Data-Driven Mobile App Design

Biplab Deka
University of Illinois at Urbana Champaign
deka2@illinois.edu

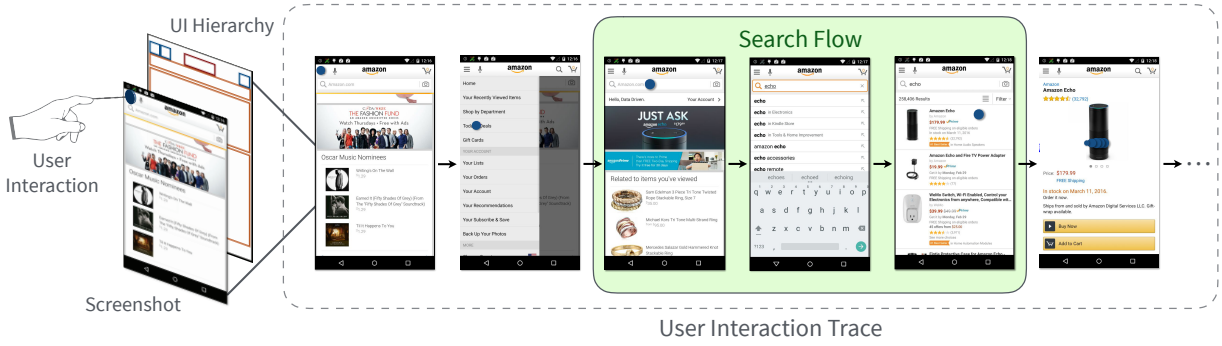

Figure 1: ERICA, a scalable system for *interaction mining* Android applications, captures *interaction traces* as users interact with an Android app. These traces contain snapshots of the app's UI over time (as screenshots and view hierarchies) as well as user interaction data. Here a *search flow* is highlighted within an interaction trace from the Amazon shopping app. The data contained in a trace enables automated identification of such semantically meaningful flows.

ABSTRACT

Design is becoming a key differentiating factor for successful apps in today's crowded app marketplaces. This thesis describes how data-driven approaches can enable useful tools for mobile app design. It presents *interaction mining* – capturing both static (UI layouts, visual details) and dynamic (user flows, motion details) components of an app's design. It develops two approaches for interaction mining existing Android apps and presents three applications enabled by the resultant data: a search engine for user flows, lightweight usability testing at scale, and automated generation of mobile UIs.

ACM Classification Keywords

H.1.2 Models and Principles: User/Machine Systems - Human Factors

Author Keywords

Data-driven design; Interaction Mining; Mobile Apps

INTRODUCTION

With millions of available mobile apps, the user experience afforded by an app is increasingly central to its widespread adoption. Designing mobile apps is a complex, layered process: researchers, designers, and developers must all work together to identify user needs, create user flows, determine layout of UI elements, define their visual and interactive properties, create design prototypes, and evaluate effectiveness of their designs both heuristically and with extensive user testing.

This thesis proposes making app design a more data-driven process by leveraging design data from the vast array of already existing apps. It advocates *interaction mining* – capturing and analyzing both static (UI layouts, visual details) and dynamic (user flows, motion details) components of an application's design. This is in contrast to previous work in design mining that has largely focused on mining static UI layouts and visual details[7, 1].

This thesis presents two approaches for scalable interaction mining existing Android apps: one uses humans to explore an app while seamlessly capturing data in the background, while the other uses an automated agent that leverages knowledge of how humans interact with UIs to perform meaningful interactions. It also presents three applications enabled by interaction mining mobile apps: a search engine for user flows, lightweight usability testing at scale, and automated generation of mobile UIs.

There are three major takeaways from this thesis:

1. *Striking the right balance between capabilities of machines and humans can enable novel interaction mining systems.* Current computational systems are limited in terms of being able to understand how to interact with the variety of

UIST'16 Adjunct October 16-19, 2016, Tokyo, Japan

ACM ISBN 978-1-4503-4531-6/16/10.
DOI: http://dx.doi.org/10.1145/2984751.2984786

UIs present in today's apps. This makes it challenging for automated systems to navigate to different UI states in an app to capture their details. A more tractable approach is to leverage humans for their strength (performing meaningful interactions with UIs) and using machines to capture data in the background.

2. *Combining multiple data streams during interaction mining results in powerful representations that capture both the static and dynamic components of an app's design.* An app's design consists of multiple components: how users move from one UI to another to complete tasks, how UIs are laid out, the visual details of the UIs and their elements, and how each UI responds to user interactions. This thesis demonstrates that it is possible to capture each of these design components by collecting, combining, and correlating multiple types of data, including both visual and structural representations of UIs and their elements, as well as user interaction details. It also demonstrates that this combination of data streams makes it possible to identify semantic meaning in apps.

3. *It is possible to perform interaction mining using a black-box approach.* The interaction mining techniques presented in this thesis do not require any changes to an app's source code. This makes it possible to collect design data from the large number of pre-existing apps. It also lowers the barrier for adoption for designers who want to interaction mine their own apps for applications such as lightweight usability testing.

INTERACTION MINING APPROACHES

Interaction Mining With Humans

ERICA is a web-based system for interaction mining that leverages the cognitive abilities of humans for understanding UIs [3]. As users interact with apps through ERICA, it detects UI changes, seamlessly records UI data (screenshots and view hierarchies) and user interactions in the background, and unifies them into a user interaction trace (Figure 1).

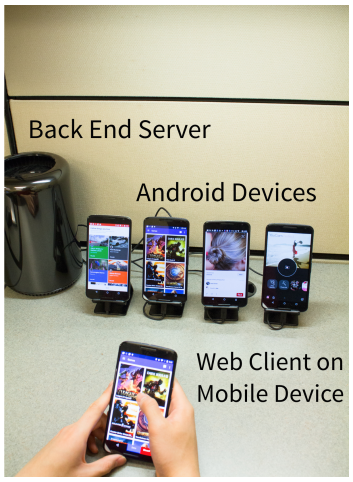

Figure 2: ERICA's data collection infrastructure. Here ERICA's web client is being used in a browser on a phone in fullscreen mode to interact with an app running on one of the other Android devices connected to the server (second from the left). ERICA collects information about the app in real-time while the user is interacting with it.

ERICA's data collection infrastructure consists of three main parts: Android devices that run apps on a modified version of the OS, a back-end server, and a web client (Figure 2). The back-end is responsible for managing the devices and relaying user inputs from the web client to a device and the state of the UI from the device to the web client. The web client is responsible for displaying the apps UI in a browser and letting users interact with it. It has a responsive layout and can be used on devices of varying form factors. On a mobile device, with the browser in full-screen mode, it can offer a near-native experience in using the apps.

The main challenge in building ERICA was being able to capture UI data without any perceived delays to the end user. Common methods for requesting a structured representation of the UI (called a view hierarchy in Android) was too slow ($1-2$ seconds). We modified the Android OS to implement a custom service that responds much faster ($0.1-0.2$ seconds). This service returns a hierarchical representation of the UI in XML format and has properties of all the UI elements (such as their positions and sizes).

Using ERICA over a few weeks, we collected 1186 user traces from 1150 apps with the help of 28 users in a lab setting. These traces contained 18.6K unique UI screens, 52K gestures, 0.5 million interactive elements and 6.7 million total elements. We are currently integrating ERICA with a crowdsourcing platform to build a much larger dataset.

Automated Interaction Mining

MIA is an interaction mining system that utilizes an automated agent to perform interactions on an app's UIs. This automated approach mitigates the main shortcoming of interaction data collected with ERICA – low coverage of an app's UI states since humans tend to mostly use only a handful of UIs in an app during normal usage.

MIA explores an app's UI states by performing the following actions: it captures the view hierarchy of a UI, uses the information in the view hierarchy to identify interactive elements on screen, and interacts with one of those elements. If this changes the UI, the whole process repeats. If not, it performs another interaction on the UI. The main challenge of this approach is that the space of all possible interactions for an app can be very large leading to high runtimes.

MIA uses several optimizations to increase the number of UI states it discovers within a fixed time budget. First, it uses a similarity metric to identify UIs it has visited before and avoids interacting with elements that it has already interacted with on that UI. Second, as it crawls an app, it keeps track of which UIs have unexplored interactive elements. This helps it use a depth-first search strategy (instead of random exploration) that backtracks to UIs with high number of unexplored elements when it encounters a dead end in the search. Third, it captures the information passed to each activity at its start and reuses that information to enable the app to be restarted at any previously seen activity enabling efficient backtracking during app exploration.

There are two additional challenges with an automated approach. First, for many apps, to capture meaningful UI states, user accounts and sometimes user data have to created. Second, many UIs require not a single but a sequence of specific

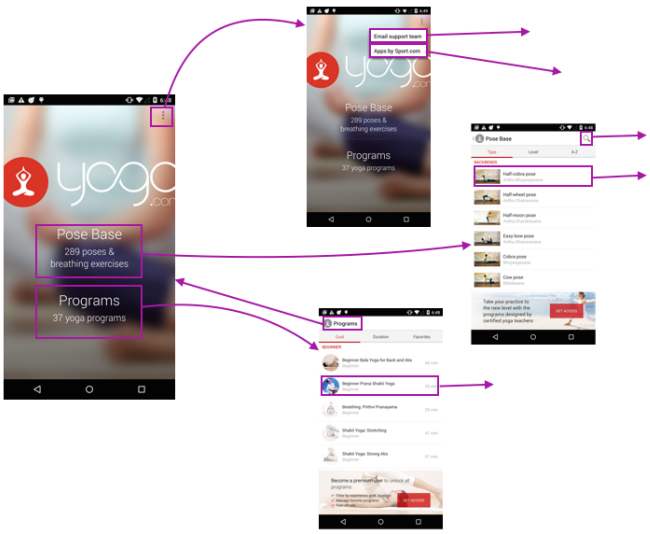

Figure 3: Interaction graphs capture how different user interactions transition an app from one UI to another.

user interactions to transition to the next UI. Not knowing how to perform such interaction sequences limits the number of states that MIA discovers.

We are developing a hybrid system that combines the strengths of ERICA and MIA to address these challenges. To handle logins and user data creation, apps are mined using MIA only after users have used them at least once on ERICA. In addition, the user traces generated by ERICA across a large number of apps are used to model how users interact with different types of common UI screens. MIA then uses these models at runtime for its interactions. Future work in this area will also explore the possibility of using automated interaction agents for UI testing in mobile apps.

REPRESENTATIONS

The data produced by interaction mining can be combined into different representations for supporting multiple tools that scaffold the entire app design process. ERICA combines the three streams of data to produce an *interaction trace* – the sequence of UI states visited by the user during a usage session (Figure 1). It contains information about which element in a UI the user interacted with to get to the next UI and helps visualizes these user actions.

Interaction traces also contain *user flows*: sequences of UIs and associated user interactions that represent semantically meaningful tasks such as *searching* or *composing*. In the UX design process, designers often focus on tasks that users want to accomplish in an app and strive to create intuitive user flows for these tasks.

The information from multiple traces can be combined to produce an *interaction graph* for the entire app (Figure 3) that captures how different UIs in an app are related to each other through user interactions. Such graphs are helpful in understanding the overall organization of an app and visualizing the popularity of different paths in the app. An interaction graph is also useful for an automated crawler like MIA to keep track of UIs that have not been fully explored during a crawl.

APPLICATIONS

Flow Search

User flows play a central role in the UX design process for an app. Like in other areas of design ([5]), designers benefit from having access to examples of existing user flows for design inspiration. This thesis presents a scalable approach for building a searchable repository of user flow examples from interaction traces generated by ERICA.

We leverage the insight that apps are generally designed with visual, textual or structural cues in their elements or layouts that help humans identify different flows. For example, a magnifying glass icon is an indicator for a search flow. To identify potential examples of flows, we identify instances where users have interacted with such indicative elements or layouts.

To demonstrate the feasibility of this approach, we identified examples of 23 common types of flows in our dataset. These were identified by using unsupervised techniques to cluster interactive elements based on visual and textual features and identifying clusters that contained semantically meaningful elements. We built classifiers to automatically identify these elements in interaction traces leading to the identification of more than 3000 flow examples.

We exposed these flow examples through an online search interface (available at `http://www.interactionmining.org`). Users have the option of searching by the name of one these flows or for performing free-form text queries. Search results show the UI that contained the indicative element for the flow in context of the UIs before and after it in the trace. Users can also visualize the UI transitions as animated GIFs.

This approach distinguishes itself from that taken by existing repositories [1] in that there is no a priori burden on users to identify desirable flows, capture them, and upload them to a repository. ERICA mines interaction data from normal, everyday user sessions: the capture happens transparently to the user. Flow identification is done offline. This low burden on the user makes it possible to build a much larger repository of examples than was previously possible.

Lightweight Remote Usability Testing

Asynchronous remote techniques analyze automatically collected user data (such as click streams) for usability testing without the need for a facilitator. They are mainly advised for summative usability testing employed during the later stages of the design process when working prototypes are available [4]. Their key advantage is the ability to collect statistically significant data (on metrics such as time on task or success rates) by using a large group of users [14].

Current tools for conducting such remote testing for mobile apps, however, require source code modifications [10, 12, 11]. In on-going work, we are building a system that leverages interaction mining to enable such testing without any

[1] Such as uxarchive.com or pttrns.com

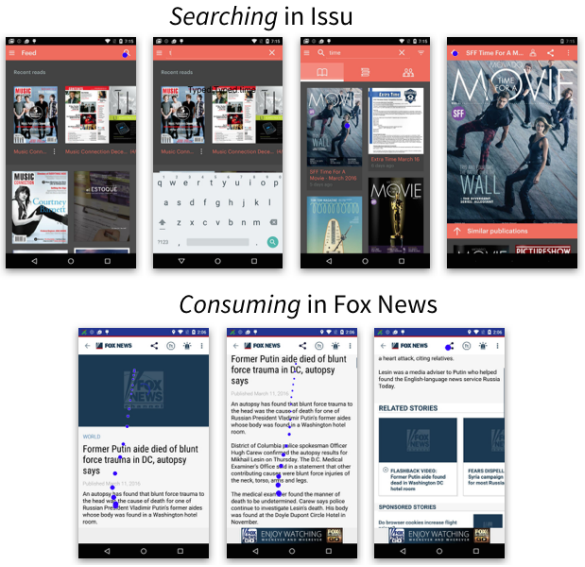

Figure 4: Examples of flows that were automatically identified in interaction traces.

source code modifications. It combines interaction traces from multiple users for the same app and helps visualize the usage data in context of the interaction graph. This approach allows designers to benefit from summative usability testing with lower development overheads. More importantly, it opens up the possibility of conducting usability testing on *any* app – even apps from one's competitors – something that was not possible before.

Automated Mobile UI Generation

Automated UI generation is useful for reducing designer effort. It can also be useful for personalization of UIs [6] or retargeting UIs from other form factors to mobile. To guide the UI generation process towards desirable UIs, conventional UI generation tools either use simple heuristics [9] that limit their effectiveness or designer selected constraints [8] that still needs significant effort from the designer. Recent work has shown that it is possible to leverage existing examples to learn the space of designs in a principled manner and generate new designs [13]. Our future work would explore the possibility of leveraging UI examples produced by interaction mining to model the space of mobile UIs towards building more effective UI generation techniques.

Applications Beyond Design

Another area of future work is to use the data captured by interaction mining techniques for applications beyond design. This includes applications such as indexing and searching on online app stores that currently use metadata provided by developers with their apps [2]. Interaction mining makes available semantic data from within apps which could help improve the performance of such systems.

REFERENCES

1. Alharbi, K., and Yeh, T. Collect, decompile, extract, stats, and diff: Mining design pattern changes in Android apps. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM (2015), 515–524.
2. Chang, S., Dai, P., Hong, L., Sheng, C., Zhang, T., and Chi, E. H. Appgrouper: Knowledge-based interactive clustering tool for app search results. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI '16, ACM (New York, NY, USA, 2016), 348–358.
3. Deka, B., Huang, Z., and Kumar, R. ERICA: Interaction mining mobile apps. In *Proceedings of the 29th Annual ACM Symposium on User Interface Software & Technology*, ACM (2016).
4. Dray, S., and Siegel, D. Remote possibilities?: international usability testing at a distance. *interactions 11*, 2 (2004), 10–17.
5. Eckert, C., Stacey, M., and Earl, C. References to past designs. *Studying designers 5*, 2005, 3–21.
6. Gajos, K. Z., Weld, D. S., and Wobbrock, J. O. Decision-theoretic user interface generation. (2008).
7. Kumar, R., Satyanarayan, A., Torres, C., Lim, M., Ahmad, S., Klemmer, S. R., and Talton, J. O. Webzeitgeist: Design mining the web. In *Proc. CHI*, CHI '13, ACM (New York, NY, USA, 2013), 3083–3092.
8. Lok, S., and Feiner, S. A survey of automated layout techniques for information presentations.
9. Lok, S., Feiner, S., and Ngai, G. Evaluation of visual balance for automated layout. In *Proceedings of the 9th international conference on Intelligent user interfaces*, ACM (2004), 101–108.
10. Ma, X., Yan, B., Chen, G., Zhang, C., Huang, K., Drury, J., and Wang, L. Design and implementation of a toolkit for usability testing of mobile apps. *Mobile Networks and Applications 18*, 1 (2013), 81–97.
11. Google Anlytics: Add Analytics to Your Android App. https://developers.google.com/analytics/devguides/collection/android/v4/.
12. Mixpanel: The most advanced analytics for Android. https://mixpanel.com/android-analytics/.
13. Talton, J., Yang, L., Kumar, R., Lim, M., Goodman, N., and Měch, R. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (2012), 63–74.
14. Tullis, T., Fleischman, S., McNulty, M., Cianchette, C., and Bergel, M.