End-User Software Development: Tool Support for Mobile Data Collections

Mathias Kühn, Peter Forbrig, Anke Dittmar
University of Rostock
Albert-Einstein-Straße 22
Rostock, D-18051 Germany
{mathias.kuehn, peter.forbrig, anke.dittmar}@uni-rostock.de

Paper presented at MIDI 2013 Conference, 24-25.06.2013, Warsaw, Poland.

Abstract

Creating mobile applications is a process of device specific programming. In this way new devices ask for new solutions. Additionally, experts with programming skills are necessary. There is an urgent need for more efficient approaches. Abstract specifications for domain-specific areas in combination with tool support for end-user development seem to be a solution. This paper presents an approach that is grounded on model-based techniques and is focused on the domain of data collection. The language UsiXML and corresponding platform-dependent interpreters for this language allow to run specifications on different devices. A set of tools will be introduced that supports the creation of model-based user interface specifications. Additionally, the distribution to different devices is supported. In combination all these tools enable end-users to create applications in their domain of data collection.

Author Keywords

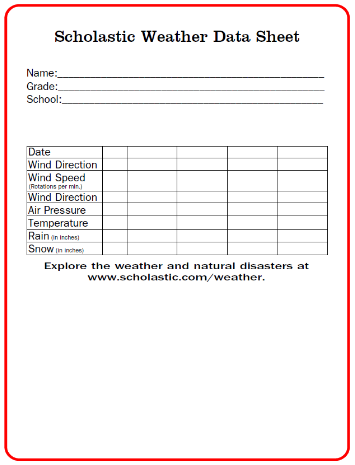
End-User Programming; Design Support; Mobile Devices.

ACM Classification Keywords

H.5.m. [Inf. Interf. and Pres. (e.g. HCI)]: Miscellaneous.

General Terms

Human Factors; Design; Measurement.


Scholastic Weather Data Sheet

Name:____
Grade:____
School:_______________________________________

Date					
Wind Direction					
Wind Speed (Rotations per min.)					
Wind Direction					
Air Pressure					
Temperature					
Rain (in inches)					
Snow (in inches)					

Explore the weather and natural disasters at www.scholastic.com/weather.

Figure 1: A sketch of a Weather Data Sheet

Introduction

The importance of mobile devices is greater in these days than ever before. Well-equipped devices allow the capturing of photos and videos in nearly all situations. Additionally, almost everyone has nowadays mobile devices like smartphones available. These devices could be used more extensively for data collection than it is done at the moment. A teacher could for instance ask students to collect weather data three times a day. The advantages are obvious. The data are available very soon. Measured data can be combined with pictures of clouds and even sound could be captured.

Manufacturers are very interested in acceptance of their products. Sometimes they distribute questionnaires on paper and ask customers for feedback. However, this runs into problems that the Experience Sampling Method [4] tries to avoid. Either feedback is not provided or the most important information is meanwhile lost. Providing questionnaires on mobile devices would help a lot to get the needed information. In most cases users are willing to provide feedback immediately. However, they are too lazy to provide it later. Immediate feedback has also the advantage that it is more realistic. Users give this feedback at the moment in the place where they are while using the product. Indeed, the need for a software to support the creation of data collections becomes apparent.

However, because of simplicity we will use within this paper the example of weather observation as motivation.

Weather Observation

A good knowledge about local and global weather helps to understand the climatic shift and hopefully avoids future impact. This example demonstrates in a simple way the type of application our tools are intended to support.

Students in Europe sometimes learn the consequences of the four seasons by gathering information about weather and other environmental influences. They document the daily weather attributes, like wind direction, wind speed, rain, snow depth, etc. in a log. Figure 1 contains a sketch of a weather data sheet that we found in the internet [1] and that is used at schools for manually collecting data. The students are asked to fill out the data sheet in a daily interval. When evaluating all data the students see changes in time which are typical for the season like the temperature or air pressure.

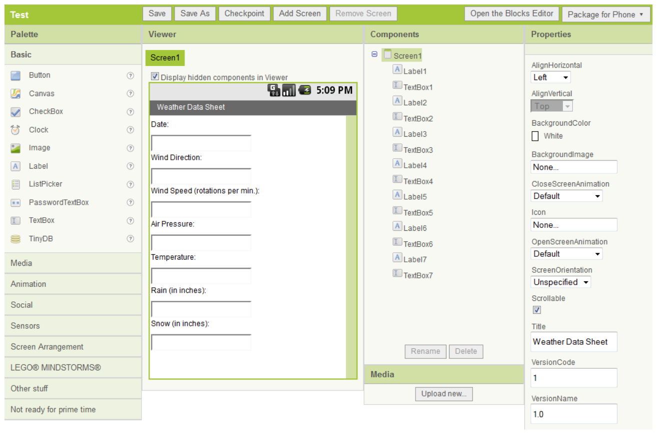
Another example of weather observation can be found in forecast. A meteorologist logs specific weather data to enhance climate models. This helps to make forecast more precise. The classical way could be that the meteorologist takes a pad of paper to make notes at the measurement station. The use of data collection forms in connection with resistant mobile devices would support the data collection process much better than the paper-based approach. It can provide a lot of data in a relatively short period of time.

Related work

In the domain of mobile data collection there is already a set of tools available that helps to make solutions for this kind of application. The following section presents some approaches that focus on the creation of mobile applications.

The first example is the App Inventor [1] which is especially designed for Android devices. This tool can be used via browsers and allows the design of graphical user interfaces (GUIs) and their behavior. The GUI design can be made in a WYSIWYG editor.

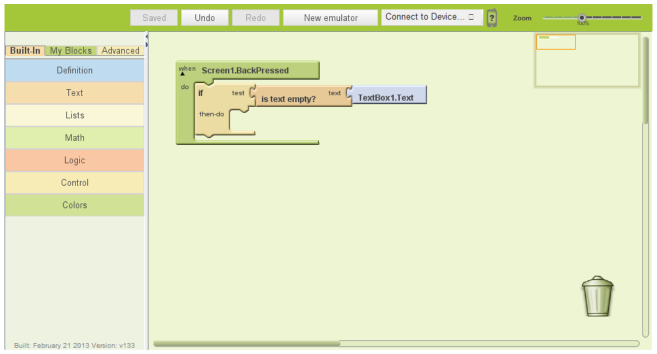
[1] `http://www.scholastic.com/weather`



AlignHorizontal: Left
AlignVertical: Top
BackgroundColor: White
BackgroundImage: None...
CloseScreenAnimation: Default
Icon: None...
OpenScreenAnimation: Default
ScreenOrientation: Unspecified
Scrollable: ✓
Title: Weather Data Sheet
VersionCode: 1
VersionName: 1.0

Figure 2: A screenshot of the App Inventor

Figure 2 shows a screenshot of the App Inventor containing some widgets on the left hand side. The behavior of the widgets can be specified via a visual language in an editor called Block Editor. Programming in the environment is like doing a puzzle. Syntactic elements of the language are pieces. The syntax matches when pieces fit.



```
when Screen1.BackPressed
do  if  test  is text empty?  text  TextBox1.Text
    then-do
```

Figure 3: A screenshot of the Block Editor

Figure 3 gives an impression how this looks like. The Block Editor contains a conditional statement that is executed when the back button is pressed. The tool provides a very interesting approach. However, it can only be applied by end-users that are able to specify algorithms on a very detailed level. Programming skills are necessary. This cannot be considered as fulfilled for most domain experts.

Another example is MobiDev [6] which is also designed for Android devices. This tool allows the creation of applications directly on the mobile device. The GUI design can be made by sketching with predefined elements. A sketch forces to use a special visual specification language. The final GUI is generated via a captured image of the sketch. This can be done with the camera of the device. Unfortunately, the behavior has to be specified in the classical way by programming on the device. This task is still very challenging for end-users.

MicroApp [3] is another approach using a visual language that facilitates the creation of applications on the mobile device. The idea is to compose sequential or parallel actions of a predefined set of actions to fulfill a task. Later the GUI is automatically generated from the specified task model. The behavior is specified in concrete action objects. At runtime the presentation is a sequence of slides containing interfaces of the underlying action objects. This modeling approach seems to be still challenging for end-users.

For the use of model-based specifications with graphical user interfaces there are several approaches available. One example is the USer Interface eXtensible Markup Language UsiXML [2, 5]. This language supports different levels of abstraction, like task models and abstract user interface specifications. One principle of UsiXML is to be completely independent of concrete input and output devices. User interface instances are generated by task-based models via abstract and concrete to final user interface specification.

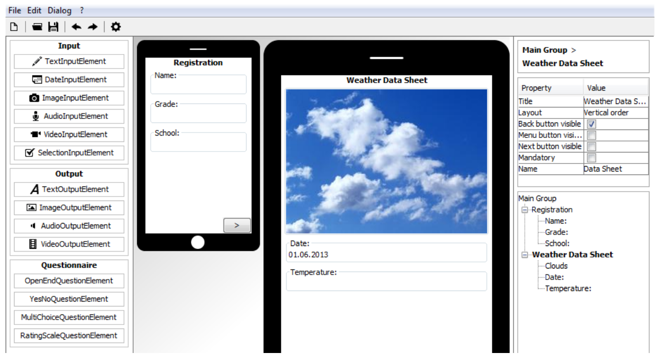
Own approach

The following section presents our tools that offer the opportunity to create applications for data collections. The corresponding editor for forms, a web application for exchanging specifications and data as well as the usage of forms on mobile devices will be discussed within this paragraph.

The editor for forms focusses on building screens containing sequences of elements. Specific elements can be selected from a predefined set of types of input, output and question elements. Output elements are of type text, image, audio or video. Input elements have additionally the types date and choice. Question elements can be of type open-end, yes-no, multi-choice or rating-scale.

In our approach output is interpreted as part of the user interface where the user will not be able to manipulate data. Input in contrast is interpreted as part of the user interface where the user will be able to manipulate data. In general question elements are interpreted like input elements, but are bound to a specific context of a question.

Screens can be grouped. All screens of all groups build a system. A system will be interpreted as sequence of screens containing sequences of widgets on the mobile device. In this way widgets of the system represent the elements povided by the editor for forms. Systems with screens containing only question elements can be considered as questionnaires. Figure 4 shows a screenshot of the editor.



Property	Value
Title	Weather Data S...
Layout	Vertical order
Back button visible	✓
Menu button visi...	
Next button visible	
Mandatory	
Name	Data Sheet

Figure 4: A screenshot of the editor for forms containing the Weather Data Sheet

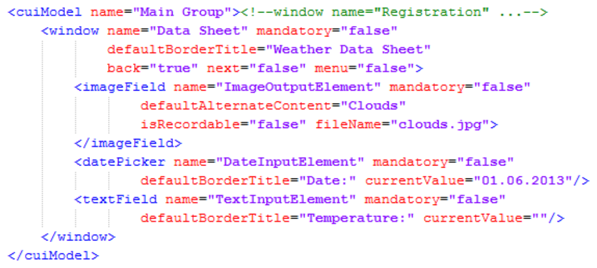
On the left hand side is the predefined set of elements. In the middle of the figure there are sketches of the screens. On the right hand side there is a breadcrumb, a table and a tree.

The breadcrumb shows the path of the current selected screen ("Weather Data Sheet") as part of the tree structure which represents the system. The table shows attributes of the selected screen and allows manipulations of them. The tree represents the structure of the specified system.

UsiXML Specification
We already mentioned that the USer Interface eXtensible Markup Language [2, 5] allows model-based definitions of graphical user interfaces. The definition of GUIs is platform independent what is nice in the context of desktop, mobile or web applications. Our editor is able to transform the designed system to a subset of UsiXML. We implemented several interpreters for this UsiXML-subsets. Using these interpreters, systems can run on a mobile device, in a browser or in the editor for forms.

One advantage of this domain-specific model-based technique is the platform independence we got. Nevertheless, this specifications are specific enough to reach a certain amount of devices with graphical user interface output. The abstract specifications of user interfaces can be instantiated in a platform dependent manner. According to this our tools need to follow some specific mapping rules for the abstraction as well as instantiation of GUIs.

To specify user interfaces we concentrated on the concrete user interface (CUI) model of UsiXML which is part of a set of different models. Some more of these models are task model, abstract user interface model, domain model or mapping model. A screen of our editor is mapped to a window-element of the CUI model. Elements of the screen also are mapped to elements of UsiXML according to their meaning, e.g. text to textfield-element or video to videofield-element of the CUI model. The hierarchy of a system is also reflected in the CUI model. The elements of the user interfaces are the leafs of the editor-tree representation as well as of the CUI model specification. Their parents are the screens of the editor as well as the parents of the CUI model tree specification which are the window-elements, respectively.



```
<cuiModel name="Main Group"><!--window name="Registration" ...-->
    <window name="Data Sheet" mandatory="false"
            defaultBorderTitle="Weather Data Sheet"
            back="true" next="false" menu="false">
        <imageField name="ImageOutputElement" mandatory="false"
                defaultAlternateContent="Clouds"
                isRecordable="false" fileName="clouds.jpg">
        </imageField>
        <datePicker name="DateInputElement" mandatory="false"
                defaultBorderTitle="Date:" currentValue="01.06.2013"/>
        <textField name="TextInputElement" mandatory="false"
                defaultBorderTitle="Temperature:" currentValue=""/>
    </window>
</cuiModel>
```

Figure 5: The UsiXML specification of the Weather Data Sheet

Figure 5 shows the UsiXML specification of the example “Weather Data Sheet” in Figure 4. The UsiXML specification is equal to the tree representation of the editor.

Web Application
We developed a web application to make the UsiXML specification available on different devices via world wide web. This application provides a browser interface as well as web service interface. The file containing the UsiXML specification needs to be uploaded to a server after it was created. The browser interface provides options to up- and download as well as to view files in browser specific presentations.

After the upload has finished the file is available for selection. A selected file will later be distributed via the web service interface. On request, a file containing the UsiXML specification can be downloaded to a mobile device. It is available for further interpretation there.

Mobile Application
To make use of UsiXML specifications we developed an interpreter for mobile devices with Android platform. This tool completely supports specifications created with our editor. The interpreter uses the web service interface of the web application to get the UsiXML specification file which is currently selected on the server.

Until now interpreted screens have a simple behavior. Users can navigate between elements of a screen by scrolling and between screens by clicking back or next button. The accessibility of these buttons can be specified both for all screens as well as for individual screens in our editor.

When the user has finished inserting all needed

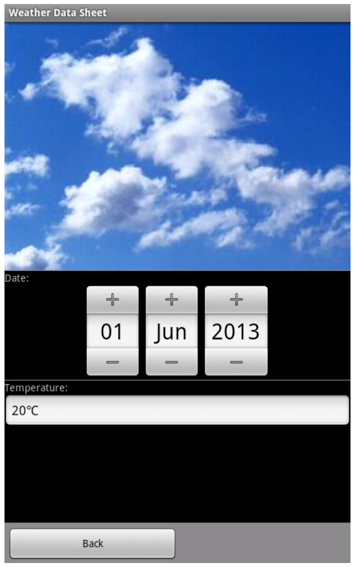


Figure 6: A screenshot of the interpreted screen "Weather Data Sheet"

information of a system the file containing the UsiXML specification together with the user data are send back again to the server via the web service interface. After that the uploaded file is available for download or for review in the browser interface. In Figure 6 is a screenshot of the interpreted screen of the example "Weather Data Sheet".

Discussion

An advantage of our approach is to use media like image, audio and video as part of the specification. In comparison to web based data collection applications the device features like microphone and camera are directly used in the mobile application. Captured data is not separated by different files. This kind of data could be part of the distributed specification as well as the submitted one. However, the model-based approach of this concept dissociates from device-specific user interfaces.

Conclusion and further work

We presented an approach that enable end-users to create specific applications for mobile devices in the domain of data collection.

An editor for forms allows the creation of UsiXML specifications from interactively designed graphical user interfaces. A web application facilitates the distribution of specifications via web service technique. This tool also offers the opportunity to view specifications via browser interfaces. The UsiXML interpreter for Android platform supports the instantiation of specifications created with the described editor and distributed via web application.

At the moment the specification of the dynamic behavior of applications like sending one picture every day or showing screens depending on entered values is limited.

Further research will show what kind of support users would like to have and which representation of such specification is usable. The statistical analysis of collected data is not focus of the tools and will be part of investigation for further work.

Acknowledgments

We thank Richard Becker for the ideas of his inquiry tool that helped us to form a concept for a mobile data collection application.

References

[1] MIT App Inventor. `http://appinventor.mit.edu/`.

[2] UsiXML. `http://www.usixml.org/`.

[3] Cuccurullo, S., Francese, R., Risi, M., and Tortora, G. MicroApps Development on Mobile Phones. In *International Symposium on End-User Development (IS-EUD)*, Springer, Heidelberg (2011), 289–294.

[4] Kubey, R., Larson, R., and Csikszentmihalyi, M. Experience Sampling Method. *Journal of Communication 46*, 2 (1996), 99–120.

[5] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Florins, M. USIXML: a User Interface Description Language Supporting Multiple Levels of Independence. In *Proceedings of the ICWE 2004 First International Workshop on Device Independent Web Engineering (DIWE 2004)*, Springer, Heidelberg (2004), 325–338.

[6] Seifert, J., Pfleging, B., Bahamóndez, E., Hermes, M., Rukzio, E., and Schmidt, A. MobiDev: A Tool for Creating Apps on Mobile Phones. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI 2011)*, ACM, New York (2011), 109–112.