AXLNET MOBILE:
USING A MOBILE DEVICE FOR LEADERSHIP TRAINING

Jacquelyn Ford Morie*, J. Galen Buckwalter
University of Southern California
Institute for Creative Technologies (ICT)

ABSTRACT

The AXLNet Mobile project builds upon the web-based leadership classroom training application, AXLNet, originally created by ICT and ARI in 2006. We ported both the functionality and pedagogical approach of AXLNet to an Apple iPhone in 2008-2009, and tested the resulting application for usability with a population of 46 Captains and 1st Lieutenants at Ft. Leonard Wood. Initial findings indicate supplemental training on the mobile device is an acceptable and possibly more engaging delivery method for today's troops.

1. INTRODUCTION

The AXLNet Mobile application is a supplemental work created to accompany the original Army Excellence in Leadership (AXL) project, a collaboration between the Army Research Institute (ARI) and the ICT (Hill et al., 2006). The original project provided a case-based methodology for new commanders to study complex decision-making scenarios, where critical thinking demanded leadership skills as well as situational and cultural awareness. AXLNet was designed for in-classroom use, was deployed via the Web 2.0, and provided a means for guided analysis of the factors and individuals involved in the scenarios.

AXLNet is a powerful educational tool developed to teach soldiers about leadership by presenting them with difficult situations and asking them to make decisions and judgments about those situations. The task in developing the AXLNet Mobile application was to transfer as much functionality of the Web-based AXLNet to a convenient mobile device, where the lessons could be accessed 24/7. These functions included: the case-based methodology, the guided analysis, the ability to go back and interview characters in the scenario to gain different perspectives on the situation, open-embedded questions about the factors that lead to problems, and finally a way to collect the student's interaction data for later upload to the AXLNet teacher-analysis tools. This work was done in 2008 and 2009, and was followed by a usability study with young commanders at Ft. Leonard Wood, Missouri, detailed in section 3 of this paper.

1.1. Design Overview

AXLNet is composed of distinct modules, each with a particular theme and pedagogical goal. Although AXLNet has several of these modules available, as well as a powerful tool that allows educators to author new modules, recreating the entirety of this content and functionality was beyond the scope of our project. Instead, we focused on porting two of the modules, "Power Hungry" and "Tripwire," to the mobile platform. In adapting these modules to mobile, we aimed to maintain the original content and pedagogical approach as faithfully as possible. More information on the procedure we followed to transition the content can be found in the section on Module Porting.

One of the hallmarks of AXLNet is its highly multimedia presentation of content. AXLNet modules can contain a variety of media objects, including video clips, audio clips, and images. "Power Hungry" and "Tripwire" both start with a short video that depicts characters facing a difficult situation and the consequences of the choices that they make. Each module contains a number of supplementary video clips that provide additional information about the characters and their motivations and understanding of the situation. We wanted to maintain this media-rich experience in our port of these AXLNet modules.

AXLNet also connects to a backend system which stores the students' answers to questions within AXLNet modules. AXLNet includes an analysis tool, which organizes and presents statistics about these responses for the course instructor in real-time. This allows the instructor to engage the students with knowledge of individual student's answers to the questions and to lead the discussion while considering the classes' collective response to the material. Although recreating the analysis tools or integrating AXLNet Mobile responses into the main AXLNet database were not feasible within the scope of this project, we felt it was important to capture this user data so that it could be incorporated into such an analysis and presentation system in the future.

We chose the iPhone (and its sibling device, the iPod Touch) as the mobile platform with which we would realize our goal of bringing these features to a personal, always-available environment. The iPhone platform was chosen because it is a popular consumer-level device that fills the technical requirements needed to effectively implement the system that we envisioned. The iPhone and iPhone Touch have been commercially available since June of 2007. The platform includes technical features such as wireless support, media playback, and includes a mobile web browser that is much more capable than most competing products. This functionality is contained in a sleek mobile phone, designed to be carried on the user's person and easily accessible at any time.

Built upon the success of the iPod line of products, the iPhone platform includes excellent media playback functionality compared to many other smartphones. We took advantage of the device's capabilities to include the video clips that focus and support the ported content modules. The iPhone platform also supports a wireless connection to the Internet, which mitigates the need for phone carrier connectivity (with associated monthly carrier fees). This Wi-Fi functionality is used primarily for the collection of user response data.

2. AXLNET MOBILE DESIGN

2.1. First Iteration

Originally, a version of AXLNet Mobile was produced that ran directly from the web in a form that could be accessed and used through the browser integrated into the iPhone platform. This first iteration of the project was based on the robustness of the iPhone's Safari browser, which interprets Javascript at a level beyond what most other devices of similar size currently support. The module content was hosted on a webserver as an HTML document and a collection of linked Quicktime video files, and streamed over-the-air to the user's iPhone browser when a user requested the page. This version of AXLNet Mobile was essentially a web application, just like the original AXLNet web application, but had some specific advantages over the original in the mobile context.

The main AXLNet website is accessible from the iPhone's Safari browser, but its usability is enormously reduced on that platform. Because AXLNet was designed with full-sized desktop-application browsers in mind, the size and layout of the page is not well-suited to viewing on the smaller screen of a mobile device. Legibility and ease of navigation are reduced to unacceptable levels. Additionally, the iPhone platform's Safari browser is not capable of running embedded videos, which are used throughout the "Power Hungry" and "Tripwire" modules on AXLNet. The iPhone platform's browser cannot play the videos at all, because it doesn't have the same codec support that desktop-application browsers have.

For these reasons, we recreated the content of the "Power Hungry" module in a more mobile-friendly format that can be read and navigated on the small screen of the iPhone. This version of AXLNet Mobile was written primarily in HTML, Javascript and CSS with a backend for data collection that uses PHP and MySQL.

The content for the module resides in a single HTML document, divided into nodes that are displayed sequentially on the iPhone's browser by a controller application written in Javascript. The HTML document was created by combing through the original AXLNet version of the module we were recreating and copying text and embedded media into the appropriate nodes of the HTML. Cascading Style Sheets were used to display the content in each node in a format that would be usable on the iPhone browser, and which incorporated interface elements that are familiar to users of the iPhone platform. The Javascript controller proscribed the user's progress through the module, displaying the contents of each node in the proper sequence.

As the user proceeded through the course, many nodes contained prompts for user input. Generally, this took the form of multiple-choice questions and textual input that could be entered using the iPhone's built-in virtual keyboard. When user input was received, the Javascript controller sent the data to a PHP page which recorded that piece of user data in a MySQL database.

This first iteration was successful at addressing many of our goals for the project. It presented the content of the AXLNet module in a format that could be better utilized on a personal, mobile device in an always-accessible manner. However, there were several major issues with the approach that prevented this solution from being ultimately acceptable. The two issues with the first version of the project were that it required continual connection to the Internet along with the fact that video streaming was slow and impractical.

This version of AXLNet Mobile lived on a webserver, and was only accessible by pointing the iPhone's Safari browser to the appropriate web address. In addition to the content contained in the HTML file and Javascript controller, network connectivity was required to download the video clips and to report user data back to the system's response database. Unfortunately, constant wireless Internet accessibility cannot be assumed in all environments where AXLNet Mobile might be deployed. In practice, this requirement would severely limit the potential usability of the system, since it could only be operated at times when the user could easily access an open wireless connection.

Even when such a connection might be available, downloading the video clips included throughout the module was problematic. While resized and

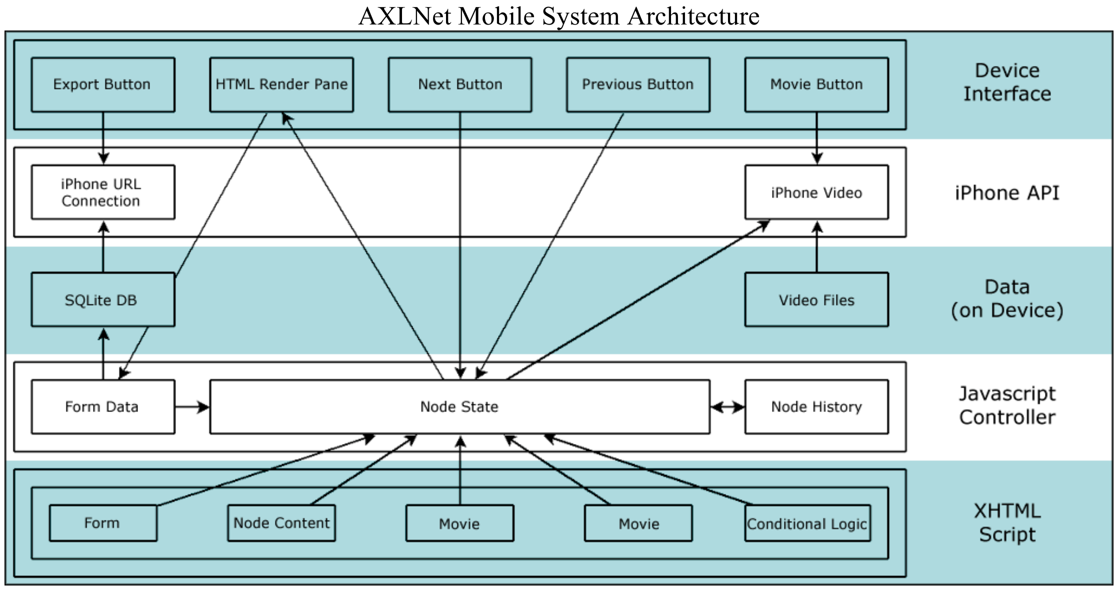


Fig. 1. Data from each content node is read from the XHTML Module Script into the Javascript Controller. The appropriate HTML code is generated and rendered by a pane on the Device Interface. Form data is entered via this interface pane and stored in an SQLite database on the device. The "Next" and "Previous" buttons affect the node state, which is reflected by the display. The "Movie" button causes the iPhone to play a video, stored on the device, using the iPhone API. The "Export" button causes data stored in the SQLite database to be sent to the central database using the iPhone API to manage the HTTP connection.

recompressed, the videos still took a significant amount of time to download over the iPhone's wireless connection. Although still theoretically usable, this caused frequent, irritating delays that interrupted progress through the module and created a less-than-ideal learning environment. Using this approach, there was no satisfactory way to preload the videos onto the device and eliminate the issue.

2.2 Second Iteration

To address these problems, a second version of AXLNet Mobile was created, this time using a different approach. In March of 2008, Apple initiated a program to allow independent development of applications that would run natively on the iPhone platform. Such applications can be installed and run on the device, utilizing an API to access the functionality of the hardware and the platform operating system.

This version of AXLNet Mobile took the form of an application residing natively on the iPhone platform. It consisted of a similar basic structure to the first iteration, with the addition of a new interface layer that ties the content directly into the operating system of the device. This approach enabled us to not only solve the problems in the first iteration of the project, but also to make improvements to the features that had already been completed.

Native applications for the iPhone platform are stored on the device's internal storage drive. Creating a version of AXLNet Mobile that was stored locally on the device rather than on a webserver solves both of the major problems presented by the first iteration of the project. Because the module content is stored on the device, there is no need to be connected to the Internet when using the program. Internet connectivity is only required for collecting user responses, which can be delayed without interrupting the user's learning experience.

Creating a new version of AXLNet Mobile as a native application was achieved by dividing system functionality between two levels of control: a content controller written in Javascript, loosely based on the controller of the previous version, and another layer, written in Objective-C, that wrapped that content in UI elements native to the iPhone platform and tied the content to functionality on the level of the operating system. The Javascript layer controls the primary display of content embedded in a UIWebView element that renders HTML with Javascript support. The two layers communicate through the return values of Javascript functions that are executed by the Objective-C code.

Most of the content of the module is contained in an HTML file organized by nodes in much the same manner as the original iteration. This content is controlled by a Javascript content controller and formatted by a CSS ruleset. The content controller stores the module's state, interprets user input, navigates from node to node, and displays the appropriate content node at the appropriate time (Fig. 1).

In addition to allowing us to utilize some of the design work done in the first iteration of the project, this type of Javascript content controller was chosen so that content could be created in a modular way, by authoring XHTML files. Each XHTML file contains a different module which can be navigated by the Javascript controller. When started, the AXLNet Mobile application displays a list of the available modules and then loads the appropriate XHTML file upon selection.

Objective-C is the primary supported language for native iPhone applications. The interface layer is written in Objective-C and serves as the entry-point and main application loop for the program. UI elements were created and placed using Apple's Interface Builder tool, and are controlled through the Objective-C code. The Objective-C layer also controls the system features that aren't handled by the Javascript controller. These features include media playback and database management.

The iPhone platform API includes media playback functionality that can be accessed through an Objective-C class. The filenames of appropriate video media are included in the nodes of the HTML module script and interpreted by the Javascript controller. The name of the video file associated with the current content node is passed up to the interface layer through a function return value when the user clicks on the "Movie" button on a page, at which point the Objective-C code invokes a call to the built-in media player.

The iPhone application also includes an SQLite database that is used to store user responses until they can be uploaded to the centralized database. Integration with the SQLite database is also functionality offered by the iPhone platform's API through Objective-C. Similar to media playback, the Javascript layer records user input on the rendered module content and then passes that data up to the interface layer through a function return. The Objective-C layer then manages the data in the database and uploads it when an Internet connection is available.

As mentioned, integration of these two layers occurs through Objective-C calls to functions in the Javascript layer which return relevant data to the interface layer. This allows almost all the logic associated with content control and display to be contained in the Javascript layer where it works with content parsed from the XHTML-formatted module script, and for the Objective-C layer to only request data from the content controller when it is necessary for functionality associated with iPhone platform API calls.

Through this approach, we successfully overcame the issues that were presented by the first version of the project, and satisfied our original project goals. The second iteration of AXLNet Mobile provides a supplementary reference to AXL course materials delivered through classroom instruction, and it does this in a convenient, personal, always-available format that maintains the important characteristics of AXLNet that we originally identified.

2.3 Module Porting

In porting the AXL content from web application to iPhone application, the FORCE team had two parallel goals of equal importance.

The first was to create an engaging mobile application that not only met baseline functional requirements, running smoothly on a standard device but also had a simple, intuitive user interface and good flow of information. Creating a satisfying, accessible and effective experience in our AXLNet Mobile prototype was a necessity; any final application would be put to use in the field, where its users would not have the cushion of a structured training environment.

Simply put, if AXLNet Mobile was at all difficult to use or hard to understand then the mobile context would only amplify these issues, and render the program effectively useless to its target user.

The second goal, of equal importance, was to be as faithful to the original AXL application as possible. By doing so, we would benefit from the research and team effort that went into the development of the AXL modules, allowing us to make valid comparisons between the effectiveness of instruction via the mobile and web platforms. Departing from the content in an attempt to “fix” something we perceived as incorrect was beyond the scope of our research, would require the consultation of education specialists and require complete revalidation of all processes.

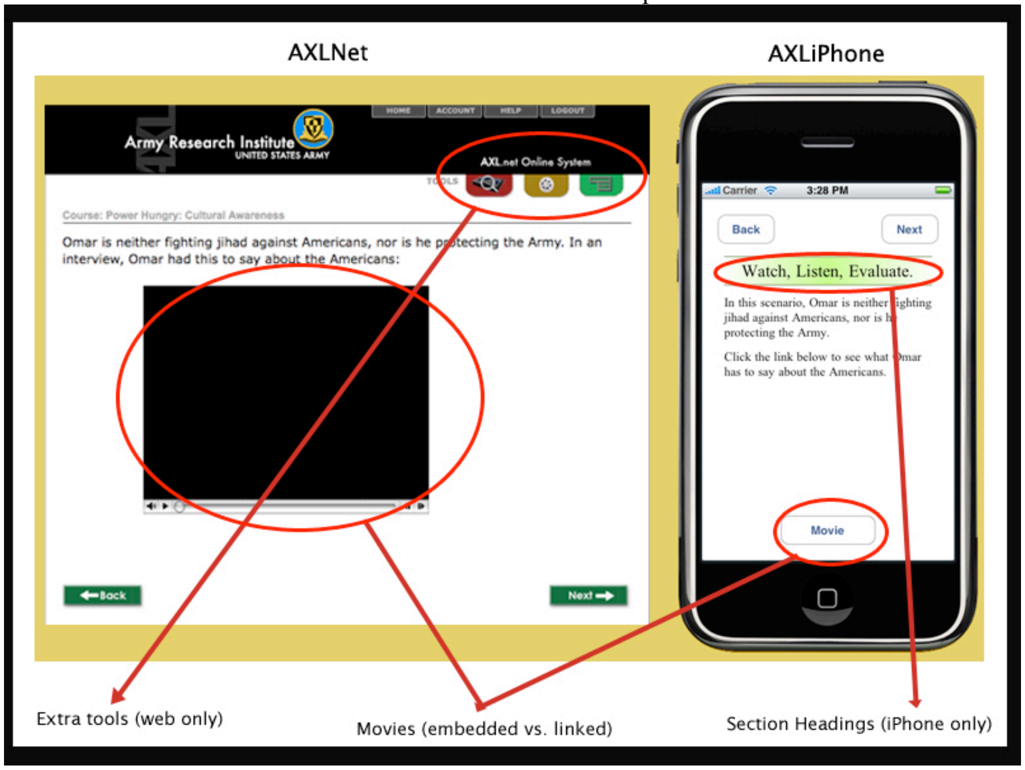
In order to maintain consistency with the original Power Hungry and Tripwire modules, we adopted a policy of providing the exact same content wherever possible, copying the text directly from the original. We also recreated the layout and logic of each individual page, dividing the content identically within reason.

In pursuing our primary goal of usability, however, we needed to diverge from the original AXL construction in a few minor but noteworthy ways.

The sheer difference in resolution between the web and mobile versions of AXL meant that, on occasion, pages that fit comfortably on the original would require several screens of scrolling on our build. In order to divide the experience more manageably for the user, some of these pages were divided into two parts, split at logical separation points. The content was not changed.

Another minor content change was necessary because of our video solution. While the web version made

Web and Mobile Interface Comparison



Omar is neither fighting jihad against Americans, nor is he protecting the Army. In an interview, Omar had this to say about the Americans:

In this scenario, Omar is neither fighting jihad against Americans, nor is he protecting the Army.

Click the link below to see what Omar has to say about the Americans.

Fig. 2. This side-by-side comparison of the interface for the web version of AXLNet (pictured left) and the iPhone version (pictured right) illustrates the similarities and differences between the versions. The demarcations highlight portions of the interface that were changed when creating the iPhone interface, in order to accommodate the specific features and technical

frequent use of embedded videos in their pages, our videos independently accessed separately from the text/navigation (as explained elsewhere). The text of the AXL modules makes frequent reference to their embedded videos. Since viewing videos is an additional step in our application, we needed to add or modify text that directed users to access the click the "play movie" button before attempting the questions.

Because of the additional navigation and greater fragmentation of pages in our modules, we added a title component that identifies and unifies themes within groups of pages. This minor divergence in content (the only original content added for AXLNet Mobile) is an attempt to fulfill our primary goal without changing any existing AXL material. The mobile module may be experienced in areas where deep concentration is unfeasible; the titles contextualize the pages, and keep the user focused on the larger sections of work across a range of pages.

In addition to the additions and modifications of page content, we did need to eliminate a few features from the original AXL module in order to meet the technical limitations of the iPhone. The cuts were made in two large database-driven features. The initial order-of-importance survey that analyzes users based on their choice of mission factors was not implemented in this version of AXLNet Mobile because it requires sophisticated text processing and access to the AXL database that fell beyond the scope of this prototype. AXL's optional "interview" panel was the other unimplemented feature: the required text processing and, more importantly, the storage space required for the interview media, made this option unsuitable for a mobile application.

On the occasions where the main page required the user to conduct an interview, these segments were removed or replaced with a more traditional segment that directs the user to the appropriate information and/or film segment. In most cases, the interview-oriented segments of AXL web modules require the user to find a specific piece of media, and the following pages provide "hints" or even show the appropriate media directly. In these instances, we simply removed the required interview and provide the information directly for our mobile module. While the exploratory element is removed, the instruction and desired content remains consistent with the original.

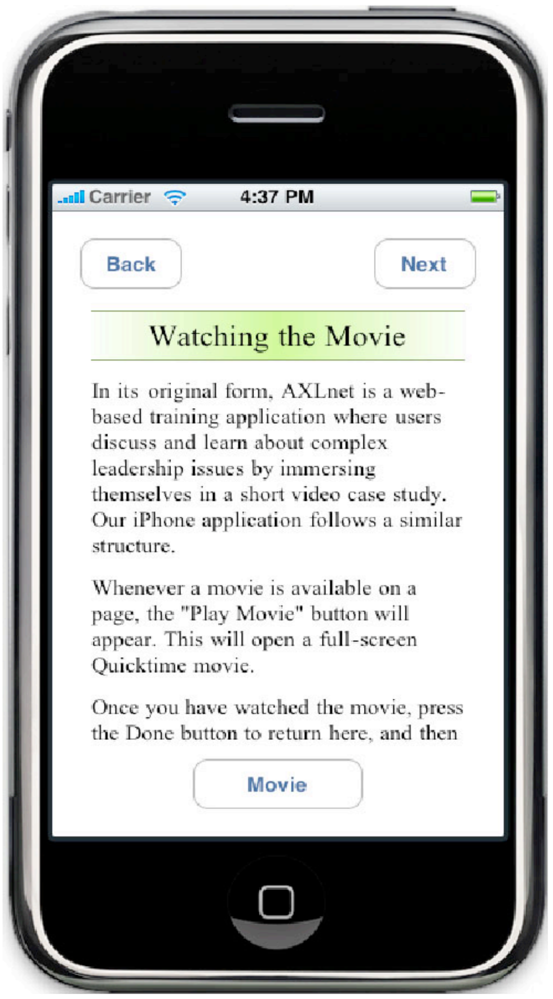
By adhering to these general principles, we were able to faithfully port two AXL modules from the web to the iPhone, keeping the experience consistent with the original while tailoring specific details to fit the new platform.

2.4 iPhone Interface

In this section, we will describe the operation of the Objective-C layer of code that connects the Javascript Controller to the iPhone device API and interface.

The AXLNet Mobile application is event-driven. The user is presented with an interface similar to the one shown in Fig. 3. The interface has two rows of buttons, one at the top and one at the bottom. Between them is the content pane, which displays rendered HTML content controlled by the Javascript controller program.

AXLNet Mobile Interface



Watching the Movie

In its original form, AXLnet is a web-based training application where users discuss and learn about complex leadership issues by immersing themselves in a short video case study. Our iPhone application follows a similar structure.

Whenever a movie is available on a page, the "Play Movie" button will appear. This will open a full-screen Quicktime movie.

Once you have watched the movie, press the Done button to return here, and then

Fig. 3

When the user clicks the Next button, the Objective-C code executes the "next" function in the Javascript content controller, which updates the HTML content displayed in the render pane. Likewise, when the user clicks the Back button, the "prev" function in the content controller is executed, which causes the previous content node to be displayed. These navigation functions are encapsulated in the Javascript code included with this document. Additionally, flow control is handled by the Javascript controller, based on the nodes set up in the module's XHTML document.

In addition to executing the Javascript function that changes the displayed content, these buttons also initiate a series of queries about the current state of the program. Several Javascript functions are executed which simply return information about the contents of the node being displayed. The askHasMovie function, for example, indicates whether there is a video clip associated with the current content. If so, then the askForMovie function returns the filename of the movie clip. Otherwise, the Movie button is removed from the interface.

All of the several Javascript functions with the prefix ask operate in the same way. These functions are designed to pass information between the Javascript layer, which interprets the XHTML module script and contains a model of the node, and the Objective-C layer, which contains the video playback and SQLite database functionality.

Media playback is handled by the Objective-C code directly. The iPhone platform has functionality for playing videos built into the API. This built-in functionality is utilized when the user presses the Movie button, which causes the askForMovie function to be executed in order to pass in the name of the video clip.

Likewise, the SQLite database is controlled through functionality provided by the iPhone API on the Objective-C layer. When the askHasData function indicates that the current node contains user-input elements, their names and any user input is stored in the SQLite database upon navigating away from the node.

When the askHasExport function returns true at the end of the course, the Export button is displayed at the bottom of the interface. Pushing this button attempts to send the data stored in the local SQLite database to a PHP script on a remote webserver, which accepts the data and stores it in a centralized database for user-data. If the device cannot connect with the remote server, then an error is reported to the user and the local data is preserved so that another attempt can be made. Otherwise, the data is removed from the local database when it is successfully uploaded.

3. USER TESTING

Forty-six subjects were recruited at Ft. Leonard Wood, with the assistance of ARI. Testing was accomplished over a three-day period, with groups of 10-13 who attended either a morning or an afternoon session. The group was told the purpose of the study was to determine if the mobile device could be used for leadership training. The procedure was divided up into 1) pre-training questions, 2) viewing the film scenario, 3) questions concerning initial responses to the film, 4) taking the full course module and 5) answering post-course questions. Finally, there were 6) questions about their reactions to the use of the mobile device, including an open-ended comment section.

- askHasMovie – Returns TRUE if a video file is associated with the current node.
- askForMovie – Returns the name of the video file.
- askHasData – Returns TRUE if the current node contains form elements.
- askForData – Returns the various elements (node, key, value) of the most recently added user data.
- askHasLink – Returns TRUE if the current node redirects the program to a new XHTML module script.
- askForLink – Returns the file name of the new module script.
- askHasExport – Returns TRUE if the current node is marked as an end-state from which exporting user data should be allowed.

Fig. 4. The *ask*-prefixed functions allow for communication between the node model in the Javascript layer and the device functionality in the Objective-C layer.

The pre-training questions included rank and deployment history. Of the 46 participants, 43 held the rank of Captain and 3 the rank of 1st Lieutenant. Parallel to this, 43 indicated at least one deployment to Iraq or Afghanistan, with 2 Captains and 1 Lieutenant having not previously deployed to either theater.

Two psychological instruments were employed: An adapted version of the PAD (Pleasure-Arousal-Dominance) scale, focusing on Arousal, and the PANAS (Positive Affect Negative Affect Scale) designed to determine a subject's self-reported affective states. These were administered at three points during the testing: at the beginning, after viewing the scenario film, and after completing the course module. Once they completed the first set of PAD/PANAS questionnaires, participants started the course module on the iPhone or iPOD Touch (module was exactly the same on either device), which began with a cultural awareness scenario (film) called *Tripwire*, detailing an IED incident in Iraq.

Post-film viewing, the students were given the PAD and PANAS tests, and a questionnaire to assess their initial responses to the situation. After completing the course, the participants were again given the PAD and PANAS, and a set of questions specifically about their experience with the mobile platform.

3.1 Results

The results of the study showed several interesting points. First, the people who were more aroused and positive at the beginning of the exercise, as assessed by the adapted PAD and positive PANAS, rated the use of the mobile device more highly. This suggests that there may be a minimal level of arousal and positive emotion needed before the AXLNET Mobile training is most effective. Overall, there was a strong feeling that the mobile device could serve as a valuable leadership training mechanism. On a Likert scale of 0-7, the consensus score was 6.11 and 6.35 for two questions comparing the Mobile Scenario with reading a scenario or seeing a PowerPoint briefing.

In terms of emotional factors, there were statistically significant increases from the first to the second set of scores (pre-training to post-film testing) on the adapted PAD (Arousal) ($p = .008$) and negative emotion from the PANAS ($p = .003$). This indicates that watching the film increased such feelings as being stimulated, excited, hostile, and upset. Follow-up questions suggest that some participants were upset by problems with the way characters handled difficult situations. Regardless of the reason, the change in emotion after watching the film indicates a high level of engagement with the scenario. It remains to be determined what specific aspects of the film caused such a strong reaction. However some of the participants left comments that they found the handling of the situation to be doctrinally incorrect.

Participant comments also indicated that the largest source of frustration with the mobile device was in typing in answers to the course module questions. However, one participant observed, despite the time it takes to type on mobile devices, that typing was made easier for her because she owned an iPhone. We expect this pattern, and we believe it will be less and less of an issue as more people have experience with these devices. Most negative comments dealt with disagreements with the details of the situation. Some students stated this could never replace classroom discussion, but this was not the intent of the mobile distribution, which is meant to supplement and reinforce classroom training during wait or off-duty times, thus capturing lost moments for training opportunities. While further studies with more doctrinally correct case studies need to be done, our initial results show good promise for the use of mobile–deployed training modules to engage soldiers. As one participant wrote, “This beats the h*ll out of death by PowerPoint!”

4. CONCLUSIONS

While the use of off-the-shelf mobile devices as training aids for soldiers is not yet widespread, their increasingly pervasive use suggests they could be useful tools in this domain. While we do not suggest that mobile-delivered training take the place of classroom and teacher-based instruction, nonetheless, we see value in supplementing traditional training with such devices. As more and more soldiers embrace mobile technology in their daily lives, in line with the general, “digital native” population, adding adjuvant applications to devices already in-hand, will provide more opportunities for easily accessible learning opportunities. As our study has indicated, today’s soldiers find value in training delivered through mobile devices. As these devices advance in functionality, we see tomorrow’s forces served through supplemental training that is accessible, up-to-the-minute, trackable, and even personalized. Instead of having to go to training, training in the future can be an integral part of a soldier’s everyday practice.

ACKNOWLEDGEMENTS

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

The original AXLNet Project was done by Randall W. Hill, Jr., and Julia Kim at the ICT, with support from Michelle R. Zbylut at the Army Research Institute. Additional contributors included Dr. David Traum, Mr. Scott Rocher, Mr. Stewart King, Mr. Reid Swanson, Mr. Sudeep Gandhe, Mr. Salvo Lavis, and Mr. Ashish Karnavat. Both *Power Hungry* and *Tripwire* were written and produced by Mr. Kim LeMasters. *Power Hungry* was directed by Mr. Chuck Bowman. *Tripwire* was directed by Mr. Kenny Johnson.

AXLNet Mobile could not have been successful without the hard work and contributions from Sean Bouchard, Dinesh Rajpurohit and Jamie Antonisse, who were the core project team. We also gratefully acknowledge the support provided by Dr. Gregory Ruark at ARI for securing participants at Ft. Leonard Wood and assisting in administering the study.

REFERENCES

Hill, R., Kim, J., Zbylut, M., Gordon, A., Traum, D., Gandhe, S., King, S., Lavis, S., Rocher, S., 2006: AXL.Net: Web-Enabled Case Method Instruction for Accelerating Tacit Knowledge Acquisition in Leaders, *Proceedings of the 25th Army Science Conference*, Nov 2006, Orlando, FL.