MobiDev: A Mobile Development Kit for Combined Paper-Based and In-Situ Programming on the Mobile Phone

Bastian Pfleging
Pervasive Computing and User Interface Engineering Group
University of Duisburg-Essen
45117 Essen, Germany
bastian.pfleging@uni-due.de

Martin Hermes
Pervasive Computing and User Interface Engineering Group
University of Duisburg-Essen
45117 Essen, Germany
martin.hermes@stud.uni-due.de

Elba del Carmen Valderrama Bahamondez
Pervasive Computing and User Interface Engineering Group
University of Duisburg-Essen
45117 Essen, Germany
elba.valderrama-bahamondez@uni-due.de

Johannes Nolte
Pervasive Computing and User Interface Engineering Group
University of Duisburg-Essen
45117 Essen, Germany
johannes.nolte@stud.uni-due.de

Albrecht Schmidt
Pervasive Computing and User Interface Engineering Group
University of Duisburg-Essen
45117 Essen, Germany
albrecht.schmidt@uni-due.de

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
In this paper we present MobiDev, a development kit that allows the creation of applications for mobile devices by developing directly on a mobile phone and by using paper-based sketches as a starting point for creating the user interface (UI). Although programming mobile applications on a computer has a well defined development structure, developing a mobile application on the mobile phone instead offers some advantages: (1) it allows people without access to a computer but to a mobile phone to create mobile applications and (2) it supports the development of applications which employ enhanced mobile phone features that are not fully supported by current desktop development environments. Users draw UI sketches on paper (similar to a paper prototype) as the initial step in an evolutionary UI development process to speed up the development of the application and to minimize the text input effort.

Keywords
Design, Programming, Mobile Phone, Mobile Application Development, Rapid Software Generation, Visual Programming.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces – Screen design (e.g., text,

graphics, color), Prototyping, User-centered Design; D.2.2 [Software Engineering]: Design Tools and Techniques – User Interfaces; D.2.6 [Software Engineering]: Programming Environments.

General Terms

Human Factors, Design

Introduction

Currently, the creation of applications for mobile devices, i.e., mobile phones, is done on personal computers (PCs). Most of the development toolkits include a mobile phone emulator where the generated applications can be debugged and tested. However, this approach of mobile programming leads to some problems:

First, in developing countries mobile phones are more widespread than PCs. According to data from [5] in average around 46% of the households in developing countries have access to mobile phones compared to only 24% of the households with access to a PC. A survey that we conducted in public schools in Panama showed that 97% of the teachers have mobile phones versus 63% with PC access; 80% of the students (5^{th} to 9^{th} grade) have mobile phones compared to 43% of the pupils with access to a PC at home. Moreover, the trend shows a continuous growth of the access to mobile phones rather than a growth of PCs. This means that in underdeveloped regions citizens currently are unable to create custom applications for their mobile phones, as they do not have access to a PC. In addition to this, researchers [4] agree that for the development of software applications in underdeveloped regions it is very important to consider the local cultural and social practices. Easy development tools running directly on mobile phones could empower those people to build successful applications based on the cultural and social context that they know very well.

Second, new communication features and behavior integrated in mobile phones, such as Bluetooth, WIFI, and GPS, cannot be fully recreated on the emulators of desktop development kits. Thus, developers code on a PC, then they debug and test on the mobile device and for changing code parts they have to return to the PC. All of this leads to a time-wasting switching between PC and mobile device and vice versa as many times as the programmer wants to test the program.

Finally, existing prototyping techniques do not allow adding full functionality of mobile applications. In paper prototyping, real interactions are very limited and higher fidelity prototypes such as those made in Power Point or Flash allow a nicer presentation but still do not permit a real interaction. Some programmers prefer then to create prototypes directly using standard programming languages. Instead of this we propose a development framework that enables users to rapidly develop prototypes and real applications directly on the mobile phone.

In this paper we will at first discuss some related works found in the literature. Later, different challenges of developing mobile applications in-situ will be analyzed. Then our concept and workflow will be explained, followed by the feedback received from an initial focus group conducted. Finally some future direction of the work will be discussed and we draw a conclusion.

Related Work

In the literature, some incipient research about prototyping in-situ for mobile environments can be found. The most prominent one is [2], [3] where a framework is presented that supports the creation of low to high

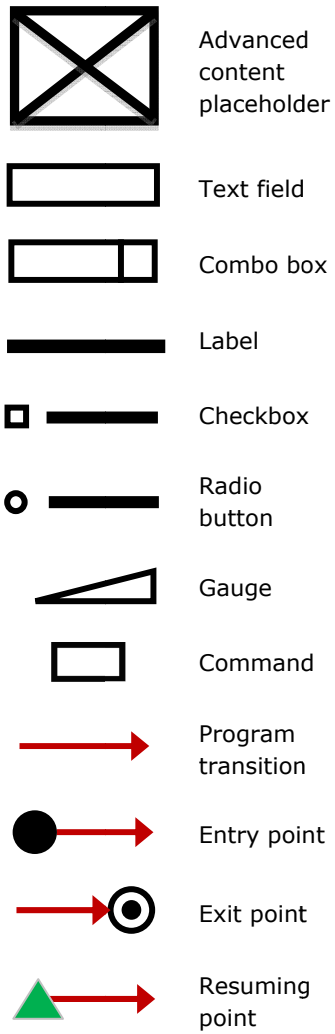

Figure 1: Currently available symbols to draw a paper-based UI sketch.

fidelity prototypes. In this approach, the application allows the combination of scanned hand drawings with digital images to create a high fidelity prototype. The sketches as well as the application flow are created on a PC and the resulting prototype runs on a mobile device. Additionally, the framework allows users to edit the properties of the graphical user interface (GUI) elements both on the PC and directly on the mobile device. The users can follow the transitions of the prototype created by the framework on the mobile device, interact with the prototype, and finally fill out a feedback questionnaire about their experiences.

iStuff Mobile [1] is another example of an in-situ prototyping framework which in contrast to the former focuses on creating low-fidelity prototypes of new sensor-enhanced interfaces for mobile phones. It uses wireless sensors which are externally attached to existing, unmodified mobile phones in order to augment their features. The framework employs software components across a network to handle phone input events and sensor events and to forward the intended output events to any (even built-in) application on the mobile phone. As the mapping between the afore mentioned events is done on a PC using a visual programming environment and as external sensors are attached to the phone, the intended target groups for this framework are mainly interaction designers and researchers.

Some Programming Environments include a visual development tool to support a rapid GUI creation. In Visual Basic users can drag and drop GUI elements, change properties, and define behavior accordingly to events raised. Netbeans for Java ME includes a Visual Mobile Designer tool, which allows the creation of GUIs and the definition of the flow between different screens.

Challenges for Mobile In-situ Development

If one intends to develop applications directly on the phone, a couple of difficulties and inconveniences arise - especially if the development method should be similar to desktop development. Primary shortcomings are caused by hardware constraints like device/screen size and interaction modalities. Big issues are the limited sizes and resolutions of most phone displays, which complicate the possibility, e.g., to overview longer parts of code, which is a common task at least while developing on a PC. Upcoming technologies like mobile phone projectors might help to overcome this problem. Additionally most mobile phones lack of an efficient typing method. This inconvenience might be solved if a full QUERTY-keyboard (perhaps foldable) can be connected to the phone, which in return limits portability. Thus, reducing the typing effort instead might be a better solution. Another feasible solution which is used in our underlying concept is to utilize visual input using the camera, which is integrated into almost every current phone. By drawing objects or writing text and then processing captured images, input can be generated to develop mobile applications.

Technology

In order to allow the development of mobile applications without a PC, we aim to reduce the technological requirements for the development process. The only device, which shall be required to be able to develop applications, is an ordinary mobile phone. This phone has to provide at least an integrated camera and has to be capable of running Java ME applications. Thus, computer vision and image processing methods can be used to transfer captured images of UI sketches into a basic version of a mobile application.

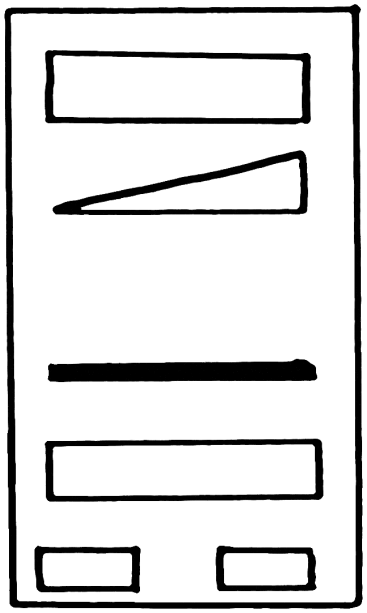

Figure 2: A sample UI sketch consisting of one form which contains the proposed symbols representing the different UI elements.

Concept

In this section, our concept will be illustrated in further detail. The key facts of our concept can be summarized as follows:

- Mobile applications are generated using just a mobile camera phone itself without the need of a personal computer for the developer.
- Mobile user interfaces can be generated by first drawing paper-based sketches of the required UI screens and then capturing images of the sketches.
- The general application logic of such a mobile application can be defined by visually connecting the drawn GUI sketches with arrows.
- Code entry can be reduced to the processing of the different user inputs and to refinements of the application logic.
- Mobile code entry can be facilitated by offering code completion, graphical programming, and optical character recognition (OCR).

As a core idea, our concept intends to minimize the efforts of developing applications for mobile phones by providing a software development kit which only requires a mobile, a pen, and a sheet of paper as a development environment. The necessity of entering big amounts of code on the phone is minimized by using the input of an integrated camera to generate the GUI. Thus, mobile application development on the phone becomes as easy as it is when using desktop development tools like the Netbeans Visual Mobile Designer.

Work Flow

According to our proposition, a mobile application can be developed by executing the following tasks: As a first step, all screens that might be visible when executing the application have to be drawn on a sheet of paper. In order to simplify the image processing methods, every screen is abstracted by using the symbols, which are explained in **Figure 1**. The available symbols correspond to the different objects which can be integrated into a typical Java ME form. A sample screen using those symbols is shown in **Figure 2.**

The second step is to define the general application logic, i.e., to specify the different transitions like, e.g., switching between or refreshing screens. The transitions are drawn as arrows on the same sheet of paper. Every arrow connects the triggering object with the screen that is shown next. Special symbols are used to visualize constructs like, e.g., loops (in future versions), entry and exit points for starting, resuming, or terminating an application. **Figure 3** shows the sketches for an example application containing all screens of the application as well as the main transitions.

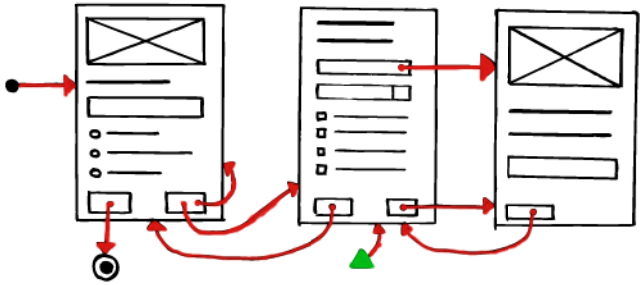

Figure 3: An Example of the complete visual input for developing a mobile application. The sketch includes all screens and the main application flow.

Once all screens and the main transitions are drawn on paper, the software development kit will be used for the final steps until the new application has been deployed. First, the application captures an image of the complete

Figure 4: A screenshot of the development kit showing the detection result of the captured UI sketch shown in *Figure 2.* The displayed items can be modified or deleted and new items can be added as well.

sketch prepared in the previous steps. The image is then processed to detect all visible elements and to generate the desired objects and transitions. In a first processing step, the different screens and the global logic are recognized and the different elements of every screen are parsed. The results for each screen are displayed on the phone one after another. **Figure 4** shows the detection result of the exemplary sketch presented in **Figure 2**. Corrections to each screen can be done by modifying or removing the displayed UI elements or by adding new elements. Additionally, the properties of each item can be adjusted like, e.g., setting a meaningful name for an item in order to allow easier access during the coding phase. As shown in **Figure 5** a preview mode is available to visualize how the screen will look like in the final application.

Whenever an appraisal of a screen is finished the transitions beginning at the current screen are examined and stored. If a transition starts at a command, it will be initiated when the command is executed. If the transition starts at a specific item on a form, the developer can select the event related to this item which should trigger the transition.

As a last step, the developer has the possibility to add specific code to each transition and thus to give life to the application itself. To reduce code entry even in this step, different supporting techniques are possible. One possibility is of course to provide code completion mechanisms while entering code. If longer pieces of code have to be entered, the user could also write the corresponding code onto a piece of paper and capture an image of the text which will then be processed by an optical character recognition web service to integrate the code into the application. If a projector phone is used, this technique could be extended to allow instant changes within the projection by using a pen.

As soon as all mentioned steps are completed, the mobile application can be started, debugged, and even distributed as a full program to other phones.

Initial Feedback

In order to get feedback about our concept, we conducted a focus group with users who had experiences in mobile programming. The participants were six males with an average age of 23.5 years. All of them were senior students in computer science at our university with experience in mobile and desktop programming.

The idea of programming directly on the mobile devices was well received by the focus group. The participants stated that they could imagine to develop on mobile phones instead of on the desktop when they are on the way or have no access to a PC: "When I have an idea to implement something and I do not have the laptop with me (…) I can just open my (mobile) phone and make the improvements." In addition the participants found that for some applications debugging and testing in-situ is much more suitable than on emulators as one participant stated: "Hardware specifics of phones are difficult to test in the emulator, for example: how do I debug a multi-touch mobile application on a PC?" Another participant complained about the permanent need to switch between the PC and the mobile phone while coding and testing. The small screen and especially the limited text input mechanisms were the main concerns raised by each participant for coding on mobile phones. The members of the focus group considered the use of paper-based sketches as the base for the implementation of GUI elements and interactions as

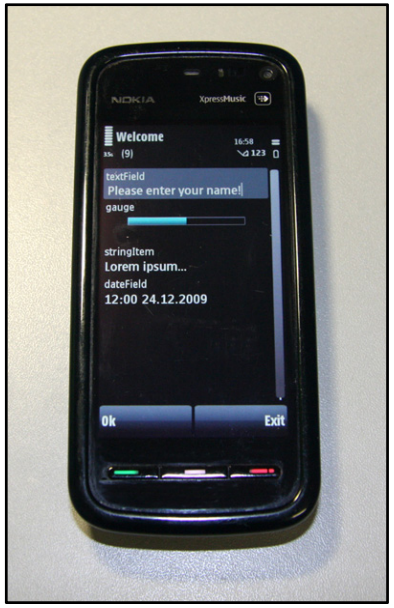

Figure 5. A screenshot of the development kit showing the preview mode for the form shown in *Figure 4.*

very useful – especially for a rapid development of applications. All participants stated that they definitely would use a tool like the proposed development kit. Without much effort and without any previous explanation the participants understood the symbols and transitions of the presented paper-based sketching method.

Conclusion and Future Work

Although current mobile phones have more power than personal computers of the middle 90s and mobile phones are very popular in developed and developing countries, programming for mobile phones is done solely on the PC. One main reason resides in the output and input limitations caused by small screens and keyboards. Here, we propose the concept of MobiDev, a development kit that combines paper-based sketches with in-situ mobile phone programming. In order to reduce the usage of the keyboard for coding, paper-based sketches are utilized as a basic visual GUI development tool. User only draw sketches of the application, then MobiDev interprets the GUI elements, interactions and transitions between the forms drawn, and converts them to a real mobile application.

Citizens from developing countries could directly benefit from such a system. As many of those people only have access to a mobile phone our system would offer them easy means to develop plain applications for the first time. For example teachers could build learning applications for their pupils, or students could learn programming, which so far has been impossible to do properly without access to a PC.

In addition, it could also benefit expert programmers as MobiDev allows to develop and to debug software which uses hi-tech mobile phone features that are not fully supported by the emulators, such as camera and GPS. Thus, environment switching between PC and phone becomes redundant as everything is done on the phone.

The first development steps of MobiDev have been completed. Currently MobiDev is able to transform the UI sketches into real Java ME forms. Next, the recognition of the general application logic as well as the code entry methods will be implemented. As soon these steps are completed, we plan to conduct user studies to test our toolkit with real users.

Further improvements, related to image processing, could be to support text recognition, and, e.g., to allow color picking through taking a picture of a texture to customize GUI colors, borders and backgrounds. Finally, the system could be extended with a web service in order to accelerate complex image processing tasks or to generate and distribute standalone JAR files of a created application.

References

[1] Ballagas, R., Memon, F., Reiners, R., and Borchers, J.: iStuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *Proc. CHI 2007*, ACM (2007), 1107-1116.

[2] De Sá, M. and Carriço, L.: A mobile tool for in-situ prototyping. In *Proc. MobileHCI 2009*, ACM (2009), 1-4.

[3] De Sá, M., Carriço, L., Duarte, L. and Reis, T.: A mixed-fidelity prototyping tool for mobile devices. In *Proc. AVI 2008*, ACM (2008), 225-232.

[4] Kam, M., Mathur, A., Kumar, A. and Canny, J.: Designing digital games for rural children: a study of traditional village games in India. In *Proc. CHI 2009*, ACM (2009), 31-40.

[5] *Measuring the Information Society – The ICT Development Index (Edition 2009)*, International Telecommunication Union ITU, Geneva, Switzerland, 2009.