2009 IEEE/ACM International Conference on Automated Software Engineering

A Methodology and Framework to Simplify Usability Analysis of Mobile Applications

Florence Balagtas-Fernandez
Media Informatics Group
University of Munich, Germany
Email: florence.balagtas@ifi.lmu.de

Heinrich Hussmann
Media Informatics Group
University of Munich, Germany
Email: heinrich.hussmann@ifi.lmu.de

***Abstract*—Usability analysis is an important step in software development in order to improve certain aspects of the system. However, it is often a challenge especially when it comes to evaluating applications running on mobile devices because of the restrictions posed by the device and the lack of supporting tools and software available to collect the necessary usability data. This paper proposes a methodology and framework to aid developers in preparing the mobile system for usability analysis. The focus is on the simplification of the developer's task in preparing the system for evaluation and the processing of the collected usability data by automating some of the tasks involved in the process.**

I. Introduction

Performing usability studies for applications running on mobile devices can be a difficult task. Typical user study scenarios collect information by means of attaching an external camera to capture a view of the mobile screen [4][7] or through logging [8]. Using an external camera to view the mobile device's screen is quite challenging because of the fact that the screen is small and most of the time the user is occluding the screen [3]. An alternative to this is to use screen capture software similar to the ones available for the desktop. However, because of limitations [12] posed by mobile devices, it is quite a challenge to find such applications that can accurately and efficiently capture user interaction with the mobile applications being tested. Logging of events on the other hand can be an accurate source of usage information. The challenge in event logging though is in the whole process of preparing the system for data collection and the extraction and interpretation of the vast amount of logged data. It would be useful to have tools that can accurately process these low-level data and give visualizations of the information that would help in the usability analysis of mobile applications.

In this paper, we would like to present a methodology and framework that aims to ease the task of the developers and analysts in the usability analysis of mobile applications. The framework we have developed only supports applications that use the Android[1] platform as of the moment. However, the set of APIs can be easily extended or ported to support other platforms as well.

II. Usability Evaluation of Mobile Applications

The goal of usability evaluation is to find out possible usability problems of a system and discover ways to resolve these problems. Some related literature [12][11] that focus particularly in evaluating usability of mobile applications provide possible steps to guide researchers in conducting their user studies. However, the guidelines discussed are very generic and most of the details discussed were more on the administrative side of the usability evaluation such as the preparation of equipment, questionnaires, etc. Detailed steps on the technical part (e.g. programming, processing low-level data) are not elaborately discussed.

In this research, we focus on the more technical side of the usability evaluation of mobile applications. We want to identify the problems encountered during the whole process and to find ways to ease the developer of the manual tasks involved.

Figure 1 shows a typical set of tasks performed by the developer, which can be grouped into four phases: preparation, collection, extraction and analysis.

The *preparation phase* involves setting up the application prototypes to enable logging of information necessary for usability evaluation. Because of the diversity of mobile platforms, it is difficult to find a general set of Application Programming Interfaces (APIs) that can be used to perform such logging tasks. Some mobile platforms such as Android have APIs for logging[2] (i.e. class Logger). However, the developer still has to come up with the appropriate formatting for the logs in order to easily port it to other applications for processing later on and also to make sure that the logs do not take up a lot of space in the device. In the *collection phase*, the task for the evaluators (who may be the developers themselves or another person who may be a usability analyst) is to make sure that the system was able to collect the necessary data. The *extraction phase* consists of extracting all the logged data during the collection phase and

[1]http://code.google.com/android/

[2]http://code.google.com/android/documentation.html

1527-1366/09 $29.00 © 2009 IEEE
DOI 10.1109/ASE.2009.12

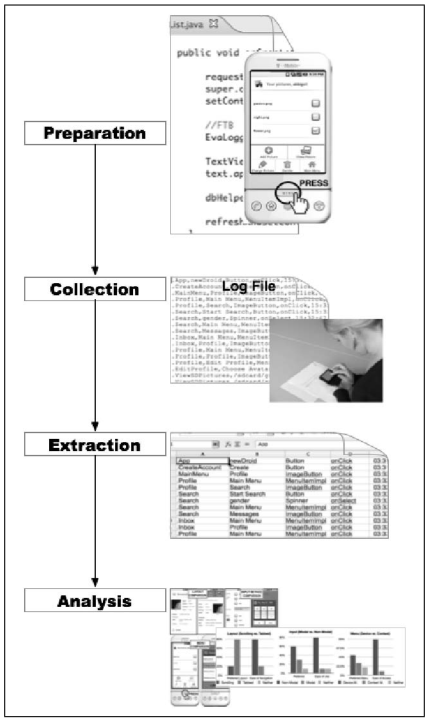


Figure 1. Typical developer tasks

porting the data to other applications for later analysis. The tasks in the extraction phase can be challenging especially if vast amount of information has to be processed. The *analysis phase* involves taking the processed information from the extraction phase and analyzing which parts of the system the users had difficulty interacting with and what can be done to improve this. Different tools [8][6][10] are available to aid the simplification of analysis of the system, but not many exist for usability analysis of mobile applications. In the next section, we will discuss our proposed approach in order to solve some of the problems mentioned during the four phases.

III. EVAHELPER FRAMEWORK: AN AID TO THE MOBILE APPLICATION DEVELOPER

In this section, we will discuss about our proposed methodology to guide and ease the developer of the tasks involved during the different phases mentioned in the previous section. As a proof of concept, we are going to introduce the EvaHelper (Evaluation Helper) framework which applies the concepts proposed.

A. Preparation Phase: Deciding Which Information to Log

Zhang and Adipat [12] carried out an extensive survey of researches focusing on the usability evaluation of mobile applications and extracted the most common questions these researches were trying to address. The authors provided a list of usability attributes (e.g learnability, efficiency, memorability, etc.) and the variables used to measure these attributes

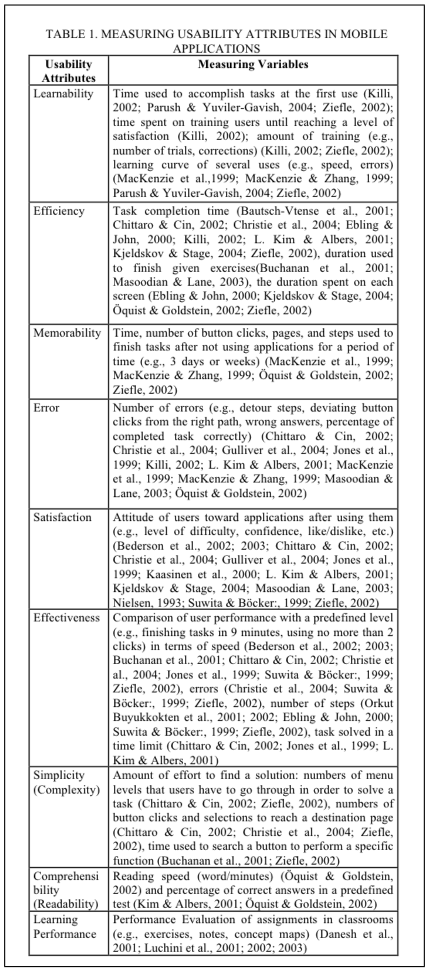


TABLE 1. MEASURING USABILITY ATTRIBUTES IN MOBILE APPLICATIONS

Usability Attributes	**Measuring Variables**
Learnability	Time used to accomplish tasks at the first use (Killi, 2002; Parush & Yuviler-Gavish, 2004; Ziefle, 2002); time spent on training users until reaching a level of satisfaction (Killi, 2002); amount of training (e.g., number of trials, corrections) (Killi, 2002; Ziefle, 2002); learning curve of several uses (e.g., speed, errors) (MacKenzie et al.,1999; MacKenzie & Zhang, 1999; Parush & Yuviler-Gavish, 2004; Ziefle, 2002)
Efficiency	Task completion time (Bautsch-Vtense et al., 2001; Chittaro & Cin, 2002; Christie et al., 2004; Ebling & John, 2000; Killi, 2002; L. Kim & Albers, 2001; Kjeldskov & Stage, 2004; Ziefle, 2002), duration used to finish given exercises(Buchanan et al., 2001; Masoodian & Lane, 2003), the duration spent on each screen (Ebling & John, 2000; Kjeldskov & Stage, 2004; Öquist & Goldstein, 2002; Ziefle, 2002)
Memorability	Time, number of button clicks, pages, and steps used to finish tasks after not using applications for a period of time (e.g., 3 days or weeks) (MacKenzie et al., 1999; MacKenzie & Zhang, 1999; Öquist & Goldstein, 2002; Ziefle, 2002)
Error	Number of errors (e.g., detour steps, deviating button clicks from the right path, wrong answers, percentage of completed task correctly) (Chittaro & Cin, 2002; Christie et al., 2004; Gulliver et al., 2004; Jones et al., 1999; Killi, 2002; L. Kim & Albers, 2001; MacKenzie et al., 1999; MacKenzie & Zhang, 1999; Masoodian & Lane, 2003; Öquist & Goldstein, 2002)
Satisfaction	Attitude of users toward applications after using them (e.g., level of difficulty, confidence, like/dislike, etc.) (Bederson et al., 2002; 2003; Chittaro & Cin, 2002; Christie et al., 2004; Gulliver et al., 2004; Jones et al., 1999; Kaasinen et al., 2000; L. Kim & Albers, 2001; Kjeldskov & Stage, 2004; Masoodian & Lane, 2003; Nielsen, 1993; Suwita & Böcker:, 1999; Ziefle, 2002)
Effectiveness	Comparison of user performance with a predefined level (e.g., finishing tasks in 9 minutes, using no more than 2 clicks) in terms of speed (Bederson et al., 2002; 2003; Buchanan et al., 2001; Chittaro & Cin, 2002; Christie et al., 2004; Jones et al., 1999; Suwita & Böcker:, 1999; Ziefle, 2002), errors (Christie et al., 2004; Suwita & Böcker:, 1999; Ziefle, 2002), number of steps (Orkut Buyukkokten et al., 2001; 2002; Ebling & John, 2000; Suwita & Böcker:, 1999; Ziefle, 2002), task solved in a time limit (Chittaro & Cin, 2002; Jones et al., 1999; L. Kim & Albers, 2001)
Simplicity (Complexity)	Amount of effort to find a solution: numbers of menu levels that users have to go through in order to solve a task (Chittaro & Cin, 2002; Ziefle, 2002), numbers of button clicks and selections to reach a destination page (Chittaro & Cin, 2002; Christie et al., 2004; Ziefle, 2002), time used to search a button to perform a specific function (Buchanan et al., 2001; Ziefle, 2002)
Comprehensi bility (Readability)	Reading speed (word/minutes) (Öquist & Goldstein, 2002) and percentage of correct answers in a predefined test (Kim & Albers, 2001; Öquist & Goldstein, 2002)
Learning Performance	Performance Evaluation of assignments in classrooms (e.g., exercises, notes, concept maps) (Danesh et al., 2001; Luchini et al., 2001; 2002; 2003)

Figure 2. Table from Zhang and Adipat [12]

as shown in figure 2. By observing the patterns in the variables used as enumerated in [12], one can observe that the most frequently used variable is *time* (e.g. task times). Other common variables are *events* (e.g. button clicks) and *navigation* (i.e. combination of the number of steps and which components are accessed to reach a destination). The usability attributes satisfaction, comprehensibility and learning performance however, can be measured by means of qualitative surveys (e.g. questionnaires) and not from low-level information such as the ones extracted from logs. Based on the survey [12], the following information are sufficient to be able to get the necessary usability measurements: the user-interface (UI) component being utilized, the corresponding action or event, and the time the action was performed.

B. Preparation and Collection Phases: Simplifying Logging with EvaLogger

One task in the preparation phase is adding the necessary code in the application to enable logging of information. To help ease this task, a set of APIs that allow easy logging

Table I
EVALOGGER METHODS

Method Name and Parameters	Description
startNewLogFile(platformName)	Tells the EvaHelper framework to create a new log file for a specific platform
setBasicLogInfo(screenName)	Set the screen name for each section of the application
log(componentName, componentType, action)	Log information
log(screenName, componentName, componentType,action)	Log information
logComment(anyLogMessage)	Log information

of information should be provided. This set of APIs should provide some guidelines on how the information should be logged and into which parts of the code should the log calls be placed. Keeping these things in mind, the EvaHelper framework consists of the EvaLogger class which contains a set of APIs that can be called from the mobile application. As mentioned earlier, EvaLogger only supports applications that uses the Android platform but can easily be extended or ported to support other platforms. Table I shows the methods present in the EvaLogger class. The following are the guidelines on how to use the methods provided by the EvaLogger class:
(1) To create a new log file, the `startNewLogFile` method should be called
(2) The `log` methods should be called inside event listeners. There are two ways to log information as shown in table I.

A `logComment` method is also included which is used to log any type of unformatted message. This log can be used to add additional comments to the log file. The message logged by this method will be treated by parsers as comments and therefore will not affect any processing done to the log files.

Logs are currently stored as comma separated values (CSV) format. This choice was made instead of the commonly used XML format in order to minimize the file size needed by the logs considering in XML, additional tags are needed to store information.

C. Extraction Phase: Simplifying Log Output Processing with EvaWriter

After preparing the system to collect the necessary usability information and conducting the user studies, the next step would be to extract and process the collected data for further analysis. In the EvaHelper framework, a class called EvaWriter is used to process the log file into some output format. As of the moment, it can output the log file into GraphML[3] file format. GraphML is an XML-based file format used to represent graphs. The EvaWriter class can be easily extended to provide other output formats as well. Currently, the EvaWriter class contains the method `printCSVDataAsGraphMLToFile` which takes a CSV file as input and transforms this CSV file into GraphML format.

[3]http://graphml.graphdrawing.org

D. Analysis Phase: Usability Analysis with Graphs

Graphs are data structures that provide great representations to aid the analysis and design of systems [9]. The choice of using graphs for visualizing user interactions instead of UML activity diagrams or state diagrams is because of its simplicity. The simple node and edge notation of graphs is sufficient to describe such events. Using graphs as visualizations for user interactions with mobile applications is viable since user interfaces and interactions are always of rather low complexity as compared to desktop applications. Another advantage of using graphs is that, algorithms (e.g graph traversal) are readily available that can be used to analyze the system. Different graph formats can be utilized, however for the purpose of this study, we used the GraphML format because of its flexibility and because of the available parsers that can read such files (e.g. the yEd Graph Editor[4], the Prefuse[5] Visualization Toolkit).

User interaction with the mobile application can be represented as a nested directed graph. Each UI component (e.g. button) accessed by the user is represented as a node in the graph. These UI components are nested inside a screen node in which they belong to. The edges of the graph represent the transitions/actions done by the user and are labeled with: a sequence number, the action/event and the time duration between each action/event. The arrows represent the direction of the next action. Figure 3 shows an example of the output graph viewed using the yEd editor.

One way to measure learnability, efficiency and memorability of the system is by measuring the time it takes for the user to accomplish a task [12]. Using colored edges [10] or different edge lengths can also be used to visualize the time spent by users to go from one part of the application to the next. One way to measure error, effectiveness and again, memorability, is by taking note of the number of steps the user has to make in order to accomplish or deviate from the task (e.g. button clicks, going from one screen to the next) [12]. The sequence number on the edges of the graph can be used to get this information. Simplicity can be measured by the amount of effort the user has to make in order to find a solution [12]. Another is by studying the current design of the actual screen in the mobile application and comparing it with the output graph to see which UI components were not utilized and can possibly be discarded in order to keep the interface simple. Another recommendation to improve the visualization of the graph is to use different colored nodes depending on the level of activity [2] (e.g. frequently used nodes are shown in red color).

[4]http://www.yworks.com
[5]http://prefuse.org

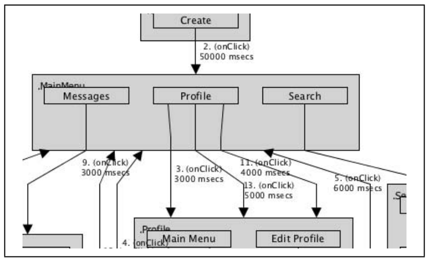
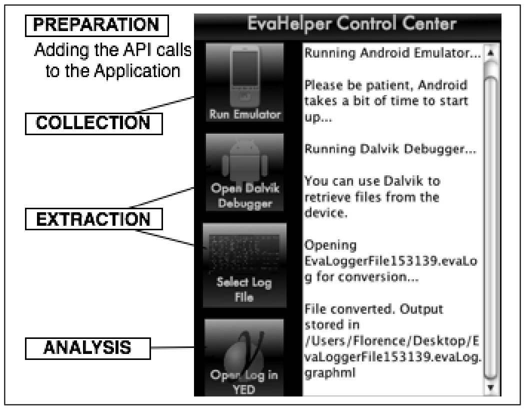
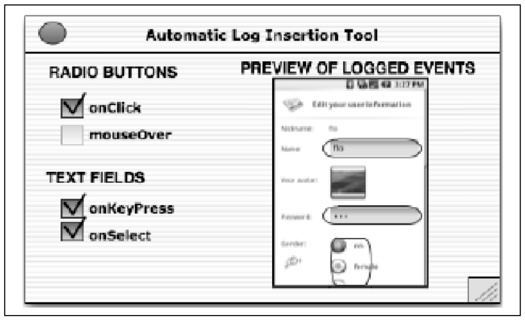


Figure 3. Graph format of the user interaction



```
Running Android Emulator...

Please be patient, Android
takes a bit of time to start
up...

Running Dalvik Debugger...

You can use Dalvik to
retrieve files from the
device.

Opening
EvaLoggerFile153139.evaLo
g for conversion...

File converted. Output
stored in
/Users/Florence/Desktop/E
vaLoggerFile153139.evaLog.
graphml
```

Figure 4. EvaHelper Control Center



RADIO BUTTONS

- [✓ onClick]
- mouseOver

TEXT FIELDS

- [✓ onKeyPress]
- [✓ onSelect]

Figure 5. Example Interface for the Automatic Log Insertion Tool

E. *EvaHelper Control Center: Guiding the Developer through Tool Integration*

We proposed that there should be a central application in which all of the tools mentioned can be easily accessed. As a proof of concept, we have developed an example of such application which we called EvaHelper Control Center. From the control center, the different tools needed in the different phases of collecting and processing the usability data can be accessed. The buttons in the control center are purposefully arranged in a sequence which indicates the order in which the different tasks should be performed. For example (see Figure 4), the emulator is first started in order to run the application being evaluated. This is the phase where usability data is collected. In cases where the real mobile device is used, this button can be ignored. The second button which runs the Dalvik Debug Monitor Service (DDMS) is used to see the debug messages and to access the files from the device (either the emulator or the real mobile device). The third button is then used to access the log file created during the user study. The last button which says 'Open Log in yEd', formats the log file to GraphML and automatically opens the formatted file in the yEd Editor for further analysis (analysis phase). What we want to emphasize in this section is that, although we recommend that the use of third party tools should be supported, there should also be a way to integrate these different tools in the form of another application such as the EvaHelper control center described. The control application should be designed in a way that it can guide the developer in the different phases of the usability evaluation through the different layout of the components, proper documentation of the functionalities and better integration of the different tools involved.

IV. Application Possibilities for Future Work

In this section, we will discuss how our methodology can be extended and what other application possibilities can be done to improve it.

A. *Simplifying Log Code Insertion through Automation*

The proposed approach in this paper involves manual coding of the log calls in the application. However, this can be a very tedious task especially when huge amounts of code are involved. To remedy this problem, we recommend automatic log code insertion to rid the developer of the manual tasks. As an example (see Figure 5), an interface in which all the UI components in the application and the possible events associated with it are shown in the tool. The events that the developer wants to be logged can be selected from the tool which then performs the necessary parsing and code transformations to add the log code to the application.

B. *Separating Logging Concerns with Aspect-Oriented Programming*

Another approach that can be done to simplify the logging part is by applying the aspect-oriented programming. Aspect-oriented programming (AOP) is a technique which aims to separate cross-cutting concerns through the use of modularized code called *aspects* [5]. The current approach with EvaHelper is that, the developer has to manually add code to all of the classes that needs to be logged. Instead of doing this approach, the idea of aspect-oriented programming can be applied. A separate module can be created to take care of the logging issues. AspectJ[6] which is an aspect-oriented Java-based framework can be applied for mobile applications based on Java platforms such as Android and Java ME. Applying the guidelines mentioned in

[6]http://www.eclipse.org/aspectj/

the methodology in section III-B, join points can be declared at the part where the developer wants to add logs (e.g. inside event listeners).

V. Related Work

Paterno et al. [8] proposed a methodology and environment that allows the remote evaluation of mobile applications. The system features a Mobile Logger that collects the information from the mobile device, and a tool called Multi-Device RemUsine that processes the logged information and provides the necessary visualizations to analyze the usability of the application. Another similar approach by Au et al. [2] was done to test usability of handheld applications. Different aspects considered in testing the usability of handheld device applications and a proposed a list of functional requirements that automated usability testing tools should have in order to be effective were discussed. The Handheld device User Interface Analysis (HUIA) testing framework was then developed that meets most of the requirements proposed.

In the area of web-based applications, usability analysis tools and visualizations are also important research areas. Waterson et al. [10] developed a tool called WebQuilt that collects information about how users interact with a web system and provides the necessary visualization that would allow web design teams to analyze the usability of the system. Graphs were used for the visualization. However, low-level data such as mouse movement and scrolling are not captured. Atterer et al. [1] on the other hand presents the UsaProxy system that collects fine-grained user interaction data (e.g. mouse movements, keyboard input) aside from the basic usage data to allow detailed usability analysis of web pages. Tools such as the UsaProxy from Atterer et al. [1] and WebQuilt from Waterson et al. [10] allows gathering of web-based usability information without modifying the code from the web pages through a proxy-based logging system. Unfortunately, this technique cannot be applied to applications on mobile platforms unless there is an observing communication channel that observes action between the application and the user.

VI. Summary and Conclusion

In this paper, we have discussed an approach to simplify the tasks involved in collecting usability information for mobile applications. We focused on helping the developer on the technical side of the usability evaluation, which involves preparation of the system for collecting usability data, and extracting and formatting that data into information which can be easily analyzed later on. Various problems regarding collecting usability data from mobile applications were discussed, and our proposed methodology and framework were described in order to solve some of these problems. In this research, we try to emphasize the importance of having the appropriate tools in the form of APIs and guidelines in order to simplify the developers' tasks. The use of third party tools in order to analyze usability data was also emphasized since such tools are already good at what they do, and also ease the burden from the developer in creating additional tools just for analyzing the collected data. Integration of the different tools involved in the whole process of usability testing and analysis were also discussed. Automatic code modification to further simplify the tasks involved is also a possible future work.

References

[1] R. Atterer, M. Wnuk, and A. Schmidt, "Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006.

[2] F. T. W. Au, S. Baker, I. Warren, and G. Dobbie, "Automated usability testing framework," in *Proceedings of the ninth conference on Australasian user interface - Volume 76*. Australian Computer Society, Inc., 2008.

[3] F. Balagtas-Fernandez, J. Forrai, and H. Hussmann, "Evaluation of user interface design and input methods for applications on mobile touch screen devices," in *Proceedings of the 12th IFIP TC13 International Conference on Human-Computer Interaction (INTERACT)*, 2009.

[4] H. B.-L. Duh, G. C. B. Tan, and V. H.-h. Chen, "Usability evaluation for mobile device: a comparison of laboratory and field tests," in *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. ACM, 2006.

[5] T. Elrad, R. E. Filman, and A. Bader, "Aspect-oriented programming: Introduction," *Commun. ACM*, vol. 44, no. 10, pp. 29–32, 2001.

[6] M. Y. Ivory and M. A. Hearst, "The state of the art in automating usability evaluation of user interfaces," *ACM Comput. Surv.*, vol. 33, no. 4, pp. 470–516, 2001.

[7] T. Kallio and A. Kaikkonen, "Usability testing of mobile applications: A comparison between laboratory and field testing," *Journal of Usability Studies*, vol. 1, pp. 4–16, 2005.

[8] F. Paterno, A. Russino, and C. Santoro, "Remote evaluation of mobile applications," *Task Models and Diagrams for User Interface Design*, pp. 155–169, 2007.

[9] G. Taentzer, "Agg: A tool environment for algebraic graph transformation," *Applications of Graph Transformations with Industrial Relevance*, vol. Volume 1779/2000, pp. 333–341, 2000.

[10] S. Waterson, J. Hong, T. Sohn, J. Landay, J. Heer, and T. Matthews, "What did they do? understanding clickstreams with the webquilt visualization system," *ACM International Working Conference on Advanced Visual Interfaces*, 2002.

[11] S. Weiss, *Handheld Usability*. John Wiley and Sons Ltd, 2002.

[12] D. Zhang and B. Adipat, "Challenges, methodologies, and issues in the usability testing of mobile applications," *International Journal of Human-Computer Interaction*, vol. 18, no. 3, pp. 293–308, 2005.