A Content Adapter Architecture for E-Learning Systems

Mark Joselli[1,2] , Eduardo Soluri[2]

[1] mark.joselli@pucpr.br

Escola Politécnica

Pontifícia Universidade Católica do Paraná - PUCPR´

[2]esoluri@nullpoiter.com.br

Nullpointer

Nullpointer Tecnologia

Abstract. In this paper, a content adapter architecture for learning system is presented. Normally, most e-learning systems are designed for web browsers. In order to be adapted for others consumers, like mobile devices, some constraints are introduced due to the different characteristics and constraints of these devices, like different screen dimensions, that would require some form of adaptation. This adaptation task can consume time and resources, since most solutions resolve it by creating different resources for different devices. In this work, a different approach is used, the content is adapted in a middle layer (in a non intrusive way), between the content provider and the device. The adaptation is done with the use of templates, which describes how the content must be adapted. This architecture consist in a backend server and a front end client. Also, the server is responsible for maintaining a version control of the content, message services, statics collections, among other features. While the client provides a thin layer, build as a native web application, which can provides interaction with the devices features and also a cache system for avoiding data transfers and also providing an offline content.

1. Introduction

The m-learning is possible by the mobile phones and tablets, providing several benefits for the students [e Dulclerci Alexandre e Clara Maria Coutinho 2006], like allowing them to study on there spare time. But these kind of devices, like smartphone and tablets, have many constraints, when compared to PC, like: hardware constraints (processing power and screen size); user input, (buttons, voice, touch screen and accelerometers); and different operating systems, like Android, Blackberry OS, iPhone OS and Windows Mobile [Joselli et al. 2012b]. All these different characteristics must be acquaintance when the content needs to be developed for this kind of device, making extremely difficult for the design and adaptation of the content for all these platforms and devices. The presented architecture provides a content adapter architecture, which can be used to adapt all the content for mobile devices.

In general, the content adaptation envelops not only the adaptation of format and types, but also different styles, dimensions, data compression and specifications, since quality of experience of the user can suffer from a not adapted (or poor adapted) media [Agboma and Liotta 2010]. Also, different contents and information can be available for

specifics devices or systems, requiring a custom adaptation or generation of the content, which the presented architecture provides.

Most of the e-learning content come from in-house or customized e-learning platforms [Ardito et al. 2006]. In order to delivery this content on mobile environments, they need also to provide solutions for this problem. The presented architecture provides an adaptation of the e-learning content for the different device with the use of templates, providing a generic non-intrusive content adaptation, as a middleware. Also, natively, most e-learning platforms do not provide cache mechanism, compression of content, and version control (for the content on the client) like the presented architecture provides.

The EAD.BR census [de Educação a Distância (ABED) 2012] shows that the use of mobile devices on distance courses is still one of the least used form of educational resource in Brazil. With the use of this architecture could facilitate the popularization of the use of mobile devices integrated with e-learning platforms, since it facilitates the integration and adaptation for these devices.

The architecture consists in two modules, a server and a client. The server is responsible for adapting the content from different providers through the uses of templates, compressing the adapted content in order to delivery to the device and controlling the different contents versions. The client is an application installed on the device, which is able to open the package, sender by the server, and render its content. Also, through the client its possible to use its native features like location, connect to social networks, gather statistics data, and more. The client is implemented as a thin client using characteristics of a hybrid application, that uses web patterns (HTML5, Cascading Style Sheets and Javascript) to provide the graphics interface, and a native code, in order to access all the available devices capabilities.

Mobile devices, normally, do not have connectivity all the time, needing some offline data in order to function when there is no network available. Also, the connectivity can still be very slow and expensive, depending on the network connection and the network carrier, requiring some form of data compression and caches techniques to reduce the data transfers, which also increases the battery life of the device. This architecture provides a cache mechanism in order to fulfill these requirements. A version control of the content is also provided, so that the content transferred between the server and the client is only the difference between the data that is on the device and the new data from the server.

The architecture was developed to be used together with e-learning platforms. This work also shows a test case of the architecture with a mobile application accessing the adapted content provided by the architecture, which were created from resources made for e-learning.

This work is divided as follows, section 2 presents the methodology and section 3 shows the related works. Section 4 presents the architecture server and section 5 shows the client. Section 6 shows the test case in order to validate the presented architecture and finally section 7 presents the conclusions.

2. Methodology

This work has been developed in some stages. This stages have not a chronological order, since the different activities were constantly revised during the development.

The first stage were the revision of the current works, considering architectures that would do some form of adaptation and also m-learning systems. With that, this work has highlighted the most important features for the presented architecture.

The second stage consist on the development of the server architecture and the client on the mobile device, showing its features and technologies used. The last step consist on the validation of the architecture in a learning application. The validation were done by tests with different users and devices.

3. Related Work

There are some works that provides a midlleware to adapt the content. Trinta et al [Trinta et al. 2008] presents a middleware to support multiplayer multi-platform games. Between the services that it provides, there is a content adaptor, which transforms the information of the game according to the actual context of the player. Another middleware service is the work Ubidoctor [Diniz et al. 2008]. The Ubidoctor is a middleware for the construction of ubiquitous application for the medicine area. It provides the service of session management, context management and content adaptation. The content adaptation transforms the resources for the ubiquitous healthcare applications.

Delicato et al. [Delicato et al. 2009] and Tummala and Jones [Tummala and Jones 2005] show works on context-awareness and location-awareness, were specific midllewares are developed for this purpose. In this kind of application the content delivery for the end-user is dependent on his context or location. The presented architecture could also be used for this purpose, since its templates can require specific data from the middleware and the thin client can provide information about localization and send it to the architecture to provide the content for the location or the context.

One main disadvantage of the works presented so far is the lack of generalization in order to be used in others contexts, like different content providers. One different approach is the CAS service [Carvalho and Trinta 2009] that provides a content adaptation, which runs as a generic service performing content adaptation of text, audio, video and image for multi-platform applications. They built its services as modular and nonintrusive, similar to the presented architecture, but the CAS lack of some of its features, like the cache, data compression and version control of content.

The wireless e-learning, mobile learning, or simply m-learning is the development of the e-learning for mobile platforms [Toledano 2006]. Despite the great popularization of mobile devices, these kind of learning platforms are still very few [Toledano 2006].

[de Morais et al. 2011] shows the a mobile game that uses m-learning for help teaching computing theory, and in [Muhlbeier et al. 2012] present a mobile comic application for mobile devices. Both of them without any integration with a e-learning platform. The presented architecture could also be used to develop this kind of applications, requiring only that all the content are embedded in the client.

[Motiwalla 2007] show a mobile platform using SMS and WAP, were the content were developed exclusive for the device. [Batista et al. 2011] shows a m-learning experience with the use of a mobile moodle plugin for simple mobile phones. Also [da S. Ribeiro et al. 2009] shows validation tests with the same moodle plugin, highlighting that a video repository is a missed feature that is required for better m-learning experience.

The presented architecture could also be used for adaptation for WAP context, but it would miss some of the features of the device, like the cache and other native features. And the architecture could be also fully integrated with video repositories, like YouTube, Vimeo or even proprietary repositories.

Earlier versions of the architecture were applied to a game content server and for web content [Joselli et al. 2012a] and for web content [Joselli et al. 2012c, Joselli et al. 2013] present some concepts of the presented architecture.

4. The Content Adapter Architecture Server

This section is devoted to explain how the server part of the architecture works. It is divided in three subsections, which core elements of the architecture are shown: the content adaptation, the content version control and the identity server.

This architecture is build as a web service that adapts and creates content from different systems based on templates. An overview of the server can be seen on figure 1.

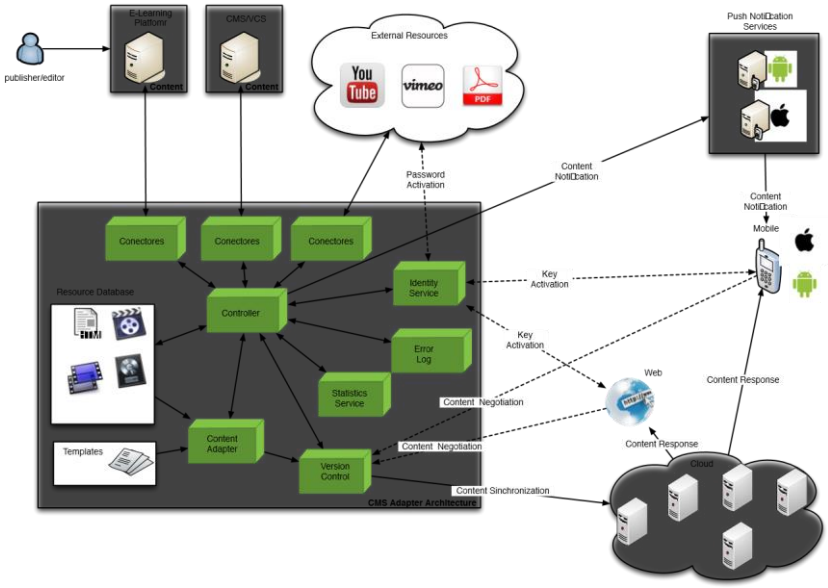

Figure 1. Overview of the Architecture Server.

In this figure, the main modules of the architecture can be seen. The organization module is shown, that is composed by the resources provides, which can be the m-learning platforms or even a web portals. These content are gathered by the controller and saved on a database. The controller is the service that connects all the different ones, and make them work together. Afterwards, the resources are adapted through the use of templates and them saved on a proprietary server or on a cloud server, like the amazon S3 and cloud front. Also, resources that come from secure services can be delivered and adapted, using the identity server. The Push notification services that are normally on the brand server, are accessed by a web service, and triggered by the controller when new content arrives.

The templates for customization and configuration of the architecture have to be registered in the controller by a developer/publisher/editor. These templates are built upon an XML based language, and they are the bases of how the architecture adapts the

contents. Also, some content can be specific for some device, like icons and logos, in this case this content can be on an outside server or it can be in the architecture.

The architecture is built upon components. On the basis of the experiences in using the architecture in different application domains, the variety of technical implementations were used to abstract and create a general architecture where different methods for context adaption, personalization and contextualization can be achieved.

The architecture server is responsible for: gather periodically content from the registered content providers; adapt content for the different platforms according to templates; implement a AAA server (Authentication, Authorization and Accounting) in order to implement a security server to communicates with servers that needs secure policy, like login to secured e-learning platforms for gather of content; manage of the user statical data, collecting from the devices and saving it for analytical statistics; send push notification to the end-user when necessary (when configured to do so, like when new content becomes available); cryptograph content based on public and private keys, if needed; keep an version control of the content on each user, and send the new content based on it; compress the content to send to the user; provides statistic report from the service done, and also from the information gathered by the client.

The architecture has a error log in the server, which is responsible for gather any error or warnings that can come from the resource gathering, adaptation of content or the generation of a new version. This error can come from different facts, like trying to gather a resource that is not on the server or adapting a content that cannot be adapted to the description that is on the template. This error logs can be delivery to registered emails, or be seen on the error report in a web report.

4.1. **Content Adaptation**

The publication of the content for the device can be done by two ways: by a content aggregation (pull) and content syndication (push). The content aggregation is the ability to pull content from content provider and adapt it through templates. The Content Syndication is done mainly by the user, in the content provider and requires the republishing of the content for the required device, in this way this content would only be copied to the architecture repository. The content syndication is used mostly only used on content that are used mainly on the device, like icons and logos.

The content adaptation is responsible for gathering the content from the content providers and adapting it to the required devices. This component normally uses a XML for the configuration of the adapter, and a series of XMLs describing how each of the content should be adapted. The process is simple, it uses a XML to gather information about the service that provides the content, and how it should gather and adapt these content.

The component can gather the information in two ways, by an push notification made by the content provider, or in an automatic manner with an timer. The push notification requires that this service is implemented in the content provider, but it has the advantage that the content can be adapted as soon as published. In the automatic manner, this implementation is not required, but the content is only published for the media when the service runs.

This service can support different contents, like sound, music, documents, video and HTMLs. The adaptations of these content are made by templates, which are made using the xml descriptor. These templates are implemented using the template method design pattern, which defines the program skeleton of an algorithm, in this case the methods needed for the content adaptation. There are a series of templates already built as components for the architecture, like images, audio and video. Sometimes, different devices implements different multimedia capabilities. With that, some devices can have different media format to be able to use it. The architecture provides the adaptation of format types and also the images proprieties.

The image adaptation is normally done by the ImageMagick library [1], which is an free open source library, that the architecture uses to change the image properties, like dimension, format and qualities. For the adaptation of audio or video, the architecture uses the FFMPEG library [2], which is an open source library, in order to convert the video/audio files, this way the adapter can change its properties, like the frame rate, format type, quality and dimensions. The configuration for each common device (mobile device) are registered in the application.

There are also adaptations of HTML pages for the devices. This is done by templates, where some tags are selected and others adapted, also the CSS and Javascripts files are sometimes modified or replaced in order to achieve a better final result, which are all define by templates.

4.2. **Content Version Control**

The content version control is responsible for keeping the control of each version of the adapted content. Every time the architecture gathers content from the content provider or if the templates are changed, it generates a new version on its repository. This component was built using a balk pattern that is a design pattern that only executes an action on an object when the object is in a particular state, in this case it only update and generates a new version of the content, when all the content gathering and adaptation has finished doing its work.

Since the content can chance frequently, and each user can have different versions, when the user updates its contents, he uses the user's version number that he has on the cache, so that the client only downloads the needed content for the up-to-date version. This way, the architecture tries to keeps the communication between the server and device to a minimum.

All the content generated by the content adaptation, are normally saved on the cloud, using Amazon S3 [3], or similar services for cloud storage. Also all the content, that is sender to the user, is compressed in a zlib format by using the zlib library [4]. This way the communication between server and client is kept to a minimum, which is good for mobile devices that sometimes suffer from slow network and expensive data plans.

[1] www.imagemagick.org

[2] www.ffmpeg.org

[3] http://aws.amazon.com/pt/s3/

[4] www.zlib.net

4.3. **Identity Server**

This server of the architecture is used for different things: an AAA server (Authentication, Authorization and Accounting); a certification generator and manager for security connection between devices and the service; a protection for special content, by doing cryptography on the files; and a statistics repository. Based on the concept of Authentication, Authorization and Accounting, this component is responsible for implementing the security requirements for data exchange and consumption of restricted provided content between clients, the devices, and the servers.

Being an Identity Server, it is responsible for provisioning the user and its devices with the available access of content. This also requires the creation and management of certification keys and tokens for the required content cryptography.

This server also gathers statistics information of each device, and of each user, of the version of the content, device capabilities and how the user consume the content. This server can provide real-time statistic report, dividing by users, devices or time.

5. **The Thin Client**

The thin client is responsible for: gather the data from the server, provide the server with the characteristics of the device, implement a cache system for the gathered data for offline use and gather of user data for statistics reports.

The client is developed as a framework, in order to be reused in others projects. It is mainly developed in object-C and Java, in order to work with the apple IOS and Google Android platforms. Others platforms are being developed, like for Window mobile, Blackberries and J2ME. This client can show different resources like, HTMLs, images, audio and video. Also, It implements components for localization system, cache management, statistics data gathering and integration with social network, like Facebook, Twitter and Google+.

This client was implemented mainly as a hybrid application, which is a mix of web app with native apps. Web apps are web sites, which all the content and logic are made exclusively for each device, but it cannot access some of the capabilities of the device. Native apps are implemented on the device, with a higher cost of implementation, but it can access all the functionalities of the devices that can be used by developers. Hybrid applications are a mix of both web and native apps, it uses mostly web for the interface, but it is still a native app, so it can use all the resources of the device, similar to the native app.

The AAA (Authentication, Authorization and Accounting) client is responsible for the implementation of the procedures of identification of the user and establishment of his permissions. It supports security communication (through https protocol) and cryptography encryption and decryption.

Also the client provides a service locator, where all the device capabilities are mapped, in order to provide the best available content for the end-user. These capabilities can be device model, screen size, location services, types of available inputs, camera, and all the characteristics of the device available for the client.

The client uses extensively the cache for its contents. It uses this cache for providing the end-user with offline content and also minimize the data transfer between updates.

This cache can also be cryptographic or compressed if needed by the application.

In order to update the data in the client, the following workflow is used: first the device, if registered gets and push notification of new content availability; then when the user opens the application, the device authenticates and it gathers from the server the needed resources for its update; in the case the user has not an available connection, the application starts without this update process, by accessing the cache resources.

6. Validation

In order to validate the architecture, a test case with a quiz HTML page were developed. It consist in a series of questions, with text, images and videos. The content provider were a proprietary web service, which is used as pull service, where the content were gathered five times a day.

The thin client used for this application was adapted in order to work on iPhones, iPads, Android phones and Android tablets. This client show the following contents, images, HTMLs, videos and integration with social media (from Facebook and Twitter). This application also gathers some statistics data that is sent from time to time to the server to analyze the use of the software by the user. These information can be used latter for investigating the use of the app.

This work has done some tests with 10 different users and 5 different devices (an low end Android phone Samsung Galaxy Y, an high end Android phone Samsung Galaxy 3, an Android tablet Asus Transformer TF100, an Apple iPhone 4S and an Apple iPad2) to see their feedback. The first tests were done without the use of the presented architecture, where the HTMLs, videos and images came direct from the content provider. The last tests were done with the use of the architecture and the native app. The feedback were done by a series of questions.

The users would evaluate the loading time and execution of the application with the question ”How was the velocity and response time of the quiz?”. All users agreed that the presented architecture had a much better overall performance. This were expected, since the architecture download all the content at the beginning of the application, in a compressed format.

The users would evaluate the layout adaptation with the question ”How was the layout and images over the quiz?”. All users agreed that in the low end Android phone the layout were very bad and that it were fixed with the architecture, and they all agreed that the architecture improved the visualization in all the devices. It could be observed that the image on the low end device, without the adaptation of the architecture, was too big, breaking the HTML layout.

The users would evaluate the text and overall legibility with the question ”How was the overall legibility over the quiz?”. All users agreed that without the architecture, the low end Android phone the text were very difficult to read, since it would require a lot of vertical and horizontal scrolling. Also two users had difficult to read with the iPhone

4S and they said that it were fixed with the architecture. All users also agreed that the architecture improved the reading in all the devices.

One example of the architecture in use that also provides the content for the web site can be seen on figure 2. In the figure can be seen the content of the quiz ((a),(b) and (c)) on different devices. In (a), the original content on the web site, in (b) the content in a smartphone and in (c) show the same content on a tablet. From these images, can be seen that different layouts and images sizes can be adapted by using the architecture.

(a) A content on the web site (b) A content a smartphone (c) A content on a tablet on

Figure 2. The Architecture in action adapting HTML content and image.

7. Conclusions

The use of e-learning in mobile devices insert many constrains on the content, that without an adaptation scheme, would suffer from bad visualization scheme. This work has presented a new adapter for content providers, that acts as a layer between the content providers and the devices, providing content adaptation of HTML, text, audio, images and videos. Also, devices can have connection constraints, like availability and this data exchange can be very expensive, and the architecture also provides a cache system, data compression and version control, needed for the offline content and to keep the network data exchange to a minimum.

This work has also validate the architecture with users comparing results without the architecture. Observing the results, can be seen that all the users had a better mlearning experience with the use of the architecture.

References

Agboma, F. and Liotta, A. (2010). Quality of experience management in mobile content delivery systems. *Telecommunication Systems*, 49(1):85–98.

Ardito, C., Costabile, M. F., De Marsico, M., Lanzilotti, R., Levialdi, S., Roselli, T., and Rossano, V. (2006). An approach to usability evaluation of e-learning applications. *Universal access in the information society*, 4(3):270–283.

Batista, S. C. F., Behar, P. A., and Passerino, L. M. (2011). M-learnmat: Aplicacao de um modelo pedagogico para atividades de m-learning em matematica. *SBIE*.

Carvalho, D. and Trinta, F. (2009). Content adaptation for multiplatform applications. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, WebMedia '09, pages 41:1–41:4, New York, NY, USA. ACM.

da S. Ribeiro, P., Franciscato, F. T., Mozzaquatro, P. M., and Medina, R. D. (2009). Validacao de um ambiente de aprendizagem movel em curso a distancia. *SBIE*.

de Educacao a Distancia (ABED), A. B. (2012). Censoead.br.

de Morais, D. C. S., Alencar, A. D. P. C., and de Souza, R. (2011). Jogo baseado em m-learning e aprendizado tangencial para auxilio ao ensino de teoria da computacao. *SBIE*.

Delicato, F. C., Santos, I. L. A., Pires, P. F., Oliveira, A. L. S., Batista, T., and P`ırmez, L. (2009). Using aspects and dynamic composition to provide context-aware adaptation for mobile applications. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 456–460, New York, NY, USA. ACM.

Diniz, J. R. B., Ferraz, C. A. G., Trinta, F. A. M., Melo, H. N., and Santos, L. M. (2008). Avaliacao de um servico de gerenciamento de sessao para ambientes de medicina ubiqua. In *Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*, WebMedia '08, pages 4–11, New York, NY, USA. ACM.

e Dulclerci Alexandre e Clara Maria Coutinho, J. B. J. (2006). M-learning e webquests: as novas tecnologias como recurso pedagogico. *SBIE*.

Joselli, M., Junior, J. R. S., Zamith, M., Clua, E., Soluri, E., and Tecnologias, N. (2012a). A content adaptation architecture for games. In *SBGames*. SBC.

Joselli, M., Passos, E. B., Junior, J. R. S., Zamith, M., Clua, E., and Soluri, E. (2012b). A flocking boids simulation and optimization structure for mobile multicore architectures. *SBGames 2012*.

Joselli, M., Soluri, E., Junior, J. R. S., Zamith, M., and Clua, E. (2012c). mcms: A content management system adapter architecture for mobile devices. In *XI Workshop de Ferramentas e Aplicac¸oes (WFA) – Webmidia 2012˜*. SBC.

Joselli, M., Soluri, E., Junior, J. R. S., Zamith, M., Clua, E., and Micheli, K. (2013). A content adapter architecture for cmss. In *The International Conference on Information Technology and Applications (ICITA)*. IEEE.

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

Muhlbeier, A. R. K., Mozzaquatro, P. M., Medina, R. D., de Oliveira, L. C., Moreira, R. C., and Antoniazzi, R. L. (2012). Mobile hq: O uso de softwares educativos na modalidade m-learning. *SBIE*.

Toledano, M. C. M. (2006). Learning objects for mobile devices: A case study in the actuarial sciences degree. In *Current Developments in Technology-Assisted Education (2006)*.

Trinta, F., Pedrosa, D., Ferraz, C., and Ramalho, G. (2008). Evaluating a middleware for crossmedia games. *Comput. Entertain.*, 6(3):40:1–40:19.

Tummala, H. and Jones, J. (2005). Developing spatially-aware content management systems for dynamic, location-specific information in mobile environments. *Proceedings of the 3rd ACM international workshop on Wireless mobile applications and services on WLAN hotspots WMASH 05*, page 14.