Android-based Mobile Assessment System

Mahtab DALIR[a*], Heiko RÖLKE[b], Björn BUCHAL[c]
[a, b, c] German Institute for International Educational Research, Frankfurt am Main, Germany
*dalir@dipf.de

Abstract: Mobile devices such as smartphones and tablet PCs have gained global popularity and are increasingly used in all areas of daily life, including learning and assessment activities. To support different kinds of learning and assessment content, a versatile and comprehensive system is desirable. In addition, other aspects of long-term assessments have to be covered such as security, data protection and adaptivity, e.g. to new schedules or items. In this paper, we present the concept and realization of an Android-based mobile assessment system especially designed for school usage. It has been used already in several research studies in elementary schools, e.g. to measure daily fluctuations of cognitive performance capabilities.

Keywords: Mobile learning, mobile assessment, ambulatory assessment, Android, QTI

Introduction

Modern daily life is more and more influenced by ubiquitous usage of smartphones and tablet PCs. This "not being tied to particular locations and times" [1] is not bound to specific domains or areas of application. The mobility is especially interesting in the areas of learning and assessment, particularly for, but not limited to, schoolchildren.

For certain kinds of studies it is important to be able to test on a recurring basis and more than once a day. In addition, the actual testing should be as smooth and non-interruptive as possible to avoid organisational problems and to ensure validity of the data. To cope with this general challenge, mobile technology can be a solution. Until recently, it was hard to find devices that were mobile, suitable for children and both affordable and powerful enough for testing purposes. As pointed out above, this has changed now. Using smartphones or tablets for assessment triggers a positive side-effect: Children are excited about using “cool” technology which might increase their motivation. Our group was asked for technical support for a psychometric study of daily fluctuations in children’s cognitive performance in the school context. The general goal was to assess children’s cognitive resources three times a day for several weeks within the school year. Approximately 120 children aged between 9 and 10 years were planned to be tested.

To make this possible, we designed a versatile and comprehensive mobile assessment (and learning) solution. Our overall system is targeted to the school context but not limited to it. We did not restrict our system in terms of types of content and aimed for open standards. Special emphasis was put on test security, autonomous test delivery, and automatic data synchronization.

The first section of this paper presents requirements for a mobile assessment system, followed by the system architecture in Section 2. Afterwards, in Section 3, we outline a case-study where our system has been used extensively. The paper concludes with a section on related work and an outlook.

1. System Requirements

As outlined in the introduction our aim is to support mobile assessments, especially in the school context. The analysis of this aim revealed three major requirement areas:

- Content
    - Wide range of item types
    - External production of items
    - Usage of standards
- Hardware and Systems
    - Inexpensive, yet reliable
    - Easy access to system internals
- Assessment Delivery
    - Autonomous (unproctored) testing, particularly outside of school (afternoon, week-ends)
    - Assessment security and data protection

We will give an overview on the requirements in this section. Possible strategies to cope with them are outlined here. Our solution(s) will be discussed in detail in extra sections.

The most important part of a generic assessment system is the content it delivers and the way it is produced. The system has to support a wide range of item types. At the same time, it should follow existing standards for content description and interoperability. This ensures that already existing content may be reused and newly created one may be used also for other purposes.

Our choice was to use the IMS Question & Test Interoperability Specification (QTI) standard [2] as a starting point. While this standard has certain deficits, it allows for reusing external (PC-based) tools as well as some assessment content. In addition, a later integration of our system into learning management systems like Moodle [3, 4] will be easier.

An important decision for any mobile application is the choice of the underlying hardware (and software) system. While it is in principle possible to develop hardware-independent mobile software – e.g., using browser technology – in reality this conflicts with the requirement of test security and autonomous testing.

So we decided to develop a native application instead of a web application and had to choose a platform. There are plenty of choices like Apple's iOS [5], Android [6], Samsung's Bada [7], Windows Mobile [8], and BlackBerry [9] to name but a few. As we aimed at a maximum of choices regarding the hardware and programming possibilities we chose the Android platform.

Assessment security was already introduced above as an important requirement. It means that it has to be as hard as possible to cheat while testing, to fake the results, to copy the items, or to gain access to other people's data. In addition, we wanted to enable unproctored testing, e.g., on afternoons or week-ends. Therefore we need to be able to lock down the mobile devices. All processes have to be completely under our control: assessment times, data synchronisation, updates and so on. We will describe our choices and the overall system design throughout the remaining sections of this paper.

2. System Architecture

To fulfil our requirements, we developed the system architecture illustrated in Figure 1. The two main applications, executed on the smartphone, are the "Assessment Controlling Application" and the "Assessment Rendering Application". Both applications are developed to be executed on Android based mobile devices. The Assessment Controlling Application is responsible for scheduling the actual assessments and for ensuring assessment security and data protection. It also enables the smartphones to connect to the Classroom Server and therefore also to the Central Server to get updates and export test results. The Assessment Rendering Application renders and illustrates the assessments. It also displays results and feedback to the user.

In addition to these applications, we have developed portable servers for each classroom to gather test results and to provide the necessary updates for the smartphones. Each of these classroom servers connects to the central server to synchronise the data. We will now describe these system parts in detail throughout the next sections.

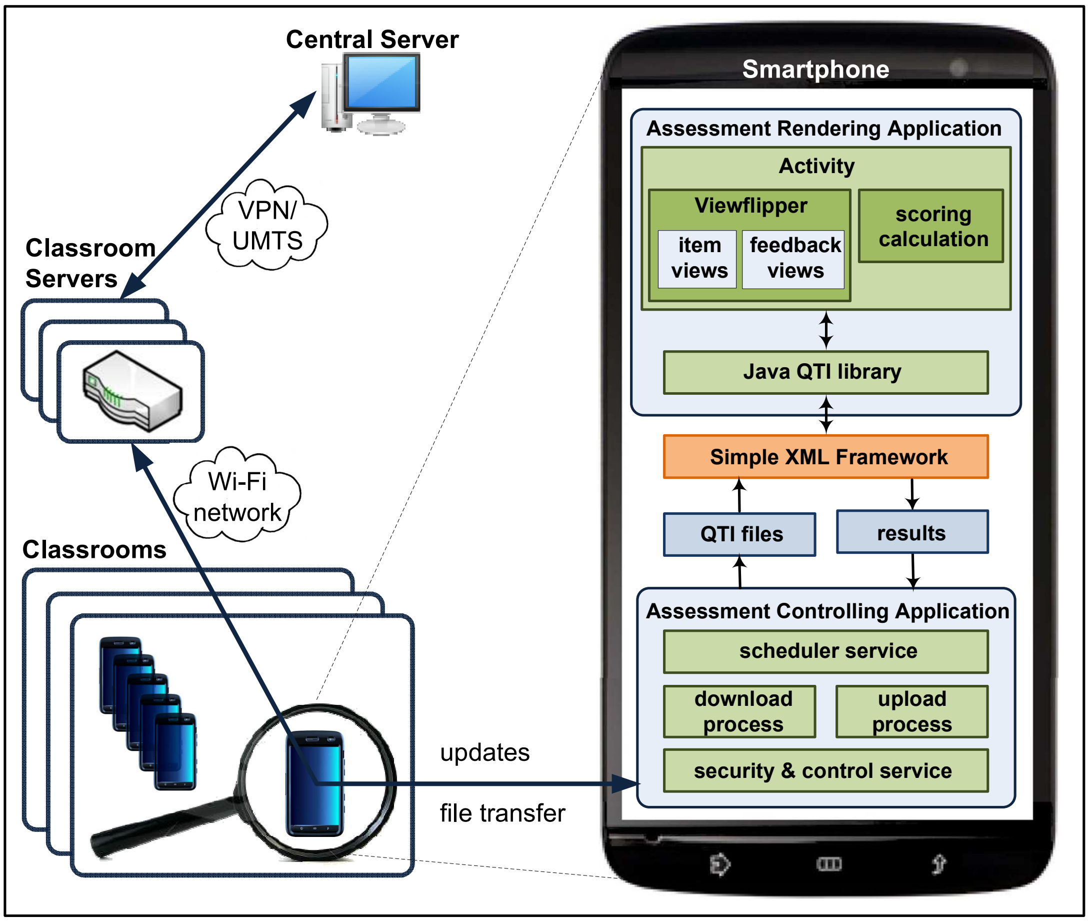

Fig. 1. The Mobile Assessment System

2.1 *Assessment Rendering Application*

Smartphones are built upon proven and existing software and technology and they are in principle compatible to usual PCs, but only up to a certain extent. There are some compatibility limitations such as screen- and memory size on the hardware side, and adaptations of the operating systems and programming languages that are directly supported on the software side. The content and the applications used for ambulatory assessment purposes therefore cannot be directly transformed from standard computerized assessment. Special design issues have to be considered. We deployed the Assessment Rendering Application upon Android system. The application delivers a framework to better support for offline ambulatory assessment and to minimize compatibility limitations.

2.1.1 Assessment Tests and Items

QTI is one of the most widely accepted specifications in the assessment and learning area. QTI defines a format for the representation of assessment content and results. It consists of a data model that defines the structure of questions, assessments and their results. Assessments can be built by combining several items (optionally combined in sections). Besides the actual item content, item descriptions specify whether the answers are used for response processing and whether feedback is provided within a test. QTI provides several templates for multiple-choice, matches, hot-spots, feedback etc [2].

2.1.2 Android Platform

Google's Android is a software stack for mobile devices. It is a combination of a free, open-source operating system for mobile devices, middleware and key applications. Android applications are written in Java as a programming language but executed by means of a custom virtual machine called Dalvik VM [10]. Dalvik is a JVM optimised for Android to cope with the mobile device limitations such as small memory, slow processor speed, and limited battery capacity. Every application runs in its own instance of the Dalvik VM [11]. Android applications consist of loosely coupled components and therefore are quite different from traditional (desktop) applications. The most important components of the Android SDK are Activities, Services, Content providers, Intents, and Widgets [12, 13]. The version we used for our implementation is Android 2.2.

2.1.3 System Design of the Assessment Rendering Application

The application is designed according to the Model-View-Controller (MVC) pattern [18]. This pattern separates the different aspects of the application and enables independent development, testing and maintenance. According to MVC pattern, the Activity components [17] are expected to take care of both the View and Controller tasks. In their role of a controller, they initialise the Model, provide the Accessors for getting data from the Model, and implement listeners. As a View they act like observers and notify automatically of any state changes.

Figure 2 illustrates the architecture of the application. It shows the main Activity of the application user interface, which includes a widget named Viewflipper [17]. A Viewflipper animates views that have been added to it. Views expose user interfaces for test items that handle screen layout and interaction with the user. To present the test items, elements such as widgets, buttons, images and texts are added to the hierarchical layouts within the respective view.

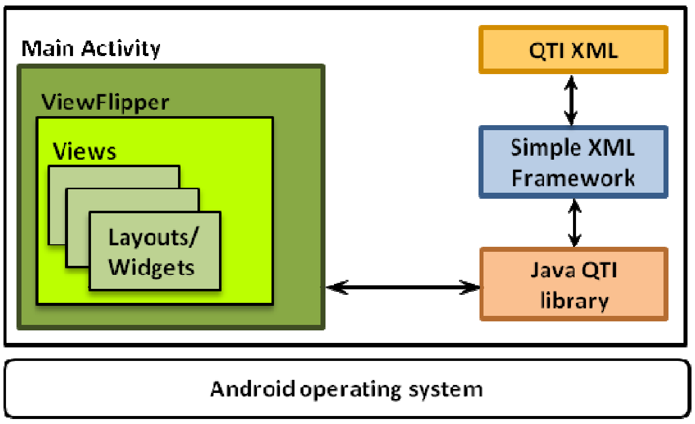

Fig. 2. Assessment Rendering Application

We use Simple [14], an XML serialization and configuration framework for Java, to process QTI XML. It supports the process of serialising Java objects to XML and de-serialising XML back to Java. Simple is provided for the Java platform and in comparison to the Android libraries available, it is easier to use and more flexible [15].

To enable interpretation and execution of the de-serialized files on a Java platform, we built a Java QTI library that reflects the QTI XML file structure. The Java QTI library serves as a domain model and illustrates both aggregations and lists of QTI tests and items. Differences to other existing libraries are described in the section on related work.

On Android, there are several ways to catch or handle the events generated by a user interaction with the touch screen of the smartphone [18]. For instance, touching a button is one possibility to generate an event within the user interface. A different possibility to capture the very same interaction (touching a predefined area of the screen) is to define hotspot points. We used the first method for implementing the widgets of the QTI interactive elements.

2.2 Assessment Controlling Application

This part of our Mobile Assessment System is a separate Android application that mainly works as a scheduler for executing instances of the Assessment Rendering Application, one for each assessment. Coordinating data transfer, managing security aspects and locking the phone are other jobs of this application.

An XML file describes the configuration and initialization settings. It contains information about the processes to be run and controls their behavior. A process can for example be an instance of the Assessment Rendering Application, a download- or an upload operation.

A scheduler file specifies the dates and time slots in which a process is to be run. There is also a timeout determined for test blocks. If the time is over, the test block is skipped. It is possible to define separate schedules and test materials for each user.

The Assessment Controlling Application establishes a wireless connection between the smartphones and the classroom servers. The connection is necessary to download new assessment content and software updates from the server to the phone and to upload the results of the assessment to the server. The smartphones need to be configured properly to be able to communicate with the classroom servers. For this purpose, there is a configuration file that contains information about the connection address, user name and password to access the server. It may also contain the user identifier. This id is used to distinguish between the uploaded files from different phones on the server. If no identifier is supplied, the device id is used which is unique for each phone.

As security plays a major role in unproctored testing, the Controlling System provides the essential security mechanisms to prevent gathering inaccurate data or faked results. An important element is changing the default home screen of the smartphone to our custom home screen. Unnecessary functionalities of the smartphones, which are not used for the assessments such as the camera, are permanently disabled. So the end-user is not able to manipulate the system.

2.3 Server Infrastructure

Besides implementing the client side of the system, developing the server infrastructure to support mobile assessment also posed a significant challenge. This includes an authentication system and a high security system for data synchronisation.

The classroom servers provide wireless LAN (WLAN) for the connection between the mobile devices and the classroom servers, based on the File Transfer Protocol (FTP). The connection between the classroom servers and the central server uses a point-to-point Virtual Private Network (VPN). It is not necessary to have a stationary internet connection in the classrooms as the classroom servers use mobile radio connections to connect to the central server. The smartphones are configured to connect to their classroom server on a regular basis to export the test results.

We decided to support two different connections between the classroom servers and the central server. The first (and preferred) possibility is to use the school's network and integrate the classroom server(s) into that network. Each classroom server establishes a connection to the central server via a VPN to ensure data security. The second networking possibility comes into play if the first connection fails or is not available in a certain school. The classroom servers are equipped with mobile broadband modems that support the advanced Universal Mobile Telecommunications System (UMTS) standard. This autonomous internet access provides the secondary VPN connection, which may also be run in parallel to the first.

The central server is equipped with two different network interfaces, one for each VPN connection. This is necessary to support both connections (UMTS and school network), because they are configured and rooted differently. Once at least one connection is established, it can be used not only for data synchronization or applying updates, it is also used to get secured remote access to the mobile server for administration or adjusting settings. For example, it might be necessary to delay the regular file transfer because of high network traffic caused by running backups.

The central server also provides a simple website. The website e.g. provides easy access for uploading updates or other files. These files will be transferred to the mobile server immediately and redistributed to the clients.

3. Case Study: FLUX

The project Assessment of Cognitive Performance FLUctuations in the School ConteXt (FLUX) deals with daily fluctuations in children's cognitive performance in the school context using mobile assessment techniques. The cognitive tasks such as working memory, processing speed, and updating as well as questions about mood, motivational aspects, and situational issues are being processed twice to four times a day for several weeks [19].

One idea behind the FLUX project is to assess children's cognitive performance in daily life. This so-called "ambulatory assessment" approach aims at increasing ecological validity of measures by assessing children in their naturalistic contexts. Another aim was to administer cognitive tasks and to assess children's answers and reaction times. After establishing tasks with satisfactory psychometric properties for this purpose, daily fluctuations in cognitive resources will be described and their role for so-called *fluid* intelligence and school achievement will be investigated at the group and the individual level applying multilevel as well as time series analyses [19]. The Dell Streak 5 was selected as a smartphone for the FLUX project.

The assessment tests and items in FLUX are defined in QTI format. The items are simple and the interactions are mostly choice, text entry, slider, hotspot, and graphic order. Figure 3 shows sample items that have been designed in FLUX. Please note that all rights on the images are with the FLUX project.

Item 1 is a single choice item. The participant has to decide whether the images shown on the left and on the right side are identical. The answer is given by pressing one of the buttons in the lower part of the screen. Doing so, the next item is shown automatically. In case of no answer the next item is shown after the timeout specified in the item definition file. Items 2 and 3 are multiple choice items, which illustrate *working memory tasks* [20]. The items assess the ability to simultaneously maintain and process information.

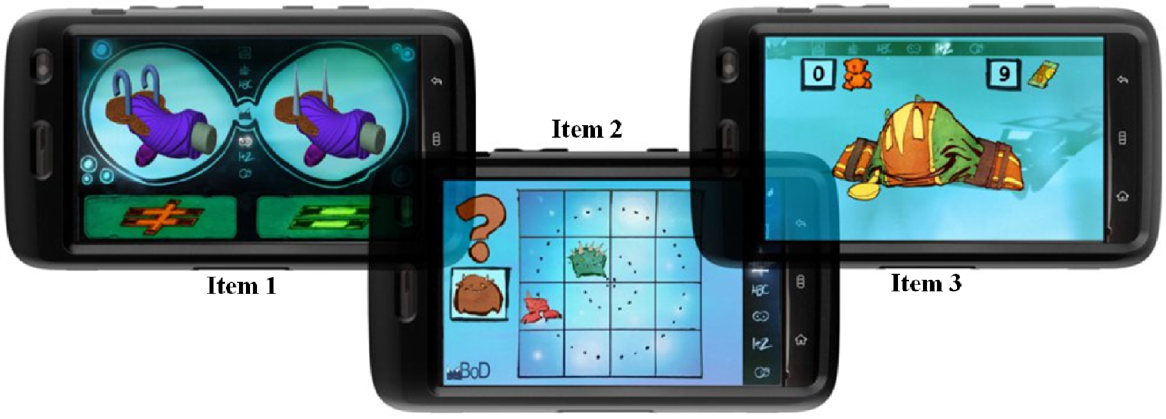

Fig. 3. Sample Items

Item 2 shows images placed in the grid at the beginning of the test. Afterwards, movements are shown. The test-taker has to remember the updated positions and place the images in the grid accordingly. In Item 3, the test-taker hast to add or subtract numbers displayed in random order.

The findings of the FLUX study offer interesting insights into the extent to which using mobile techniques might be applied for supporting ambulatory assessment. It also highlights a number of major challenges that this format raises for the design of such resources.

4. Conclusion and Outlook

We have presented an integrated approach to enable and support mobile assessment in schools. Special emphasis was put on the usage of open standards and on test security. Our solution has been tested in several smaller and in one larger study where more than 120 students are tested three times a day over several weeks.

We do not only provide software but integrate it into a generic approach consisting of a hardware/software system with a central server, multiple classroom servers and the smartphones (or tablets) used for the assessment. We aim for unsupervised assessment; the phones can be handed over to the students for the assessment period without manual intervention necessary.

Our future plans are manifold. We want to extend the support for additional item types. This may go beyond the QTI standard, so we have to investigate additional standards usable for our purpose. Assessments are an integral part of learning. We want to extend the possibilities of the system to better support e-learning activities. Parts of our software development, especially for the test security, have been thoroughly tested only on one type of smartphone (the Dell Streak). We plan to extend the basis of supported smartphones, especially to more recent models. However, having in mind the innovation speed in the mobile sector, this is an on-going effort without end. Moreover, we plan to include additional hardware support into our system. Most smartphones offer sensors like GPS or a camera that might be used for assessment purposes as well.

5. Related Work

As we mentioned above, important parts of our system are based on QTI XML. We have implemented an Android Java QTI library. A similar work called Mobile QTI playr [21] has been done at the University of Southampton. This application is an adapted version of a QTI engine [22]. The QTI engine uses a web interface to browse through an assessment. It transforms QTI XML files via XSLT [23] in two steps into XHTML files. An XHTML renderer displays these output files in a browser. In the *Mobile* QTI playr, the XHTML renderer has been redesigned and re-implemented for mobile devices. To display XHTML files, it is necessary to install a mobile web server, i-jetty [25], on each Mobile device [21]. The QTI library that we implemented uses an XML serialization framework, which is Android compatible, to de-serialize QTI XML directly to Java objects and to serialize the objects back from Java to XML. The items of the assessment are implemented in Java using the Android SDK. So they can be run directly as an App on any Android device.

Our design decision was mainly based on test security considerations. This is easier to accomplish having full control over the assessment application. In addition it is desirable to have as few processes run on the smartphone as possible so that full access to all resources can be ensured for the main process.

References

[1] A. Holzinger, A. Nischelwitzer, M. Meisenberger (2005). In proceeding of: Pervasive Computing and Communications Workshops: Mobile phones as a challenge for m-learning: examples for mobile interactive learning objects (MILOs)

[2] IMS Global Learning Consortium, Inc. (2011). IMS Question & Test Interoperability Specification, http://www.imsglobal.org/question/.

[3] Respondus Inc. (2011). Question Import/Export Format: Respondus QTI Importer, http://moodle.org/mod/data/view.php?d=13&rid=986.

[4] Moodle Trust (2011). IMS QTI 2.0 format, http://docs.moodle.org/20/en/IMS_QTI_2.0_format.

[5] www.apple.com/de/ios/

[6] http://www.android.com/

[7] www.bada.com/

[8] www.microsoft.com/windowsmobile

[9] ca.blackberry.com/apps-software/blackberry7/

[10] Reto Meier (2010). Professional Android 2 Application Developement (pp. 14).

[11] Android Open Source Project (2011). Application Fundamentals, http://developer.android.com/guide/index.html.

[12] Android Open Source Project (2011). What is Android?, http://developer.android.com/guide/index.html.

[13] Sayed Y. Hashimi, Satya Komatineni, Dave MacLean (2010). Pro Android 2 (pp. 31-33).

[14] Simple XML Serialization Inc. (2011). Tutorial, http://simple.sourceforge.net/.

[15] Fabrizio Chami (2011), Android XML Binding with Simple Framework Tutorial, http://simple.sourceforge.net/articles.php.

[16] Reenskaug, Trygve (2011). MVC XEROX PARC 1978-79, http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html.

[17] Android Open Source Project (2012). Reference, http://developer.android.com/reference/packages.html.

[18] Android Open Source Project (2012). Android 4.0 release Document, http://developer.android.com/guide/topics/ui/ui-events.html.

[19] DIPF (2011). FLUX Project, http://www.idea-frankfurt.eu/homepage/idea-projects/projekt-flux.

[20] Akira Miyake, Priti Shah (1999). Models of Working Memory.

[21] Pei Zhang, Gary Wills, Leser Gilbert (2010). IMS QTI Engine on Android to support Mobile Learning and Assessment. http://eprints.soton.ac.uk/271485/.

[22] http://wiki.qtitools.org/wiki/QTIEngine

[23] XSL Transformations (2012). http://www.w3.org/TR/xslt.

[25] code.google.com/p/i-jetty/