CAA: A Content Adapter Architecture for Content Management Systems

Mark Joselli, Micheli Knechtel
Escola Politécnica
Pontifícia Universidade Católica do Paraná - PUCPR
Paraná, Brazil
mark.joselli@pucpr.br

Jose Ricardo Silva Junior, Marcelo Zamith, Esteban Clua
MediaLab, IC-UFF, Brazil

Eduardo Soluri
Nullpointer Tecnologias, Brazil

***Abstract*—This paper presents a Content Adapter Architecture (CAA) for Content Management Systems (CMSs). A CMS is a software application that maintains and keeps track of every piece of content of a website. In the past, CMS has been mainly used in developing websites and portals for desktop PCs. For mobile devices such as smart phones and tablets, the web content developed for PCs needs to be adapted in order to provide the best possible user experience due to different characteristics and constraints such as screen dimensions. This adaptation task can be time consuming and resource intensive. Most solutions resolve it by creating different resources or CMS plugins for different devices. This paper proposes a novel approach to adapt CMS content in a non-intrusive way, through the use of templates which specify how the content should be adapted. The CAA consists of a backend server and a front end client. The server is responsible for the adaptation and provides version control of content, amongst other features. The client, built as a web application, is a thin layer which provides interaction with the device features, and acts as a cache system to reduce data transfer.**

Keywords—CMS; content management systems; content adaptation; multimedia; mobile devices; software architectures

I. INTRODUCTION

Mobile devices, like smartphone, tablets, and digital TVs have many constraints [1], when compared to PCs, including hardware constraints (processing power and screen size) [2], user inputs (buttons, voice, touch screen and accelerometer), and operating systems [3], (Android, Blackberry OS, iPhone OS, Symbian and Windows Mobile). Due to all these different characteristics, developing and adapting content for all these devices and platforms become extremely difficult. In addition, the context, user preferences, user locations, and time of use, must also be taken into account [4]. The presented CAA architecture was developed to adapt all content for mobile devices.

In general, content adaptation includes not only the adaptation of format and types, but also styles, dimensions, data compression and specifications. The quality of user experience can suffer from a non-adapted or poorly adapted media [5]. Also, different content and information can be made available for specific devices or systems, requiring a custom adaptation or generation of the content, which the CAA can provide.

Developing websites with CMSs offers many advantages including better content organization, increased access to resources, and greater organizational effectiveness. CMSs can be used, natively or with plugins, for others devices such as TVs and mobile devices, which can require the input of new content. The CAA provides an adaptation of the web content for the different devices with the use of templates, providing generic, non-intrusive content adaptation. Also, natively, CMSs do not provide cache mechanism, content compression, and version control (for the content on the client) like the CAA provides.

The CAA was developed to be used together with popular CMSs like JOOMLA and DRUPAL, and also with proprietary CMSs. This paper presents a test case of the CAA with a mobile application accessing the adapted content, which were originally created from resources made for web sites developed by a proprietary CMS and a JOOMLA CMS. Earlier versions of the CAA and their applications and concepts can be found in [6-8].

Mobile devices usually do not have continuous connectivity. When the network is not available, mobile devices may still require some offline data in order to function properly [9]. Also, the connectivity can still be very slow and expensive [10], depending on the network connection and the network carrier, requiring some form of data compression and cache techniques to reduce data transfers. The CAA 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 CAA consists of two modules, a server and a client. The server is responsible for adapting the content from different CMSs though the use of templates, compressing the adapted content in order to ensure delivery to the device, and controlling the different content versions. The client is an application installed on the device, which is able to open the package, sent by the CAA, and render its content. Also, through the client it is possible to use the native features such as location, social network connection, statistical data collection, etc. The client is implemented as a thin client using a number of web technologies (HTML5, Cascading Style Sheets and JavaScript) to provide the graphical user interface,

and a native code, to access all the available device capabilities. In summary, CAA has the following features:

- ability to gather resources from different content sources, such as CMSs;
- adaptation of this content for different device characteristics;
- creation and control of content versions;
- collection of statistical data, in order to create users profiles;
- a cache mechanism, in order to provide offline data;
- data compression for network data exchange;
- and a thin client application for content visualization and server communication.

This work is divided as follows: Section II presents the related works. Section III presents the CAA server and Section IV shows the client. Section V shows the test case in order to validate the presented architecture, and finally Section VI presents the conclusions and future works.

II. Content Management System

Content and content management are current trends in the world wide web [11]. A CMS is a computer system that has the main objective of facilitating the maintenance of content [12], [13]. The CMSs help in publishing, editing, and modifying content as well as the overall maintenance from a central page or control. It is normally used in a collaborative environment, so most CMSs provide a collection of procedures, automatic or manual, for managing the workflow.

The content managed by a CMS includes almost anything from documents, movies, text, and pictures, to many others. The CMS acts as a central repository for all this content, and sometimes as a version control for the content.

The functionalities of a content management system can be broken down into several main categories, such as content creation, content management, publishing and presentation. The presentation component compiles and presents the information that the user sees [14]. The CAA was developed to work with CMSs, providing a layer after the presentation, where the data that would be present normally to a website is adapted to mobile devices.

III. Related Work

CMSs can be used to provide web content for different types of devices. This can be done in different ways. Normally it requires the publishing of different versions of content for different platforms. Another common solution is the development of plugins or add-ons for the CMS that could achieve the same effect by adapting the resources. Nurminen et al. [15] described the use of a Drupal CMS with some add-ons for mobile website provisioning.

Some companies have developed CMSs specific for mobile platforms, like Magnolia CMS [16], Cellular CMS [17], Mobile CMS [18] and mFabrik [19] to name a few. They work in a similar way to the traditional CMSs that are customized for mobile content. Manashty et al. [20] presented the ARMrayan MCMS, a CMS designed for J2ME mobile applications. Moreover, the paper by Hildebrand et al. [21] described a CMS platform for e-learning specific for mobile devices.

There were other projects that used middleware to adapt content. Trinta et al. presented a middleware to support multiplayer multi-platform games [22], [23]. A content adaptor was used to transform the game information according to the actual context of the player. Ubidoctor was another middleware service [24] developed for the construction of ubiquitous applications in the medical 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. [25] and Tummala and Jones [14] described their work on the context-awareness and location-awareness problems, where specific CMS or middleware were developed for these purposes. In this kind of application the content delivery for the end-user is dependent on his context and/or location. The CAA could also be further developed to include this service, as the CAA templates can require specific data from the CMS, and the thin client can provide information about localization and send it to the CAA that would then deliver the content for the matched location or the context.

One main disadvantage of the works presented so far is the lack of generalization for use in other contexts, like different content providers. One different approach reported was the CAS service that provides content adaptation as a generic service performing content adaptation of texts, audios, videos and images for multi-platform applications [26], [27]. The CAS service is modular and non-intrusive, similar to the CAA, but lacking features such as cache, data compression and version control of content.

In relation to the cache feature of the CAA, Li et al [28] presented a statistical mechanism for maintaining cache consistency in mobile environment. It tried to deliver a new cache mechanism since most existing cache consistency strategies assume reliable communication between mobile terminals, and cannot adequately handle frequently offline devices (a common situation on mobile devices). Our work also uses cache, and tries to avoid inconsistency in the cache. However, the CAA uses a simpler mechanism that requires the verification of the resource hash code to achieve this purpose.

IV. The Content Adapter Architecture Server

This section explains how the server part of the CAA works. It is divided into three sub-sections to describe the core elements, namely the content adaptation, the content version control, and the identity server.

The CAA server is built as a web service that adapts and creates content from CMS based on templates. The main modules of the CAA server are shown in Figure 1.

The organization module at the top left corner is composed of the resource providers, which can be CMSs or web portals. All content is gathered by the controller and saved to a database. The controller is the key service that connects all the

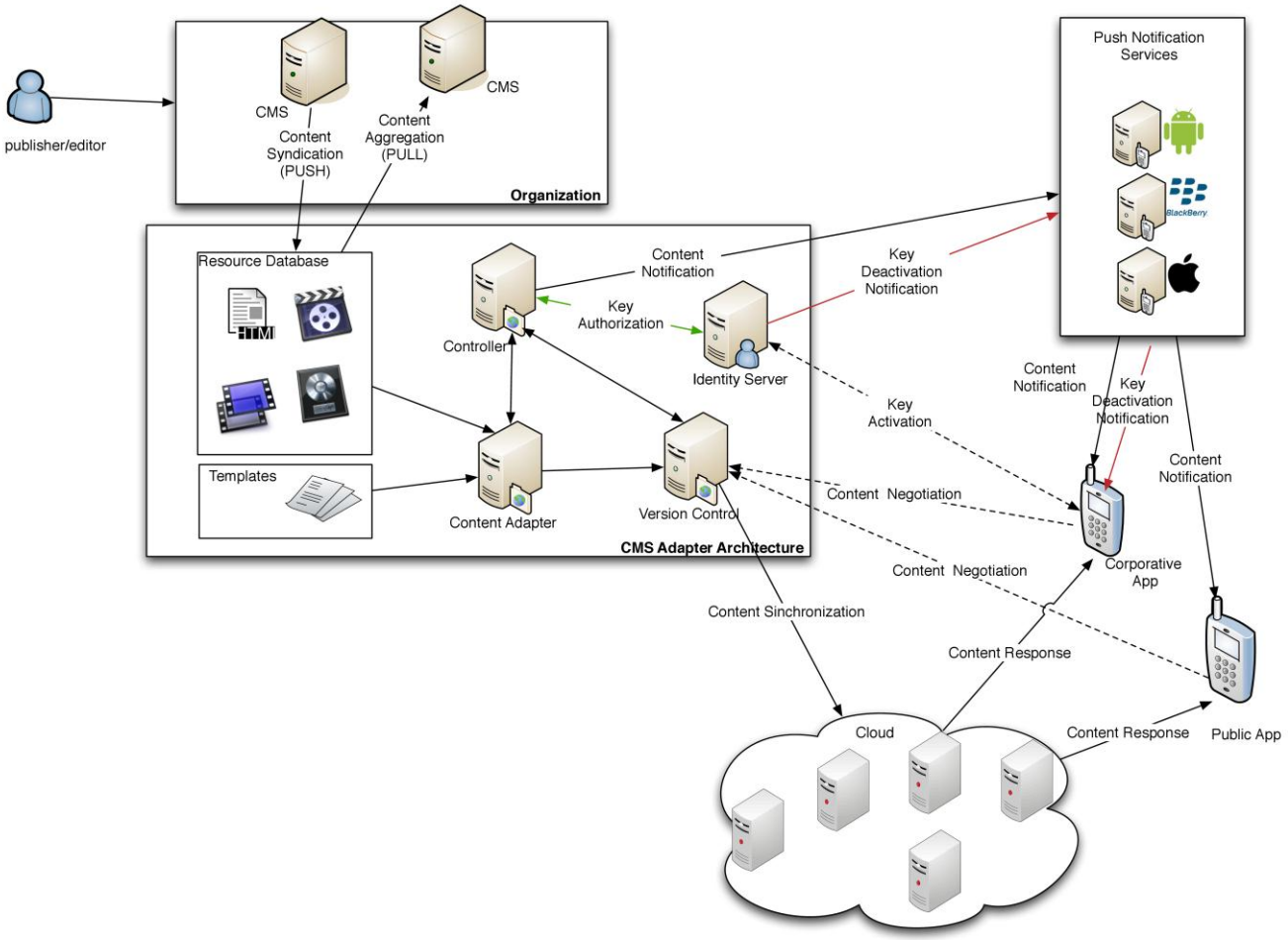

Fig. 1. Overview of the CAA Server

components, and makes them working together. The resources are then adapted through the use of templates and saved on a proprietary or cloud server, such as the Amazon S3 and Cloudfront. Resources that come from secure services can also be delivered and adapted, using the identity server. The push notification services are accessed by a web service, and triggered by the controller when new content arrives. The process flow is illustrated in Figure 2.

The templates for customization and configuration of the CAA have to be registered in the controller by a developer, publisher, or editor. These templates are built upon an XML based language, and they dictate how the CAA adapts the content. Some content like icons and logos can be device specific. In these cases the content can be stored in the CMS or CAA server.

The architecture is built upon components. Based on our previous experience in applying this architecture in different domains, a 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 CAA server is responsible for:

- periodically gathering content from the registered CMSs;
- adapting content for the different platforms according to templates;
- implementing a AAA server (Authentication, Authorization and Accounting) in order to implement a security server to communicate with servers that need secure policies, for example, login to a secured CMS for content;
- managing user statistical data; collecting it from the devices and saving it for statistical analysis;
- sending push notifications to end users when necessary (when configured to do so, like when new content becomes available);
- cryptographing content based on public and private keys, if needed;
- keeping an version control of the content on each user, and sending new content accordingly;
- compressing content to send to users;

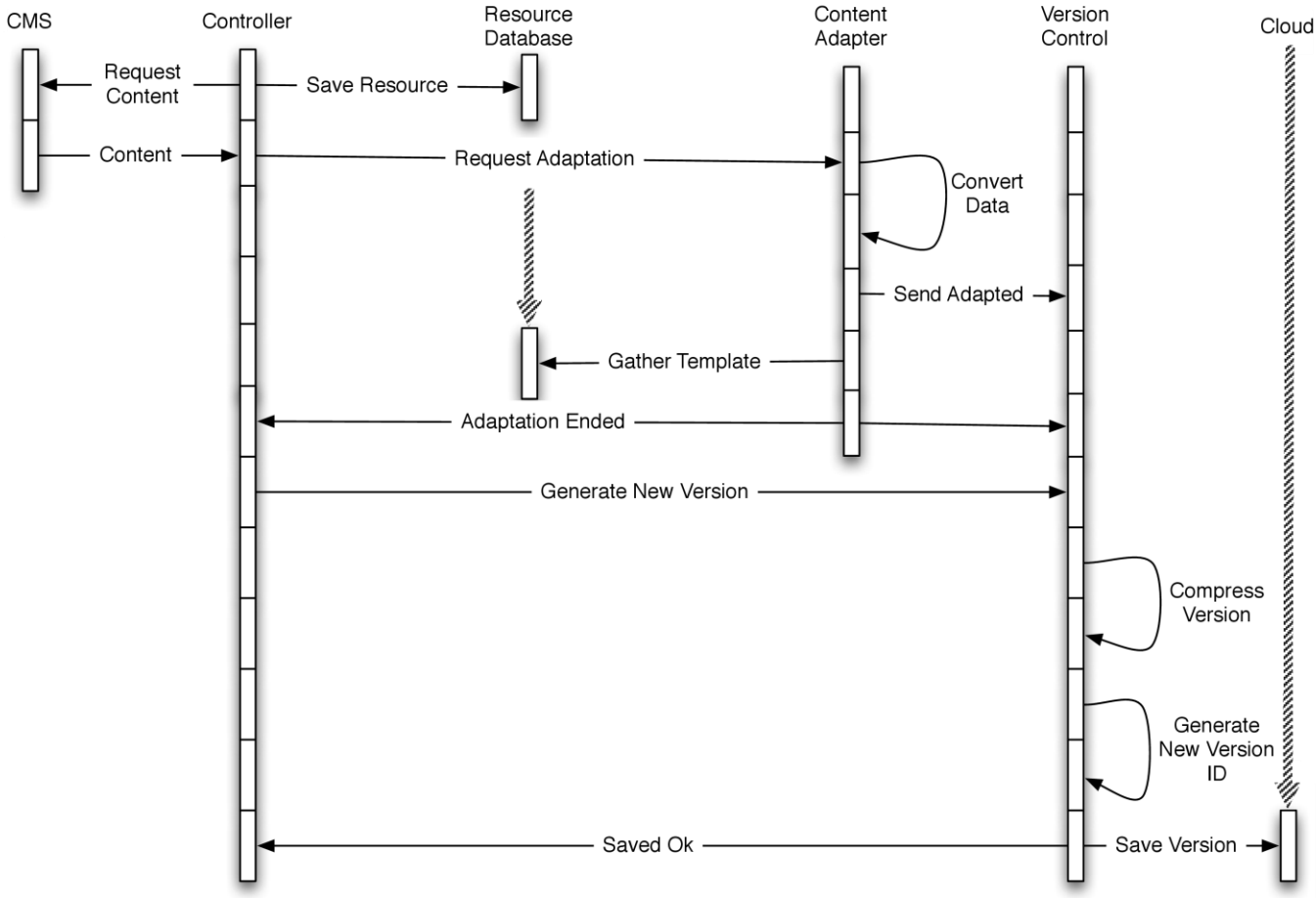

Fig. 2. Process Flow of the CAA Server

- generating statistical reports from the service logs, and information gathered by the client.

In addition, the CAA has an error log keeping records of any error or warning coming from resource gathering, adaptation of content or new version generation. For example, an error is generated when attempting to gather a resource that is not available on the server, or when adapting content that cannot be adapted according to the template. These error logs can be delivered to registered emails, or be shown as web pages.

A. Content Adaptation

The publication of content for the device can be done in two ways: by content aggregation (pull); and content syndication (push). Content aggregation is the ability to pull content from CMS and adapt it though templates. Content syndication is done mainly by the user in the CMS and requires republishing of the content for the required device. In this way, the content would only be copied to the CAA repository. The content syndication is mostly used on content such as icons and logos.

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

The component can gather the information in two ways, by way of a push notification made by the CMS service, or in an automatic manner with a timer. The push notification requires that this service is implemented in the CMS, 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 content types, like sound, music, documents, videos and HTMLs. The adaptations of these content types are specified by templates using XML descriptors. 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.

```
<Config>
  <Contents cms='Joomla'>
     <content type='automatic'>
```



```
        <service timer='2h'>
http://201.200.1.132/cinema.php
        </service>
        <paramters>
          <param type='datetime'>
date
          </param>
        </paramters>
        <template>
            cinema.template.xml
        </template>
      </content>
  </Contents>
</Config>
```

With the XML above, the CAA gathers information from a Joomla CMS using a scheduler of two hours, i. e., it will run every two hours to check for new contents. When new content is available, it will gather the content using the http://201.200.1.132/cinema.php web service, which requires the datetime as an input. For the adaptation of the content it will use a template which is defined in the file cinema.template.xml.

Scheduler pattern is used in order to call each CMS service, when they are configured as a timer for content gathering. The command pattern is used in order to encapsulate all the information needed for the service to be called, when the time comes.

It can also be seen in the XML tag <Contents> in which each content piece is described, and how each should be adapted for the desired platform using another XML. As an example the XML <Template> below uses an image adaptation to present a 320x240 image for the end-user.

```
<Template>
         <method>imageAdapt</method>
        <input type='image'>imageSrc
        </input>
        <output>
                <type>
                image
                </type>
                <format>
                png
                </format>
                <dimension>
                320x240
                </dimension>
                <quality>
                high
                </quality>
                <alpha>
                true
                </alpha>
        </output>
</Template>
```

The XML above will use an imageAdapt method to adapt the resource and it takes as input an image. As the output it will provide an image with the following characteristics: a png format with 320x240 of size, with high quality and alpha channel.

There is a series of templates already built as components for the architecture, like images, audio and video. Sometimes, different devices implement different multimedia capabilities; and therefore, they may require different media formats. The CAA provides the adaptation of format types and image properties.

The image adaptation is normally done by the ImageMagick library, a free open source library, used by the CAA to change the image properties, like dimension, format and quality. For the adaptation of audio or video, the CAA uses the FFMEPG library, another open source library, to convert video/audio files, and at the same time, to change properties such as frame rate, format type, quality and dimension. The configuration for each common device (mobile device) is registered in the application.

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

These classes were implemented as a factory pattern, illustrated in Figure 3, where it takes an XML as input, and gives the object with the characteristics of the generated new resource as output. This way the architecture keeps the code clean by creating objects without specifying the exact resource class of object that will be created.

In this UML, there are the main classes of content adaptation. The image class is responsible for converting the images and it has the following children: the sprite generator, which creates a sprite though combining a series of images into one single image for 2D animations; the Image2square, which is created by inserting blank pixels on the image until it has the dimension of power of 2, required for the use on some mobile openGL devices; the image aspectRatio, which converts the images, with different aspect ratio, by resizing it and inserting or removing transparent pixels of the image; the Crop Image, which crops the image to a different size; and the imageResize, which resizes the image according to a scale factor. There is also a video adapter, which converts the video dimensions, bit-rates, codecs and encapsulations; and the text adapter for the conversion of text, by removing and changing characters, from different char-sets. The HTML adapter transforms HTML to different platforms, by removing features or even changing the code. The sound adapter converts the sound from different bit-rates and different formats.

B. Content Version Control

The content version control is responsible for keeping control of each version of the adapted content. Every time the CAA gathers content from the CMS, or when the templates are changed, it generates a new version in its repository. This component was built using a balk pattern, which 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 updates and generates a new version of the content, when all the content gathering and adaptation has been completed.

The content can change frequently, and each user can have different versions. When users update their content, they use their own version numbers that were previously saved in the cache. Therefore, their devices only download the needed

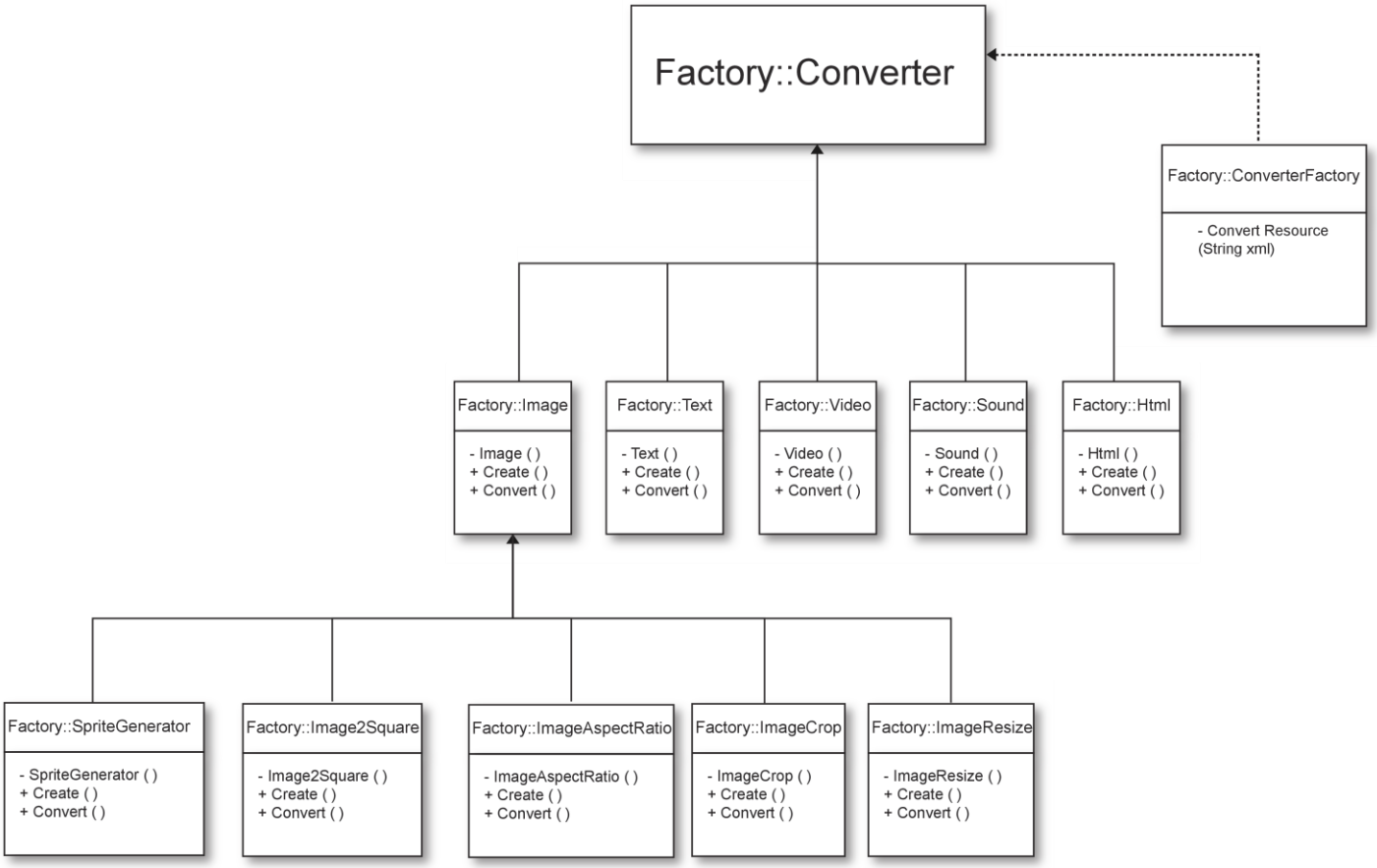

Fig. 3. UML of the Content Adapter

content for the latest version, keeping the communication between the server and device to a minimum.

All the content generated by the content adaptation, is normally saved in the cloud, using Amazon Simple Storage Service, or similar services for cloud storage.

Additionally all the content sent to the user is compressed in a zlib format by using the zlib library. Compression helps to reduce traffic, which is good for mobile devices that sometimes suffer from slow network and expensive data plans.

C. Identity Server

The identity server is used as: 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 cryptography; 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 use of restricted content among the clients, devices, and 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 collects statistical information of each device, each user, content versions, device capabilities and how the user uses the content. This server can provide a real-time statistical report, grouping by user, device or time.

V. THE THIN CLIENT

The thin client is responsible for: gathering data from the CAA server; providing the server with the device characteristics; implementing a cache system for the gathered data for offline use; and gathering of user data for statistical reports. An overview of this client can be seen in Figure 4.

The client was developed as a framework, in order to be reused in other projects. It was mainly developed in object-C and Java, in order to work with the Apple IOS and Google Android platforms. The clients are also being developed for other platforms, such as Windows mobile, Blackberry and J2ME. This client is capable of displaying different resources like HTMLs, images, audios and videos. It can also process and show a specific data structure for maps, which are used in the validation. In addition, components were also implemented for the localization system, cache management, statistical data

Fig. 4. Overview of the Thin Client

collection and social network (Facebook, Twitter, and Google+) integration.

Most of the customization in the application is specified via XMLs, such as the XML below, which configures how the client should download the data and where it should save it.

```
<ApplcationPush>
   <server timer='2h'>
   http://201.200.1.210:8000/ContentArch.js
   </server>
   <parameters>

         <param type='datetime'>
         date
         </param>

         <param type='model'>
         iPhone
         </param>

         <param type='gameengine'>
         cocos2d
         </param>

         <param type='cryptografy'>
         true
         </param>
      </parameters>

   <cache>
         <maxSize>
         100MB
         </maxSize>
         <diretory>
         APPDIR/ContentArch/Cache/
         </diretory>

         <param type='cryptografy'>
         false
         </param>

   </cache>

   <statistics>
       <timeBetweenUpdates>
       true
       </timeBetweenUpdates>

       <resourcesViewed>
       true
       </resourcesViewed>

       <dateTimeAppUse>
       true
       </dateTimeAppUse>

   </statistics>

</ApplcationPush>
```

The XML above will configure the client services for data, cache and statistical data collection. The XML sets the application to call the server every two hours (only if the application is running and has a network connection) on http://201.200.1.210:8000/ContentArch.js web service that requires a datetime as input, the device model (iPhone), the game engine used (cocos2d) and a variable saying that it will

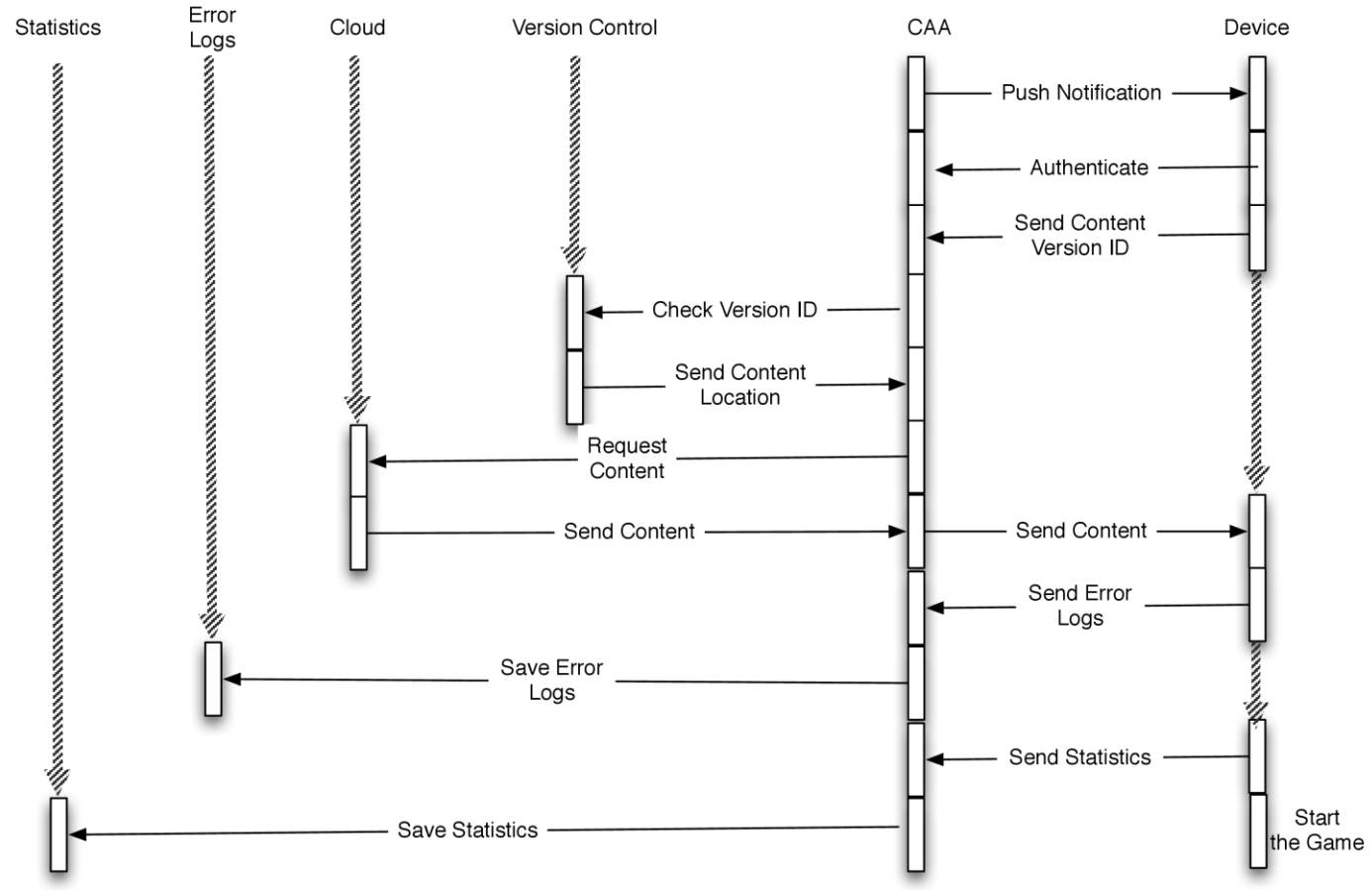

Fig. 5. Process Flow of the CAA Client

not require cryptography. This XML will also configure the cache that will have a maximum of 100 MB, no cryptography, and will use the directory APPDIR/ContentArch/Cache/ to save the content. The statistical data collection is configured by having the XML monitoring the time between updates required by users, the resources users view and the date/time of user interactions with the application.

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

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

The client also 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, available input types, camera, and all the characteristics of the device available for the client.

The client uses extensively the cache for its content. It uses this cache for providing the end-user with offline content and also minimizes the data transfer between updates. This cache can also be cryptographic or compressed if needed by the application. The cache consistency can be a problem in some devices when the network data exchange fails [28]. This client only updates its version after gathering all the data and verifying its hash code. This client also uses a service to save the session data, and to collect statistics if needed by the server.

To update the data in the client, the following workflow is used: first the device, if registered gets a push notification of new content availability; next when the user opens the application, the device authenticates it and gathers from the server the needed resources for its update; if the user has no available connection, the application starts without this update process, by accessing the cache resources. Figure 5 illustrates this process.

Fig. 6. The CAA in action adapting HTML content and images for the web site, tablet, and smart phone.

VI. VALIDATION

In order to validate the architecture, the CAA was used in a commercial project for shopping centers. The application is a hub containing mini-applications of different malls, with different CMSs. These mini-applications have the responsibility of being an information channel of the mall, where users can obtain information such as cinemas, news, sales, and also a mapping system for locations of stores and car parks.

One of the main problems of these kinds of applications, without the use of the CAA, is that the content may come from different services and different formats. With that, the CAA uses different configurations to gather content from different providers, but to show the same result on the client. In this application, there were two different CMSs, a Joomla CMS and a proprietary web service. These were existing CMSs used for the mall web sites. The same content from these sites was used by the application as pull service, where the content was gathered five times a day. The use of the CAA architecture saves some rework when customizing difference platforms for releasing different contents.

The thin client used for this application was adapted in order to work on iPhones, iPads, Android phones and Android

tablets. This client shows the following content types: images, HTMLs, videos, maps, and is integrated with the social media sites Facebook and Twitter.

This application also collects statistical data that is sent from time to time to the server for usage analysis. This information can be used later for purposes of promotions, sales and other events of the malls.

Figure 6 shows an example of the same news content being displayed on different devices - the original content on the web site, the adapted content on a tablet, and the adapted content on a smart phone. From these images, it can be seen that different layouts and image sizes can be adapted by using the CAA architecture.

This work has not been compared to other architectures due to different paradigms. One possible comparison is the CAS reported in [26], [27], but the authors of this work did not have access for the project. The CAS only provides content adaptation of texts, audios, images and videos, but not HTMLs. Also, the CAS does not provide caching, data compression and version control.

VII. Conclusions

Devices nowadays have many different characteristics and constraints, requiring specific content. With that, these devices need the publishing of specific content of the CMS or the adaptation of already published content. This paper has presented a new adapter for CMS content adaptation that provides a layer between the CMS and the devices.

Moreover, devices can have connection constraints, like availability, and also this data can be very expensive. This presented architecture also provides a version control system and a cache system in order to provide offline data and also keep the data communication to a minimum.

Future works consist of adapting and evaluating the platform to be used as a content provider for a multi-display e-learning platform [29] and the adaptation of the content adapter to a cloud GPU computing architecture using CUDA [30].

References

[1] M. Joselli and E. Clua, “grmobile: A framework for touch and accelerometer gesture recognition for mobile games,” in Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment (SBGAMES ’09), Washington, DC, USA, IEEE Computer Society, 2009, pp. 141–150. [Online]. Available: http://dx.doi.org/10.1109/SBGAMES.2009.24

[2] J. R. da Silva Junior, M. Joselli, E. Clua, M. Pelegrino, and E. Mendonça, “An architecture for new ways of game user interaction using mobile devices,” in SBGames, SBC, 2011.

[3] M. Joselli, E. B. Passos, J. R. S. Junior, M. Zamith, E. Clua, and E. Soluri, “A flocking boids simulation and optimization structure for mobile multicore architectures,” in Proceedings of SBGames 2012, pp. 83–92.

[4] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness,” in Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, (HUC ’99), London, UK, UK: Springer-Verlag, 1999, pp. 304–307.

[5] F. Agboma and A. Liotta, “Quality of experience management in mobile content delivery systems,” Telecommunication Systems, vol. 49, no. 1, pp. 85–98, 2010.

[6] M. Joselli, J. R. S. Junior, M. Zamith, E. Clua, and E. Soluri, “A content adaptation architecture for games,” in Proceedings of SBGames 2012, pp. 17–25.

[7] M. Joselli, E. Soluri, J. R. S. Junior, M. Zamith, and E. Clua, “mCMS: A content management system adapter architecture for mobile devices,” in XI Workshop de Ferramentas e Aplicações (WFA) – Webmidia 2012, SBC, 2012.

[8] M. Joselli, E. Soluri, J. R. S. Junior, M. Zamith, E. Clua, and K. Micheli, “A content adapter architecture for CMSs,” in Proceedings of the International Conference on Information Technology and Applications (ICITA), IEEE, 2013, pp. 80–85.

[9] M. Zamith, M. Joselli, E. W. G. Clua, A. Montenegro, R. C. P. Leal-Toledo, L. Valente, and B. Feijo, “A distributed architecture for mobile digital games based on cloud computing,” in Proceedings of the 2011 Brazilian Symposium on Games and Digital Entertainment (SBGAMES), IEEE, 2011, pp. 79–88.

[10] M. Joselli, J. Ricardo da Silva, M. Zamith, E. Clua, M. Pelegrino, E. Mendonca, and E. Soluri, “An architecture for game interaction using mobile,” in Proceedings of the IEEE Games Innovation Conference (IGIC), Rochester, New York, USA, 7–9 September, 2012, pp. 46–50.

[11] T. Nguyen and C. Girimohan, “Global content management systems,” Multilingual Computing and Technology, vol. 45, no. Supplement 45, pp. 8–13, 2002.

[12] J. Souer, D.-J. Joor, R. Helms, and S. Brinkkemper, “Identifying commonalities in web content management system engineering,” International Journal of Web Information Systems (IJWIS), vol. 7, no. 3, pp. 292–308, 2011.

[13] J. Souer, P. Honders, J. Versendaal, and S. Brinkkemper, “A framework for web content management system operations and maintenance,” Journal of Digital Information Management (JDIM), vol. 6, no.4, pp. 342–347, 2008.

[14] H. Tummala and J. Jones, “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), Cologne, Germany, 2–Sept, 2005, pp. 14–22.

[15] J. Nurminen, J. Wikman, H. Kokkinen, P. Muilu, and M. Grnholm, “Drupal content management system on mobile phone,” in Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC 2008), Las Vegas, Nevada, USA, 10–12 January, 2008, pp. 1228–1229.

[16] M. Internation, “Magnolia CMS,” Jun. 2012. [Online]. Available: http://www.magnolia- cms.com/

[17] C. C. M. Services, “Cellular CMS,” Jun. 2012. [Online]. Available: http://www.cmslive.com/

[18] MobiManage, “Mobile cms,” Jun. 2012. [Online]. Available: http://www.mobimanage.com/mobile-cms.cfm

[19] mFabrik, “mfabrik web and mobile cms,” Jun. 2012. [Online]. Available: http://webandmobile.mfabrik.com/

[20] A. R. Manashty, M. R. A. Raji, Z. F. Jahromi, and A. Rajabzadeh, “Armrayan multimedia mobile cms: a simplified approach towards content-oriented mobile application designing,” in Proceedings of the International Conference on Wireless Communication and Mobile Computing (ICWCMC 2010), World Academy of Science, Engineering and Technology, vol. 62, pp. 62–67, 2010. [Online]. Available: http://arxiv.org/abs/1009.5347

[21] A. Hildebrand, T. C. Schmidt, and M. Engelhardt, “Mobile elearning content on demand,” Information Sciences, vol. 5, no. 2, pp. 94–103, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.123.9176

[22] F. Trinta, D. Pedrosa, C. Ferraz, and G. Ramalho, “Evaluating a middleware for crossmedia games,” in Proceedings of the 5th international workshop on Middleware for pervasive and ad- hoc computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference, (MPAC ’07), New York, NY, USA, ACM, 2007, pp. 43–48. [Online]. Available: http://doi.acm.org/10. 1145/1376866.1376874

[23] F. Trinta, D. Pedrosa, C. Ferraz, and G. Ramalho, “Evaluating a middleware for crossmedia games,” Computers in Entertainment (CIE), vol. 6, no. 3, pp. 40:1–40:19, Nov. 2008. [Online]. Available: http://doi.acm.org/10.1145/1394021.1394033

[24] J. R. B. Diniz, C. A. G. Ferraz, F. A. M. Trinta, H. N. Melo, and L. M. Santos, “Avaliao de um servico de gerenciamento de sessao para ambientes de medicina ubqua,” in Proceedings of the 14th Brazilian Symposium on Multimedia and the Web, (WebMedia ’08), New York, NY, ACM, 2008, pp. 4–11.

[25] F. C. Delicato, I. L. A. Santos, P. F. Pires, A. L. S. Oliveira, T. Batista, and L. P`ırmez, “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), New York, NY, USA, ACM, 2009, pp. 456–460. [Online]. Available: http://doi.acm.org/10.1145/1529282.1529381

[26] D. Carvalho and F. Trinta, “Content adaptation for multiplatform applications,” in Proceedings of the XV Brazilian Symposium on Multimedia and the Web, (WebMedia ’09), New York, NY, USA, ACM, 2009, pp. 41:1–41:4. [Online]. Available: http://doi.acm.org/10.1145/1858477.1858518

[27] F. T. Diego Carvalho, “Servico de adaptacao de conteudo para aplicacoes multiplataforma,” in Anais do 3o Simpsio Brasileiro de Componentes, Arquiteturas e Reutilizao de Software (SBCARS 2009), Natal, RN, Set 2009.

[28] W. Li, E. Chan, D. Chen, and S. Lu, “Maintaining probabilistic consistency for frequently offline devices in mobile ad hoc networks,” in Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems, (ICDCS ’09), Washington, DC, USA, IEEE Computer Society, 2009, pp. 215–222. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2009.23

[29] J. Delgado, M. Joselli, S. Stanzani, S. M. Sadjadi, E. Clua, and H. Alvarez, “A learning and collaboration platform based on sage,” in Proceedings of the 14th Western Canadian Conference on Computing Education, ACM, 2009, pp. 70–76.

[30] M. S. Doost, S. M. Sadjadi, J. R. S. Jr, M. Zamith, M. Joselli, and E. Clua, “Architecture of request distributor for gpu clusters,” in Proceedings of the Third Workshop on Applications for Multi-Core Architectures (WAMCA), IEEE, 2012, pp. 13–18.

