Acta Polytechnica Vol. 47 No. 4–5/2007

Developing E-learning Courses for Mobile Devices

R. Szabados, K. Sipos

The recent and rapid development of mobile devices and the increasing popularity of e learning have created a demand for mobile learning packages and environments. We have analyzed the possibilities of adapting the existing content for mobile devices, and have implemented two fundamentally different systems to satisfy the demand that has arisen. One of the systems creates e learning courses from existing materials and adapts them to the specified platform (this system realizes the functionalities of the Content Management System). The other system is a modified version of the Moodle Learning Management System, which can adapt existing courses right before displaying them. This paper discuses the fundamentals of e learning, the design considerations and investigates various methods of scalable video coding. Finally the realization details of the two systems are presented.

Keywords: E-learning, SCORM, mobile devices, adaptation of learning packages, video coding.

1 Introduction

The recent development of computers has opened up new opportunities in education. Today, in computer assisted education modern multimedia technologies and the Internet are used to create place and time independent learning possibilities by using a uniform framework. The present form of computer-aided learning, or e-learning, is a result of continuous development. Nowadays most of the existing e-learning systems and courses are conformed to one of the widespread e-learning standards or recommendations, so the learning systems are independent of the learning material, and are interoperable with each other. The key to the popularity of e-learning is the richness of multimedia contents in the courses, and the extensive accessibility of the Internet, where these materials are usually presented.

Due to the development of mobile devices such as PDAs (Personal Digital Assistants) and smartphones, mobile Internet access has become more and more common, and thus a demand for mobile e-learning systems and courses has arisen. However, most of the existing learning packages were designed to be run on PCs. This means that these materials are usually rich in multimedia contents that fit the size of a typical desktop computer's display. Also the formatting and wrapping of the texts are designed presuming this display size. Although mobile devices can generally display these courses, real time shrinking and displaying a video with fine resolution and good quality to a small display requires more computational power than that is available in these devices. There is, therefore, a need for specialized learning packages for mobile devices. Most of the course editing and management softwares have some support for PDAs or smartphones, but this only involves limiting the displaying area. The creator of the learning package (let us call him the lecturer) has to pay attention to the size and quality of the multimedia contents for the best playback compatibility; he has to resize or reencode the images and videos with another program. Since the lecturer is usually not a computer specialist, it would be ideal if the learning content development system would pay attention to these properties of the content, and it would adapt them automatically to the given device. As the generation of a learning package takes much time, and there are already many existing courses designed for PCs, there is also a need for proper playback of these courses on mobile devices. To achieve this, automatic adaptation is required in the e-learning system: after the construction phase of the package and before delivery. To solve these problems, our goal was to analyze the possibilities of adapting courses to mobile platforms, and to develop a system capable of this adaptation.

In this paper after introducing the structure of the most common e-learning systems and the quasi-standard SCORM (Sharable Content Object Reference Model) [1] recommendation, we review and evaluate the adaptation possibilities. After this, we present two systems that we have implemented to solve the problem of adaptation. We finish this paper by discussing the evaluation of these systems and presenting the possibilities of their further development.

2 Fundamentals of e-learning

The very first e-learning courses were developed together with their delivery system; therefore, both of them should have been changed if the learning content had been changed. In addition, there was no way to transfer a learning content to another system, so in the early days e-learning was not very flexible. A new impulse was given to e-learning when independent organizations worked out recommendations for the structure of learning systems and for cooperation between the delivery system and the course itself.

The most commonly used recommendation is the SCORM, which was created by combining different widely used standards and recommendations. According to SCORM, three parts of an e-learning system can be separated: the course, the course editing module, and the runtime environment. This is shown in Fig. 1. In an e-learning system we talk about the Learning Management System (LMS), which is responsible for displaying the learning content, for authenticating, and for tracking the user's progress; the Content Management System (CMS), which manages and generates the learning packages; and the content itself. Most commonly the LMS is an on-line, web-based system. After the login, it provides different levels of access to the courses according to the user's rights, which can scale from basic read-only access to system administrator privileges. The CMS

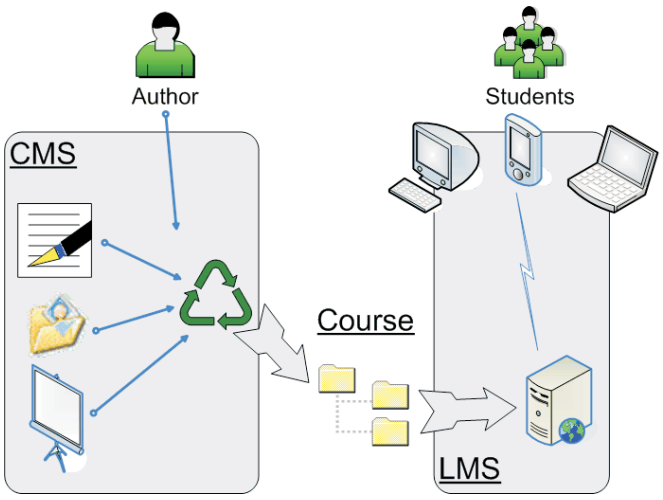

Fig. 1: Main parts of an e-learning system

is usually available off-line, and provides course generation facilities. Most of the CMS programs support the creation of different e learning package formats ranging from off-line courses (which can be distributed on CDs) to SCORM conformed packages.

The SCORM specifies, among others, the structure and content of a learning package. A SCORM package – i.e., a SCORM compatible learning package – is made up of independent learning objects, called SCOs (Sharable Content Object), to enable the reusability of the learning contents already made. The hierarchy of the course is created by organizing the SCOs into a tree structure. SCOs are like chapters and subsections in a book. The lowest level SCOs are built up of HTML pages, which actually contain the learning materials such as text, picture, video, etc. A learning package includes, among others, these HTML files, the multimedia files, and a so called manifest file. This latter file contains course-related metadata, such as the language and the logical structure of the course, and the location of the files appearing in the course. The name of this file must be imsmanifest.xml, and it has to be placed into the root folder of the package, because the LMS displays the course relying upon this file.

According to SCORM, the adaptation of the multimedia content can be realized in the CMS during the construction of the package, and also in the LMS during the presentation of the course. The two solutions offer different kinds of adaptation. In the case of adaptation during course production, the learning package will be compatible with all of the SCORM-conformed LMSs. In this way it produces platform specific SCORM packages, which contain the multimedia files adapted for the specific device. On the other hand, these packages can only be displayed appropriately on the specified device; therefore, different learning packages have to be generated for different platforms. To support the multiple package generation of the same course, the lecturer only has to name the platform to which the package is created, and the CMS takes care of all the rest. In the case of adaptation in the LMS, all SCORM conformed packages have to be supported, including the existing courses designed for PCs. In this situation the learning package contains the multimedia content only once, and the system has to adjust it to the device before delivering the package. This way the lecturer does not have to focus on the properties of the learning material. Since the two ways of adaptation grant different advantages, we decided to implement both of these methods.

3 Design considerations

Content adaptation is necessary because the different mobile devices have different technical capabilities. During the design of the systems we considered the following properties:

- Bandwidth: when using mobile devices, the available bandwidth can be relatively low, or the quality of the connection can be poor. An example is the GPRS based (General Packet Radio Service) Internet access. This may cause problems when trying to send higher quality or longer videos over the network. This problem can also occur while using a notebook.
- Computational power: PDAs have relatively slow CPUs, which can also cause problems during higher quality or longer video playback.
- Screen size: The small screen size causes problems on PDAs or smartphones when trying to display large images or videos. When the device displays the images in full size, intelligibility decreases because the user has to scroll on the page. Also, in the case of large videos, real-time shrinking requires much computational power.
- Animations: The electronic learning courses are popular for being rich in vivid multimedia contents; therefore, the presence of flash animations in these courses is common. However, this can cause problems if a device cannot play these files.

As the range of existing mobile devices is quite wide and they have rather different capabilities, we defined three profiles corresponding to three common device types for adaptation in our systems. For desktop computers we defined the "PC" profile presuming big screen size, high bandwidth and a fast processor. This is the base profile; no adaptation is necessary for this. We defined a "notebook" profile for computers with low bandwidth Internet access. Finally, for the "PDA" profile we assumed an average PocketPC with 320×240 pixel screen size, low bandwidth connection and a slow processor. As nowadays smartphones have quite small displays, and extremely varied capabilities, we omitted the "smartphone" profile for now. However, the realizations of our systems provide an easy way to insert new devices later on.

According to the considerations mentioned above, we reviewed and evaluated the possible adaptation methods of the following multimedia contents:

- Text: The only reason for adapting text is the small display size, so it would be essential solely for the PDA profile. As in SCORM the course pages are HTML files; the browser can automatically adjust the text to the width of the screen. Thus, no other adaptation is required.
- Picture: As real-time resizing of an image is not so difficult, and their transfer also does not require high bandwidth, the main reason for the adaptation of images is again the small display size. When the device displays a large image in full size, it is difficult to understand it as a whole, because the user needs to scroll in order to view the different parts of the picture. On the other hand, displaying these images

in a reduced size can cause a very congested image full of small, unreadable parts. That is why we decided to shrink the image to the display size of the device, and also to leave an opportunity for the student to see it in full size. We ensure this by changing the original image to a reduced version that is also a link to the original-sized one.

- Flash animations: The lack of appropriate animation player softwares would be the reason for adapting animations, but there are implemented players (for example the Macromedia Flash) for PDAs too; therefore, we do not deal with this issue any more.
- Video: Probably the most important problem is adequate adaptation of video files to the given platform. To achieve this, we have to consider the available bandwidth, screen size and computational power in all of the profiles. No adaptation is required in the PC profile. In the notebook profile the bottleneck is the low bandwidth; hence, the video has to be reencoded to cope with this. Adaptation is also necessary in the PDA profile, because of all the constraints mentioned above.

Due to the large number of different video codecs, there are numerous ways for video adaptation. The storage of video streams requires large disk space, so we tried to choose a solution that does not require the multiple storage of the same stream. In the case of adaptation during course generation, this is an unreachable goal, because the SCORM package has to contain all the relevant multimedia content. Thus, the video streams have to be stored in each learning package. In this case we tried to find an effective video coding method, and decided to use the MPEG 4 [2] codec, due to its comprehensive support and effectiveness. In the case of adaptation just before displaying the content, our goal is achievable because the SCORM package has to contain only the best quality video stream, and the LMS can adjust it to the appropriate quality. In this case we can also use coding and delivery methods that require special server applications, by integrating them into the LMS.

The simplest way of adaptation is to encode and store the video stream for all of the profiles, as we have already discussed above. However, with this solution the needed storage capacity is huge, and the user has to wait for the whole file to be transferred to his device before starting the playback. Some PDAs do not even have enough capacity to receive the whole file, so it is not playable at all. The streaming transfer method is appropriate for avoiding this problem. This delivery method is based on a continuous data flow transmission that is decodable immediately; therefore, the user does not have to wait for the download to finish; the playback can be started after a short buffering time. Traditionally, streaming is a broadcast transfer method, but there is a special form of streaming, called Video on Demand (VoD), that sends out the stream only on request.

Using streaming specifies only the mode of the data transfer, but the actual encoding, and thus the storage capacity needed, can be various. Storing multiple video streams using the MPEG-4 codec, and transferring it with VoD is an easily realizable mode of adaptation in the LMS, so we compared the other emerged solutions to this method of adaptation.

One of the possible solutions was to store only the best quality video stream, and to reencode it real-time for the other platforms only on demand. This adaptation requires less storage capacity, but much more computational power, which seriously limits the number of simultaneous accesses. This is why we discarded this idea. Another solution is to use multiple bitrate video streaming, which involves packing several video and audio streams with different bitrates into one file, and to use a special program to select which stream to transfer to the user on demand. To open a video, a single hyperlink is enough; the server automatically chooses the proper stream according to the properties of the Internet connection. This solution is not suitable for adaptation in the LMS system, because the stream choosing algorithm considers only the bandwidth, so we cannot adjust the video to the screen size of the device. Furthermore, the required storage capacity is not significantly less than it is in the solution applied in the CMS.

A way to reduce the needed storage capacity is to use a scalable video codec [3]. There are two different realizations. The first one uses layer-level, the second one bit level scalability for improving the quality. The main point of the layered scalable video coding method is to create different layers of the video stream: a base layer and some enhancement layers. The base layer requires small bandwidth, but has poor quality and occasionally low resolution. The enhancement layers are for improving the quality or resolution. For decoding the video stream, the whole base layer is essential, and according to the available bandwidth and the desired quality, additional enhancement layers are transmittable. The more enhancement layers are received, the better quality the video has. The disadvantage of this method is that the quality improves only if a whole enhancement layer has been received, otherwise it does not improve at all.

The enhancement layers commonly improve only one property of the video stream. These properties can be seen in Fig. 2. The *spatial scalability* controls the resolution of the video stream. Decreasing the resolution results in smaller picture size, which is a disadvantage when the video is played on a PC or notebook, but it is favourable when sending the video to a PDA. *Temporal scalability* modifies the number of frames sent in a second. The fewer frames are sent, the smaller the trans-

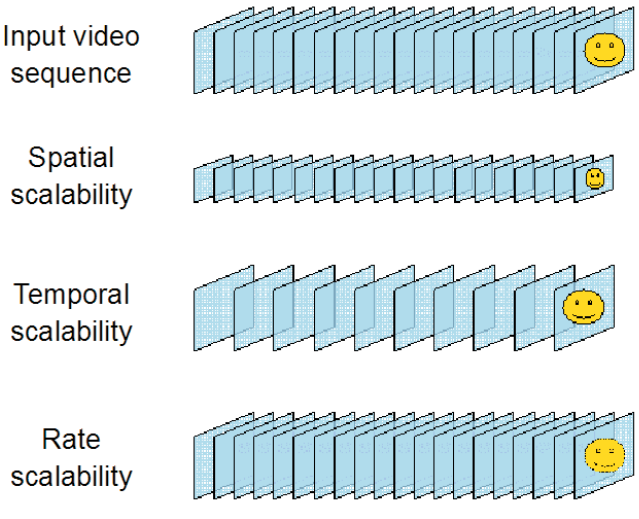

Fig. 2: Different ways of scalability

mittable stream is. However, decreasing it too much brings on discontinuity in the video, and it becomes less enjoyable. This gives a solution for the low bandwidth problem, but not for the small screen size; therefore, this kind of encoding is good for notebooks, but not for PDAs. *Picture quality scalability* (also known as *rate scalability*) reduces the quality of the frames to achieve lower bit rates and faster transmission of the video stream. Just as in the case of temporal scalability, it is beneficial for adapting the video to notebooks, but this does not reduce the geometrical size of the video frames either; therefore, leaving the problem of small screen size unsolved in the case of PDAs.

From the above mentioned properties of the different kinds of scalability, it is clear that none of the scalability methods is able to fit all of our requirements by itself. There are existing combined scalable video codecs, as well. These achieve the best quality on a specific resolution first, and switch to higher resolution only after that. The server programs consider only the available bandwidth while selecting the number of enhancement layers to be transmitted. In our case this behavior would cause a PDA with high bandwidth Internet connection to receive the video stream in too fine resolution.

Fine Granular Scalability (FGS) is a special form of scalable video coding that realizes bit-level scalability. In FGS only one enhancement layer exists and all of the bits in this layer improve the video quality; as a result the enhancement layer can be trimmed at any point; in this way, the transmission can adapt well to the available bandwidth.

In the LMS, with scalable codecs the adaptation for the available bandwidth could be managed, but we would have to store multiple streams for the different resolutions. For the defined profiles it means two streams instead of three, which would also significantly reduce the required storage capacity. The only implemented FGS codec that we found was the MPEG-4 FGS reference codec and server software realized by MoMuSys (Mobile Multimedia Systems). [4] As it was created as a reference implementation, the functionality of this program was quite limited and ineffective. Since the implementation of this reference software, the MPEG-4 community has recognized that the MPEG-4 FGS may not be efficient enough; therefore, the community is now working on a new MPEG-4 based scalable codec, called the AVC Scalable Extension (Advanced Video Coding), but this is still under development.

Our goal to use a scalable video coding method failed due to the lack of appropriately-operating codecs, so the implemented solution was the simplest way of adaptation that we used in the CMS as well. For the purpose of reencoding the video files and Video on Demand, we searched for an existing program, and we decided to use the open source program called VLC (VideoLan Client) [5], which provides both functionalities.

4 How to adapt in a CMS

In this section we present in detail our solution for adapting the course by modifying the CMS. At the time we started this project, we looked for existing open source software to use as a start-up for both of the realizations. That was how we found on a website of the Warsaw School of Information Technology the code of SCOmaker [6], which we used as a starting point for our program.

SCOmaker is written in the Ruby programming language. In the state we found it, it was not a completely working software. It was able to build the hierarchy of the course (from SCOs) and assign the appropriate pages to the appropriate SCOs. Navigation within the pages was also possible, and it generated the SCORM 1.2 compatible manifest file as well. On the other hand, the lecturer had to modify the program code to define the hierarchy, and also it was not able to find the resources. This means that the original software did not fill the pages with content; it only created empty HTML files. Furthermore, it was not able to adapt the courses to different platforms – like most CMSs, it created learning packages for PCs. Also, originally the lecturer had to run four programs to create the learning package.

We have modified the SCOmaker so that it is able to find the resource files, and from them create HTML files that are not empty any more, but contain the learning material. At this time, raw text files (.txt files), web pages containing only formatted text (.htm, .html files), pictures and illustrations (.bmp, .jpg, .jpeg, .gif, and .png files), and videos (.avi, .mpg, .mpeg, .mp4, and .wmv files) can be used as resource files. Another goal was to be able to create the learning package for different platforms, such as notebook with poor quality Internet connection, or PDA. While designing the software, we tried to keep in mind that we are creating a program that is going to be used by people who are not computer specialists. This is why we tried to keep the user interface as simple as possible. Thus, the lecturer only has to modify one tag at a given place in an XML file to choose to which platform he wants to create the course. Then he has to run only one program (instead of four), and he does not need to change the program code to define the hierarchy, he just has to create folders and copy the source files into them.

Because the original software, in the state we found it, was only able to run on Linux platform, we decided to keep this characteristic, and port the program to Windows later. Also, though it was not the latest version of the SCORM recommendation that the SCOmaker implemented, we decided not to modify the SCORM compatible part of the software, because this is the most prevailing version today.

Now let us overview the operation of our software. To run the program, two parameters have to be given. One is the path of the folder containing the course hierarchy; the other is the path of the configuration file, called config.xml. This file contains the parameters needed by the program, such as the path of the template HTML pages, where to put the created learning package, etc.

First the configuration file is processed. Originally, the lecturer had to hard-code the package hierarchy into one of the program files. We modified the program so that the lecturer does not have to modify the program code. Now he only has to create a folder structure that represents the hierarchy of the course (SCOs and pages), and copy the resource files into the corresponding page folders. The resources are placed on the pages in alphabetical order; therefore, the lecturer has to name them accordingly. From this folder structure, a temporary file is created, which later is used to create the imsmanifest.xml file that is mandatory for every SCORM

package. To achieve adaptability, we placed a new tag into the config.xml file, so that only the name of the profile has to be specified in order to adapt the course.

Now let us see how the empty HTML pages are filled with content. The resource files belonging to the appropriate page are collected from a temporary file, processed and inserted into the HTML page according to the extension of the resource file.

When inserting text into the HTML page, the only change made is that the "`\n`" or "`\f`" (new line, new page) characters are replaced by the "`<br>`" HTML tag.

In the case of pictures and illustrations, if no adaptation is needed, the following tag is placed into the HTML code: "`<img src=\"#{fname}\">`" where `#{fname}` is the name and the extension of the file containing the picture or illustration. This file is then copied to the folder of the corresponding SCO. Picture adaptation may only be necessary when the course is created for PDAs. In this case a PHP script is called to check if the size of the picture exceeds the permitted maximum (it is 230×300 pixels – the effective display size of most PDAs), in which case adaptation takes place. Adaptation means that a reduced version of the picture is placed into the page as a hyperlink, which points to the original-sized version. In this case the original picture and the picture reduced with the PHP script are both copied into the folder of the corresponding SCO.

At the insertion of web pages, the HTML code between the "`<body>`" and the "`</body>`" tags are inserted. As in the case of raw text, the automatic text wrapping of the browser provides a proper appearance on small displays.

Fig. 3: E-learning course on a PDA

Placing a video into the course can be done by inserting the "`<a href=\"#{fname}\"> Start video </a>`" line into the HTML code. As a result, a link is placed into the page with a "Start video" label. If no adaptation profile is given, the `#{fname}` is replaced by the original filename, thus the original, unchanged video file is placed into the course. If the PC adaptation profile is provided, the video is reencoded with the MPEG 4 codec because of its efficiency, but the properties of the video (bitrate, resolution, etc.) are not changed. The `#{fname}` is replaced by the original filename, but with .mp4 extension. If the "notebook" adaptation profile is given, the video is encoded with a lower bitrate; in the case of the "PDA" profile, it is encoded with both a lower bitrate and smaller resolution into MPEG 4 format. In all cases the reencoding is done by a separate program, and only the new, reencoded video files are placed into the correct SCO folders. Also the references in the HTML codes point to these files.

Once the hierarchy, the pages and the imsmanifest.xml file are ready, the very last step is to create a .zip archive from the learning course. Besides the fact that SCORM requires the learning packages to be archived into an archive format such as zip, jar, cab, tar, etc., there are several advantages of creating this archive file. The main advantages are: it is easier to upload one archive file than a folder containing several files; sending the learning packages through the Internet consumes less bandwidth, and also the archive requires less hard drive capacity.

By the modifications described in this section, our program is capable of creating complete learning packages with multimedia contents; also the adaptation of these contents is done simply by providing a single extra parameter. Such a package can be seen in Fig. 3.

5 Adaptation in the LMS

For implementing the adaptation during the playback of the course, we searched for an open source Learning Management System that conforms to SCORM. Though there are several open source LMSs, only a few of them are able to display SCORM packages properly. We chose the Moodle [7] system because of its modular program structure, the continuous support and its high level SCORM compatibility.

After uploading a learning package onto the LMS server, the Moodle system checks the SCORM conformity because the playback presumes it, and displaying a non conform course could cause errors. There are two ways to realize this kind of adaptation. One way is to fit the course to the platform after the verification by modifying the displaying module of the Moodle system. The other way is to modify the package itself before the verification by creating a new, adapted SCORM conformed course. Since we had gained much experience of SCORM compatibility and course generation while developing the CMS, we decided to implement the latter method.

Besides producing an adapted learning package, we had to be able to recognize the user's device, in order to know which device to adapt to. We also had to solve the video transmission through streaming. Therefore we installed the VLC program (mentioned in Section 3.) onto the server in addition to Moodle. For streaming a video by VoD, first the video file has to be registered into the VoD server. This is done through a telnet interface, by assigning a unique identifier

to the video file. After that, the video stream is available at the rtsp://servername:portnumber/identifier hyperlink. The video file can be registered to the VLC at the same time with the multimedia adaptation, and the hyperlink can also be inserted into the course at this time, so no other modules of Moodle have to be modified.

The used profile depends on the user's device and also on the quality of the Internet connection; hence, we have modified the login screen to grant the user the opportunity to choose the profile. After this step the program stores the profile in a cookie on the user's device. The remaining modifications of the Moodle system were placed into a separate module.

In order to construct an adaptive SCORM package, it is first necessary to analyze the whole course. Depending on the results of the analysis, the multimedia contents have to be adjusted; the displayed HTML pages and the course descriptor XML file have to be modified. The analysis and the adaptation of the course are based on the processing of the imsmanifest.xml file. The adaptation of the multimedia content takes place when a reference to this content is reached for the first time while reading the manifest file. The adaptation in the LMS is executed for all of the profiles. First, the software checks whether adaptation is required. The only situation when no adaptation is required is when the multimedia content is an image that is smaller than all of the display sizes of the different profiles. After checking this, the program reencodes or resizes the content, if necessary, as described in the previous section. The names of the newly generated files are always formed as `filename_profilename.extension`. The program registers the video files into the VLC server at this point and it stores the new unique video identifiers. The identifiers for the same video but different platforms are formed as `identifier_profilename`. In this way we can ensure that similar video file names will not cause problems in video streaming. After the adaptation of the material, the program refreshes the manifest file to keep the SCORM compatibility. During the processing of the imsmanifest.xml the program collects all of the HTML pages for all multimedia contents that can refer to it. When the analysis is finished, the adaptation of media files is completed, and the course descriptor XML file is consistent with the new package. After that, only the modifications of the HTML pages occur. The program searches for the hyperlinks of the media files in the HTML codes, and replaces them with short JavaScript functions. These functions automatically produce the proper hyperlink using the original filenames (or video identifiers) and the profile name from the previously stored cookie.

After the above modifications of the HTML pages, the new package is adaptable, but still SCORM compatible, so it can pass the verification of the Moodle system. For proper playback it requires the profile name to be stored in a cookie in the user's device, and a VoD server, which provides the video streams. Because of these requirements, only our modified Moodle system can correctly display an adapted course.

6 Conclusion

With the use of modern technologies, our lives are changing. As a result, educational methods are changing as well; computers are used to aid teaching, and mobile devices to develop ubiquitous learning. As e-learning is spreading, the need appears for learning packages to be playable on these mobile devices as well. To satisfy this need we implemented two fundamentally different systems capable of adapting the e-learning content and packages for mobile devices. In this paper, after discussing the fundamentals of e-learning, we presented the design considerations and the different possibilities of scalable video coding. We also explained and justified our implementation choices. Finally, we briefly described the realized systems.

The CMS, that we created for demonstrational purposes, can create an e learning package using existing text, image, video and HTML files and is able to adapt these contents to the given platform. This system, as it is now, meets the requirements of proper generation of platform-specific learning packages. The courses created by our CMS were tested on several LMSs operating on Windows and on a PDA emulator. The most important property of this system is that the constructed courses are displayable by all SCORM conformant LMSs.

Another way to support the use of mobile devices is to convert existing courses that were created for PCs. This approach is advantageous, as it supports every package regardless of the CMS used to create it. Our modified version of Moodle adapts and displays SCORM compliant learning packages properly. We used several courses either created by different CMSs or found on the Internet to test the system. Since different CMSs may apply different ways of inserting a video into the course, problems may arise during playback. However, Moodle handled the video files in the packages that we created with the two CMSs that were used for testing.

To sum up, both of the systems that we implemented are working properly and are capable of learning package adaptation, but there is still room for improvement. The set of supported multimedia contents and the supported profiles could be increased. When a suitable, efficient scalable video codec appears, it can also be easily applied in the LMS system. Implementing version 1.3 of the SCORM recommendation (aka. SCORM2004) could also be considered.

Acknowledgments

The work described in this paper was supervised by Krisztián Németh, and was supported by the High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics.

References

[1] SCORM 1.2 Documentation, October 2001, http://www.adlnet.gov/scorm/history/12/index.cfm

[2] ISO/IEC JTC1/SC29/WG11(MPEG working group), doc no. N4668 The MPEG 4 Overview, March 2002, http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm

[3] Wang, Y., Ostermann, J., Zhang, Y.: *Video Processing and Communications*. Prentice Hall, 2002. ISBN 0-13-017547-1

[4] The home page of the Mobile Multimedia Systems project: http://cordis.europa.eu/infowin/acts/analysys/products/thematic/mpeg4/momusys/momusys.htm

[5] The homepage and the documentation of VideoLan Client: http://www.videolan.org

[6] SCOmaker: http://mxf.wsisiz.edu.pl/scomaker/doc/SCOMAKER

[7] The homepage and the developer's documentation of the Moodle system: http://www.moodle.org, http://docs.moodle.org/en/Developer_documentation

Réka Szabados
e-mail: sr556@hszk.bme.hu

Katalin Sipos
e-mail: sk494@hszk.bme.hu

Department of Telecommunications and
Media Informatics

Faculty of Electrical Engineering
Budapest University of Technology and Economics
2 Magyar Tudósok krt.
Budapest, H-1117, Hungary

