# MobiDev: A Tool for Creating Apps on Mobile Phones

**Julian Seifert[1], Bastian Pfleging[2], Elba del Carmen Valderrama Bahamóndez[1], Martin Hermes[1], Enrico Rukzio[1], Albrecht Schmidt[2]**

[1]Paluno, University of Duisburg-Essen, Schützenbahn 70, D-45117 Essen, Germany

[2]VIS, University of Stuttgart, Pfaffenwaldring 5a, D-70569 Stuttgart, Germany

[1]{firstname.lastname}@uni-due.de, [2]{firstname.lastname}@vis.uni-stuttgart.de

## ABSTRACT

Currently, the development of mobile applications heavily relies on using conventional computers as development platform. MobiDev enables people in emerging countries without access to a computer but to a cell phone to develop their own locally relevant applications. The goal of the MobiDev project is to simplify development and deployment of applications directly on mobile phones. As a first step, we focus on the design of applications and try to support the computer science curriculum in developing countries to bootstrap the mobile developer culture and community. MobiDev allows the creation of graphical user interfaces (GUI) using various concepts. We present the results of a first system evaluation that show how people perceive the concepts for UI creation of MobiDev.

### Author Keywords

Mobile phones, application development, visual programming, code generation.

### 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.

### General Terms

Human Factors, Design.

## INTRODUCTION

While even in emerging countries around 60% of the people own a cell phone [4], the penetration of desktop computers in emerging countries is rather low: Only 22% of the households have access compared to 70% in developed regions [4]. The lack of conventional computers in those areas hinders programming localized applications as well as teaching computer science and programming in schools. The One Laptop per Child initiative could support the latter but the number of distributed laptops (~2 m) is much smaller than the actual demand [6]. Considering the trajectory of current smart phone prices, we assume that smart phones will be affordable and widely available for users in emerging countries in the near future.

Therefore, we see a high potential of using available mobile phones for programming purposes at school and at work. In addition we think that it is important to enable non-programmers, amateurs, and end users to build easily their own local applications. We developed MobiDev – an Android-based system for mobile application development that allows for the implementation of small applications directly on the cell phone without any additional hardware. The system allows users to create applications with graphical user interfaces (GUI) in three ways: (1) by defining the UI in code, (2) by using a graphical GUI designer and (3) by drawing a sketch of the desired UI on a piece of paper that is photographed with the mobile phone's camera and further transformed into a UI.

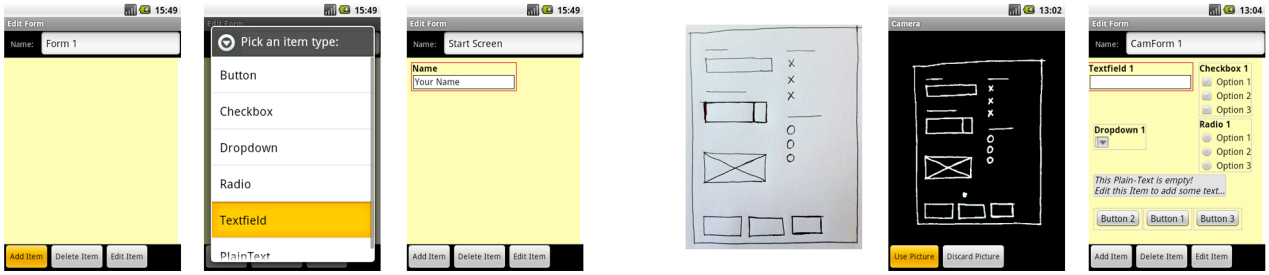
Developing applications on a phone holds a number of obvious problems. First, it is very cumbersome to type program code due to the small (virtual) keyboard. Second, only a small amount of program code can be visible due to the limited size of the screen. However, developing applications on phones is certainly an interesting option when no other alternative is available.

### Related Work

Research related to the concepts of MobiDev can be categorized in the field of (mobile) prototyping, simplified programming, paper prototyping and GUI building. An example of the first category is [1]. It allows users to create prototypes by drawing a UI design on paper, taking a picture of it with the phone's camera and augmenting these pictures on the mobile phone with interactive areas. Scratch is a visual programming language where users can compose little applications on a desktop computer by assembling visual programming blocks that define the functionality of the application [3]. The App Inventor from Google is based on this approach [2] and allows the user to create mobile apps for the Android platform on a conventional computer. Some of the conventional development environments provide a GUI builder for supporting the design of user interfaces for desktop and mobile applications (e.g., NetBeans GUI Builder). In the literature, we have not found mobile GUI designers running directly on the cell phone. With TouchStudio [6] a scripting environment is available that



*MobileHCI 2011,* Aug 30–Sept 2, 2011, Stockholm, Sweden.




a) b)

**Figure 1 a) UIBuilder: First, the user presses "Add Item" to select a new item. Then the new item is displayed at a default location and can be moved to the desired position. Second, item-specific properties and corresponding code can be added or modified. b) SketchBuilder: A picture of the drawing is taken using the integrated camera of the phone. This image is analyzed to extract all UI elements and possible transitions. The result can be inspected and modified using the UIBuilder or the code view.**

uses touch screen concepts to facilitate the creation of scripts on the phone.

## CONCEPT OF MOBIDEV

The concept of MobiDev [5] is based on the following key aspects. The system allows creating fully functional GUI based applications on a mobile phone without the need for additional hardware. That is, all program logic is created on the mobile phone and the resulting applications can be directly deployed. Users can publish applications on the web or share them with others.

MobiDev supports in particular the development of applications with a GUI and a back-end. The back-end containing the program logic is developed by typing code via the phone's keyboard into separate source code files. Users can also add program logic directly to UI controls (e.g., a button) which is executed when an event occurs. MobiDev allows users to create UIs in three ways. These are (1) typing the code that describes the UI, (2) using the graphical UIBuilder, and (3) using the SketchBuilder. The latter two are designed to accelerate and facilitate development compared to typing all the required code.

The UIBuilder (UiB) is a graphical editor that enables the user to create UIs directly on the mobile phone. Adding widgets to a UI using the UiB requires the user to take two steps as shown in Figure 1a. First, the user selects "Add Item". A list of available widgets appears from which the user selects one. For placing the widget on the canvas, the user can use the arrow keys or simply drag the widget to the desired position with a finger.

The SketchBuilder (SB) is used for transforming UI sketches created on paper into actual UI objects on the mobile phone. MobiDev provides a visual UI description language (UIDL) for defining UIs in paper-based sketches (see Figure 2). The SB can transform only UI sketches that are based on this UIDL into executable code. After creating a UI sketch on paper, the user selects "Create UI from sketch" for starting the SB. The first step is taking a picture of the UI drawing with the built-in camera of the mobile phone. The SB algorithm automatically analyses the captured image and generates the corresponding UI objects. The description language also contains arrows for the definition of the general program flow in the paper prototype. In a first step, especially screen transitions for events like starting or resuming an application as well as reactions to UI events (button press, item selection, etc.) can be visually defined. This information is transferred automatically as well. An example of a (static) UI sketch and its translation into real UI objects is shown in Figure 1b. On top of recognizing UI widgets, the SB supports creating basic application flows based on flow elements of the UIDL as shown in Figure 2. For example, one program transition arrow pointing from a button on one screen to another screen leads to the creation of corresponding code. This facilitates the design process and it spares the user entering code that is often reoccurring.

In order to add user-defined code, the developer has two possibilities: (1) From the central overview page of a project a code view (text editor) can be accessed where global methods and variables can be defined. (2) Widget-dependent code may be added from the detail page of the corresponding widget similarly for the widget-specific events. The code is entered as JavaScript as the Android platform provides a corresponding runtime environment without requiring a compiler. Objects can be accessed by their name which is defined in the UiB. Currently, all code parts need to be typed manually.

In the following, we illustrate the application of MobiDev's concepts and a typical workflow by means of a scenario: Kim and Alex are working together on a project in which they want to interview people on the street. For this purpose, they decide to create a small mobile application for collecting the answers with MobiDev. Together, they draw the sketches for the needed UIs on paper. Once they finished sketching the individual views of the app, Kim takes photos of the drawings with the camera phone. MobiDev transforms the pictures into views of the new application. The basic structure and workflow is polished using the graphical UIBuilder of MobiDev. They add program logic by typing additional code. Once finished, Kim saves the project and shares it with Alex via Bluetooth. Now both can start interviewing with their custom made application.

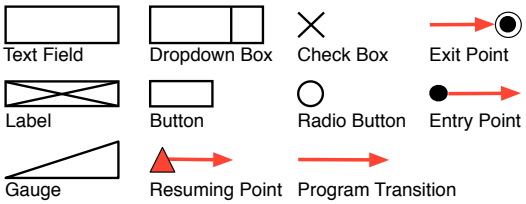



**Figure 2. Elements of the MobiDev UI description language (UIDL). UI sketches based on these elements can be transformed into UIs and basic program flows are detected.**

## IMPLEMENTATION OF MOBIDEV

As a proof of concept and for evaluation purposes we built a first prototype of MobiDev using Android 2.1. The user can create apps that are based on HTML and JavaScript. Program logic (i.e., JavaScript code) is edited in a code editor view.

In order to transform a captured UI sketch into UI objects, a set of processing steps has to be executed. First, the captured RGB image is converted into a gray level image. The result is inverted to its negative complement. As the next step the background of the image is calculated through the operations erosion and dilation. In order to correct the illumination the resulting background is subtracted from the image. Later, a threshold is applied to generate a binary image that distinguishes objects and background. The operations opening and closing are used to reduce noise. A simple classification algorithm based on geometric features maps each drawn symbol to its corresponding widget type. JavaScript code is added to the application that will insert a new widget of the recognized type into the Document Object Model of the application on each startup.

## EVALUATION OF MOBIDEV'S INTERFACE BUILDERS

We conducted a user study to investigate how users of the MobiDev prototype get along with the two alternative approaches for the creation of UIs that do not require to type code. Additionally, we wanted to gain insights how users welcome the concept in general.

Therefore, we recruited 16 participants. All participants were students from our computer science department. Two of the participants were female. The participants were in average 25.4 years old (SD=2.4).

The study was conducted in a laboratory environment. A video camera was used to record the interactions with the system. Additionally, a time logger recorded the time of each activity performed by the participant with the system.

The study was organized in three parts and took on average one hour to complete the whole procedure: First, the participants were introduced to use MobiDev by completing a training task. In this task the participants created a small MobiDev project, which consisted of two views. The first view contained text fields, radio buttons, check boxes, and buttons. The participants used the UiB to create the first view. For the creation of the second view, which was similar to the first one, they used the SB. Here, participants first made sketches on paper for the UI, then used the SB to transform the sketch into a UI. This made sure that each participant got to know the characteristics of the visual UIDL. For example, one common issue is that users tend to leave little gaps open at the corners of squares that denote a control. Gaps in the elements of the sketch reduce the accuracy of the recognition process.

In the second part, the participants had to perform two practical tasks with the system. In task A, participants created a simple currency converter application. The UI for this application consists of one view with widgets for entering an amount of money, a drop down menu for selecting the target currency, a button for triggering the calculation, and a text field for displaying the result. In task B an input interface for adding contact information to an address book was created. The UI consisted of three text fields for names and address information, a radio button group and a button for generating a welcome screen from the given information. In both tasks, the participants were given screen shots of how the final UI should look like. Also the code for the program logic was given on the instruction sheet. The mobile device used for the practical part was a Samsung Galaxy Spica I5700. The order of the tasks was randomized. One task was to be performed with the help of the SB. In the other task, participants had to use the UiB. Thus, the 16 participants were randomly distributed in four groups with a certain task order: G1 ($A_{SB}$, $B_{UiB}$), G2 ($B_{UiB}$, $A_{SB}$), G3 ($A_{UiB}$, $B_{SB}$), G4 ($B_{SB}$, $A_{UiB}$).

In the last part of the study the participants filled in a questionnaire concerning their impression of the system in general and comparing the aspects of creating UIs with the UiB or the SB.

### Evaluation results

The mean task completion time for task A with SB ($A_{SB}$) was 935.5s (SD=244.9) and for task A with UiB ($A_{UiB}$) 921.6s (SD=203.1). The mean completion time for task $B_{SB}$ was 1237.6s (SD=346.6) and for task $B_{UiB}$ 1167.6s (SD=316.1). The differences of average completion times for both tasks are statistically not significant ($p > .05$).

The biggest amount of time in the tasks was spent writing the code for the program logic (for task $A_{SB}$: M=618.3; $A_{UiB}$: M=657.0; $B_{SB}$: M=801.1; $B_{UiB}$: M=818.1). When focusing only on the actions that are specific for the UI creation approach it appears that users needed more time with SB. The difference is only statistically significant for task B as a paired-samples t-test shows ($p < .05$). Figure 3 shows next to the times for the UI creation the time that was required for drawing the sketches and taking the picture. The time for the image transformation into the UI was in average 4.7s and is included here.

The accuracy of the image transformation process for the SB was 98% correctly recognized widgets. This is surpris-

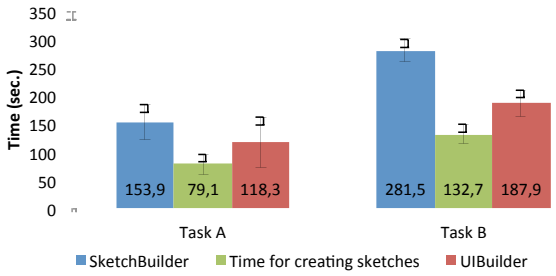



**Figure 3. The diagram shows time for creating UIs with SketchBuilder and UIBuilder. Further, the time for creating UI sketches is depicted as separate series. It appears that drawing sketches takes a significant amount of time.**

ingly high, and is likely because the given UI contained only few elements. On the UI canvas, participants made only corrections in the alignment of the widgets. The need for corrections in the placement of the widgets and the considerable amount of time for the drawing of the sketches make the SB approach slower compared with the UiB. Even though it took longer time to complete the tasks, 12 of 16 participants stated that they would prefer to work with the SB. As reasons they stated "working with the SketchBuilder is fun" or "it is faster". They also rated (Likert scale: 1=very low; 5=very high) the effort for working with SB lower (M=2.0, SD=0.8) as with UiB (M=2.9, SD=0.9).

Regarding the general concept, participants found that learning to use MobiDev is simple (M=4.6, SD=0.5) and the concept is easy to understand (M=4.1, SD=0.7). One participant suggested that sketches could be easily reused for other projects and also for working together with others. Another indicated that starting to develop an application could be done without the use of a mobile phone. Another participant suggested, "MobiDev would be great for teaching beginners how to write programs".

## CONCLUSIONS

In this paper we introduced MobiDev, a mobile application development tool that allows the creation of mobile apps without the need for hardware other than a mobile phone. The evaluation showed that it's possible to transform paper prototypes into executable code with a very high recognition rate of 98%. Further results show that creating UIs with SB is slower than working with the UiB. However, the great majority of participants stated that they would prefer the SB. From the qualitative feedback, we conclude that one reason for this is that working with the SB is fun and more pleasant compared to the UiB. Which approach for creating UIs is to prefer cannot be answered in general. As participants suggested, sketches of UIs can be reused. Also several people can collaborate and draw sketches together, which would be transformed to UIs by one mobile phone later. Therefore, SB and UiB are two approaches that supplement to each other. The latter is, to the best of our knowledge, the first mobile implementation of graphical user interface builder, which supports creating functional UIs.

A mobile application development tool as MobiDev will never be able to replace fully-grown integrated development environments for desktop computers. Limitations of mobile phones as a developing platform are too strong. However, for people in parts of the world where the mobile phone is the only available computing device MobiDev creates new options. It can be used for educational purposes as well as for creating applications for the own benefit.

The current prototype does not implement all aspects of the concept. Thus, next steps would be to investigate how the SB could be extended to recognize application flow charts. Additionally, coding assistance tools like code templates or methods as shown in TouchStudio [6] could be integrated to facilitate the coding process. Another interesting aspect is how mobile phones with integrated projectors could be used for creating UIs. Augmenting the interface of MobiDev with projected parts could help to overcome issues that are inherent to the size of mobile phones.

## ACKNOLEGEMENTS

The presented research was conducted in the context of the Emmy Noether research group "Mobile Interaction with Pervasive User Interfaces" and within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart, both funded by the German Research Foundation (DFG).

## REFERENCES

1. De Sá, M. and Carriço, L. A mobile tool for in-situ prototyping. In *Proc. MobileHCI'09.* ACM Press (2009).
2. Google Inc., *About – App Inventor for Android.* http://appinventor.googlelabs.com/about/.
3. Malan, D. J. and Leitner, H. H. Scratch for budding computer scientists. *SIGCSE Bull.* 39, 1 (March 2007), 223-227.
4. *Measuring the Information Society 2010*. ITU (2010), Geneve, Switzerland.
5. Pfleging, B., Valderrama Bahamóndez, E., Schmidt, A., Hermes, M., and Nolte, J. MobiDev: a mobile development kit for combined paper-based and in-situ programming on the mobile phone. *Ext. Abstracts CHI 2010*, ACM Press (2010), 3733-3738.
6. Tillmann, N., Moskal, M., and de Halleux, J.: TouchStudio - Programming Cloud-Connected Mobile Devices via Touchscreen, Microsoft Technical Report MSR-TR-2011-49, Microsoft Research, 2011. http://research.microsoft.com/apps/pubs/default.aspx?id=147663
7. One Laptop per Child, Deployments. http://wiki.laptop.org/go/Deployments