A General Framework for Developing Training Apps on Android Devices

Jeremy Ludwig, Robert Richards, Bart Presnell, Dan Fu
Stottler Henke Associates, Inc.
San Mateo, CA
ludwig@stottlerhenke.com, richards@stottlerhenke.com
bpresnell@stottlerhenke.com, fu@stottlerhenke.com,

ABSTRACT

Handheld applications (apps), such as those run on Android and iPhone devices, hold the possibility of revolutionizing military training by increasing the availability and engagement of training material. This paper describes progress on software design and development towards a general framework for deploying Android training apps. A primary objective is to allow nonprogrammers to reuse existing content to create training apps that make full use of the capabilities offered by mobile devices. The described prototype implementation includes a web page where the end user fills out a form, uploads content, and receives an email with a link that they can follow (and share with others) to download their app directly to their device. The main contributions of this paper are: The requirements that led to the framework design, the description of the implemented framework, and a summary of qualitative feedback received from targeted demonstrations. While this framework has been developed with a focus on military training, it is broadly applicable in a civilian educational setting as well.

ABOUT THE AUTHORS

Jeremy Ludwig, Ph.D. is a scientist and project manager at Stottler Henke, where his research areas in artificial intelligence include mobile & adaptive training systems, intelligent user interfaces, behavior modeling, and planning & scheduling.

Robert Richards, Ph.D. is a principal investigator and project manager at Stottler Henke. He has a wide range of research and application area interests, ranging from: training system development including intelligent tutoring systems; applying automation and artificial intelligence techniques to decision support tool development for life-critical situations; to development of advanced critical chain project management techniques.

Bart Presnell is an engineer at Stottler Henke, where his research areas in artificial intelligence include game based training systems, behavior modeling, and planning & scheduling.

Dan Fu, Ph.D. is a group manager at Stottler Henke. His research interests include serious games and intelligent tutoring systems. He holds a computer science Ph.D. from the Univ. of Chicago and a B.S. from Cornell Univ.

A General Framework for Developing Training Apps on Android Devices

Jeremy Ludwig, Robert Richards, Bart Presnell, Dan Fu
Stottler Henke Associates, Inc.
San Mateo, CA
ludwig@stottlerhenke.com, richards@stottlerhenke.com
bpresnell@stottlerhenke.com, fu@stottlerhenke.com,

INTRODUCTION

The use of electronic learning is well established in US military training programs. Mobile learning (m-learning) takes this one step further (Traxler, 2009), enabling military personnel to access instructional content anytime, anywhere (Lin and Andrew, 2011; Matthew, 2010). The intuition behind m-learning is that student achievement will be improved by making it easier for students to access course content and communicate with one another. Motivated students carrying their study materials with them everywhere on devices they already use heavily will be more likely to make use of the instructional content – i.e., they will study more because it is easy. Students can take their curriculum content to their work areas [e.g., a mechanic could take it to the shop or a pilot could take it to an aircraft (on the ground)] to see the actual context of a lesson. The availability of mobile devices can also support the transition of training material from the schoolhouse to deployment by allowing students to take their curricula with them out into the field, onboard a ship, etc. Finally, these devices can better-deliver reference materials, replacing stacks of three-inch-thick binders with Portable Document Format (PDF) files that can be automatically updated. Moving from intuition to evidence, Pollara & Broussard (2011) in a careful review finds positive benefits of m-learning across studies such as increased achievement shown by higher test scores and enhanced sense of engagement among learners.

An initial problem with m-learning is the need to update existing e-learning content for use on mobile devices. Developing custom apps is expensive, often only covers a limited set of content, and often leaves individual teachers with limited control over the content. As a result, a standard method of developing m-learning apps is to repackage existing course materials. Using the open-source Moodle course management system (www.moodle.org) as an example, there are apps on iOS and Android devices that reformat the Moodle web pages to increase their usability on mobile devices. That is, an instructor can create a Moodle curriculum for the desktop and this material will be available on mobile devices without any additional effort on the instructor's part. However, this material competes for a learner's time and attention with millions of other mobile apps and games. Therefore merely repackaging content might not be sufficient. The app also needs to motivate learners to engage with it.

Another challenge with m-learning is pointed out by Kinash et al. (2011). They found that making content available on mobile devices is not always by itself enough to ensure improved student learning. While this has been found to improve outcomes for specific courses (e.g., McConatha et al., 2008), it cannot be assumed in general. Kinash et al. (2011) also found that "While there are numerous educational apps available for download, these are not what this group of students downloaded. Those students who downloaded apps, downloaded games." Their research suggests that making content available is not enough – it needs to be engaging as well.

This paper describes how we designed and developed a general framework for deploying Android training apps that addresses these two challenges. The primary objective for this framework is to allow non-programmers to reuse existing content to create training apps. The implemented solution includes a web page where the end user fills out a form, uploads content, and receives an email with a link that they can follow (and share with others) to download their app directly to their device.

The deployed app delivers the training content using motivational features found in popular "casual games" such as Angry Birds (http://www.rovio.com/). The distinguishing features of casual games include both simple gameplay and support for playing the game in short bursts; such features have resulted both in a wide audience appeal and in utilization in military training domains (e.g., Sanchez, 2010). The deployed app also includes game-based multi-player support to allow for

impromptu contests that can tap into students' competitive instincts.

The remainder of this section describes existing work relating to this functionality. In the Methods section we list the requirements we developed based on discussions with military instructors. Following this we provide an overview of the authoring tool and underlying framework implementation. The Results section includes a graphical description of an example app generated in this framework and a summary of qualitative feedback received from a demonstration. The Conclusion summarizes the major themes of this paper and outlines future work.

The main contributions of this paper are: The requirements that led into the framework design; the description of the framework; and the qualitative feedback summary. While this framework has been developed with military training in mind, it is still broadly applicable to civilian training.

Related Work

There are two categories of related work. The first consists of app authoring tools accessible to authors who are not computer programmers. The second includes existing apps that have functionality similar to what is described by the requirements.

App Authoring Tools

While general Android app builders like App Inventor (www.appinventor.googlelabs.com) and Sencha Touch (www.sencha.com) would not be considered relevant because of the relative difficulty of authoring, there are a number of more constrained authoring tools. Of these, AppBreeder (www.appbreeder.com) has an authoring kit called AppTeach designed specifically to author training apps. This tool supports developing apps that contact the instructor, provide event notification, and show videos, photos, and lessons. However, this authoring tool does not meet the requirements we present for developing training apps. Specifically, it does not support test-prep questions, motivational concepts from games, etc.

Existing Apps

There are a number of specific apps aimed at particular education topics. One example is Test Assassin (www.testassassin.com) for LSAT prep. It contains tutorials, drills, and practice problems. However, most of these types of apps are tuned to one specific subject and unrelated to the general-purpose solution we are investigating. Two apps we found to be more similar to what we investigated during this project are Flash Card Maker Pro (www.flashcardmakerpro.com) and My Pocket Prof (www.mypocketprof.com). Flash Card Maker Pro enables users to import their own flash cards from a file through a process similar to our input requirements; however, it does not help the instructor to either author a custom app or share resources. My Pocket Prof allows students to create and share notes, documents, test questions, etc. It is aimed at supporting student collaboration toward developing and sharing the course material. The described framework combines functionality from both of these types of apps with a focus on supporting military training.

METHODS

In this section we list the set of requirements that were developed based on discussion with instructors and on prior experiences developing mobile training apps. Then we present an overview of underlying implementation.

Requirements

The first goal is that the system should have utility for military users – these include people in training, acting as instructors, in the field, etc. The first five requirements were developed with this goal in mind. Only requirements that can be supported with the described authoring capability are included.

1. The app needs to display reference documents in order to replace stacks of binders.
2. The app needs to be updateable so that new reference documents can easily be pushed out to the end user.
3. The app should support self-study for multiple-choice questions tests as these are commonly given to military personnel throughout their careers.
4. The app should also support a multi-player quiz show interaction mode to allow for small-group competition and interaction. Self-study often occurs at home in the evening – discussion with trainees indicated it would be useful to have someone to virtually study with.
5. The framework needs to include simple techniques for increasing the user's engagement. One of the interesting things about popular mobile games is their use of a scoring system that helps keep users engaged in trying to both unlock all of the levels and replay past levels to achieve a better score. The apps should automatically include simple techniques such as these.

The second goal, which is that apps be easy to author, is based upon both discussions with military training personnel about this specific project and our general

experience working in the military training domain. The following numbered requirements were developed out of this goal:

6. Support a simple authoring interface so that any individual with content can create a basic app. This minimal amount of content will take the form of:
   a. Meta-information – the title of the app, an icon, etc. to allow a limited amount of customization.
   b. References – reference documents were a requirement of all of the stakeholders with whom we spoke.
   c. Questions – multiple choice, fill-in-the-blank, or true/false questions typed up in an Excel spreadsheet or Word document are the primary forms of readily available content.
7. Be easier to use than existing app builders.
8. Include adaptive drill / practice techniques that do not require any additional authoring.
9. Include a reporting system that does not require any additional specification.
10. Include pre-specified games that can ingest the available content.
11. Game-based features, such as scoring and locked content, should work without placing any additional burden on the app author.

Framework Implementation

This section briefly describes the underlying implementation. The section is broken into two subsections. First we review the app authoring tool with which the end user works to create a training app. Second we provide an overview of the training app framework implementation, which is a collection of Java code that converts the supplied content into an Android app.

App Authoring

This section describes the prototype app authoring tool and provides a brief overview of implementation. The goal is to allow non-programmers to supply content and create an app. The prototype solution includes a web page where the end user fills out a form, uploads content, and receives a link via email that they can follow (and share with others) to download their app. The tradeoff for this simplicity of authoring is that the content is limited to what can easily be predefined.

From an author's perspective, two steps are involved in deploying a training app. The first step is to gather the content. As part of gathering the content, the author must:

1. Decide on an app name.
2. Find or create a PNG image file that will serve as the app's icon.
3. Identify the PDF or PowerPoint references to be included and create a zip archive of these references.
4. Generate XML question files:
   a. Identify and group the multiple-choice test questions to be included in the app.
   b. Place these questions into the provided Excel template (shown in
   c. **Figure 1**) via exporting from another program, cut and paste from a Word document, or typing.
   d. From the **Tools** menu in Excel select **APP1 | Write XML File** while the spreadsheet is open. This will generate a XML question file for the group of questions.
5. Create a zip archive of XML question files.

	A	B	C	D	E	F	G
1	**Group Name**	Chapter 1					
2							
3	**Question**	**Answer**	**Response 1**	**Response 2**	**Response 3**	**Response 4**	**Image File**
4	This is a multiple choice question	1	a. Right	b. Wrong	c. Wrong	d. Wrong	
5	A multiple choice question with fewer responses	2	a. True	b. False			
6	A fill in the blank question.	Right					

Figure 1. Excel template for creating test questions

The second step is to create the training app. The author goes to a web page as shown in
Figure 2, where they enter the title and upload the icon, PDFs, and XML questions. When the **Send** button is pressed, the app is created and the download location emailed to the given email address. Once the app is downloaded to the Android device, it can be installed from the *Downloads* app included with Android.

The underlying implementation makes use of an HTML form to allow the author to provide the content for the app. This data is sent to an application server where a servlet processes the given information. The servlet produces a copy of the Java code that makes up the training app framework. The servlet then updates the application manifest with the given title and copies the icon, PDF references, and XML questions to the appropriate locations. After all of the modifications have taken place, the servlet executes an Ant process that compiles the customized framework code and creates a signed, executable, app. The app is then renamed and moved to a location where it can be downloaded. Finally, the servlet sends an email to the given address containing the HTML link to the training app.

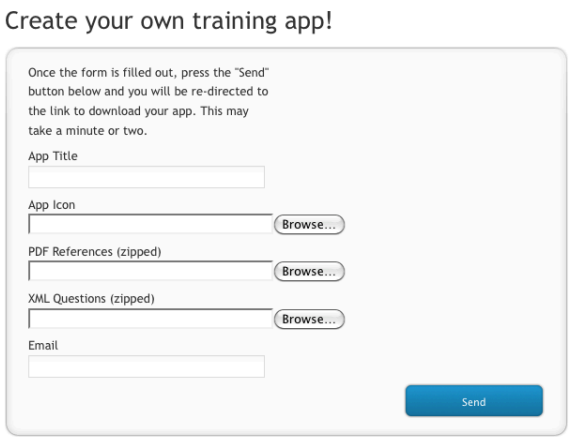


Figure 2. Web-based app authoring

App Framework

The training app framework is a collection of Java code that converts the supplied content into an Android app. The framework was developed in Java using the Android Software Development Kit (SDK) and the Eclipse and Ant development environments. The bulk of the training app framework is composed of a number of Java classes that form Android *activities*. Each *activity* represents a single, focused, user-interaction. For example, the main screen of the app is an activity with pre-determined content. However, most of the developed framework activities are content-driven to support training in any domain. The activity that displays the list of references is a good example. It creates the list dynamically based on the contents of the internal assets of the application. Similarly, the multiple choice questions displayed in the Flash Cards or Quiz Show activities are dynamically created from the internal assets.

The training app framework also makes use of external activities. The primary example of this is displaying PDF and PowerPoint files. In these cases, a request is launched that asks for an available activity that can perform the desired function. External services, Android programs without user interfaces, are also used to support the framework. The only example of this is the use of AllJoyn to support ad-hoc WiFi networking for the multi-player Quiz Show (www.alljoyn.org).

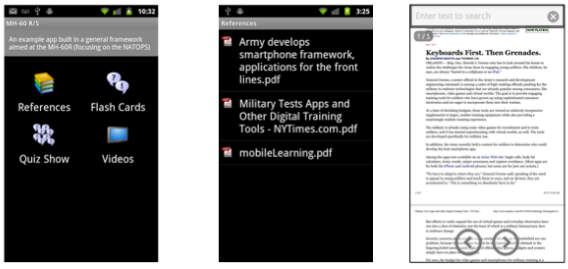
RESULTS

The Results section includes a graphical description of an example app generated in this framework and a summary of qualitative feedback received from a demonstration.

Example App

We developed an example training app in the framework for helicopter pilots and aircrew. Specifically, we used the general framework combined with the course-specific content to create an Android training app. The example app is presented as a series of choreographed screenshots in an attempt to capture an interactive process in printed form. While screenshots from the smaller Google Nexus S phone are provided in this paper, the app framework was primarily designed for larger screens and demonstrated on Motorola Xoom tablets. Additionally, helicopter-specific questions and references have been replaced with example questions and references for this paper.

The main page and the *References* activity are shown in **Figure 3**. The main page allows the student to log in and presents a grid of buttons that will start the other activities. This page also displays the title of the app along with optional restrictions, description, and logo. The *References* activity bundles the relevant training manuals and slides and makes use of external activities to actually view the material. This leverages all of the features of existing PDF viewers, such as search.



Press *References* to bring up the list. **Click on an item to view.** **Search in the document.**

Figure 3. Main Page and References activity

The *Flash Card* activity is shown in **Figure 4**. It presents test questions to individual users. Questions are grouped based on the question files created by the instructor. Initially, only the first group is unlocked (represented by a green icon). Once the user achieves at least a 1-star rating (70%) on a group then the next group in the sequence is unlocked. Currently this level of performance is hard-coded, though the app author could set the level in future versions. A user selects an unlocked question group as seen in the left image and is asked each question in order. The center three images depict answering an individual question, where the first image asks the question, the second shows the available answers, and the third illustrates when an incorrect answer is selected. A correct answer is shown in green and a wrong answer is shown in red

with the correct answer highlighted in green. After all of the questions are answered the user will see a summary of their score as shown in the rightmost image. The score page also contains buttons to go back to the list of question groups, repeat the same group to try to achieve a better score, or to proceed to the next question group. Scores are saved across training sessions.

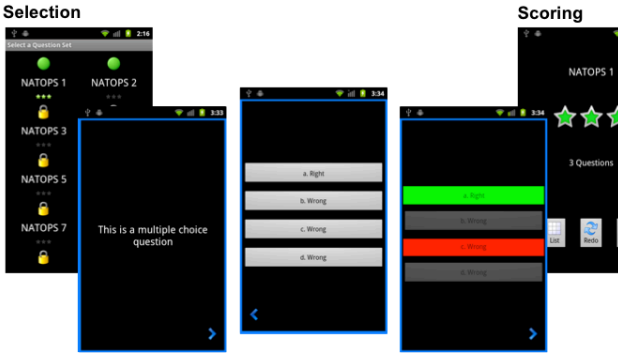


Figure 4. Flash Card activity

The *Quiz Show* activity presents the same content as *Flash Cards* but allows players to compete against one another to see who answers the questions first. The user starts the activity and selects to **Start** or **Join** a game as shown in
Figure 5. In the start sequence the user selects a question set and waits for others to join. Technically there are no limits to the number of people who can join, but ad-hoc networks initiated on the device are limited to five external connects (i.e., six players). Alternately the join sequence automatically searches the network for any games that have been started on other devices.

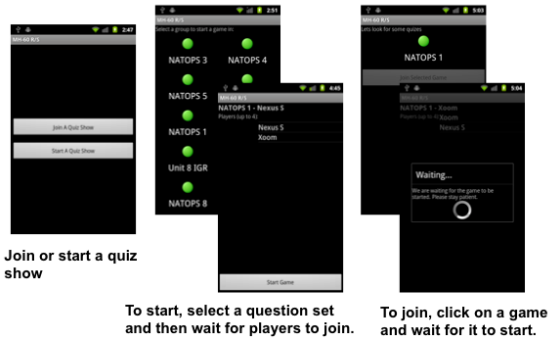


Figure 5. Setting up the Quiz Show activity

Once setup is complete, the game-play is very similar to the *Flash Card* sequence with the exception that the first person to answer gets the point. Other changes include an answer screen that is shown to all of the non-winners and a running scoreboard at the top of the questions. When all of the questions in the group have been answered, all players see a screen with a larger scoreboard and the game ends.

Feedback

The demonstration included more formal presentation and discussion with 14 participants as well as informal discussions with others who saw the tablets and wanted to know more. Most of the participants were educators, instructor pilots, test pilots, or advanced students. None of the participants was an introductory student. Three Motorola Xoom tablets and two Google Nexus S phones were available to the participants for hands-on testing. The demonstration participants were generally enthusiastic about the possibility of using phones and tablets, not only for training but also in carrying out their normal duties. Other than asking where the power button was, participants had no further questions about how to use the software when handed a tablet or phone. In every case, participants also had ideas for other apps they would like to have on the devices. GPS maps, charts, and approach plates topped the list.

There was a strong interest in the training app; especially the *References* activity coupled with a more advanced authoring tool that supported updating the references as they change. There was general interest in self-study for various certification and career advancement tests, though the participants could see even greater utility for entry-level students. At first, participants were not sure about the utility of the *Quiz Show* outside of entry-level training where study groups are more likely to occur. However, participants in the meeting pointed out that the best use of this feature at their career stage would be for it to support social studying with colleagues in the evening from home in order to help make a dreary task more interesting. This priority points towards a need in future work for server-based networking as well as for peer-to-peer networking.

Several other improvements were also suggested. The participants make heavy use of computer-based-training lessons and would like to see the lessons and associated tests be made available on the device to free them from having to work in the computer labs. Another feature of interest was our providing more organizational structure to the references, given the volume of manuals and presentations they deal with. Finally, participants would like to see the framework expanded to include other smartphones. The implemented version of the framework supports Android phones and tablets.

Participants would like to see this expanded to include iOS devices as well.

CONCLUSION

This paper described progress on software design and development towards a general framework for deploying training apps for Android devices based on existing content. The specific results of the paper were: Generating a set of requirements for the framework, the development of a prototype framework for creating training apps based on existing content, and feedback from a demonstration of an example app. While the results from this prototype are promising, there will be significant future work required to develop a complete training app framework and demonstrate its utility.

The first set of tasks involves iterative improvement to the existing framework and authoring tool. Some examples of framework improvements are polishing the user interface (e.g., prompts, sounds), expanding existing activities (adaptive drill, more entertaining multi-player, reference organization), adding new activities (lesson, videos) and including more social networking (sharing of user-created content, study groups, facilitating communication). The authoring tool features both iterative improvements and significant new functionality. Iterative improvements include allowing the author to better-customize the app (logos, restrictions, etc.) and include more types of content for new activity types. More significantly, the app authoring website needs to be expanded to include the ability to manage and update apps in addition to just creating them.

A second set of tasks involves carrying out a pilot program and evaluation to demonstrate the efficacy of the device with respect to military training. These tasks include selecting a site for a pilot program and identifying the end users, how they will be able to make use of the Android device, and how any security issues will be addressed. Tasks also include developing the criteria that will be used to evaluate the app, carrying out the pilot program, and analyzing the results.

ACKNOWLEDGEMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. W31P4Q-11-C-0006. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

DISTRIBUTION STATEMENT A: Approved for Public Release, Distribution Unlimited.

REFERENCES

Kinash, S., Brand, J., Mathew, T., & Kordyban, R. (2011). Uncoupling mobility and learning: When one does not guarantee the other. In *Enhancing learning through technology: Education unplugged: Mobile technologies and web 2.0*.
Retrieved from http://epublications.bond.edu.au/tls/25

Martin, L., & Martin, A. (2011, May 1). Keyboards First. Then Grenades. *The New York Times*.
Retrieved from
http://www.nytimes.com/2011/05/02/technology/02wargames.html?pagewanted=all

Matthews, W. (2010). Apps That Answer Military Needs; Push Is On To Make Smart Phones Work for Combat Troops. *Defense News, 2010, April 18*. Retrieved from:
http://www.defensenews.com/story.php?i=4588104

McConatha, D., Praul, M., & Lynch, M.J. (2008). Mobile learning in higher education: An empirical assessment of a new educational tool. *Educational Technology, 7*(3), 15-21.

Motiwalla, L. F. (2007). Mobile learning: A framework and evaluation. *Computers and Education, 49*(3), 581–596.

Pollara, P., & Broussard, K. K. (2011). Mobile Technology and Student Learning: What Does Current Research Reveal?. *International Journal of Mobile and Blended Learning (IJMBL), 3*(3), 34-42.

Sanchez, A. (2010). Taking Cues from Industry: Using Casual Games for Learning at DAU. *Learning Solutions Magazine. 2010, November 30*. Retrieved from:
http://www.learningsolutionsmag.com/articles/595/taking-cues-from-industry-using-casual-games-for-learning-at-dau

Traxler, J. (2009). Learning in a Mobile Age. *International Journal of Mobile and Blended Learning, 1*(1), 1-12.