An Intelligent Agent for Adapting and Delivering Electronic Course Materials to Mobile Learners

Mohamed Ally, Ph.D.
Athabasca University
mohameda@athabascau.ca

Fuhua Lin, Ph.D.
Athabasca University
oscarl@athabascau.ca

Rory McGreal, Ph.D.
Athabasca University
rory@athabascau.ca

Brian Woo
Athabasca University
brianwoo@yahoo.com

Abstract

The concept of mobile learning is often defined as learning that takes place with the help of mobile devices to access course materials; however, these devices operate in different ways and have different capabilities. To make course materials available on these mobile devices as well as on desktop systems, a software program is required to allow course material to be delivered in heterogeneous computing platforms. Because of these heterogeneous computing platforms, some course materials may not be in a format that different mobile devices would accept. This project developed an intelligent software agent capable of adapting to the heterogeneous mobile computing environment. The agent can search for a conversion tool according to the desired format and convert the course materials automatically. The agent is able to understand mobile clients' capabilities. In order for the server to know what type of course material the client wishes to receive, the client needs to feed information on the software and hardware capabilities of the device to the server. However, devices do not normally carry any information about their capabilities. We have chosen the Resource Description Framework (RDF) represented Composite Capabilities/Preferences Profile (CC/PP) promoted by the W3C to describe the device capabilities and user preferences.

Keywords: Mobile Learning, Intelligent Agent, M-learning, Mobile Devices

1. Introduction

With the proliferation of mobile technology in society and business, many individuals and organizations are seriously looking at using mobile devices for learning and training. As a result, educators must develop new courses and re-purpose existing courses for mobile learning. The concept of mobile learning (M-learning) is often defined as learning that takes place with the help of mobile devices. Ally (2004) defined M-learning as the delivery of electronic learning materials on mobile computing devices to allow access from anywhere and at anytime. However, different mobile devices operate in different manner and have different capabilities. The design for mobile devices has to be flexible to allow electronic learning (E-learning) materials to be delivered in heterogeneous computing platforms. Because of these heterogeneous platforms, some course materials may not be in a format that mobile devices would accept. The objective of this research is therefore to develop an intelligent software agent capable of adapting to the heterogeneous mobile computing environment. The agent should be able to search for a conversion tool according to the desired format and convert the course materials automatically through understanding the mobile clients' software capabilities and hardware limitations.

In order for the server to know what type of E-learning material the client wishes to receive, the client needs to provide the software and hardware capability of the device to the server. However, a device initially does not carry any information about the capability. This project experimented with the Resource Description Framework (RDF) represented Composite Capabilities/Preferences Profile (CC/PP), which is promoted by the W3C (2005) to describe the device capabilities and user preferences. Jena as an RDF parser was used to obtain the device information, providing a query language called Resource Description Query Language (RDQL, 2005) to query for device information. Sun's Wireless Toolkit and PalmOS Emulator were used as the testing environment. Also, an open and service-oriented architecture was used to develop the agent and open application interfaces to enable interaction and integration seamlessly between learning objects repository, course repository, and learning services.

2. Related work

Gaedke et al. (2005) proposed an approach to automate Web content conversion. Similarly, Castellanos and Sanchez (2003) have designed a framework to allow small devices (e.g. PDA and cellular phones) to download digital library (DL) resources. Gross (2001) has developed a prototype system PRAVTA to monitor the presence of users. The aim of the prototype was "to develop a system that provides users with adequate awareness information anytime and anyplace." The presence of a user can be found by sensors and reported by indicators. The entire "awareness" process is automated. The information is updated on a Web server and can be queried through a Web service. This automated information update principle has been used in this project. Lum and Lau (2002) proposed a content adaptation system that decides the optimal content version for presentation and the best strategy for deriving that version. They developed a prototype PDF documentation adaptation system.

3. Architecture Design and Methodology

3.1 Design Considerations

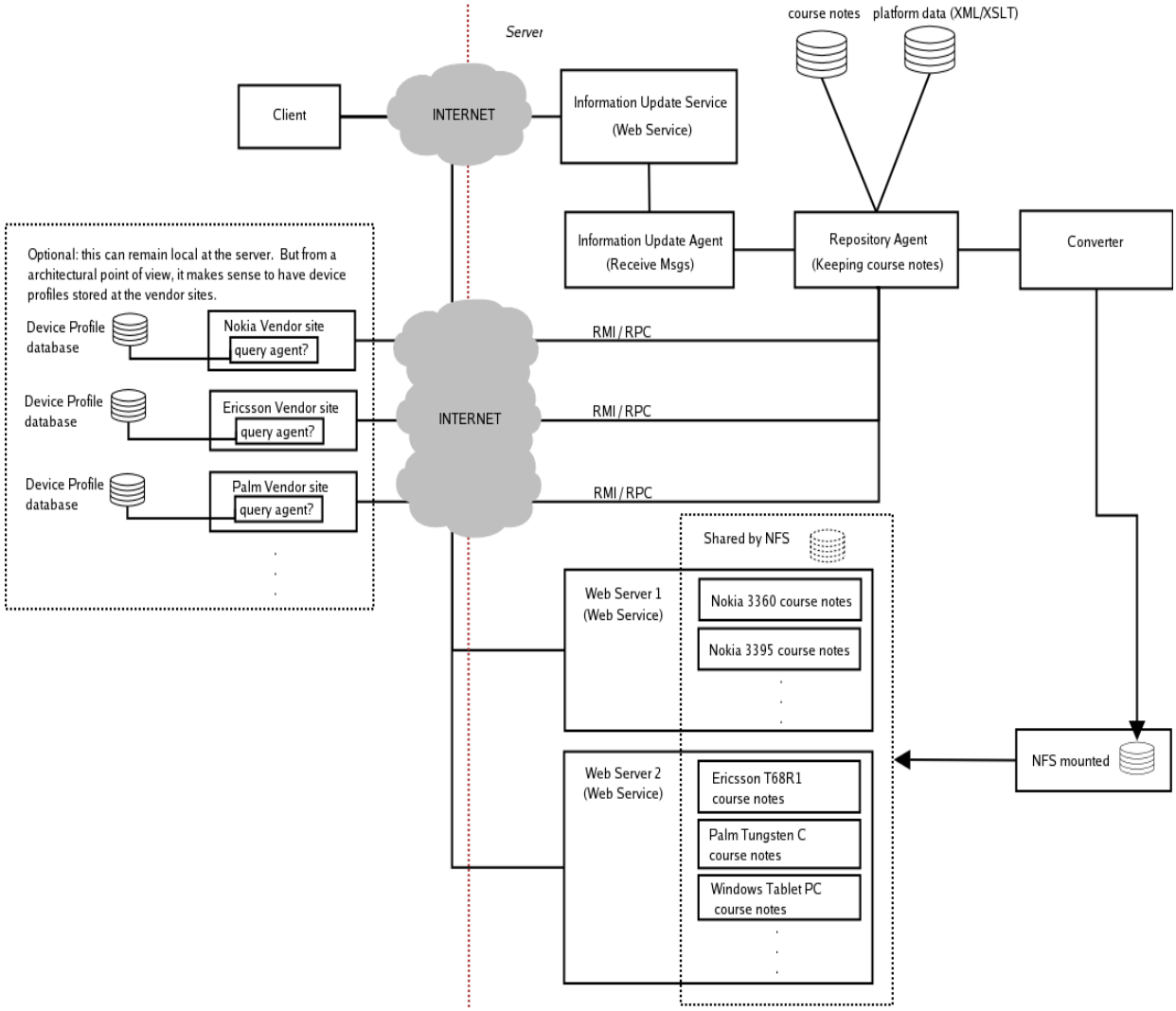
This project considered the following issues in heterogeneous devices.

- *Software Portability.* A Java-based cross-platform software framework is one of the most important features in this project.
- *Limited Computing Power and Memory Consumption.* We designed computing intensive tasks to execute on the server side. The portable computing devices act as a thin client. The thin client design model precludes the need for having large executable files and user libraries stored on the client mobile device.
- *Display Properties.* The system provides support for different devices by separating the presentation layer from the data content. A number of different devices can be supported without the need of modifying programming logic and data content.
- *Development Costs.* This project targets Athabasca University (AU). Therefore, the software should be designed to integrate seamlessly into AU's existing infrastructure.
- *Design Flexibility and Scalability.* The system is designed in a distributed architecture that prevents a single point of failure, improves structural scalability and performance. Besides the distributed architecture, a Web service is deployed on the server to allow thin clients to access the backend data. Web services are platform independent; therefore, virtually any type of client can construct a Remote Procedure Call (RPC) to execute code at the server side.
- *Caching to Save on Limited Resources.* With the support of the caching design, users can read the cached course notes on the device to avoid additional downloads.
- *Software Agent Support.* Software agents can be used to facilitate automated routine tasks, residing in both the client device and the server environment.

3.2 Architectural Design

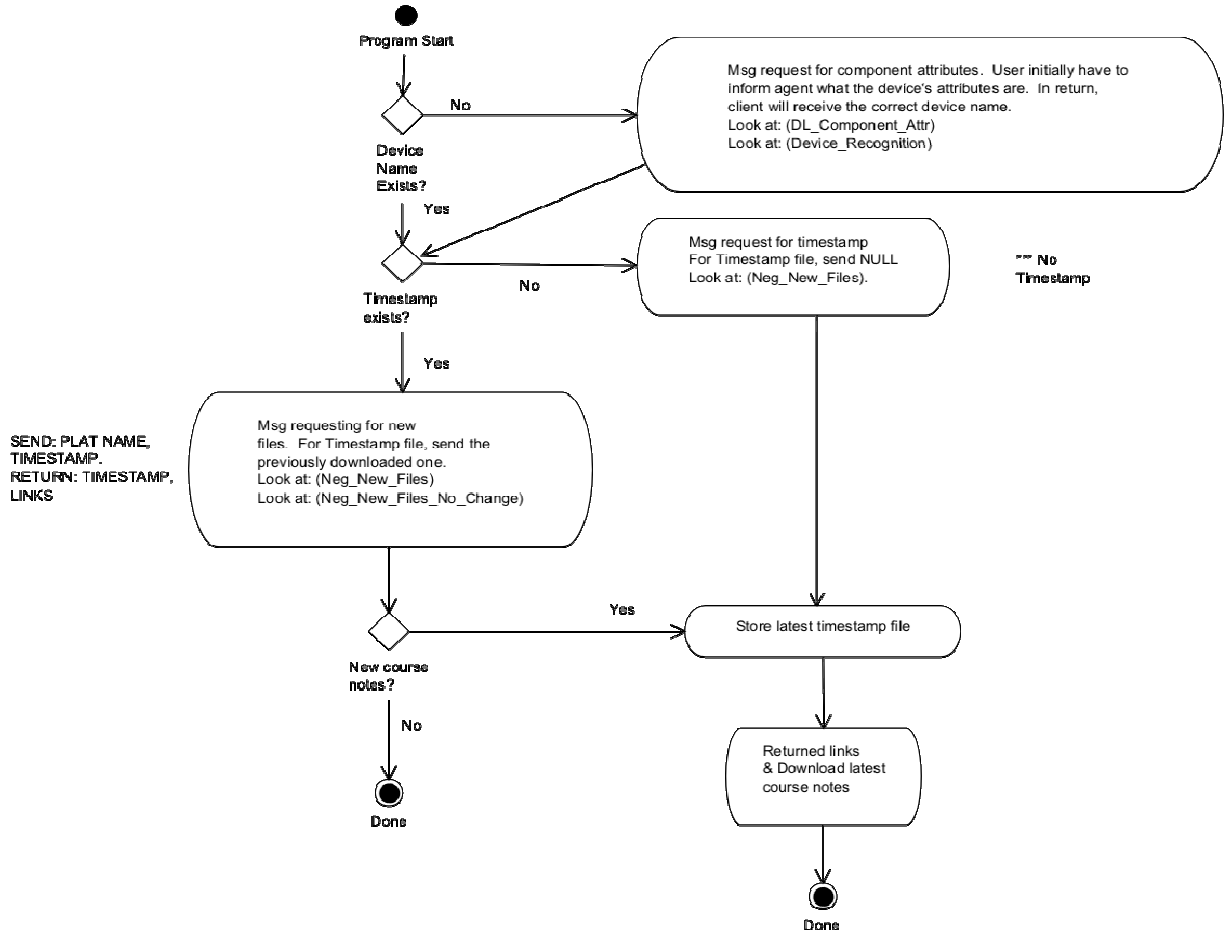
The architecture can be divided into two main areas: Client and Server (Figure 1). The client shown in Figure 1 connects to the University server through the Internet, running the Information Update Client Agent. Figure 2 shows how a client negotiates with the server to retrieve updated course notes. There are two main activities in this Client-Server Communication: Device Recognition Negotiation and Negotiation for Updated Course Notes Files.

Figure 1. The overall architectural diagram.



Device Recognition Negotiation. The reason for having device recognition is that the server needs to know what type of course notes content the client wishes to receive. In order for the server to achieve this, the client needs to provide the information on its software and hardware capabilities to the server. Unfortunately, a device initially does not carry any information about its supported hardware and software capability. Our solution to obtaining this device capability information is to utilize Composite Capabilities/Preferences Profile (CC/PP) promoted by the W3C (2005). CC/PP has been written in Resource Description Framework (RDF) which is represented in Extensible Markup Language (XML). We used Jena (2005) as an RDF parser to obtain device information. Jena provides a query language called Resource Description Query Language (RDQL) to query for device information. RDQL is similar to SQL used in relational databases. CC/PP contains a list of component attributes.

Figure 2: Activity Diagram: Client Server Negotiation.



The attributes contain information about the processor type, amount of memory, operating system version and sound and image capability. The client starts with a request to the server for a set of component attribute criteria, which include a series of questions the user has to answer. Using the answers from the user, the server can recognize the device by matching these criteria. When there are more devices added to the database in the future, additional component attributes might be added to the schema. It is an advantage that component attribute criteria can be re-fetched instead of stored on the device permanently. In other words, the server can easily manage the addition and deletion of component attributes and the client will adapt to the changes immediately. By default, there are four downloadable component attribute criteria in the configuration file. The criteria are a series of questions listed in Table 1.

Table 1: the questions for the component attribute criteria

Attribute Criteria / Question	Sample Answer / Settings
1. Vendor Name?	E.g. Ericsson
2. Model Number?	E.g. T68R1
3. Is this device image capable?	E.g. Yes/No
4. Can this device display color?	E.g. Yes/No
5. Can your device output sound?	E.g. Yes/No
6. Can your device accept downloadable software?	E.g. Yes/No

Some component attributes, for example "HTML version supported", will not be downloaded because not all users will understand the meaning of this attribute. Only the simple attribute criteria questions that a normal user can understand, such as "Can your device output sound?" will be downloaded. The component attribute criteria are stored in XML format. The criteria can be changed during a software runtime and the new attribute criteria will be downloaded to the client. The client will send a message to the InformationUpdateAgent which then acquires the RepositoryAgent. The RepositioryAgent will invoke the ComponentAttrReader class to retrieve the component attribute criteria. This information will be delivered back to the client. Upon receiving the component attribute criteria, the human user has to provide some information about the device for device recognition. The component attribute settings the user specified will be sent to the InformationUpdateAgent to search for the appropriate device name. The RepositoryAgent will fetch all available device profiles with the DeviceProfileReader class. RDQL will be used to query for the device name and the correct device name will be sent to the client.

There are two steps in the device recognition process. In the first step, the server agent will try to recognize the device by its vendor and device names. If the server agent cannot find that device directly from the database, it will try to recognize the device by the component attributes. The advantage of this strategy is that, if the user already knows the device model, this can be entered into the device. This strategy saves the extra computing time on the server to perform device recognition. If this information is not recognized, the server agent will have to match the device component attributes for the correct device name.

Component attribute settings can also be overridden to suit the users' preferences. For example, some users cannot tolerate a slow network speed and therefore, they prefer to download Web pages without images. The "image capability" attribute flag can be turned off even if the device has the ability of displaying images. After the user has entered the component attributes, the client agent can communicate with the server agent to perform device recognition.

There are three main components on the server side: Information Update Service, Information Update Server Agent, and Repository Agent. First, the Information Update Service provides a Web service for the client to connect and request for a specific service to execute. The Web service is provided by a Java Servlet which is run on the Apache Tomcat Web server. The Web service provides a light version of SOAP, called KSOAP (2005), as the communication protocol. Second, the Information-Updating Service automatically marshals and un-marshals KSOAP objects, which are delivered to the appropriate destinations. When the servlet has successfully received a message from the client, the message will be synchronously delivered to the Information Update Server Agent. The Information Update Server Agent executes method calls according to the message types. In addition, the Information Update Server Agent will communicate with the Repository Agent in fetching data from the repository. Third, there are two main functions for the Repository Agent: File Conversion and Fetching updated files for the remote client.

File Conversion: Each platform requires a transformation file (XSLT) for data conversion. Transformation files are made device specific and contain layout and formatting information. A transformation file transforms a generic data file to a file in a specific format.

Negotiation for Updated Course Materials Files. This functionality is used where the client has to communicate with the Repository Agent to determine what files have been updated. InformationUpdateClient sends a message to the InformationUpateAgent which redirects the request to the RepositoryAgent for requesting new course materials. The message includes the platform name and the timestamp last obtained from the server. The timestamp file contains timestamps of the entire list of course files currently stored on the client device. The RepositoryAgent first

retrieves all timestamp data from the timestamp file on the server. The retrieved data will be compared against the timestamp data passed in from the client. The agent can obtain a list of updated files with a timestamp comparison. With the same mechanism, the files deleted on the server will also be detected. Next, the RepositoryAgent will try to find the location of the updated files based on the platform name obtained from the client. In return, the RepositoryTimeUtil class will fetch and return the links to the RepositoryAgent. The RepositoryAgent will also request the latest timestamp file in byte array from the RepositoryTimeStampReader class. Then three items returned to the client: links to download latest course materials, deleted files, and the latest timestamp file. If there are no files being updated or deleted, a message will be returned to the client indicating there has been no file change.

3.3 Testing Environment

There are two pieces of software used in this project to save testing time: Sun's Wireless Toolkit (WTK) and PalmOS Emulator. The wireless toolkit provides an easier way to build a software package for a Java compatible device. The toolkit also offers package integrity checking to ensure the package and other files are generated correctly from Java class files. This toolkit also provides emulation of a few Java-enabled cell phones. The compiled package can run on these emulated cell phones (Figure 3). The PalmOS Emulator provides the ability to run the J2ME package on the emulated PalmOS environment. With these testing tools, the testing image is not required to run on-target; instead it can be run in an emulated environment.

Figure 3: Screen Shots

	Users have to enter the information to the device for device recognition. For example: Vendor Name == Ericsson Model Number = T39m If the users do not know their device model number and vendor name, they can leave the fields blank. Press down to enter other options for device recognition (next figure).		When the users press down, they will see a list of questions to be answered. These are the component attributes. Some of the options have 2 choices like the one shown in the figure.

4. Results

There are a number of advantages of distributing course materials using the agent over the current method. Table 1 shows the total time an average person would need to create course notes.

	First Page		Second Page	
	With Automatic Software Updater	Conventional method of distribution	With Automatic Software Updater	Conventional method of distribution
Compose course notes (1 page)				
- *Windows*	N/A	30 mins	N/A	30 mins
- *Linux*	N/A	30 mins	N/A	30 mins
- *PDAs*	N/A	30 mins	N/A	30 mins
- *Cell phones*	N/A	30 mins	N/A	30 mins
- *Platform-in specific*	30 mins	N/A	30 mins	N/A
XML Transformation StyleSheet				
- *Windows*	30 mins	N/A	0 mins	N/A
- *Linux*	30 mins	N/A	0 mins	N/A
- *PDAs*	30 mins	N/A	0 mins	N/A
- *Cell phones*	30 mins	N/A	0 mins	N/A
Packaging	N/A	10 mins	N/A	10 mins
Total:	150 mins	130 mins	30 mins	130 mins

Table 1. Time needed for instructor to create Page 1 and Page 2.

With the agent, there might be a slight learning curve at the beginning. However, using the agent provides the time savings in the long run. The second and third columns of Table 1 show a scenario where the person needs to create the first page of the course notes. The fourth and fifth columns of Table 1 show a scenario where the same instructor needs to create a second page of course notes.

With the help of the updater, an instructor only requires creating another file, which contains the course materials data; that is, there is no page formatting included in the file. The same transformation stylesheets previously created can be reused. The stylesheets also allow the instructor to provide a consistent look and feel across all the pages. Note that there is a significant amount of time which can be saved especially when more platforms are added. Transformation is performed before deployment to avoid the lagging file access time introduced by Just-in-time transformation. Therefore, the entire pages ready to serve on the Web servers should already have been converted, then the users' wait time being reduced tremendously. Without the help of the agent, Web pages will have to be transformed at least once (by caching the transformed pages) during runtime.

The updater also helps instructors save time from packaging and deployment. Instead of packaging and deploying each set of files manually, the updater will copy the files to the appropriate platform folders for deployment. The updater also ensures that students will get the latest files on the server. Therefore, if there is an addition, deletion, or update for the course files, instructors will not need to send out any notifications to their students.

5. Conclusion

Because of the increasing use of mobile devices in society and businesses, they will be increasing emphasis on the

use of mobile learning in training and education. As a result, new courses have to be developed or existing courses have to be revised for delivery on mobile devices. The project described in this paper developed and introduced the Automatic Software Updater. The results gathered from using the Automatic Software Updater proves that the Automatic Software Updater provides a substantial timesaving in a heterogeneous platforms environment. Not only the time for developing course notes but also the time for packaging and deployment has been shortened. The Automatic Software Updater generates a substantial timesaving when there are a number of device platforms to support. Organizations that have a large user base should consider adopting this technology for their future benefits.

6. References

Ally, M. (2004). Using learning theories to design instruction for mobile learning devices. *Proceedings of the Mobile Learning 2004 International Conference*, Rome.

Castellanos, N., J. A. Sanchez. (2003) "PoPS: Mobile Access to Digital Library Resources," *JCDL*, vol. 01, p.184.

CC/PP (2005). At http://www.w3.org/Mobile/CCPP/, 25 June 2005.

Gaedke, M., M. Beigl, H-W Gellersen, C. Segor, (2005). Web Content Delivery to Heterogeneous Mobile Platforms, 1998. University of Karlsruhe, At http://www.teco.edu/~gaedke/paper/1998-lncs1552.pdf, June 12, 2005.

Gross, T. (2001). "PRAVTA: A Light-Weight Mobile Awareness Client." SIGGROUP Bulletin **22**(1): p. 3-7.

Jena. At http://jena.sourceforge.net/, June 25, 2005

KSOAP. At http://kobjects.sourceforge.net/ksoap2/, June 5, 2005.

Lum, W-Y & F C.M. Lau. (2002). A Context-Aware Decision Engine for Content Adaptation, IEEE Pervasive Computing, July-September, p. 41- 49.

RDQL (2005). At http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, 12 June 2005.

Woo, B. (2005). E-Learning: Automatic Software Updater in Heterogeneous Environment, M.Sc. thesis, Athabasca University.