likely item at the top of the list, i.e. rank order. Therefore, we must evaluate the benefits of stability on the homescreen to determine its value as the adaptive model becomes less easy to understand. This will help interaction designers decide when stability should be included in the design of an adaptive homescreen.

Adaptive Homescreen

The homescreen is the most common way of navigating apps on a smartphone, as found by Hang et al. (2013) in their study of app launching habits. Böhmer et al. (2011) found that users spend an average of 59.23 minutes per day on their device, with app use spread intermittently throughout the day. This presents a different use case to a desktop application, where the system is used in concentrated periods. Furthermore, apps can be installed and uninstalled on a smartphone, changing the range of functions that can be displayed over time (Shin et al., 2012). Mobile devices are used in a variety of contexts, and this also affects the apps that are likely to be used (Böhmer et al., 2011). To make accurate predictions, the adaptive model must update frequently to keep up with the continuously changing context. Therefore, it is important to consider the design of an adaptive menu in a mobile context, and understand how frequent adaptations will affect usability.

Research Aims

We investigate the impact of stability as the adaptive model becomes less easy to understand. The adaptive model controls the movement of apps in the adaptive layout, and we control stability by varying the order: alphabetical order (*alph*) and ordering by the rank of the model (*rank*). Our research question asks: When items are added and removed very frequently, should we increase stability by displaying apps in alphabetical order? Our hypotheses are as follows:

1. Rank order will decrease the perceived predictability of both models;
2. Alphabetical order will improve selection time with both models;
3. Alphabetical order will be preferred overall for both models.

We proceed by describing our research methods, including our stability measurements and data collection. Then we discuss the results of our main usability experiment with 12 participants, and conclude with directions for future work.

RESEARCH METHODS

There are many factors that could affect how a user will perceive stability on the homescreen. For example, one person might perceive an interface to be less stable when the target app moves, whereas another might find it more noticeable when movements surround the target app. This makes it difficult to quantify stability in a single measurement. We adopt a variety of measures to compare the stability of the interface to account for this. In this section, we will define the measures that we use to control stability in our experiment.

Similarity Measurements

List comparison measurements differ in two ways: weightedness and conjointness. In our ranked list of 8 apps, items can be inserted or removed, and so we require non-conjoint measures. Weightedness depends on the user: if she is more likely to look at the start of the list then we should consider a weighted measure, or a non-weighted measure if all positions are equally likely. A weighted measure would be appropriate when we are ordering by rank, and non-weighted when ordering alphabetically.

Learnability (Cockburn et al., 2007) is a stability measurement between 0 and 1 that considers how possible it is to learn the position of items in a menu. This is calculated as 'one minus the average distance that items move as a proportion of half of the total menu length' (p. 629). A stable interface would have a value of 1, whereas an interface that moves items randomly would have a value of 0. We use Learnability to compare the overall stability of our experiment conditions.

We can also compare the distance between two lists using similarity measures. In their review of similarity measures, Webber et al. (2010) identify *Average Overlap* (AO) as a weighted non-conjoint measure, that applies weight to higher ranked items by averaging over matches in list prefixes of length 1 - *k*. *Kendall's tau* is a non-weighted measure, and Fagin et al. (2003) describe a way to transform this into a non-conjoint measurement, by considering a penalty for the case where items exist in one list but not the other. This penalty can be set to 0 to find the minimum distance, *K (min)*, or 0.5 to find the average distance, *K (avg)*. We will use AO and K (min) to compare the similarity between adaptations in our experiment.

We would also like to measure the magnitude of movements in the list. Zhang et al. (2013) define UI difference to be the number of positions apps change between launches. This is a conjoint measurement, as it considers all apps in the app drawer. However, we require a non-conjoint measure that accounts for items that are inserted or deleted. One option is to consider items that do not appear in the top-*k* to have moved to position $k + 1$, an approach taken by Fagin et al. (2003). Another approach is to assign a penalty of half the total menu length, as used by Fitchett and Cockburn (2012) in their calculation of Learnability. Rather than assigning a large penalty to insertions or deletions, we select a penalty of 1. This represents the appearance or disappearance of an item, rather than its movement from the end or middle of the list.

UI Displacement

We define the *UI displacement* to be a non-conjoint measure that assigns items that are inserted or removed to have a penalty of 1. We calculate this to be the mean sum of changes to the position of apps, where a change is:

$$\begin{cases} |(_i - _j)| & \text{if position changed between i, j} \\ 1 & \text{if the app is inserted or removed} \\ 0 & \text{if the app position did not change} \end{cases}$$

We use both weighted and non-weighted UI displacement measures to compare our lists, to account for both orders: alphabetical and rank. The displacement magnitude (*DM*) measures the total movement of items, and the weighted displacement magnitude (*WDM*) increases this value as movements occur towards the start of the list, where they might be more noticeable. WDM follows the approach of AO, but altered to account for the size of displacement in the UI. In an experiment environment, we also know what the next app to be selected will be, and so we can measure the target displacement magnitude (*TDM*). These measurements are defined as follows, with i_t = position of an app at time t, k = the number of apps in the grid, and n = the total number of app launches in the series:

$$\mathrm{DM} = \frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{k}|(\mathrm{i}_t - \mathrm{i}_{t-i})|$$

$$\mathrm{WDM} = \frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{k}\frac{1}{w}\sum_{w=1}^{i}|(\mathrm{w}_t - \mathrm{w}_{t-1})|$$

$$\mathrm{TDM} = \frac{1}{n}\sum_{t=1}^{n}|(\mathrm{target}_t - \mathrm{target}_{t-1})|$$

Table 1 illustrates the use of each of these measurements with an example. These measurements help us to control stability in the conditions of our experiment.

Pilot Study and Data Gathering

To measure the time to select an app, we must record when a user starts to search for an icon and when the icon is selected. We must also eliminate distractions during the search. This requires a controlled experiment. However, it is important to consider how the adaptive homescreen will be used outside a research environment, with a user's own apps. To make the distribution of apps in our experiment dataset less contrived, we chose to use naturally generated data, the same approach as Findlater and McGrenere (2004). We ran a pilot study to gather this data, which was also an opportunity to gain insight into the use of the adaptive homescreen in the wild.

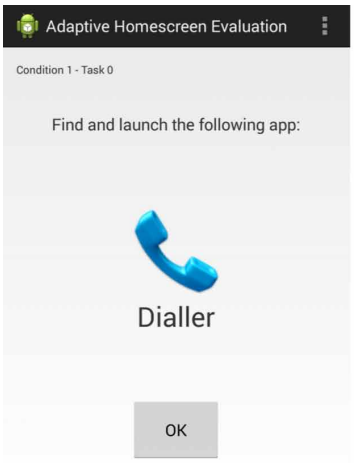
We recorded the launch history of 6 Android 4.0+ smartphone users over a period of 4 weeks. App usage was detected on their personal device every 3 seconds (0.3 Hz), and the name and timestamp of the foreground app was stored in a database, as well as whether it was launched from the widget. Test subjects were aged 27 - 61, all had experience with an Android smartphone for at least 6 months, and two were female. After 2 weeks, our adaptive widget displayed in Figure 1(b) was placed on the homescreen.

All test subjects stated that they rarely arranged apps on their homescreen, and rarely uninstalled any. In the screenshots that we captured before and after we added the widget, the majority of homescreens had at least one duplicate app, which is another indication that there was little interest in arranging app icons. In the feedback, S1 said that he 'tr[ies] to avoid scrolling through the full list of installed apps on my phone as I have lots of them that I never use'. An adaptive homescreen could

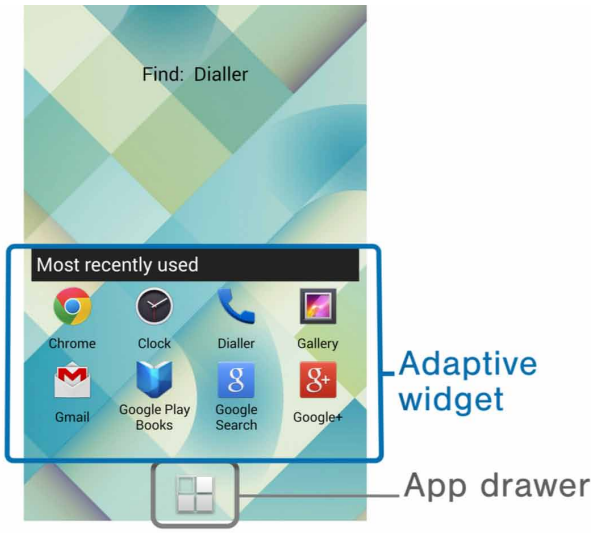
Table 1. An example calculation is demonstrated by launching an app shown in the adaptive homescreen widget in Figure 1(b), which we will consider to be using a most recently used model ordered by rank. If the Dialler is launched at index 2, three apps will change position: Dialler moves up 2 positions to index 0, resulting in a TDM of 2. Chrome and Clock then shuffle down one position each, resulting in a total DM of 4. These movements are weighted towards the start of the list which is more noticeable with rank order, and so the WDM is relatively high.

Measure	Example: Dialler Launched	Result
K (min)	$\frac{1+1}{0.5 \cdot 8 \cdot 7}$	0.07
AO	$\frac{\left(\frac{0}{1}+\frac{1}{2}+\frac{3}{3}+\frac{4}{4}+\ldots+\frac{8}{8}\right)}{8}$	0.8125
Learnability	$1-\frac{\left(\frac{4}{8}\right)}{4}$	0.875
TDM	$\lvert 2-0 \rvert$	2
DM	$\lvert 2-0 \rvert+\lvert 0-1 \rvert+\lvert 1-2 \rvert+(0*5)$	4
WDM	$\frac{\lvert 2-0 \rvert}{1}+\ldots+\frac{\lvert 2-0 \rvert+\lvert 0-1 \rvert+\lvert 1-2 \rvert+(0*5)}{8}$	8.371

Figure 1. The experiment app



(a) The experiment app.



(b) The custom homescreen, with the adaptive widget in the lower section. Above it, a separate widget displays a reminder of the app to select: in this case, 'Find: Dialler'.

assist users who install many apps on their device. Overall, the widget was used for an average of 9% of all app launches in the second 2 weeks (SD=5.7%). Test subjects found it 'to be very helpful' [S1] and they 'want[ed] to keep this widget as I like it' [S2].

An average of 161 apps (SD=46) were installed on the devices, and 33 (SD=8) were actively used over the 4 weeks. This provides evidence that many installed apps were unused. We also found that 16.2 mins per day was spent in apps (SD=5.6 mins), and 8.7 mins on the homescreen (SD=6.3 mins).

An interesting insight from the pilot study was that, prior to installing the widget, average homescreen visits lasted 28s before an app was selected (SD=24.7s), or 8s before turning the screen off (SD=3.3s). Afterwards, average visits lasted only 15.4s (SD=13.8s), or 8.2s before the screen off event (SD=4.5s). Adaptations could have the potential to reduce selection time, by 12.6s in this case. However, the time to select an app from the homescreen can be influenced by interacting with widgets or by external distractions, among other reasons. This means that we are unable to verify timings on the homescreen without a controlled environment, and provides motivation for our experiment.

Though we have chosen to focus on smartphones, one participant of our study also allowed us to record app use on his Nexus 7 tablet. In general, he used it to launch fewer apps. However, he used the widget for 15% more launches on his tablet than on his smartphone. An adaptive widget has the potential to be valuable on a larger device.

Dataset Selection

To make the distribution of apps in our experiment more representative of natural launch behaviour, our dataset contains actual app history that we collected during our pilot study. We selected the history of one test subject to use as the dataset for all participants of our experiment. Using a single dataset allowed us to compare participant selection times. Additionally, we were unable to combine the launch history of multiple test subjects into a dataset as this would disrupt the natural app cycles that may be important for the MFU-n model.

The launch history of S2 was chosen as our dataset since his smartphone usage was most representative of the target user group for an adaptive homescreen: S2 used the most downloaded apps (32 apps), he had 147 apps installed on his device, and he launched an average of 146 apps per day (SD=65). His most common app launches belonged to the Built-in, Social and Communication categories. After we eliminated system apps, such as the Launcher, and launches that were repeated more than 3 times, the app launch history dataset contained 1,275 launches.

RESEARCH DESIGN

We designed a usability experiment to investigate the impact of stability on the homescreen as ease of understanding decreases. The experiment was conducted in a controlled lab environment and lasted 30 minutes. A within-subjects design with four conditions was used: MRU and MFU-*n*, ordered alphabetically and by rank. These were counter-balanced using a Latin square. In this section, we will describe the design of our experiment, including the adaptive models, the adaptive layout, and the experiment task.

Adaptive Models

With even a small number of features, it will become difficult to understand the behaviour of an adaptive model. We compare two models in our experiment which control the selection of apps in the menu. The models depend on features related to app use so that we can control them in a usability study, and they are accurate enough for short experimental conditions:

- **Most Recently Used (*MRU*):** Considers the last time that each app in the history was launched;
- **Most Frequently Used Next (*MFU-n*):** Counts the number of times that each app has been launched after the previously used one, i.e. sequentially used.

We considered Most Recently Used (MRU) to be easier to understand than Most Frequently Used Next (MFU-n), since it is difficult to keep track of which apps are commonly used after each other. In contrast, MFU-*n* has the potential to be more accurate over a longer period of time as it reacts to two pieces of context: launch frequency and the previously used app. These models vary in ease of understanding, and change frequently enough to be noticed in a short lab experiment. However, we did not know how their UI displacements would compare. To calculate the UI displacements, we used the dataset to simulate the adaptive homescreen. This allowed us to verify that MFU-n caused more UI displacements and updated more frequently than MRU.

We found that MRU updates slowly over short-term use. In comparison, MFU-n can display a completely different set of apps after each interaction. We also found a sparsity issue with MFU-n, resulting in an under-populated menu. This means that some apps are followed by fewer than 8 distinct apps in a single session, where a session is the period between turning the screen on and off again. We could avoid this by using a hybrid model: for example, the overall most frequently used apps could fill the remaining spaces. However, this might not be desirable, as fewer apps reduce the clutter of irrelevant apps on the homescreen, which could reduce selection time. In our experiment, we allow for periods of low menu population and investigate the impact of this on usability.

Adaptive Algorithm

The following process occurs each time an app is selected:

1. The model updates its ranking;
2. If the ranking changes, then apps may or may not change position depending on the order;
3. If any app changed position, then the layout is updated.

For example, the rank of MRU updates when a selected app differs to the previously used app. If the rank changed, then the alphabetical order will only update if the most recent app is not already in the layout. Any updates to the UI execute before the user returns to the homescreen.

Adaptive Layout Design

Our adaptive widget is designed as a grid layout, which is available for Android 4.0+ smartphones. This can be seen in relation to the app drawer icon in Figure 1(b). The label above the grid reveals the model that is in use: it states either the name of the model, or for MFU-*n*, a reminder of which app was used previously. Positions in the grid are indexed left-to-right, top-to-bottom, as one would read a page of English text. Though Zhang et al. (2013) found that users do not necessarily read grid items as a list, this is a common approach to grid indexing that is used in the app drawer and that smartphone users are most familiar with. Alternate approaches are possible, such as prioritising items at the edges or the middle.

We placed the widget in the lower section of the screen, where it is comfortable to access icons in case of one-handed interaction (Böhmer & Krüger, 2013b). It also features above the app drawer, which minimises its distance to the full selection of apps, similar to a Split Menu (Sears & Shneiderman, 1994). Though split menus recommend a maximum of 4 items for users to scan quickly, we chose 8 apps, as used in a split interface (Gajos et al., 2008), to increase the accuracy of our simple models to 85%, and to create more opportunities to interact with the widget during our short lab conditions. As a model becomes more accurate, the need for a second row of items may become unnecessary. However, there could still be benefits to increasing the number of items, including to promote stability, or to outweigh the potential cost of entering the app drawer. The impact of varying the number of homescreen apps on accuracy is discussed in more detail by Shin et al. (2012). Though it is also possible to adapt folders or homescreen pages (Böhmer & Krüger, 2013b), we did not consider this since they require additional interactions that increase selection time (Hang et al., 2013).

Participants

The 12 participants were aged 21 - 40, all were smartphone users from a computing science background with no prior experience of our adaptive widget, and one was female. We did not allow test subjects from the pilot study to take part in the experiment.

Equipment

The experiment app was installed on the Nexus 4 smartphone that was used by all participants. 150 apps filled 6 pages of the app drawer, which is close to the average reported by Shin et al. (2012).

Experiment Task

Participants were asked to perform tasks with an adaptive homescreen widget that changes the position of apps in a layout according to our chosen models and orderings. The task was designed as a game, similar to the one proposed by Böhmer and Krüger (2013a). There were 4 sets of 60 selections, using the widget or the app drawer as required. We used a custom homescreen to ensure that apps could only be selected from these two menus, which we implemented with the open source ADW Launcher.

Participants were presented with a task UI that displayed the icon of the app that they should select next. Screenshots of the task UI and adaptive widget are shown in Figure 1. When a participant selected the correct app, the task UI launched to display the next app in the series. If a selection was incorrect, participants were immediately informed by a short notification near the bottom of the screen. To ensure that the widget did not update if an incorrect app was selected, adaptations were controlled using a predetermined list of widget states, which was compiled by simulating the widget with the app launch history dataset.

As the participants of our experiment were required to use another person's apps, our task provides time to become familiar with the target app before each selection task. Between tasks, the 'OK' and hardware 'back' buttons were disabled for 2.5s to enforce a minimum pause.

Participants could familiarise themselves with the icon of the next app to be selected before continuing with the selection task. The selection time for the previous task was also displayed during this period. This break replicates natural app use behaviour, since there is usually a period between clicking on an icon and returning to the homescreen where a user interacts with the app.

Participants were informed that the first 10 selections of each condition were for practice, which left (50*4*12) = 2400 selections to be analysed in our results. We wanted participants to select apps as quickly as possible,so we offered a prize for the user with the lowest overall selection time.

STATISTICAL DESIGN

This section explains how we measured our dependent variables in the experiment, and how we controlled our independent variables by selecting blocks of app launch data for each condition.

Task Dataset Selection

The task dataset comprises 4 blocks of 60 app launches, and each non-overlapping block is used in the selection task. Blocks were chosen from the second half of the dataset that we collected during the pilot study. The whole sequence of app launches leading to the start of each block was used to populate the model, and defined the starting state of the layout for each condition.

To evaluate the impact of stability, we required a noticeable difference in the average number of UI displacements between the conditions. We chose sequences of app launches that fit our criteria by running a simulation of the widget. To ensure that each condition had a high accuracy (approximately 85%), we selected blocks of app launches that had the required number of hits. We controlled stability by selecting a difference in DM of at least 25% between alphabetical and rank order conditions. Our

independent variables are summarised in Table 2 and the comparison of stability in each condition is visualised in Figure 2.

Table 2 shows that the average *target position* in each condition is lower when items are ordered by rank. This is as we would expect, since the most relevant apps are more likely to be near the start of the list, i.e. position 0. The table also shows that the *grid population* is lower on average in the MFU-*n* conditions. This is because fewer than 8 distinct apps can be used after the previous one, as found when we performed the model simulation.

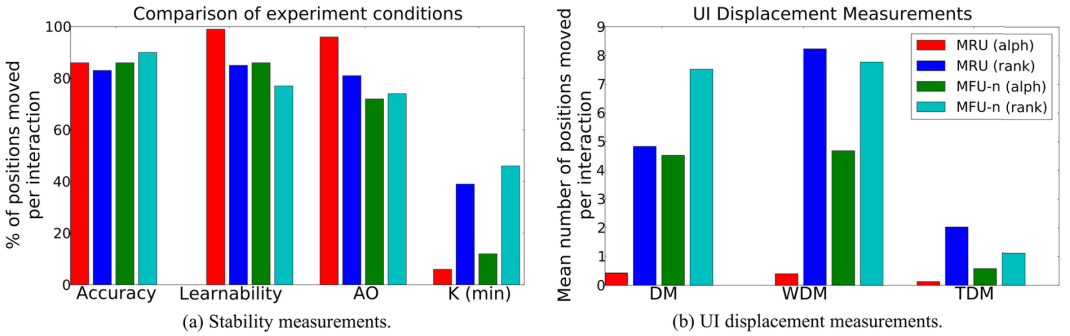
In Figure 2, we see that Learnability is greater than 0 in all conditions, which means that they are more stable than a random algorithm. We can also see that Learnability is higher in the alphabetical conditions compared to rank order, and the UI displacements are lower. Based on these measurements, we are confident that alphabetical order makes our models more stable.

We notice that both orders of MFU-*n* have a similar AO. This means that a similar number of items change position near the start of the list, which might be surprising for alphabetical order. Changes towards the start of the list could increase if the layout is under-populated, and also if a completely different set of apps are displayed after each interaction. If we were to search from the start of the list, the stability of MFU-*n* could appear to be similar for both orders. However, the target position for MFU-*n* (alph) is likely to be somewhere in the middle of the list, and the apps surrounding the target will act as alphabetical landmarks that will influence search in this condition. Therefore, we

Table 2. Summary of experiment conditions, with the mean of measures. The mean target position is lower with rank order, and mean grid population is lower with MFU-n.

	Conditions			
	MRU (alph)	**MRU (rank)**	**MFU-*n* (alph)**	**MFU-*n* (rank)**
Accuracy	0.86	0.83	0.86	0.9
Target position	4.82	0.02	4.10	0.88
Grid population	8	8	6.78	7.23
K (min)	0.06	0.39	0.12	0.46
AO	0.96	0.81	0.72	0.74
Learnability	0.99	0.85	0.86	0.77
TDM	0.13	2.02	0.57	1.12
DM	0.42	4.83	4.52	7.52
WDM	0.40	8.23	4.68	7.77

Figure 2. Alphabetical order is more stable than rank for all measures, except AO



(a) Stability measurements.

(b) UI displacement measurements.

expect K (min) to be more reliable than AO, and that MFU-*n* (alph) will be perceived to be more stable than MFU-*n* (rank).

Task Measurements

For tasks where the app is in the widget, we record the *selection time*. This is measured as the time between clicking the OK or back button to clicking on the correct app icon. For tasks that require the use of the app drawer, we record the *decision time* of when it is opened. This denotes the time that a participant realises the app is not in the widget and that they have to use the app drawer. We also record *errors*, which are any incorrect selections and extra swipes or clicks that are performed.

At the end of each condition, we asked users to rate their subjective opinions on a 7-point Likert scale (1=low, 7=high). After the experiment, we asked participants to rank the conditions in order of preference (1=high, 4=low). We also recorded any comments made by participants.

Statistical Methods

We performed a 2 *(alphabetical vs. rank order)* x 2 *(MRU vs. MFU-n)* ANOVA to test the significance of selection time, with $p < 0.05$. We used a log transformation to control for non-normal distributions in this timed data. A two-tailed Mann-Whitney U-test was used to test our ordinal data, with $p < 0.05$ and $U \leq 37$ selected for our 12 participants. To compare the correlation between selection time and UI displacements, we used the Pearson coefficient, with Evan's (1996) guide for interpreting the r value ($0.2 \leq r \leq 0.39$ for weak correlation).

RESULTS

Table 3 provides a summary of our results. The number of errors made during the experiment were negligible and so we do not report on them. We report the selection time, predictability and overall preference.

Table 3. Summary of results: mean of measured data, median of subjective data and mode of overall preference. The results of our two independent variables (order and ease of understanding) are isolated in the right side of the table by averaging between the two orders and models.

		Conditions				Averaged over Ordering			Averaged over Understandable		
		MRU (alph)	MRU (rank)	MFU-*n* (alph)	MFU-*n* (rank)	Alph.	Rank	*Sig.?*	MRU	MFU-*n*	*Sig.?*
Time	Selection (s)	1.21	1.44	1.45	1.42	1.33	1.43	*	1.33	1.43	*
	Decision (s)	1.46	2.15	1.87	2.04	1.66	2.09		1.80	1.95	
Subjective	Awareness	5.5	7.0	6.5	5.5	5.75	6.0		6.0	6.0	
	Predictable	5.0	7.0	2.5	3.0	3.75	5.0	*	5.75	3.0	*
	Useful	6.0	6.0	4.5	5.0	4.5	5.25		5.75	4.25	*
	Satisfied	5.0	6.0	3.5	4.0	4.25	4.75		5.5	3.75	*
	Efficient	6.0	6.0	4.5	5.0	4.75	5.5		6.0	4.25	*
	Control	5.0	6.0	2.0	3.0	3.5	4.5		5.5	2.5	*
	Frustrating	2.0	2.0	2.5	3.0	2.5	2.5		2.0	3.0	*
	Overall Pref.	1.0	2.0	4.0	3.0	2.5	2.5		1.5	3.5	*

Selection Time

The overall mean selection time was 2.08s (SD=2.7s), and 1.39s (SD=0.9s) using the widget alone. Figure 3(a) compares the widget selection times to the UI displacements. It is clear that the mean selection time increases with the magnitude of the previous UI displacement. Selection time increases above the overall mean selection time when there are 6 or more UI displacements. The Pearson correlation for this effect is weakly significant (Evan, 1996) (r=0.208, p<0.001).

Figure 3(b) displays Boxplots of the widget selection times and decision times. Widget selections were significantly faster in the MRU (alph) condition compared to the others (p<0.001), with a mean selection time of 1.21s (SD=0.71s). Decision time was also faster (1.46s) in this condition, but we would need a larger sample of selections from the app drawer to test if this result is significant.

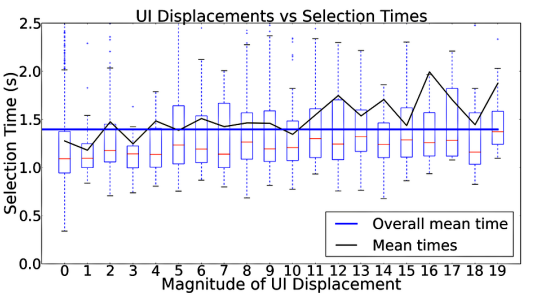
Predictability

The subjective ratings can be viewed in Table 3. As we expected, both MRU conditions were rated more *predictable* than MFU-*n* (U=2.5, p<0.001), and participants felt more in *control* when using this model (U=9.0, p<0.001).

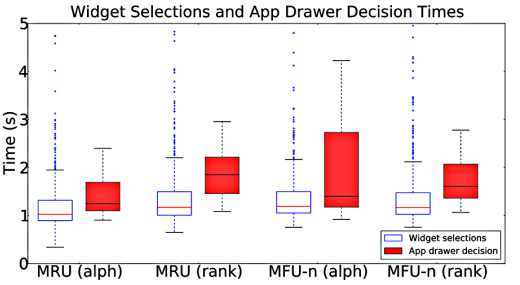
The MFU-*n* conditions were rated significantly lower in *satisfaction* (U=13.0, p<0.001), *usefulness* (U=26.5, p=0.008), *efficiency* (U=30.5, p=0.015) and were more *frustrating* (U=34.5, p=0.029). Despite the MFU-*n* model being as accurate as MRU, participants were 'not sure how it was adapting' and 'didn't like that the homescreen changed unpredictably sometimes' [P9].

The perceived predictability significantly increased when apps were ordered by rank, particularly in the case of MRU (U=29.0, p=0.01), as displayed in Figure 4(b). This was mentioned in the feedback

Figure 3. Selection times

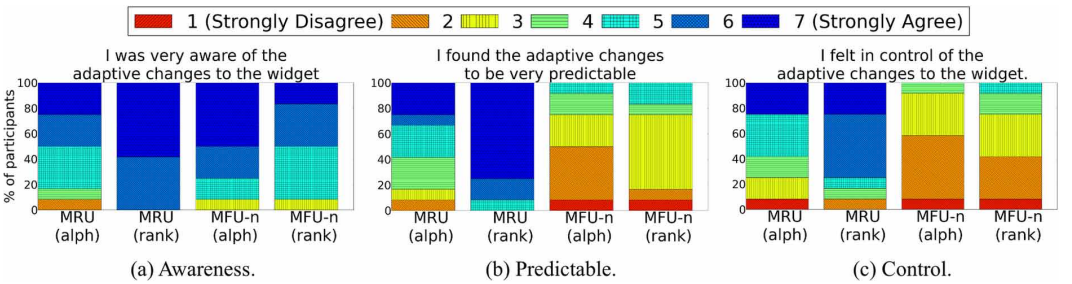


(a) Widget selection times increase with increasing numbers of UI displacements (DM), and is above the overall mean widget selection time (1.39s) when there are 6 or more UI displacements. (Note: this chart is capped at 2.5s to reduce the visual effect of outliers).



(b) Widget selections were significantly faster in MRU (alph) compared to the other conditions.

Figure 4. MRU (rank) is rated significantly greater in: (a) Awareness; (b) Predictability; and (c) Control



(a) Awareness. (b) Predictable. (c) Control.

to the MRU (alph) condition, where one participant commented, 'it was difficult to remember what the least recently used app was [and so] it was difficult to predict which would be dropped when a new app was opened' [P10]. Conversely, another participant noticed that MRU (rank) 'seemed to sort the frequently used ones within reach of my right thumb' [P9]. Ordering MRU by rank did have its drawbacks, as some participants felt that it caused apps to 'move from the end of the widget to the start [and] was quite annoying' [P8]. Additionally, P2 'could [only] predict the adaptive change for apps I had just used but could not remember the last 8 apps used'.

The ratings for the *awareness* of the adaptations are displayed in Figure 4(a). For MRU, participants were significantly more aware of updates when apps were ordered by rank (U=33.0, p=0.017), which is consistent with the measured number of UI displacements. In comparison with MFU-n, participants were more aware of movements when apps were ordered alphabetically, when the measured number of UI displacements was lower. Though this awareness was not significantly more (U=46.5, p=0.124), it is still an interesting result, as it is in conflict with our expectations and it is consistent with the measured AO for MFU-*n*.

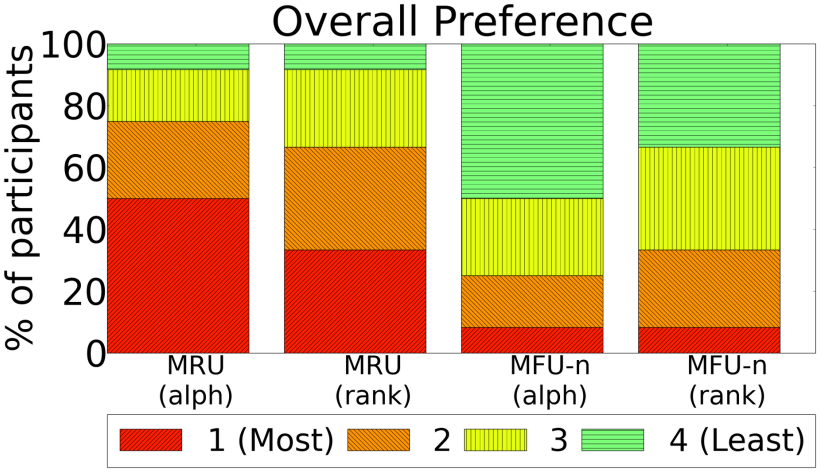
Overall Preference

Figure 5 displays the ratings for overall preference. In addition to being the fastest, participants preferred MRU (alph) overall, with 50% of participants rating this most preferable, and 25% rated this their second preference. This combination of model and ordering was the 'best version of adaptive widget because [it was] very predictable' [P9].

At the lower end of the scale, MFU-*n* (alph) was both the slowest and the least preferred overall, rated least preferable by 50% of participants and third preferable by 25%. In this condition, participants said, 'often I had seen the position of an app just before I was told to press it, however it had then moved' [P2], and 'the widget was not always fully populated and it felt as though major changes (>2 apps) occurred frequently' [P10].

There was no significant preference between MRU (rank) and MFU-*n* (rank). Although MRU (rank) was preferred more than the alphabetical order of MFU-*n* (U=32.5, p=0.018), it was not preferred significantly more than MFU-*n* with rank order (U=40, p=0.056).

Figure 5. MRU (alph) is preferred most overall, and MFU-n (alph) is least preferred



Summary of Results

Our results can be summarised as follows:

1. Rank order improved perceived predictability for both models;
2. Alphabetical order only improved selection time when the model updated infrequently;
3. Alphabetical order was preferred overall only when the model updated infrequently.

DISCUSSION

Our results have several implications for the design of adaptive homescreens, which we highlight in this section:

- **Alphabetical order is effective when updates are infrequent:** Participants preferred alphabetical order for the MRU model, which added and removed items very infrequently. This result is unsurprising since it is the version that is most commonly used for quick-launch menus. Additionally, decision time was lowest with MRU (alph), and highest with MRU (rank), which highlights the effectiveness of alphabetical order when a model updates items infrequently;
- **Rank order is effective when updates are frequent:** Participants were faster and preferred when the MFU-*n* model was ordered by rank, the condition that caused the most UI displacements. This result is contrary to our hypothesis, and the suggestion of Scarr et al. (2013). We believe that it is important to support a search strategy that is easy to understand, even if this strategy does not increase stability. When items update frequently, participants found it easier to look towards the start of the list than to use alphabetical order. This result will inform the design of more reactive, adaptive homescreens. Further work will be required to test if this will generalise to other approaches of stability;
- **Rank order increases predictability:** Our subjective results show that predictability increased when both models were ordered by rank. This result was contrary to our first hypothesis, as we expected alphabetical order to increase predictability by stabilising the interface and allowing items to be indexed by app name. However, rank order was more transparent about which app would be dropped when another was inserted. In comparison, alphabetical order did not reveal the relative value of items, and so items could unpredictably appear and disappear at any position. MFU-*n* was even less predictable when ordered alphabetically, since many items could appear and disappear at once, making the alphabetical landmarks very unstable;
- **Large movements were irritating:** Though participants could understand how apps moved in the layout, some found it irritating when the magnitude of movement was large, particularly when items moved from the end of the menu to the start in MRU (rank), which has a DM of 14. Further work will be required to investigate this effect, particularly as users use their apps intermittently in the wild. A compromise might be to use a stable ranking algorithm, such as AccessRank (Fitchett & Cockburn, 2012), but allow items to move towards the start of the list while minimising large movements;
- **Fewer grid items are easier to recall:** With MRU, participants were unable to recall all 8 recently used apps, and this limited their ability to know whether an app was still in the menu. If a user expects an app to be in the list when it was not, then this could impact selection time and subjective opinion. Fewer items in the menu could reduce this effect, such as the 4 items recommended for a Split Menu (Sears & Shneiderman, 1994). This number of menu items will impact the accuracy of the adaptive model, which should be considered in the design of the adaptive homescreen;
- **Fewer grid items reduce decision time:** The time to decide that an app is not in the layout was faster in both MFU-*n* conditions compared to MRU (rank). We expect that this is because the

MFU-*n* model did not fully populate the layout, as fewer items to check would make this decision easier. The decision time could be faster with rank order if there were fewer items. However, fewer items will reduce the accuracy of the adaptive model. This suggests a trade-off, as an accurate model creates fewer occasions that an app is not in the grid. This trade-off should be considered when selecting the number of items on the homescreen;

- **UI displacements increase selection time:** Selection time increased with the magnitude of the previous UI displacement, and was above average when there were around 6 or more displacements, which was close to the mean DM for MFU-*n* (rank). We found this to be weakly significant, and further work would be required to find this threshold. With an estimate of how frequently the interface can update before negatively affecting performance, practitioners will be able to decide whether alphabetical order will improve the usability of their adaptive menu.

Research Contribution

We summarise the contributions of this work as follows:

- We present an adaptive homescreen widget that displays a selection of app icons;
- We define a non-conjoint similarity measure for comparing UI displacements in an adaptive split menu;
- The results of our experiment show that participants were faster and preferred rank order when the adaptive model updates items frequently.

CONCLUSION AND FUTURE WORK

An adaptive homescreen can help mobile users to make better use of their limited screen space. However, adaptations make the UI less stable and less easy to understand. In this work, we considered the impact of stability on selection time and subjective rating as items are added and removed frequently. We defined UI displacement measurements that help us control stability in a usability experiment. We demonstrate that the combination of the layout order and the dynamics of the model cause stability to decrease. A pilot study allowed us to gather naturally generated data to use in our experiment, and to test our adaptive widget in the wild. We used the UI displacement measurements to analyse this dataset, and to select blocks of app launches for our experiment conditions. Our results show that when items update infrequently, participants were faster and preferred items ordered alphabetically. However, when updates were frequent, participants were faster and preferred rank order. This result will inform the design of more reactive adaptive homescreens.

In our experiment, we only considered simple models of app launch history. As we make UIs increasingly sensitive to context, we decrease the ease of understanding for the user. Future work in this area will involve testing the impact of UI displacements on predictive models. Additionally, as our results are limited to a controlled usability environment, we must evaluate stability in adaptive homescreens with more users, over a longer period of time, and with each users own app launch data. In our pilot study, we found that one participant used the widget for more app launches on his tablet than on his smartphone. The impact of UI displacements on different form factors will be another important area for future work.

ACKNOWLEDGMENT

This work is funded by EPSRC grants EP/P505534/1 and EP/P504937/1.

REFERENCES

Böhmer, M., Hecht, B., Schöning, J., Krüger, A., & Gernot Bauer (2011). Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage. MobileHCI '11 (pp. 47-56).

Böhmer, M., & Krüger, A. (2013a). Gaming the Android OS for improving the design of smartphone launchers. MobileHCI `13.

Böhmer, M., & Krüger, A. (2013b). A study on icon arrangement by smartphone users. *CHI*, *13*, 2137–2146.

Cockburn, A., Gutwin, C., & Greenberg, S. (2007). A predictive model of menu performance. *CHI*, *07*, 627–636.

Evans, J. D. (1996). *Straightforward statistics for the behavioral sciences*. Pacific Grove: Brooks/Cole Pub. Co.

Fagin, R., Kumar, R., & Sivakumar, D. (2003). Comparing top k lists. *SODA*, *03*, 28–36.

Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. *MobiSys*, *10*, 179–194.

Findlater, L., & McGrenere, J. (2004). A comparison of static, adaptive, and adaptable menus. *CHI*, *04*, 89–96.

Findlater, L., & McGrenere, J. (2008). Impact of screensize on performance, awareness, and user satisfaction with adaptive graphical user interfaces. *CHI*, *08*, 1247–1256.

Findlater, L., Moffatt, K., McGrenere, J., & Dawson, J. (2009). Ephemeral adaptation: The use of gradual onset to improve menu selection performance. *CHI*, *09*, 1655–1664.

Fitchett, S., & Cockburn, A. (2012). AccessRank: Predicting what users will do next. *CHI*, *12*, 2239–2242.

Fukazawa, Y., Hara, M., Onogi, M., & Ueno, H. (2009). Automatic mobile menu customization based on user operation history. *MobileHCI*, *09*, 1–4. doi:10.1145/1613858.1613921

Gajos, K. Z., Czerwinski, M., Tan, D. S., & Weld, D. S. (2006). Exploring the Design Space for Adaptive Graphical User Interfaces. *AVI*, *06*, 201–208.

Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., & Weld, D. S. (2008). Predictability and accuracy in adaptive user interfaces. *CHI*, *08*, 1271–1274.

Gustafson, S., Holz, C., & Baudisch, P. (2011). Imaginary phone: Learning imaginary interfaces by transferring spatial memory from a familiar device. *UIST*, *11*, 283–292.

Hang, A., De Luca, A., Hartmann, J., & Hussmann, H. (2013). Oh app, where art thou? on app launching habits of smartphone users. *MobileHCI*, *13*, 392–395.

Hui, B., Partridge, G., & Boutilier, C. (2009). A probabilistic mental model for estimating disruption. *IUI*, *09*, 287–296.

Scarr, J., Cockburn, A., Gutwin, C., & Malacria, S. (2013). Testing the robustness and performance of spatially consistent interfaces. *CHI*, *13*, 3139–3148.

Sears, A., & Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. In ACM Transactions on Computer-Human Interaction (pp. 27–51). doi:10.1145/174630.174632

Shin, C., Hong, J. H., & Dey, A. K. (2012). Understanding and prediction of mobile application usage for smartphones. *UbiComp*, *12*, 173–182. doi:10.1145/2370216.2370243

Tsandilas, T., & Schraefel, M. C. (2005). An empirical assessment of adaptation techniques. *CHI EA*, *05*, 2009–2012.

Webber, W., Moffat, A., & Zobel, J. (2010). A similarity measure for indefinite rankings. In ACM Transactions on Information Systems (pp. 20:1–20:38). doi:10.1145/1852102.1852106

Zhang, C., Ding, X., Chen, G., Huang, K., Ma, X., & Yan, B. (2013). Nihao: A predictive smartphone application launcher. In Mobile Computing, Applications, and Services (pp. 294–313).

Lauren Norrie received her BSci in Computing Science from the University of Glasgow in 2010. As a PhD student in the Inference, Dynamics and Interaction group in the School of Computing Science at the University of Glasgow, her work has focused on inclusive and unobtrusive interaction with mobile devices, and adapting and situating the mobile interface in the context of personal places. In 2015 she was hired as a Software Engineer at Google.

Roderick Murray-Smith received a BEng in Information Engineering and a PhD in Computer Science from the University of Strathclyde in 1990 and 1994, respectively. He has held positions at Daimler Benz Research, MIT, the Technical University of Denmark, Nokia, the Hamilton Institute, and is now Professor of Computing Science in the School of Computing Science in the University of Glasgow. His research interests include machine learning, mobile human-computer interaction, uncertainty in interaction design and sensor-based interaction.