Project description

Introduction

The world has entered an era of design. Through their phones and tablets, everyday people interact with digital design *constantly*: mobile apps are disrupting every industry --- healthcare, transportation, business, finance, agriculture, education. They serve as vehicles for collecting data and deploying interventions at unprecedented scale, and have catalyzed major technological and social innovations in the last decade.

While tens of billions of dollars are spent on digital design each year, it remains an endeavor mostly based on intuition and opinion, as well as one in which results achieved are only weakly correlated with capital deployed. Why? **The central challenge in design has always been tying the complex and vast space of decisions to measurable outcomes.** A/B and multivariate testing are useful tools for understanding causal relationships between design choices and key performance indicators; however, these techniques require large number of active users to achieve statistical significance in a reasonable amount of time, as well as the engineering resources to build out alternative solutions to test. While companies like Google, Amazon, and Facebook can effectively employ A/B testing given the size of their user base and engineering teams, most organizations are relegated to running inconclusive tests for months on the simple design variations they can afford to build.

This proposal introduces a **platform that automatically captures and aggregates design and interaction data across mobile apps without any code integration, allowing designers to run tests at scale over apps they do not own or did not build.** With the millions of mobile apps available for download today, it is likely that *any* given UX problem a designer encounters has been tackled by a number of existing apps. By leveraging crowd workers and this rich space of extant apps, the proposed platform will allow design teams to cheaply explore and test a wide range of satisficing designs.

The platform allows users to find relevant third-party apps and define performance and usability experiments. It crowdsources these experiments over the apps, automatically capturing design and interaction data during usage, computing specified metrics, and presenting designers with visualizations and comparative analyses that tie design decisions to outcomes. Designers can use these data-driven insights to motivate their design decisions and communicate with stakeholders. In addition, by aggregating recent, equivalent tests together and performing meta-analyses, the platform can go beyond an individual result and present users with accrued design knowledge --- current trends and best practices.

The proposed platform comprises three technical innovations. First, we introduce scalable systems for capturing design and interaction mobile app data streams, and combining them into representations that can be post-processed by machine learning and knowledge discovery techniques. Second, we describe how to develop data-driven models to semantically tag important UI/UX elements in the collected data to support design exploration, and aggregation within and across tests. Third, we sketch interfaces for finding relevant designs, defining experiments and metrics, and presenting aggregated results and predictive insights to users.

Background and Motivation

Throughout the mobile app design process, designers seek to understand the artifacts they build and the experiences those artifacts confer on users. In the exploration phase, designers attempt to evaluate the relative merits of potential satisficing designs. During the execution phase, designers endeavor to detect usability issues before design specifications are sent to the engineering team. Once an app is implemented, designers attempt to optimize user performance metrics and benchmark them against competitors.

To gain this understanding, designers employ a number of testing techniques which generally fall into one of three categories: A/B, usability, and analytics-driven testing. The types of questions that can be answered depend on the *usage context* of the testing technique (natural or scripted), the *types of data* it generates (qualitative or quantitative), and whether it *supports comparisons* with different versions of an app and its competitors (Figure 1).

A/B and Multivariate Testing

A/B testing is used to answer questions of the form "which design performs better?" A typical A/B test might involve dividing user traffic amongst two experimental conditions (A and B), logging usage behavior, and looking for statistically significant differences in key performance indicators (KPIs) such as conversion rate between the two conditions. Multivariate testing, similarly, allows designers to test multiple variables at once to determine if any combination produces an optimal design. The goal of A/B and multivariate testing is to **understand causal relationships between design decisions and business goals**; hence, companies have widely adopted these techniques for *continuously* optimizing digital design, replacing existing designs with new ones if they perform better. The tradeoff is that, while A/B tests can determine *if* one design performs better than another, they cannot help designers understand *why* (King et al. 2017). In addition, A/B and multivariate tests require large user bases to achieve statistically significant results in reasonable amounts of time, and the engineering resources to build out the variations to test.

		A/B & MULTIVARIATE	USABILITY	ANALYTICS
USAGE CONTEXT	Natural	●	○	●
	Scripted	○	●	○
GENERATED DATA	Quantitative	●	◔	●
	Qualitative	○	●	○
COMPARISON TYPE	Own App	●	●	○
	3rd-party Apps	○	◔	○
DESIGN PHASE	Exploration	◔	○	○
	Execution	○	●	○
	Evaluation	●	●	●

Figure 1. A testing technique's usage context, the data it generates, the types of comparisons it enables, and the stages of the design process it supports determine the types of questions it can answer.

Usability Testing

Usability testing studies *how* users perform representative tasks in an app: **whether users can understand a design and use the app as intended to accomplish their goals**. Usability testing helps designers answer questions like "*is* there a usability issue with a particular user interaction flow?" and, if so, "*what* is the issue and *why* does it exist?" Usability testing leverages qualitative data collected through direct behavioral observation (e.g., researchers watching people use the app) and self-reporting (e.g., surveys) to pinpoint usability issues and understand user satisfaction. Unlike A/B tests, usability tests can leverage partially functioning prototypes and as few as 3-5 users (a technique known as *heuristic evaluation*) to identify usability issues with a design (Nielsen & Molich 1990). Accordingly, usability testing is widely used to iteratively refine prototypes before implementing apps. While usability testing *can* involve quantitative measures that capture user performance metrics such as task completion rate and time, achieving reasonable confidence intervals can be costly given that data collection and aggregation is performed manually (Nielsen 2001).

Analytics-Driven Testing

App analytics data is generally geared toward business and marketing intelligence, but it can also be useful for user experience research (Cardello 2013). Since analytics platforms usually log *all* user interaction events, they can answer questions such as "How do users spend their time in the app?" **Understanding characteristics of *natural* usage informs design strategy**, helping designers recognize emergent use cases and prioritize features. Similarly, log data can help identify usability issues on critical conversion paths by capturing points of significant user drop-off, although extracting these insights from large volumes of logs can be challenging for designers without significant technical assistance.

Practical Use Cases and Challenges of Design Testing

These three classes of testing are complementary and can be used in conjunction with one another to assess and improve user experience. For example, usability testing can be employed to understand the *what* and the *why* behind an issue identified via app analytics; after implementing a fix for the issue, A/B testing can confirm that the design fix works.

While each of these techniques provides value to designers and helps elevate the practice of design beyond guesswork and intuition, there is often a gap between what is theoretically possible to test and what is tested in practice. In theory, A/B testing allows users to compare large design variations; in practice, companies test small changes because the engineering effort required to build divergent alternatives will be wasted if the change is not adopted. In theory, usability testing supports quantitative, comparative analysis, allowing designers to benchmark user performance metrics against their competitor's apps; in practice, running usability tests with sufficient participants to achieve statistical significance is prohibitively expensive given the manual effort required to collect and aggregate data.

This gap between theory and practice highlights the central challenge of design testing: **current testing techniques send designers along gradients to local optima**. Existing techniques force focus on small, incremental improvements and fail to put tests in the requisite context to answer **the fundamental question of app design**: "*given the space of possibilities, which solution performs best*?"

Closing the Testing Gap

This lack of global context is particularly unfortunate given the millions of mobile apps across thousands of categories that have been built, released, and tested at scale by users. In this world of ubiquitous interaction design, it is likely that *any* UX problem a designer encounters has already been considered and solved --- perhaps countless times --- by apps that have come before. What is missing is **a way for designers to locate these existing solutions and understand their performance**.

This proposal closes the testing gap by introducing a crowdsourced platform to open-sources mobile app analytics. This platform will allow designers to cheaply collect detailed design and interaction data at scale over Android apps they *do not own* and *did not build*, and correlate this data with *quantitative metrics* and *qualitative feedback*. The platform's goal is to become a self-sustaining resource for design knowledge: aggregating fine-grained design analytics at scale to identify successful trends and best practices.

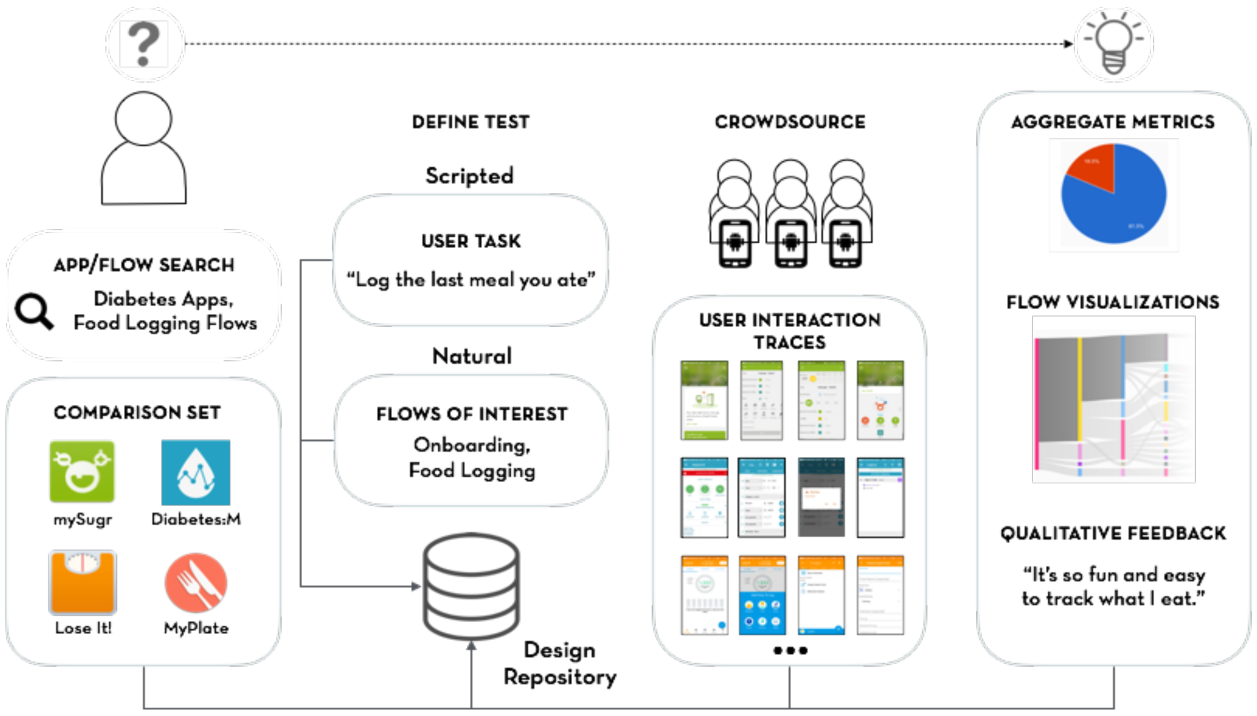


Figure 2. The platform's testing pipeline comprises four stages: creating a comparison set, defining tests, deploying them to crowd workers, and aggregating the results. At each stage, the platform stores generated data (gray boxes) that can inform future design tests and drive data-driven models of design.

While current testing and analytics platforms require source code instrumentation, the proposed platform supports both scripted and in-the-wild testing with *zero* app integration: it requires no access to code, and can be deployed by *any* user over *any* app in the Android store. This somewhat miraculous capability is made possible by a novel data-capture system that inserts itself between a user and an Android app via a browser or the OS, snapshots design and interaction data during usage, and combines the two data streams into a multi-modal representation of user traces. The system **reduces the cost of capturing structured app interaction data by an *order of magnitude***: to around fifty cents a tester from $50 on a usability testing platform.

Designing with Open-sourced Mobile App Testing

Suppose Jane is prototyping a *food logging* flow for a diabetes management app and wants to understand the space of performant and navigable designs. Jane can use the proposed testing and analytics platform (Figure 2) to drive her design process.

First, Jane must find a set of relevant apps to analyze. Using the platform's topical search interface, Jane searches for "diabetes" and finds a number of existing diabetes management apps such as *mySugr* and *Diabetes:M*. Since the search interface also allows her to run queries with functional semantic keywords that describe the tasks supported by an app, Jane also searches for "food logging" and finds food journaling apps such as *Lose It!* and *MyPlate* which also contain food logging flows. This type of scalable, functional search gives designers superpowers, allowing them to **find apps in distal categories that contain relevant design features**.

Based on her search results, Jane creates a *comparison set* containing both diabetes management and food journaling apps to measure user performance on their food logging flows. She defines a *scripted* usability test, where she instructs crowd workers to log the last meal they ate using the app they are assigned from the comparison set. Additionally, she specifies a set of open-ended survey questions to elicit feedback around the task experience. Jane also sets up a second test to collect in-the-wild analytics over a comparison set comprising just the diabetes management apps. By examining *natural* usage, she hopes to learn how often people actually log their meals while tracking their glucose and insulin levels.

After defining the tests, Jane deploys them on the platform. For the scripted tasks, the platform recruits crowd workers from Amazon's Mechanical Turk and allows them to complete the tasks on a device of their choosing through a web app emulator. As the crowd workers perform their chosen tasks, the platform captures design and interaction data streams in the background, and processes them into multimodal representation of user traces. For the in-the-wild tests, the platform engages crowd workers who use the diabetes management apps that interest Jane and have installed the platform's monitoring app on their devices. Through this monitoring app, the platform prompts these crowd workers to participate in Jane's trial, and collects analytics.

Once the tests complete, Jane can review aggregate metrics and visualizations over the collected user traces, as well as the collated qualitative feedback. The performance results from the first test reveal that the meal logging flows found in the food journaling apps have statistically higher *completion rates* than those in diabetes management apps. Jane uses the platform's novel *interactive flow visualization* to quickly identify and inspect interaction traces where users could not finish the task. Jane finds that many of these users had a hard time initiating the food logging flow in the diabetes apps, and notices that these medically-oriented apps have information dense UIs. In contrast, Jane observes that the meal logging UIs in the food journaling apps are more minimal, although the flows themselves involve more steps. Moreover, the qualitative feedback indicates that crowd workers found the task "fun and easy" on the journaling apps.

Usage analytics from Jane's second test reveal that only 5% of the diabetes apps' users record meals on a daily basis. Based on aggregated analytics data from these apps, the platform states with a 95% confidence-level that users spend 83% of their time engaging with the apps' bolus calculators to compute insulin doses, which Jane notices require users to input food they are about to consume. Based on the data from the two tests, Jane decides to build a meal logging flow that is integrated with the insulin treatment flow. She hypothesizes that this strategy will reduce the overall complexity of the app and allow users to derive the benefits of meal tracking automatically whenever they are computing insulin doses.

Scenario Meta-Analysis

This scenario illustrates how understanding the performance of existing solutions can transform the design process. The platform's ability to capture design and interaction analytics over *any* existing app --- and semantically analyze and aggregate behavior over current and previous tests --- allows Jane to make large-scale design decisions backed by data *before* implementing expensive solutions.

The next three sections of the proposal describe the technical innovations necessary to enable the novel data-driven interactions discussed above. First, we describe how the platform can capture interaction and design data from any Android app without requiring source code instrumentation. Second, we describe how to train functional semantic embeddings over the components of mobile app design, which allow the platform to meaningfully classify and aggregate user interactions. Finally, we describe how functional semantics can power novel search, aggregation, and insight generation over the collected user traces.

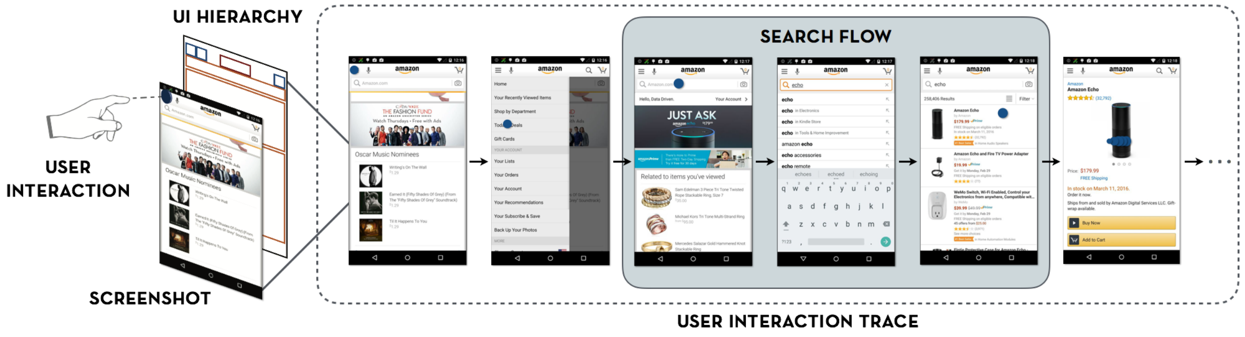


Figure 3. An overview of ERICA, our scalable system for mining design and interactions in mobile apps. As users interact with apps through ERICA, the system detects UI changes, seamlessly records multiple streams of design data (screenshots, view-hierarchies, user events), and unifies them into a user interaction trace.

Zero-Integration Approaches to Crowdsourcing Mobile App Analytics

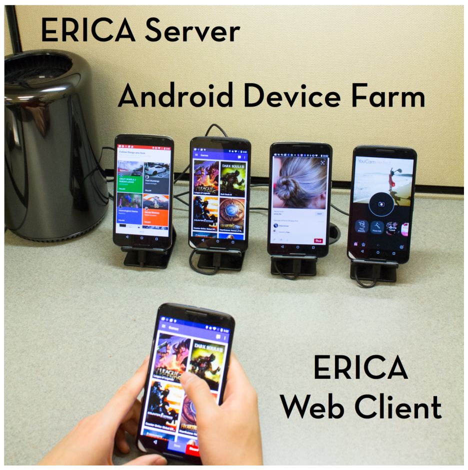
This genesis of this proposal is a scalable system for capturing the diverse, heterogeneous data streams that comprise mobile design, and merging them into representations that can be post-processed by machine learning and knowledge discovery techniques**.** While data-rich domains like text and natural images each comprise one single stream of data (words and pixels, respectively), design is much more complex. Mobile design, for instance, is described simultaneously through structured source code, visual hierarchies, rendered screenshots, and programmed interactions that respond to user behavior. This representational complexity is both a bane and a boon: while mapping between data streams to produce a unified representation can be technically challenging, the different encodings of the same concepts often provide stronger semantic understanding than a single kind of data alone.

Prior Work: Capturing and Representing Design and Interaction Data

Previous attempts to mine design data have analyzed apps both statically and at runtime. Static methods inspect code or binaries, and cannot capture design components such as result lists or feeds that are generated while the app is running (Alharbi & Yeh 2015). Dynamic approaches mine design data at runtime, and explore different states in an app by programmatically interacting with its UIs (Azim & Neamtiu). Dynamic mining agents, however, can be stymied by UIs that require complex interaction sequences or human inputs. Encoding heuristics for handling common UIs can help, but these heuristics are difficult to scale given the diversity of UIs and apps.

Since humans use prior knowledge and contextual information to effortlessly interact with complex UIs, we have developed human-powered exploration to mine design data from mobile apps. Our approach is manifest in ERICA, a system that *enables real-time interaction capture from Android* apps (Deka et al. 2016). As users interact with apps through ERICA, the system detects UI changes, seamlessly records streams of design and interaction data, and unifies them into a user interaction trace. (Figure 3). **The key technical innovation of ERICA is its black-box approach to mining interaction and design in mobile applications, requiring no instrumentation of an app's source code.**

ERICA captures three streams of data: XML files that represent the hierarchy of UI elements (view hierarchies), screenshots captured from the UI at a high rate, and the stream of interactions performed by the



user as she navigates the app. For each interaction event, we record the type such as "tap" or "scroll," and the UI element that reported it. By reconciling these three streams of data, ERICA computes a representation of the user trace composed of a sequence of distinct UI views, where each view is connected to the next through a performed user interaction. Each view comprises a screenshot and visual hierarchy of UI elements that can be queried for render-time properties. Screenshots snapshotted in-between interaction events are also recorded so that animation and motion design data is available for analysis.

Crowd workers interact with apps on the devices of their choice through an ERICA web client. Komarov et al. demonstrated that crowdsourcing is a feasible approach to collecting performance data for user interfaces (Komarov et al. 2013). ERICA's architecture supports scripted tests, where crowd workers are asked to perform specific tasks on apps. When a test is deployed, the platform installs and runs the apps specified in the comparison set on a mobile device farm. The devices running the apps connect to a server hosting the ERICA web client, which continuously streams app screens to crowd workers' browsers. As users interact with the app screens on their browsers, their interactions are sent back to the platform's mobile devices, which perform the interactions on the app (see inset). To minimize bandwidth requirements, the testing platform uses an adaptive frame rate and a high-compression ratio.

Proposed Research: Crowdsourcing In-The-Wild Analytics

To move beyond scripted tests and capture in-the-wild app analytics, we propose to leverage ERICA's base technology to build a background Android monitoring app that continuously logs user interaction and design data in every app that it is granted permission to monitor. Although commercial systems such as App Annie and research systems like PACO (Baxter 2015) log on-device app usage, they capture only high-level user behavior (e.g., what app a user launched). Our monitoring app will wrap ERICA and provide detailed app analytics over *every* interaction a user performs in an app, as well as the complete design state of the app during each interaction. This data is powerful because it enables designers to dynamically query and generate design insights in a post-hoc fashion, asking new questions of user interaction patterns as they aggregate on the platform.

To support this style of crowdsourced testing, we propose to recruit a set of crowd workers through social media advertising and platforms such as UpWork for longer-term engagements than Mechanical Turk tasks. These crowd workers will download and install the ERICA monitoring app, which will track the set of apps that are installed on the workers' devices and notify them of trials they can join. By joining a trial, a crowd worker will agree to have his device usage logged during for the duration of the trial, and be paid a fixed fee for the data that he generates during normal use.

Unlike the ERICA web client, this native monitoring app will not immediately stream data to our servers. Instead, it will require workers to verify that no personally identifying information (PII) is being leaked and approve the data before it is uploaded. While workers who participate in scripted tasks can be provided proxy personal data for PII fields, workers will inevitably use their own data in natural testing. The native monitoring app will use machine learning classifiers to automatically detect PII fields, ignore text entered in them, black them out in screenshots, and allow workers to inspect and remove any part of an interaction traces that contains sensitive data before submitting it for reimbursement.

Functional Semantic Embeddings for Mobile UI/UX components

Designers describe and reason about mobile designs using function-based semantic handles (e.g., *onboarding* flow, *profile* screen, *back* button). To support core capabilities in design search, aggregation, and insight generation, we propose a set of functional semantic embeddings for mobile UI/UX components. Once computed, **these embeddings can be used to semantically classify design components in a user interaction trace, or determine whether two components are similar enough to be merged together when computing aggregate statistics**. Within the user interaction traces collected through our capture systems, we can classify semantic subsequences of screens which represent user flows such as *search*, *login*, *onboarding*, *checkout*, etc. UI screens can share the same labels as the flow they belong to, or have labels that describe specific states in the app such as *home*, *settings*, and *profile*. Similarly, semantic labels for interactive elements can describe the user task they initiate, specific app states they link to, or other self-contained actions (e.g., *like* icons, *back* buttons). To train embeddings for these three components of mobile app design, we can leverage the hierarchical relationships between them and their shared functional semantics.

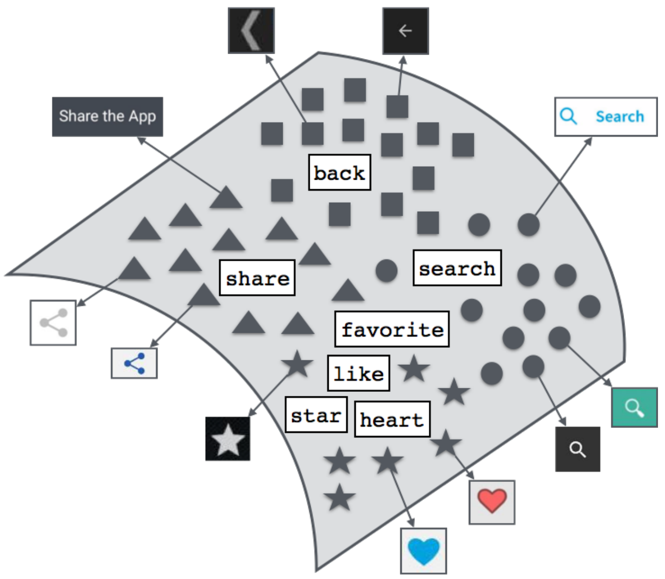


Figure 4. We can train a multi-modal embedding to semantically classify interactive elements found on mobile UI screens. These embeddings can encode similarity relationships between visually distinct elements that serve similar functions, like a button containing the word "search" and the magnifying glass icon.

Prior Work: Creating a Training Dataset to Bootstrap Semantics

To bootstrap these semantic embeddings, we collected an initial set of training data using ERICA. We crowdsourced user traces for 10K popular apps from 26 categories on the Google Play Store: **a collection 50 times larger than the largest preexisting manually-curated repository** (Deka et al. 2017). In a second pass, we recruited 13 participants from Upwork to use each app for 10 minutes and perform typical tasks, generating more than 10K user interaction traces and 72K unique UI screens. We demonstrated that this scale of interaction data is sufficient to support deep learning techniques by training and evaluating an autoencoder for UI layout similarity, and have released the dataset to other researchers in the community.

Proposed Research: Learning Multimodal Embeddings through Design Hierarchies

To build semantic embeddings for flows and screens, we propose to use the same "design scents" that humans employ: visual, textual, and structural cues in interactive elements that communicate which actions those elements enable humans to perform. For instance, a magnifying glass icon is a universal signifier for *search*: when a user encounters an element containing such an icon, she knows to click on it and type in a query to initiate a search flow. Similarly, a large "sign in" button positioned in the middle of the UI signifies a *login* screen. These examples suggest that models that distinguish between classes of interactive elements can inform screen and flow embeddings.

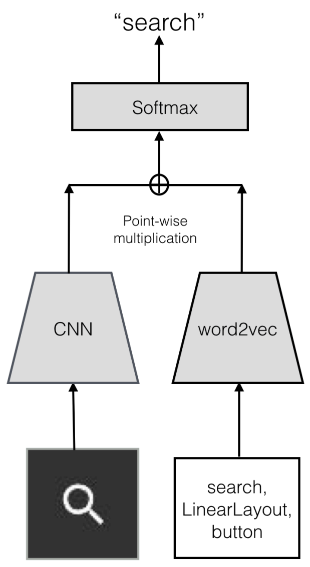
To learn a functional embedding over interactive elements, we intend to leverage the rich representation captured by RICO's user interaction traces, which encodes visual, textual, structural, and interactive information about each UI element. First, we can identify all the elements that users have interacted with in the traces. Next, we can extract visual representations of each element by combining its bounding box with the screenshot of the UI it belongs to. Finally, we can extract `class` and `resource-id` element properties specified by the app creator as well as any text contained within. This textual data often provides semantic clues about the element's functionality, and can function as *weak supervision*: Yi et al. leverage a similar form of weak supervision --- artist's annotations contained in their scene graphs --- to successfully classify semantic parts of 3D models (Yi et al. 2017).

Given this multimodal data, we can train a multimodal embedding on the classification task of predicting the text label of an element given its image representation. Multimodal embeddings are a popular approach for captioning and semantically annotating images, video, and audio (Baltrusaitis 2017). Such embeddings can encode tasks and functions that are similar to each other such as *favorite* and *like*, as well as UI elements that are visually distinct but functionally similar (Figure 4).

Such an embedding over interactive *elements* can in turn inform functional embeddings for *screens* and *flows*. We can train a screen embedding that minimize distances between UIs based on their visual, textual, and interaction similarity. Two login screens from two different apps may be visually distinct; however, in both screens, users would click on the *login* button to proceed to the next screen. Similarly, a flow embedding can learn from both the element and screen embeddings, and minimize the distances between flows that contain similar screens and user interactions.

A key property of these embeddings is that information can propagate in both directions along the hierarchy during training. For example, interactive elements that co-occur on a single screen should not be close together in the element embedding since they are likely to represent different UI functions. Similarly, the flow embedding can generate additional training data for improving the screen embedding: from two nearby flows, we can generate similar screen pairs that were not part of the original training data**. We can formalize this approach of iteratively improving the different models by propagating new training data up and down the design hierarchy as an instance of co-training (Blum and Mitchell 1998).**

Once computed, the testing platform can leverage these functional semantic embeddings to power design search, aggregation, and insights. Given a user interaction trace, the platform will first process the elements that a user interacted with, projecting each element's visual and textual features into the embedded space, element-wise multiplying the vectors, and running them through a softmax classifier trained to output a distribution over functional labels (see inset) (Agrawal et al. 2016). These results can then be fed into a similar process to label each screen in a user interaction trace.



Designers can use these semantic handles to identify relevant sections of arbitrary interaction traces, which will in turn allow them to identify relevant, distal apps for their comparison sets. Similarly, the platform leverage these handles to identify successful trends and best practices: "*onboarding* flows with fewer than 5 screens have the lowest drop-off rates." The platform can also utilize the computed embeddings to support nearest-neighbor "query-by-example" style searches for design. Lastly, the platform can align screens in different user traces based on similarity in the embedding space, showing an app's aggregate user behavior.

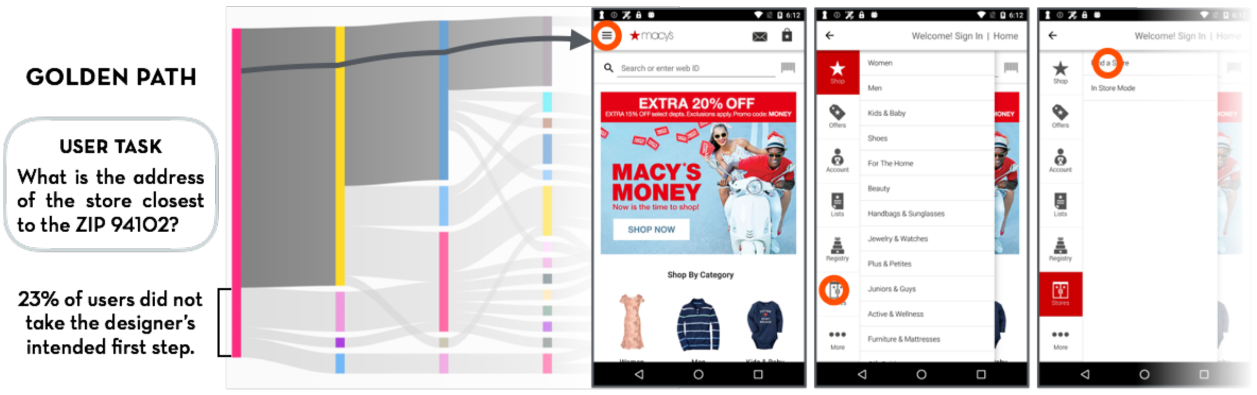

GOLDEN PATH
USER TASK
What is the address of the store closest to the ZIP 94102?
23% of users did not take the designer's intended first step.

Figure 5. Interactive flow visualizations allow users to quickly understand aggregate interaction data, as well as inspect individual traces to identify usability issues. The darker colored edges indicate *golden paths* --- a designer's intended interaction sequence to accomplish a task.

Design Search, Aggregation, and Insights

With a deployed capture system and functional embeddings in hand, we can hope to power a number of novel interactions in search, aggregation, and insight generation. While we have recently created rough prototypes of some aggregation and insight generation interactions to understand the design considerations involved (Deka et al. 2017), we propose to develop and deploy these techniques at scale once we have collected sufficient data to train strong semantic design embeddings.

Finding Relevant Flows

To help designer's find relevant apps to test, the platform will make the Google Play Store searchable through **faceted keyword search over app metadata (name, category, etc.) and design semantics** as well as support **query-by-example search over app components.** The learned embeddings enable both semantic keyword and query-by-example search. In addition to diverse types of queries, the platform can support different views of the results set. Depending on the query, it can present result galleries of app icons, flows, screens, and elements. Since ERICA captures screens at a high frame rate, flows can be presented as sequences of of screenshots or animated gifs. **As part of this proposal, we will identify the searching techiques, query types, and design data views that are most useful to designers.**

Aggregating User Interaction Traces

To help designers compare and contrast the different paths taken by users to accomplish similar tasks, we propose a novel, interactive visualization based on Sankey flow diagrams (Riehmann et al. 2005, Schmidt et al. 2008). Color-coded nodes representing different screens in a user interaction trace can be arranged sequentially along the horizontal axis, and the nodes in the visualization connected by bands whose thickness is directly proportional to the number of users who took the path defined by its node endpoints (Figure 5). To construct these diagrams, the platform must compute the set of screens that are semantically similar in screen embedding space at every interaction step and merge them together into a single node.

These diagrams can allow designers to quickly understand aggregate interaction data, as well as inspect individual traces and screens to identify usability issues. For example, designers can interact with the nodes can to view screenshots and with the bands to see user interaction traces. Similarly, a designer can create a flow by demonstration in an app, and use it to highlight paths of interest in the visualization.

For example, by defining a *golden path* for a task --- the intended path a user should take to complete a task --- and highlighting this path in the diagram, a designer can quickly determine the screens that cause the most confusion and are most likely to prevent users from completing the task.

Generating Summative and Comparative Usability Insights

The platform will also allow designers to compute aggregate performance statistics for usability tests, including both quantitative and qualitative metrics**. For standard quantitative measures like *completion rate*, *error rate, time on task, average number of interactions*, etc., the platform can provide confidence-intervals based on statistical analysis of the collected user traces**. When a designer requires more certainty about user behavior than the platform can provide, new tests can be dynamically generated and deployed, sampling flows that have inadequate coverage.

To calculate qualitative measures like *perceived difficulty* and *user satisfaction*, **the platform can estimate success and error rates based on designer defined end screens**, using the content-agnostic screen embeddings to detect whether the final state of a user trace matches the set of success states specified by the designer. Alternatively, the designer can also construct usability tests with built-in verification, requiring a user to provide an answer to a question in addition to performing a task in the app (e.g., what is the address of the store closest to ZIP 94041).

These measures can also be used for comparative testing: detecting statistically significant differences in the performance of flows that accomplish the same task. Although the designs being compared may be quite different, comparative performance testing can help designers build intuition about the relative effectiveness of design patterns, and set benchmarks for their own apps. A designer who evaluated our prototype usability testing platform mentioned, "*I am generally interested in if these reusable components in different apps share similar usability issues.*" Similarly, another designer said, "*if adding a record for food takes ten second on my competitors app, that would be my target...if I have a set of core tasks that can be matched up with those in other apps then I can understand how well I am doing*" (Deka et al. 2017).

Deriving Design Insights and Best Practices

Because the proposed monitoring app will capture *every* user interaction performed in an app along with its associated design context, designers will be able to query the platform after the fact to derive high-level insights. By leveraging the semantic classification of screens and elements over user traces, the platform can uplevel information about the different tasks users performed within flows. A designer can ask questions like "what percent of app users perform activity X every day?", "on average, how much time does a user spending doing activity X?", and "is there a significant difference in behavior X between these two user cohorts?"

This highlight a key power of the proposed platform: **it enables scalable, precise meta-analyses by combining raw data from prior trials**. Rather than restricting meta-analysis to aggregate data that may not serve to answer a designer's specific question, the platform can leverage all the prior tests that were performed to support one-stage individual participant data (IPD) meta-analyses (Debray et al. 2013).

This ability to aggregate raw testing and interaction data after the fact also makes it possible to identify best practices and design principles in a data-driven way. **By clustering performant traces across apps and performing facet analysis, designers can leverage the platform to identify UX trends and interaction patterns that are provable hallmarks of good design**.

Educational Plan

As an educator, I have three overarching pedagogical goals: developing curricula that integrate design, innovation, and entrepreneurship into computer science; empowering students to begin research early in their academic careers; and leveraging design to increase diversity among students who study computer science.

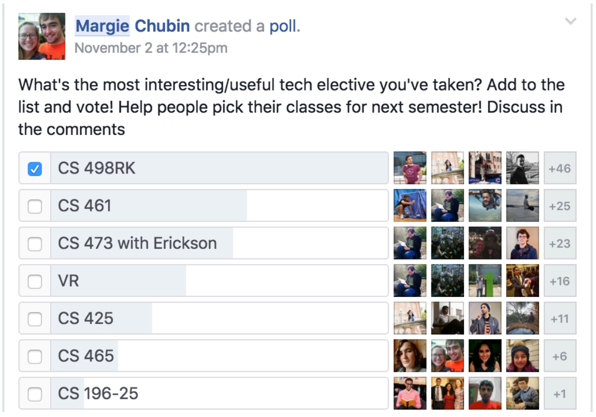
In my first three years at UIUC, I've created and taught three courses in furtherance of these goals: one targeting at first-year undergraduates, the second for upperclassmen, and the third for graduate students. My classes teach practical web and data science skills that are sought after in industry, and offer entry points into research for students who are planning to pursue Master's degrees or PhDs.

In addition, I conceived of and established a joint project between the Computer Science Department and the School of Art + Design called the *Underground Unicorn Program*. The goal of this program is to bring computer science research and product design together for undergraduates in the major. The program is driven by real-world problems in domains that are not often associated with traditional engineering (e.g. fashion, interior design, social media), and accordingly attracts students from non-traditional backgrounds who are interested in designing *solutions* through data science and computation.

Course Development

Data-Driven Design: A course geared towards Master's and PhD students that explores the use of data-driven methods to support creative design processes by examining recent research in human computer-interaction, product design, cognitive science, machine learning, graphics, vision, and natural language processing. Students read and discuss papers from these fields, and work in teams on a multi-week project to build data-driven tools to solve real-world design problems. The course has been offered three times (including the current semester); the average enrollment is 25 students; the curriculum for the current offering can be found here. Several past course projects have turned into publications at top-tier HCI conferences.

The Art of Web Programming: A course geared towards third- and fourth-year undergraduates that presents the client- and server-side technologies that enable modern Web applications. Topics include the building blocks of the Web (HTML, CSS, the Document Object Model, JavaScript) and data exchange (HTTP, JSON, RESTful APIs, and SQL/NoSQL databases). Students work in teams to design, implement, and deploy a full-featured web application. The course has been offered three times, and the curriculum from last semester can be found here. In the past I have capped the enrollment to 100, but **due to popular demand I am scaling the class to 200 students in the fall.** The course is theoretically challenging and time-consuming, but dedicated students often land competitive summer internships and full-time jobs after taking the course (Figure 6). The course also organizes events for current students and course alumni to connect with industry professionals (press).



Margie Chubin created a poll.
November 2 at 12:25pm

What's the most interesting/useful tech elective you've taken? Add to the list and vote! Help people pick their classes for next semester! Discuss in the comments

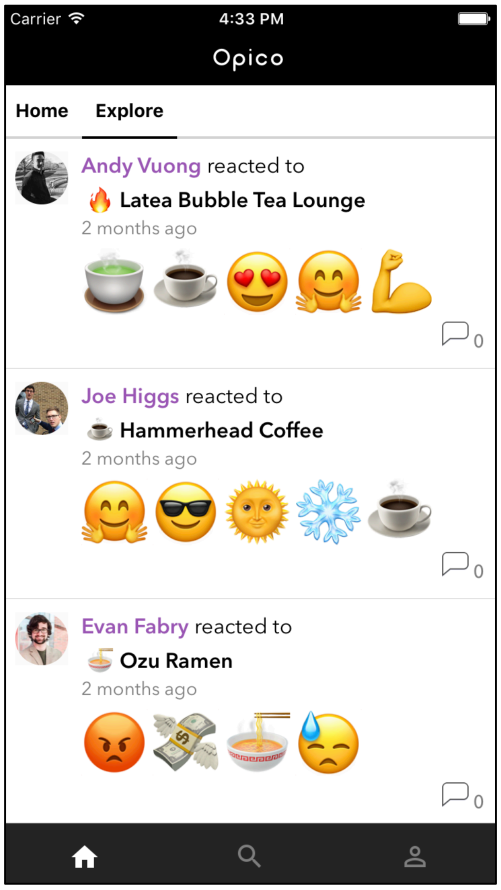
✓ CS 498RK	+46
CS 461	+25
CS 473 with Erickson	+23
VR	+16
CS 425	+11
CS 465	+6
CS 196-25	+1

Figure 6. A student poll with 150+ votes on the UIUC Computer Science Facebook group ranked the Art of Web Programming class (CS498RK) as the most interesting/useful tech elective.

Research with Design, Data, and the Web: An introduction to design thinking, data science, web programming, and research methodologies geared towards freshmen and sophomores. Students read and discuss introductory materials, accessible research literature, and relevant articles from popular media. In addition, students work in teams to frame research questions, and design and prototype solutions. This course was offered once in Spring 2015; out of the ten undergraduates who took the course, six continued working in my research group. **For this class, I was named to UIUC's List of Teachers Ranked as Excellent by their Students.**

The Underground Unicorn Program

I created the Underground Unicorn Program (UUP) with Eric Benson (Associate Professor & Chair of Graphic Design) as a joint project between the Computer Science Department and the School of Art + Design. In the tech industry, unicorns are individuals who can both design and develop: a rare combination that is sought after in industry. **The goal of the UUP is to teach top undergraduates in Computer Science and Design the requisite set of complementary skills to become unicorns.** The UUP is a year-long program: during the first semester, students take sister courses in web/mobile programming and design methods; in the second semester, students enroll in a tech transfer project course where they design, develop, and deploy user-facing products based on research done at the university (Figure 7). **The first UUP class comprised 17 students of which 8 students were from underrepresented minority groups**. Three of the projects resulted in research papers that will be submitted for publication this fall, and all four remain in active development by students who hope to turn the functional prototypes they have developed into legitimate product launches.


Carrier 4:33 PM

Opico

Andy Vuong reacted to
🔥 Latea Bubble Tea Lounge
2 months ago
🍵☕😍🤗💪
0

Joe Higgs reacted to
☕ Hammerhead Coffee
2 months ago
🤗😎🌞❄️☕
0

Evan Fabry reacted to
🍜 Ozu Ramen
2 months ago
😡💸🍜😓
0

Figure 7. Screenshot from an UUP project: an emoji-based review app. The Underground Unicorn students plan to deploy the app this summer and collect data to learn a grammar for emoji communication.

CS + X

As part of UIUC's CS + X initiative to create interdisciplinary majors that marry a background in computer science with disciplines in the arts and sciences, I am collaborating with colleagues to develop a curriculum for CS + Art and Design. The curriculum will place a strong emphasis on studio practice, and combine fundamental courses in design methods, type, imaging, and digital interaction with computer science courses in data structures, web programming, human computer interaction, data mining, data visualization, and applied machine learning. One of the original motivations for developing the UUP was to create a lightweight testbed for understanding how to transfer knowledge between computer science and design.

Broader Impacts

Once built, the testing platform will serve as a self-sustaining resource for designers, developers, digital marketers, UX researchers, and educators in the design and HCI communities. **We intend for the platform to democratize design testing, dramatically lowering the cost of drawing data-driven inferences about design tasks, and making these inferences economically feasible for teams with limited access to engineering resources, user research, or large preexisting user bases**. I will also integrate this platform into the web & mobile app course curriculum at my institution and make it available to other educators to train students in the emergent field of computational design.

The First Design Resource of its Kind

In his 2017 Design Tech Report, John Maeda --- former president of RISD and venture partner at Kleiner Perkins Caufield & Byers --- writes, "Design isn't just about beauty; it's about market relevance and meaningful results" (Maeda 2017). **We hope that, by developing techniques for *explicitly* tying design decisions to business goals, more companies will invest in better design, resulting in better products and smoother interactions for their customers.** The goal of this testing platform is to make data-driven design accessible and economical, enabling industry professionals to more efficiently produce higher-quality designs where success can be measured concretely and communicated to key stakeholders.

In 2013, I co-founded a data-driven design company --- Apropose --- based on my dissertation work on mining web designs (Kumar et al. 2013), and raised $2.3M in seed funding from top venture capital firms Andreessen Horowitz and New Enterprise Associates. The inspiration for this proposal came from dozens of first-hand interviews with digital design agencies and the design wings of tech companies I conducted while serving as Apropose's Chief Scientist. Time and time again, we found that once a company was locked into a particular design, they were heavily disincentivized from considering divergent alternatives, and instead continuously optimized through small changes and A/B testing. **We intend this platform to allow designers to cheaply explore and validate large-scale design changes with data for the first-time.**

The economic argument for open-sourced design has two facets. First, it is inexpensive to recruit crowd workers who are paid simply to use their devices as they normally would: the pool of available workers is much larger than the set of Amazon's Mechanical Turkers. Second, each time a designer pays for a test, she is *implicitly* contributing data to hundreds of related tests, multiplying the effect of every dollar spent.

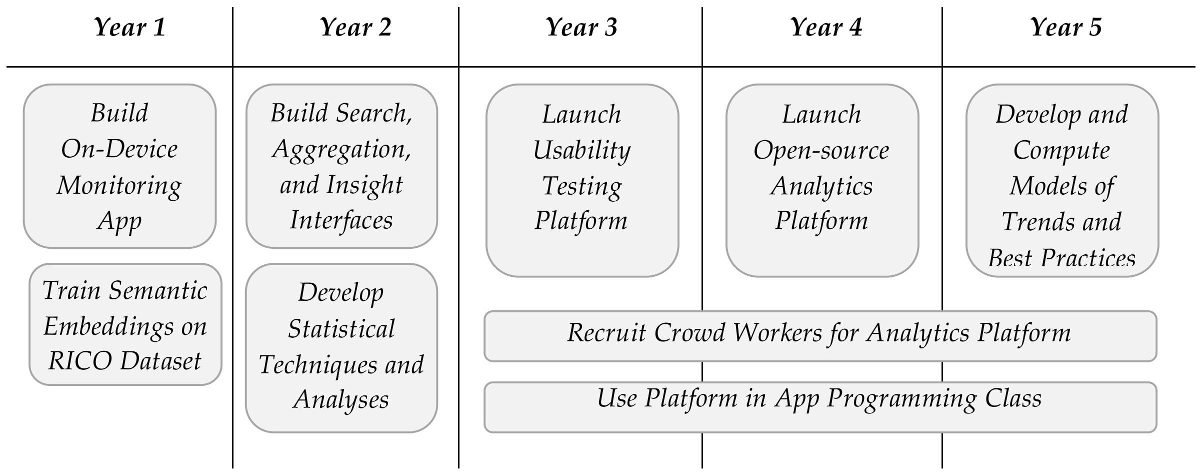
Curricular Integration

We also intend for the platform outlined in this proposal to serve as a valuable resource for educators teaching data-driven design. At UIUC, I am in the process of transitioning the *web* programming course into an *app* programming course, where the last third of the semester will focus on mobile app design and programming. The testing platform will serve as a teaching tool for user experience and interaction design, supporting students as they conduct comparative usability tests to optimize the user flows they design and build.

Illinois' Graphics Design Department is also interested in using the platform for its design search capabilities: design instructors often find it difficult to find illustrative examples for concepts that they are teaching in class. Similarly, the UUP teams building mobile apps can leverage the platform during exploration and testing. In fact, one of the UUP projects we have planned for the spring semester is building the on-device monitoring app itself, which will allow other UUP teams to "dogfood" the app and give feedback as it is being built.

Broadly, we believe that this platform is essential for training the next generation of designers. John Maeda propounds that *computational* design is the future of design: “designing for billions of individual people and in realtime” (Maeda 2017). For computational designers to have impact at this scale, it is essential for them to be able to leverage data-driven techniques to understand how design decisions are tied to business outcomes.

Timeline and Deliverables


Year 1
Year 2
Year 3
Year 4
Year 5
Build On-Device Monitoring App
Build Search, Aggregation, and Insight Interfaces
Launch Usability Testing Platform
Launch Open-source Analytics Platform
Develop and Compute Models of Trends and Best Practices
Train Semantic Embeddings on RICO Dataset
Develop Statistical Techniques and Analyses
Recruit Crowd Workers for Analytics Platform
Use Platform in App Programming Class

Results From Prior NSF Support

The PI has no previous NSF support.