AT-EASE: A Tool for Early and Quick Usability Evaluation of Smartphone Application

Wei Yang
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
Email: wei.yang@ncsu.edu

Xiao Yu
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
Email: xyu10@ncsu.edu

Abstract—We present our tool AT-EASE: A Tool for Early and Quick Usability Evaluation. The tool can help mobile app designers simulate their low-fidelity prototypes on real mobile devices. Designers are able to perform iterative design based on the usability data provided by AT-EASE. To enable a more realistic simulation regarding screen navigation, AT-EASE provides multiple ways to override the default behavior that just shows a screen to another. To provide meaningful usability evaluation to designers, we design a set of experimental metrics derived from the raw data collected by AT-EASE. We demonstrate the basic usage of AT-EASE, and present two experiments with four human subjects on two Android apps to show the effectiveness of AT-EASE. The experiment results show some possible correlations between the usability problems observed on real apps and revealed during the AT-EASE study.

I. INTRODUCTION

The growing usage of smartphones in many scenarios is projected to overtake the usage of desktop computers in the very near future [1]. Lower costs and advancements in the hardware capabilities of smartphones have led to a decisive shift in numerous application areas such as email, social networking, entertainment, and e-commerce [1], [2]. This trend has led to an explosive growth of the number of mobile applications being developed. As of April 2012, both Apple and Google claimed that there were more than 800,000 third-party apps on their online markets, respectively [3]. Given the scale of online markets, users typically face the choice between the apps with similar functionality. Thus developers need to develop high quality apps in order to be competitive. In contrast to desktop software, smartphone apps have a much broader customer base, and most of apps are frequently used in daily life. This makes the usability more critical in the app quality. As a result, companies are realizing the benefits of designing and developing products in a usability-oriented manner rather than a purely functionality-oriented manner.

The improvement of usability should be considered in the every stage of mobile apps’ life cycle. A suggested practice is the iterative design with user involvement. Gould et al. [4] proposed an iterative methodology where user should be involved with prototype in the design process to uncover problems, then designer should fixed these problems, and allowed user to test the prototype again. Such methodology is also applicable to smartphone apps. On mobile devices, the limited size of screens usually requires the balance between different design choices and compromises that affect the behaviors of reading, navigation, input, etc. Thus, the iterative design is essential to ensure the usability of mobile apps.

Testing and evaluating the mobile app usability demand attention to the real devices and location aware usage scenarios. Ideally, testing should be performed on multiple devices by a wide variety of users in all kinds of usage scenarios. Field study, lab experiments, and hands-on measurement all require apps running on real devices [5]. So usability testing normally is carried out after the completion of development or near the end of development, so that users can be provided with a complete product as app under testing (AUT) to test in real usage scenarios.

However, performing usability testing after the completion of development will cause iterative design to be impractical in real mobile development setting. First, as the competition is fierce on mobile-app markets, it’s very important for a team to release their product before their competitors. The limited development time makes the iterative design infeasible. Second, mobile apps are usually developed in relatively small-scale projects which may not be able to support extensive and expensive usability testing. Above all, even if the team could afford usability testing and they indeed get the feedback from the testing, a slight modification of the design may trigger a lot of changes in the implementation and testing. Every iteration in the iterative design will requires a significant mount of efforts and time to prune, finesse and perfect things which initially seemed to be less important. In other words, the later the usability defects can be detected, the higher the costs of fixing such defects will be, the less likely the iterative design could be achieved.

Another problem of performing usability testing after completion of the development is that it cannot provide enough evidence to root the cause of usability defects. As the product provided to carry out usability testing is the combination of design and implementation, there would be many factors affect the testing results. The usability defects detected can be caused by the oversight of design, the flaws in the implementation or the limitation of underlying infrastructure. The developers and
designers have to work together and spend significant efforts to identify the possible root causes of these usability defects.

In addition to the dilemma between iterative design and the need for real testing context, the existing usability techniques are not adequate for mobile apps. Traditional guidelines and methods used in usability testing of desktop applications may not be directly applicable to a mobile environment. Usability testing of mobile applications should take into account the characteristics of mobile devices, such as portability, location awareness, and accessibility. Ideally, the usability testing should cover most usage scenarios in a mobile environment. However, previous research showed that traditional techniques used in usability testing, including controlled laboratory experiments and field studies, have various limitations in a mobile environment, such as ignoring mobile context or insufficient procedural control.

In this paper, we present a tool called AT-EASE to facilitate the early detection of usability defects of smartphone apps on real devices by using low-fidelity prototypes. Our tool can alleviate the challenges mentioned above of state-of-art usability testing of mobile app. First, our tool provides usability evaluation at design phase for designers. Designers can leverage our tool to compare the usability of competitor’s products against their own products to make critical design decisions. Such information is obtained in the early phase of software development, thus has the advantage of reducing time and costs. Second, our tool segregates the design from the implementation of the app under development. Designers can import the design of apps into our tool, and make users try the design out live on mobile devices, without having to code anything. By doing this, the usability testing results will not be affected by side effects of implementation and designers can focus only on their design to locate and fix the root cause of usability issues. Last but not least, we incorporate mobile context in the usability testing by implementing the tool as an Android app that can simulate semi-real behaviors of the AUT on real devices.

Specifically, this paper makes the following contributions:

- A quick usability evaluation method using low-fidelity prototypes to assist iterative design of mobile apps in the early phase of development.
- A set of usability metrics that can measure the usability of smartphone apps.
- A tool implementing this method to perform usability evaluation as a smartphone app on real Android device.
- An empirical study evaluating this tool on several real-world Android apps for demonstrating its efficacy at providing accurate usability evaluation result.

II. BACKGROUND

A. Low-fidelity prototype

Low-fidelity (lo-fi) prototyping is drawing the design of a software’s GUI on either on pieces of paper physically or on computers virtually, and testing them with real users. Previous research suggested that prototyping can be used in obtaining usability data during the UI designing phase. We use the lo-fi prototypes (essentially a group of pictures of design) as the input to our tool because it helps bring results early in development. The advantages of lo-fi prototyping makes our design-phase usability testing more promising than usability testing after completion of products. First, the lo-fi prototypes are easy to build and easy to change. This will allow more iterations in design process, thus will bring more improvement. Second, using more complex artifact will affect testing process and result. The bug in product or even hi-fi prototype by programming may halt the usability test. As we previously mentioned, the detected usability defects could also caused by mistakes in programming, and using picture of design will get rid of such problem. Last, developer team may resist drastic changes in design after spending enough efforts in programming, while our approach allow them make important changes before programming.

The current practice of usability testing using lo-fi prototype involve significant human efforts. These practice generally have four roles: real users, observers, the human computer, the facilitator. Real users are the ones who use the tool in the test. They are asked to “think aloud” along the testing to give context to observers. Observers are the ones who observe and interpret what user said (the users’ interactions). The human computer, as its name suggests, is simulating the computer’s behavior. He/she should remain silent and react to the users’ interactions to provide feedback. Facilitator is the one to pre-process and document the issues in testing. Such approach require human labor but cannot provide real user experience, while AT-EASE tool could serve as all roles involved except real users and provide an almost same user experience as running the app on real device.

Computer-based and paper-based low-fidelity prototypes are two types of widely used lo-fi prototypes, and our AT-EASE tool can work on both types. Figure shows both types of prototypes. Previous studies have compared these two types of lo-fi prototype and suggested that the detected usability defects will not be affected by types of lo-fi prototypes being used. Sefelin et al. further concluded that the human subjects involved are likely to give comments and suggestions to computer-based lo-fi prototype. Based on such result, we believe that designers are likely to use computer-based prototype instead of the paper-based one, so we implement our approach as the AT-EASE tool to enable computer-based lo-fi prototypes.

B. Usability Testing of Mobile Application

Our work falls under the general category of usability testing of mobile application. Ryu proposed a systematic approach to develop usability questionnaires for electronic mobile products. Specifically, the purpose of usability evaluation based on such questionnaires is to assist the decisions making among constantly changing competitions on the end-user markets. Gafni invented a set of usability metrics specifically for mobile context. They carried out four different experiments to validate these metrics. Ma et al. created a toolkit to
automatically log interface events from users. The logged data are used for usability evaluation. However, their tool can only be used after the completion of apps’ development, which makes iterative design unaffordable. Waterson et al. \[18\] conducted a pilot study aiming at comparing the laboratory-based experiment with remote log-based analysis tool on mobile apps. They found out while remote log-based tool showed its advantage in getting content-related usability data, the lab-based testing can capture more issues related to devices. Zhang et al. \[7\] highlighted the existing techniques and challenges of usability testing of mobile apps, and extracted a general process for conducting such testing. They also argued that the laboratory-based testing is more suitable on standalone mobile devices that don’t require network connection.

C. Wireframing

The idea of our tool comes from a commonly used web-design methodology called wireframing. Wireframing is referred to a set of drawings that show the basic navigation and layouts of a website without concrete contents involved. Wireframing allows designers to easily design and demonstrate the hierarchy and layout of each web page based on their intentions on how page elements work together to guide users fetching displayed information. The motivation of wireframing is to help designers document their designs, and evaluate the usability of navigation on the website. The outcome of wireframings could be a reference for designers and other team members to ensure the completeness of navigation and layout features on the website.

On the other hand, the expressiveness of wireframing limits designers from directly leveraging it to design mobile apps. Although wireframing is useful in determining the interactions between users and the displayed screen, it is usually unable to capture the transitions and animations affected by the persistent data states across different screens. Such transitions and animations are very common on Android and other mainstream mobile platforms. For example, the user interaction on one screen depends on the user inputs from other screens. Our tool alleviates this expressiveness limitation by introducing different kinds of overrides for the default navigation between screens.

\[1\] From http://thegrid.soup.io/tag/Wireframe

III. Using AT-EASE: An Example

Figure 2 describes how designer can make use of AT-EASE to carry out iterative design. The app designer will first express his or her design idea on lo-fi prototypes. The lo-fi prototypes are basically image files, which can be imported into AT-EASE tool. On AT-EASE tool, we provide a screen editor to help designer further enrich the prototypes with screen transitions. After finishing editing, the prototypes become operable, i.e., a set of image files that can react to user actions, and simulate app’s behaviors. As user interact with the operable prototypes, AT-EASE will log user behaviors, and collect usability data. AT-EASE matches our predefined usability metrics with collected usability data to get the evaluation results. The app designer can spot the usability defects suggested in evaluation results, and make necessary modifications to the original design. After redesigning, the modified lo-fi prototypes can be used for next iteration.

We use Figure 3 to further illustrate how screen editor helps designer simulate the behaviors of the app under developing. The image in the screen editor on Figure 3 is one screen from an Android task app Any.Do \[19\]. The designer could add rectangle by button ’+Rect’ to circle out the operable GUI component. In Figure 3 buttons “Share”, “Remind”, “Add”, and “Done” have been circled out by rectangles. The operable components can be deleted by using “-Rect”.

For each rectangle on the screen, if the action on the corresponding GUI component will trigger a screen transition, i.e., jump to a different screen interface, we can add link of another image to the rectangle by “Link” button to
simulate the behavior. Also, designer can link screens with slight changes to present the animation effects which are very common on mobile apps. To alleviate the work of circling out all components, designer can use "Import" button to import specified rectangles from another similar screen, and make slight modifications. After all the edition to the screen is done, the designer can tap "Finish" button to exit the screen editor.

It is worth being noted that, on Figure 3, the designer didn’t circle out all the actionable components on the soft keyboard. This is mainly because these components will not bring a conspicuous change on the screen, or designer have not designed screens or actions related to these components. Still, if needed, designer can later review the data collected by AT-EASE to identify the possible actions on the unspecified components. Our experiment shows that the evaluation results will not be affected if the components unrelated to usability task are not circled out.

![Fig. 3: The Screen Editor in AT-EASE](image)

**IV. IMPLEMENTATION HIGHLIGHTS OF AT-EASE**

In this section, we present two important features that enable the prototype simulation and analysis in AT-EASE: the semi-realistic simulation for app behaviors, and the event logging for usability analysis.

**A. Simulating App Behaviors**

In previous section, we mentioned how designers can indicate the transitions among different screens so that AT-EASE can simulate the behaviors of the app under development. Specifying those transitions is only one side of the story. By default, the specified transitions can only react to one type of actions named as forward actions [20]. The other type of actions is back action triggered by the back button or other backward operations on Android devices. Those back actions are usually determined by the *back stack* in Android, but developers can provide app-specific back behaviors to override the default behavior.

The AT-EASE tool provides different ways to simulate the default back behavior as well as the app-specific ones. On Android devices, the back stack is managed at the level of activity. According to the Android Developer Guide [21], “When the user presses the Back button, the current activity is popped from the top of the stack (the activity is destroyed) and the previous activity resumes (the previous state of its UI is restored)”. However, in AT-EASE, the navigation is conducted at the image level, and the relation between activities and images are one-to-many, i.e., several different image files could come from the same activity of the app under development. AT-EASE maintains a back stack at image level, and incorporates the behaviors of back stack in Android. To resolve the issue caused by one-to-many relation between activities and images, we propose three solutions for designers to better simulate different back behaviors in addition to the default one. These solutions are built upon an reasonable assumption that in Android developers would likely not change the default state-saving behavior, in which the state of the activity that is being switched out would be automatically saved.

- **Animation effect.** AT-EASE requires designers to name the images in the same activity with the same prefix. By doing this, the image for animation effect will not be pushed into the back stack directly. Instead, it will be used to replace the top of the stack. For example, a screen Example.jpg contains animation effects when opening a menu. Another screen that can illustrate the expected animation effects will be named as Example.openmenu.jpg. Under such circumstances, Example.openmenu.jpg will be displayed, and the top of the back stack, Example.jpg in this case, will be replaced.

- **Single top.** This solution simulates the behavior similar to "singleTop" launch mode in Android, which essentially means if an activity at the top of the back stack will be revisited, instead of place another instance of the activity on the top of the back stack, Android will route back to the existing instance. For instance, when a screen SomeTask.jpg that is initiated by Example.jpg is being closed by a GUI component whose transition points back to Example.jpg, Example.jpg will be displayed, but it will not be pushed on the back stack again.

- **Persistent data change.** This solution aims to solve the scenario happened when persistent changes have being made on activity B will substantially changed the appearance of activity A. There are two cases for users to navigate between activity A and B: either press the button provided on activity B, or press the back button to go back to activity A. To handle the first case, for activity A, AT-EASE requires two image files that represent the states before and after displaying activity B, respectively. The two files must have the same root name,
for instance, A.before.jpg and A.after.jpg. In activity B, the GUI component that links back to activity A should go for A.after.jpg. In such case, A.after.jpg will be displayed, and A.before.jpg on the back stack will be replaced. To handle the second case, AT-EASE introduces an override for the back button. It requires designers to name the image file for this back button override as [Root name of the screen backing to].BF – [Name of the screen backing from]. For example, a screen Task.jpg links to another screen SubTask.jpg. When the user presses the back button from screen SubTask.jpg, the content on Taskjpg may need to be changed. By using our name convention, when the user presses the back button, instead of backing to Task.jpg, Task.BF-SubTask.jpg will be displayed if it exists, and Task.jpg on the back stack will be replaced as well.

B. Logging Events

To provide accurate data for usability evaluation, AT-EASE records every touch on the screen when the user is interacting with the simulated screens. Particularly, AT-EASE collects three kinds of raw data from user touches, and derives more sophisticated metrics from these raw data. The raw data consist of the coordination and the time stamp of each touch, with the name of the screen on which the user was operating. The coordination and time stamp are obtained by implementing the onTouchListener interface to capture touch events.

Given the raw data, AT-EASE derives several metrics that are important to usability evaluation. Combining the coordination of touch and the location of GUI component, AT-EASE can determine whether a touch hit the GUI component (a valid touch) or not (an invalid touch). Besides, given a series of coordination and time stamps, AT-EASE can calculate the distance and the time interval between two touches. Designers may use the distance between touches to examine the layout of a screen, and use the time interval to examine whether a user can easily identify a specific sub task on a screen. Combining with the navigation between screens, designers may infer the route to all visited screens to examine the rationality of a specific task flow.

V. EVALUATION

A. Evaluation Setup

We carried out two isolated experiments to evaluate the effectiveness of AT-EASE in usability evaluation. The first experiment asked human subjects to perform a pre-designed task on two Android apps that are similar in functionality, and to answer two questionnaires for the two apps, respectively. The second experiment asked subjects to complete the same task by using the prototypes of the two apps simulated by AT-EASE. If the usability problems revealed from the second experiment correlate with the real usability problems that are encountered by the subjects in the first experiment, we will have high confidence that AT-EASE can likely help designers identify usability problems earlier.

We selected four human subjects that were separated into two groups, and each group was assigned only one of the two experiments. For each experiment, we asked subjects to complete an “adding task” task flow on two different apps (Any.Do [19] and Astrid [22]), respectively. Both of the experiments were conducted on a Nexus 7 Tablet.

For both experiments, each subject was given 5 minutes to go over a guideline. The guideline contains a rough task flow that is eight steps as follows. Each step represents a sub task to be accomplished to reach the final goal.

- **Add task.** Add a task whose name is ”Add a task”.
- **Set task priority.** Set the priority to red color (high priority).
- **Set task due date.** Set the due date as April 26.
- **Set a task reminder.** Set the reminding time as ”6 p.m.”.
- **Set the task recurrence.** Set the recurrence as ”every 1 month”.
- **Create a folder.** Create a list/folder ”new folder” if not exist.
- **Set the task to the folder.** Set the task to the list/folder ”new folder”.
- **Add a note to the task.** Add the note/description “add a note” to the task.

In the first experiment, we sit besides subjects to observe the reactions that subjects would have, such as complaining the apps. Once the subject completed or gave up the task, he or she was given two questionnaires to fill out for each app. We also checked whether the task was completed or not. The questionnaire evaluated the usability of the app from four aspects: attractiveness, understandability, learnability, and operability.

First, to evaluate the attractiveness, we asked subjects whether they feel the interaction or display tedious. By *tedious interaction* we mean if subjects feel too many unnecessary interaction in one task. By *tedious display* we mean if subjects need to navigate through many screens when completing the task.

Second, we elaborated three metrics to measure understandability and learnability: *display burden, clarity of operation possibilities,* and *feedback consciousness*. Subjects should rate the display burden by their feeling towards the crowdedness of items displayed on one screen. The clarity of operation possibilities measures whether subjects were aware of their current location in a task flow and could easily identify the right action to be performed on the screen. The feedback consciousness, as its name suggests, judges whether subjects could easily identify an action was successfully completed or not.

Third, operability is evaluated based on the following entries on the questionnaire: *completeness of operation possibility measures* if there are sufficient operable choices on each screen; *ease of input entering* is judged by whether there were frequent uses of auto filling, and fewer places required input entering when subjects were completing the task; *ease of manipulation* evaluates whether subjects felt easy to tap on desired the UI component; *ease of action sequence* measures
whether subjects felt easy to perform a sequence of actions in a task.

In the second experiment, we first asked human subjects to complete a small training task on some simulated app screens, in order to let them be familiar with AT-EASE. After that, the subjects were asked to complete the task flow, as the subjects in the first experiment did, but on simulated screens instead of real apps. Meanwhile, AT-EASE recorded all subject actions on the screen as usability data. In addition, to judge whether and how the task was completed, we manually created some expected action sequences using AT-EASE. Once we collected all usability data, we derived state machines of action sequences, and compared the subjects’ state machines against the expected ones to reveal potential usability problems.

B. Experiment Results

1) Experiment on real apps: Table I shows the results from the experiment using real apps. Two subjects carried out the same task four times on the two apps, and only subject B gave up when he or she failed to complete one of the subtasks. However, for other three executions of the task, all of them failed to complete one sub task, which means that the subjects were not aware of the incompletion of the task. Notably, the uncompleted task is the same for the three executions. They all performed incorrectly in the step “set the task to the folder”. It is worth being noted that such finding is not shown on the feedback consciousness entry of the questionnaire because subjects were unaware of the problem when filling out the questionnaire. Such situation likely indicates a potential usability problem regarding the set folder feature in both apps. Without checking whether the folder has been set or not, subjects are unable to know the completion of the task. Subjects are still very likely to spot the problem in future use. This also expose the limitation of the questionnaire-based study that questionnaire may not reflect all usability problems occurred during the study. It can only reflect the subjects’ feeling by a short experiencing of an unfamiliar app. Subjects themselves may not notice some of the problems, and just pass such ignorance to questionnaire and the whole study result.

Table II shows the evaluation result extracted from the questionnaires. For each entry on the questionnaire, subjects can give ratings from 3 to -3 to express their feelings towards corresponding usability metrics. We place the sum of the rating into Table II for every corresponding entries. Overall the evaluation from questionnaires are valid. For example, from the result we can see that the ratings of Clarity of Operation Possibilities are very poor, which is consistent with the performance of all subjects, and the subjects’ complain on "incapability of finding the place to add a new folder”.

2) AT-EASE based usability evaluation: We present the action sequences for another two subjects, C and D, and the standard one of accomplishing the task on two apps in state machines form on Figure 4 and Figure 5. The state machines decompose the eight steps of task flow into a smaller granularity based on the screens of different functionality, and reflect the deviations between subjects’ action sequences and our standard ones. The lack of some states on subjects’ sequences indicate the incompletion of some sub tasks, whereas the extra states or multiple transitions among some states could indicate wrong actions towards completing specific sub tasks.

Table III and Table IV present the usability data collected and derived from AT-EASE. Those data indicate possible
(a) Standard action sequence  
(b) Subject C’s action sequence  
(c) Subject D’s action sequence  

Fig. 4: Action sequences to accomplish task on Any.Do in state machine form

(a) Standard action sequence  
(b) Subject C’s action sequence  
(c) Subject D’s action sequence  

Fig. 5: Action sequences to accomplish task on Astrid in state machine form
correlations between the results of AT-EASE and questionnaire in three dimensions.

First, for attractiveness, we can find that for both Subject C and Subject D, the valid/invalid touches on Any.Do are more than Astrid. This correlates with the questionnaire entry No Tedious Interaction. The same principle also applies to No Tedious Display, the backward/forward steps taken on Any.Do are more than Astrid.

Second, for understandability and learnability, the Clarity of Operation Possibilities entry on the questionnaire correlates to our results. Subject D had given up during the task completion on Any.do. And Subject C’s correct flow ratio on Astrid is higher than his or her correct flow ratio on Any.do. That means the Astrid shows more clarity when subjects need to consider the next operation.

Third, for operability, AT-EASE shows consistency at Ease of Action Sequence entry on the questionnaire. Except unfinished execution of Subject D, both Subject C and the standard execution show that the average touch distance of Any.Do is longer than the average touch distance of Astrid. It is a reasonable assumption that longer distance between subsequent taps will bring difficulties to perform an action sequence to the designed task.

C. Threats to Validity

We carried out a real-app based study to compare against AT-EASE’s results. However, the study results based on the questionnaire have a lot of limitations. The subjects may not be objective enough when filling the questionnaire. We observed that during the real-app study on Any.Do, some subjects showed frustration of being unable to find a way to perform the task. This could not only decrease the subjects’ rating to related usability problem, in the meantime, it could also affect the usability metrics which may not even related to their frustration.

Another major threat is that we only conducted the experiments on four subjects. The generality could be undermined, since with limited subjects the study may not be able to provide accurate results. We are planning to conduct the experiments on more human subjects in future.

VI. CONCLUSION

In this paper, we propose a quick usability evaluation method using low-fidelity prototypes to assist iterative design of mobile apps in the early phase of development. To better assist our evaluation methodology, we provide a set of usability metrics that can measure the usability of smartphone apps. We have implemented such methodology into a tool to perform the usability evaluation as a smartphone app on real Android device. At last, we provide an empirical study evaluating this tool on several real-world Android apps for demonstrating its efficacy at providing accurate usability evaluation result.

REFERENCES