Analysis and Design of a Subtitling System for Ambient Intelligence Environments

Aitor Rodriguez-Alsina, Guillermo Talavera, Pilar Orero, Jordi Carrabina
Qc-2090D. Escola d'Enginyeries. Campus UAB
08193 Bellaterra, Spain
{Aitor.Rodriguez, Guillermo.Talavera, Pilar.Orero, Jordi.Carrabina}@uab.cat

Abstract — **The development of ubiquitous applications for ambient intelligence environments needs to also take into account some usability and accessibility issues in order to ensure a proper user experience and to overcome the existing content access barriers. A proper access to video subtitles, for instance, is not always available due to the technical limitations of traditional video packaging, transmission and presentation. New Web standards enable more featured applications with better multi-platform definition, so they are suitable for building ubiquitous applications for ambient intelligence environments. This work presents a video subtitling system that enables the customization and adaptation of subtitles. The benefits of Web applications compared with device-specific native applications for building the solution as well as its current platform support are analyzed. Finally, three different application use cases are presented.**

Keywords - Multimedia Synchronization; Subtitles; Ambient Intelligence; Accessibility; TV; HTML5; SVG; SMIL

I. INTRODUCTION

In Ambient Intelligence (AmI) environments, as with any multimedia context, one of the key elements to ensuring proper access to video content is the service of subtitling and captioning. Subtitles are a multifunction service since they allow the breaking down of many barriers such as those related to language and audio content for groups such as disabled people, the elderly, immigrants, foreign language students and children. They can also be a powerful teaching tool. Subtitle services enrich multimedia content, but the mechanisms for their presentation (as well as other accessibility data like audio-descriptions and sign language) do not yet conform to the accessibility requirements of specific groups of users. In ubiquitous home environments, where users share and interact with a variety of smart devices (e.g. TV, mobile, tablet, PC, laptop, console), in order to get video content the underlying technological differences between connected devices can be overcome through the use of Web technologies. This enables the implementation of more ubiquitous systems by the widespread support of web applications over the variety of proprietary runtime environments that are provided by device manufacturers. However, most of the web video content consumed does not include selectable subtitles, and non-standard embedded video players do not allow independent management of subtitles with the aim of enabling adaptation to the user (or application) accessibility requirements. Most subtitles distributed on the Internet are described in text files that follow the SubRip (.SRT) format, considered "perhaps the most basic of all subtitle formats"[4]. Figure 1. shows an example of an .SRT file. Each subtitle entry consists of the subtitle number, the time the subtitle should appear on the screen, the subtitle itself, and a blank line to indicate the end.

```
1
00:00:20,000 --> 00:00:24,400
Altocumulus clouds occur between six thousand

2
00:00:24,600 --> 00:00:27,800
and twenty thousand feet above ground level.
```

Figure 1. Example of subtitles defined using the SubRip (.SRT) format.

While the standardization of subtitles as part of HTML5 video and audio components is still under discussion, novel approaches for video subtitling in web-based environments are required for the exploration of video accessibility in ambient intelligence environments. This paper presents an alternative design strategy for implementing customizable subtitle systems in a variety of platforms and devices. It takes advantage of the native support to Scalable Vector Graphics (SVG) and the Synchronized Multimedia Integration Language (SMIL) [5] within HTML5 documents for presenting synchronized subtitles on any web-capable device with the appropriate browser. The current platform support for the proposed solution is also analyzed for both desktop environments and for mobile and TV. Furthermore, in order to demonstrate the viability and support for the proposed solution, three application use cases for video subtitling have been constructed: (1) a web widget for desktop applications, which extends the functionality of an HTML5 video component to enable user modifications to the language and format of subtitles; (2) a TV application that improves subtitle readability when the user interface combines video with other interactive content; and (3) a mobile application that extends the screen of the media center in a home environment (as an example of AmI environment) to receive the TV subtitles on a mobile device.

The rest of the paper is structured as follows: Section 2 analyzes the pros and cons of deploying Web applications compared with native applications. Section 3 presents a new

approach to synchronizing web-based video subtitles using SVG and SMIL. Section 4 analyzes the current platform support for the proposed approach and Section 5 outlines the paper's conclusions and takes a look at the future.

II. WEB APPLICATIONS VS. NATIVE APPLICATIONS

With the explosion in the popularity of smart devices and application markets on the one hand, and on the other the enhancement of Web technologies led by HTML5, it seems natural to analyze the pros and cons of building native applications versus the development and maintenance of Web sites. This section compares the two, and justifies our choice for building the presented subtitling system.

A. Platform support

Native applications need to be specifically built for every target platform. Thus, application providers must take into account the number and types of target platforms in order to address the application to the individual needs of each of them. It is not only a matter of the application environments (e.g. desktop, mobile, interactive TV) or brands (e.g. iPhone, Android, BlackBerry), but also a question of the platform versions (e.g. Android experiences a cloud of versions with different features depending on the device manufacturer). When the application needs to be supported by a new platform, it must be written again using its native programming language and its native look and feel functions.

By contrast, Web applications are based on Web standards and are deployed on a Web browser, which acts as a bind between an application written using shared standards and the specific runtime environment of each target platform. This enables Web applications to be deployed on any browser-capable platform regardless of the application environment of the specific device. However, differences still exist when using the same platform with different browsers due to different feature implementation in the browser engine (e.g. individual browsers support a different HTML5 set of features, and this may also depend on the browser version). Moreover, the universal support of Web applications can be limited when using special device features such as the accelerometer for mobile devices and the tuner for TV platforms. Although HTML5 provides support for accessing some of these features (e.g. geolocation) through specific Application Programming Interfaces (APIs), it is quite clear that it will not support all the concrete features for all devices. When a Web application needs to be supported by a new platform, nothing (theoretically, based on the principle "build once, run anywhere") has to be done more than ensure that content fits properly within the new environment.

B. Installation and maintenance

The installation of native applications can depend on the application environment and device brand, but the current trend for the discovery and delivery of native applications on various platform environments such as smart mobiles and smart TV platforms, is in their acquisition from 'app stores'. Applications are published to an app store (or market), reviewed by the store managers, approved for delivery and installed on the device when a user requests them. Updates also require the same review and approval, and they must be done specifically for each different platform.

Web applications on non-desktop devices are accessed in a variety of ways depending on the target platform. The so called "smart devices" (i.e. smart phones and smart TV devices), as well as mobile phones, allow free searching and access to any site available on the Internet. In contrast, some other interactive TV experiences only allow the access to the Web applications included in standard markets. However, hybrid solutions are also possible, such as the placement of Web applications (for both mobile and TV) inside native applications which are then sold through the common application stores. Web applications do not need to be reviewed and approved by the managers of any application market, and they are updated simply by modifying the Web application on the server side. All the platforms then receive the updated version without modifications on the client system.

C. User experience

Native applications are usually faster because they reside within the device platform and can take advantage of the system UI functions. These functions also enable a set of visual effects that are shared by the rest of the system applications. This results in a better user experience by facilitating the application usability on small screens.

With the arrival of HTML5, Web applications can be more integrated within the system (especially on mobile devices), their structure can be semantic and more functional, and JavaScript functions can effectively meet the requirements of most applications. While some device functions such as accelerometer, video and audio are now supported in HTML5, other device specific features are still under discussion. Two examples of this are the access to phone services (e.g., contacts, agenda, etc) and the management of device-associated interfaces (e.g. webcams, microphones, USB devices) through the <device> tag. This can be sufficient for most applications, but the rest still need to be natively implemented, especially for non-mobile devices. HTML5 is still evolving and new features will be added but at the moment it is not supported by the vast majority of non-smart devices.

D. Development

The development of ubiquitous services as native applications requires specific knowledge for each different platform due to the varying underlying programming languages, SDKs, style guidelines and development infrastructures. This implies that there are usually higher development costs because of the need for a higher number of developers with different skills, and the specialization of developmental tools and frameworks.

For building Web applications, developers mainly need to know HTML, CSS and JavaScript, thus avoiding the need

to learn new programming languages to code native applications. This allows the reusability of the existing development teams and development infrastructure for any new target platform. However, any platform may require specific content adaptations to the supplied Web sites in order to achieve the best user experience.

E. Our conclusion

While platform support and user experience may still be better for native applications, Web applications can offer a more attractive solution for application providers by facilitating (and reducing the costs of) the distribution, updating and development of ubiquitous applications. Moreover, the platform support and user experience is being quickly improved with HTML5, and users will become increasingly used to Web applications on non-desktop devices as new applications become available. Finally, we strongly believe that the ubiquity achieved by Web applications through the principle of "build once, run anywhere" is the key to implementing ambient intelligence user applications and supports its choice for building the subtitling system proposed in this paper.

III. WEB-BASED, SCALABLE AND CUSTOMIZABLE SUBTITLES FOR AMBIENT INTELLIGENCE

Taking advantage of the underlying Web ubiquity, we have designed and successfully implemented a video subtitling system that enables subtitle format customization and its adaptation to needs of the user/application. It is based on HTML5 and the new features that enable SVG and SMIL. Since the standardization of the HTML5 video component and its acceptance by the major Web browsers, there has been some implementation of the synchronization and presentation of subtitles attached to a <video> tag.[6] Most of these changes make intensive use of JavaScript through third party libraries which can imply a penalty on the resulting performance and portability of the application. A different approach can be achieved by taking advantage of SMIL time properties for the multimedia content synchronization. Instead of managing the content timeline programmatically through JavaScript, a web programmer can make use of the event management and timeline running on the SMIL engine of a Web browser. This can dramatically simplify the development of a time-based application like subtitling by delegating the time management (i.e. the presentation time of each subtitle or caption) to the browser, instead of purely managed by JavaScript.

A. Related Work

Jan Gerber was one of the first to develop a subtitle synchronization system for a HTML5 <video> tag, which resulted in the JavaScript library *jquery.srt.js* [7]. This library parses the HTML document looking for subtitle containers which have been previously defined with the "srt" class, and attempts to load the related subtitles. Subtitles can be written directly in .SRT format into the subtitle container or accessed remotely through the address defined as a custom parameter into the HTML container. This implementation works correctly but relies on custom HTML attributes.

Taking Gerbers's approach, Michael Dale proposed an evolution that demonstrates the benefits of including some HTML5 elements as child nodes of the HTML5 <video> container to associate a video component with time-aligned text [8]. This eliminates the custom defined attributes of the previous approach but still relies on the same JavaScript library as subtitle synchronization for time-related issues.

Philippe Le Hegaret adapted Gerber's JavaScript library to demonstrate the viability of using Gerber's proposal with a different format for the presentation of time-based text in his HTML5 DFXP Player prototype [9]. He used the Distribution Format eXchange Profile (DFXP) of the Timed Text Authoring Format (TT AF) [10], which was created by the W3C for the conversion, exchanging and distributing timed text information amongst legacy distribution content formats in use for subtitling and captioning.

Alex Danilo proposed a completely different approach to multimedia synchronization in web-based environments [11]. This work presents an example of multimedia and graphic synchronization using SMIL to present SVG subtitles that are displayed on top of an SVG video. SMIL facilities allow the direct definition of time marks on subtitle text elements and specify the begin time and duration for each subtitle. No JavaScript code is needed because the SMIL engine is responsible for maintaining the synchronization between video and subtitles. This is only a basic illustrative example, but it opens up a new and interesting pathway for implementing video subtitles in web-based environments.

B. Proposed approach

Inspired by Danilo's approach for synchronizing SVG subtitles through SMIL, this paper proposes a comprehensive solution for video subtitling based on new Web technologies around HTML5. This solution enables the acquisition and parsing of remote subtitle files and allows dynamic modifications to the selected language and text format during video playback. It is robust against changes on the video timeline, i.e. the subtitles remain synchronized after functions such as pause, fast-forward, direct jump, etc. are used. Subtitle text components are written in SVG in order to improve scalability and allow SMIl animations on them; JavaScript is used for parsing functions, client-server communications, the dynamic creation of subtitle text elements, and the event management of the <video> container. SMIL manages the time-related functions and decides the subtitle text visibility according to the 'begin' and 'end' attributes defined by the SMIL <animation> element. These time marks correspond to the times defined in the related .SRT subtitle file. Figure 2. shows a subtitle definition using an SMIL animation on

HTML5 documents. Text opacity is initially set to 0 to hide the text, and the <animation> element specifies when this opacity attribute must be changed to set the text to visible. The time format is the same as in the .SRT subtitle files (i.e. hours:minutes:seconds,milliseconds), so no calculations or JavaScript are required in the translation. The SMIL engine of the Web browser manages the rest of the process in order to keep subtitles synchronized with the video content.

One interesting novelty provided by HTML5 is the support for embedded SVG and SMIL code in the document as for any other HTML tag. These elements are also included as part of the Document Object Model (DOM), which means that they can be accessed and modified through JavaScript and CSS in the same way as any other HTML tag. By including SVG subtitles as part of the Web document, any web application can present customized subtitles natively, almost as in those platforms with the suitable web browser. However, this basic implementation is insufficient to support user-driven changes in the video timeline (e.g. pause, seek in time) because the timeline managed by the SMIL engine must also be notified about these changes. This is important for keeping the subtitles synchronized with the video content. When the user modifies the video playback timeline, any web application can automatically apply the suitable modifications to the SMIL timeline through the JavaScript API provided by HTML5 for the SMIL time-management. For a basic playback management, three basic video properties are required: *onplay*, *onpause* and *onseeking*. Figure 3. shows an eexample of the JavaScript functions required to control the SMIL timeline. When the user pauses, plays or seeks on the video, the HTML5 <video> container fires these JavaScript functions. The SVGRoot element of the example stands for the root element of the entire SVG document (i.e. <svg>) and the variable *video* references the <video> container itself.

```
<text opacity=0>
  <animation attibuteName='opacity'
    begin='00:00:20,000'
    end='00:00:24,400' values='1'/>
  At the left we can see...
</text>
```

Figure 2. Code example of an SVG text element with an SMIL animation for presenting a subtitle.

```
function onPlay() {
 SVGRoot.unpauseAnimations();}
function onPause() {
 SVGRoot.pauseAnimations();}
function onSeek() {
 SVGRoot.setCurrentTime(video.currentTime);}
```

Figure 3. Code example of a SVG text element with a SMIL animation for presenting a subtitle.

IV. ANALYSIS OF THE PLATFORM SUPPORT

The platform support for the proposed solution depends on the availability of a web browser and its support for some HTML5 features such as the video and audio management, SVG and SMIL animations. Although a variety of browser plugins and SVG players exist, we focus on the native support for SVG by the main web browsers in order to ensure maximum ubiquity. In this section we analyze the support for these features in three different application environments: desktop, mobile and TV.

A. *Desktop environments*

Nowadays, all major modern desktop web browsers provide support for HTML5 and SVG to the level required by the proposed solution, with the exception of Internet Explorer (IE) which can present some problems with SMIL animations. Google's announcement on August 2010 that it had begun to index SVG content (standalone files as well as embedded in HTML5 documents) and cases of successful applications such as Wikipedia (which presents SVG images) all push for a wider acceptance of the format. However, not all the features are completely supported by the current versions of some browsers and slight adaptations should be applied when implementing the proposed solution for video subtitling. Table I summarizes the SVG (and SMIL) browser support for the main features that use the proposed subtitling solution. As shown in the table, the inclusion of inline SVG tags within HTML5 documents is not completely supported, but this can be easily solved by externally referencing the SVG document. Moreover, IE does not support native SMIL animations so a third-party plugin is required.

TABLE I. HTML5 AND SVG SUPPORT IN DESKTOP WEB BROWSERS

	Main desktop web browsers[a]				
Required feature	***Fi***	***Ch***	***Op***	***IE***	***Sa***
HTML5 documents	yes	yes	yes	yes	yes
HTML5 Video	yes	yes	yes	yes	yes
SVG basic support	yes	yes	yes	yes	yes
Inline SVG	yes	yes	no	yes	no
SMILanimation	yes	yes	yes	no	yes

a. Firefox 4.0 (Fi), Chorme 12 (Ch), Opera 11.1 (Op), Internet Explorer 9.0 (IE) and Safari 5.0 (Sa)

B. *Mobile environments*

New mobile web browsers are supporting more and more advanced features but they are still some way back when compared to the development of desktop browsers. In particular, the only mobile web browser in the study that provides full support for the proposed solution to video subtitling is Mozilla Firefox 4.0 Mobile. The lack of SVG support by the Android default web browser (based on the WebKit browser engine) and the lack of support for SMIL animations by the rest (with the exception of Firefox) limit

the deployment of the proposed solution to only one of the major web browsers. One reason for this delay in providing support on mobile web browsers is evident in the lack of variety of applications that request these features on mobiles devices. At the moment, most mobile applications are deployed natively. However, in the same way that users are currently totally familiar with desktop web applications, it is only a matter of time (and more available applications) before users also become used to mobile web applications.

TABLE II. HTML5 AND SVG SUPPORT IN MOBILE WEB BROWSERS

	Main mobile web browsers[a]			
Required feature	***iOs***	***An***	***Fi***	***Op***
HTML5 documents	yes	yes	yes	yes
HTML5 Video	yes	yes	yes	yes
SVG basic support	yes	no	yes	yes
Inline SVG	no	no	yes	no
SMILanimation	no	no	yes	no

a. iOS Safari 4.2 (iOs), Android Browser 2.3 (An), Firefox 4.0 (Fi), Opera Mobile 11 (Op)

C. *TV environments*

The interactive TV situation is even more scattered than for mobiles because the technological details of what constitutes a 'smart TV' have not yet been clearly defined. Currently, three main approaches coexist as smart TV platforms. (1) Device and Internet operators have presented their own solutions. In the case of Apple and Google (which have respectively created AppleTV and GoogleTV) the first provides an integrated Set-top Box (STB), whilst the second provides a framework deployable on prepared STBs and TVs. Both support the main features of HTML5, but they do not yet support SVG animations. (2) TV manufacturers build Internet-capable devices that combine the TV video flow with web applications and widgets. Their browser implementations provide little support for HTML5 features as they have very limited resources (even less than mobile devices). (3) Some Internet video service providers (e.g. Netflix, Boxee) are interested in collaborating with TV and STB manufacturers to deploy their services on a variety of terminals. In any case, the support for HTML5 and SGV animations on smart TV devices depends on the available device resources to deploy a web browser that supports the required features. Fortunately, the Open IPTV Forum recommends the use of SVG (SVG Tiny 1.2) for declarative environment applications on the client side [12] and some initiatives for the standardization of hybrid digital TV platforms such as the Hybrid Broadcast Broadband TV (HbbTV) follow this model. Even so, and according to Ericsson, the SVG specification could add some features in order to make SVG more suitable for TV (e.g. support for TV-related remote control keys, ability to seek in time and speed for fast forwarding/rewinding, etc).

V. APPLICATION USE CASES

The proposed approach for advanced video subtitling has been validated through its implementation on three different use cases in order to also demonstrate its usefulness and ubiquity. The first one extends the functionality of any HTML5 <video> container onto desktop browsers; the second offers a solution for video subtitling in web-based interactive TV environments; and the third makes use of mobile devices to present personalized subtitles on a TV's second screen.

A. *The SVG Subtitle Access Bar (SVG-SAB)*

SVG-SAB is a web widget that extends the functionality of any HTML5 <video> component to enable advanced subtitle functions. It facilitates the access to video content on the Web by the adaptation of subtitles to the needs and preferences of the user. This widget applies the proposed solution to synchronizing subtitles and video content on the client side using SMIL animations. This keeps contents synchronized even if the user dynamically changes the language, the subtitle text format and the position of the text on the screen. Figure 4. shows a snapshot of the SVG-BAR attached to a video component playing a subtitled video. Subtitle languages can be selected and switched dynamically from remote .SRT files. Each subtitle is added as a DOM node with its corresponding presentation time, and SMIL manages these times in order to show each piece of text on the screen. Developers need not take into account either the current playback time or the subtitle sequence order to maintain the content synchronization. When a language switch is applied, the widget removes the previous subtitles and adds the new ones without synchronization delay.

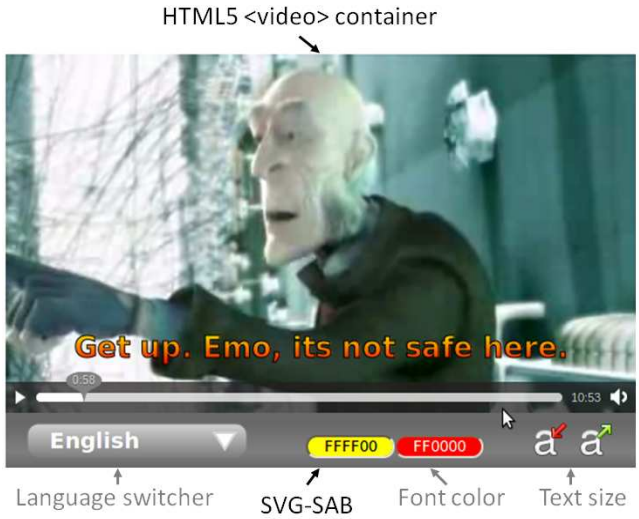


Figure 4. Screenshot of the embedded SVG-SAB presented on a desktop browser (Firefox 4).

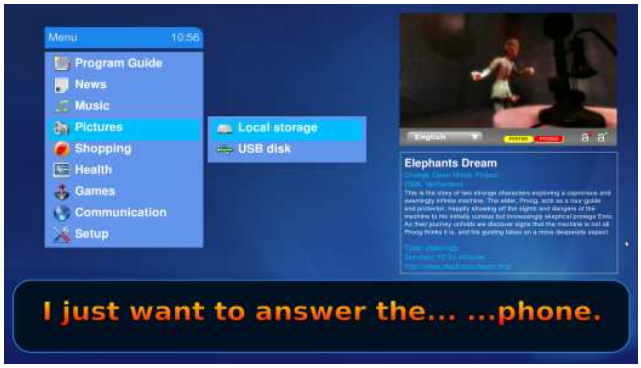
The SVG-SAB widget includes a set of tools for changing the text color and size in order to enable the subtitle's personalization and its adaptation to the video or application requirements, e.g. text color can be adapted to

achieve a better contrast. Adaptation of the subtitle can also be necessary to overcome user difficulties in accessing web content. The use and functionality of these subtitles is multiple: for language teaching and language therapy [13], for people with visual impairments or color blindness, for the elderly and people who speak other languages, or for those with literacy or comprehension difficulties [14][15].

B. *Positioning Subtitles on Interactive TV*

On TV, subtitle customization enables better access to video content, especially within the environment of interactive TV. Subtitles can be placed and sized properly to ensure maximum readability, while the video content remains playing in an interactive user interface. Subtitle position can also be modified in order to place it wherever is most comfortable for the user [16]. This feature has been applied, as in Figure 4, to construct an interactive application for TV that presents video subtitles that are always readable even if the video window is reduced in order to display the rest of the interactive contents. Due to the time management of the SMIL browser engine, content is always synchronized even if it is visually separated on the screen.

(a)



(b)

Figure 5. Two screenshots of the positioning of subtitles within an interactive TV environment. In (a), the user is watching a subtitled video content on the TV. He then accesses the interactive element and the video playback is reduced to a corner of the screen as shown in (b). Here, subtitles are presented in normal size and positioned in the same place as before.

C. *Presenting subtitles on a second screen*

Within the home environment, a mobile device can be used as a second screen [17] of the TV platform onto which to extend its content and management options. Mobile devices can become an advanced remote controller, as well as the support for presenting the same contents as on the TV set or some complementary contents. The access to video content can benefit from this feature as it allows the personalized access to video subtitles by each user through their own smart mobile phone. This can facilitate the access to content for people with special requirements (e.g. the elderly, children and disabled people) and thus improves the family balance. We have built a web-based mobile application to present video subtitles that are synchronized with TV contents, extending the TV screen to a mobile device. In a home environment, this mobile application connects to the media center, which serves the TV interactive application with the synchronized video subtitles (shown in the previous section), through the WLAN network. The user receives the subtitles of the current video content on his mobile device and can change the language, size and color of subtitle texts independently from the TV stream or any other connected device. This mobile web application applies the same SMIL subtitle synchronization principle proposed in this paper. The main difference with regards to implementation from the previously highlighted use cases is the synchronization requirements between the media center and the user device acting as a second screen. In our implementation, the user device application receives the current playback time from the media center, which acts as server. Media Center and device SMIL timelines are then synchronized and the mobile browser engine can present each subtitle at the correct time. Figure 6. shows an example of its use when accessing from an Android 2.2 platform and Firefox 4 web browser.

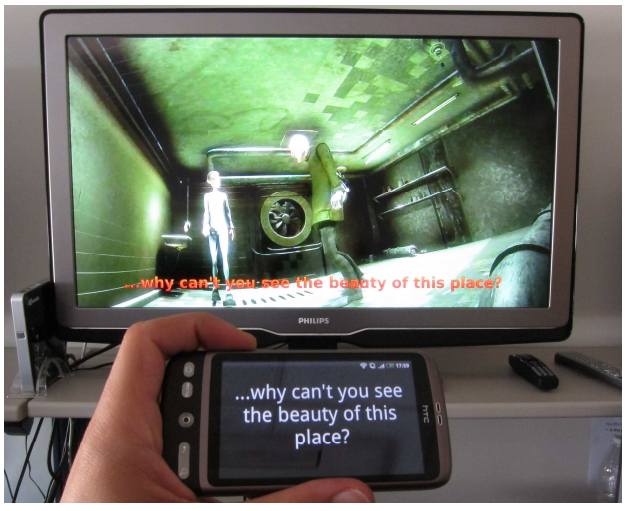


Figure 6. Example of 2nd screen application for subtitles.

VI. CONCLUSIONS

New web standards facilitate the building of new user applications that meet the tough requirements of ubiquitous environments that users are currently requesting. The present work has focused on the use of some of these standards (HTML5, SVG and SMIL) to improve the accessibility to video subtitles in web-based environments by allowing their customization and adaptation to personal needs. Furthermore, we strongly believe that the achieved flexibility of the subtitle styling can also be very useful for research studies on subtitle usability and accessibility on new platforms. However, the analysis conducted of the platform support reveals that, although high feature browsers are available on the main multimedia environments, not all browsers natively support SVG and SMIL animations. This can still be an issue for deploying the proposed subtitle approach on smart devices with low featured web browsers, but this situation is constantly changing. In addition, in this paper we have also presented some application use cases for customizable video subtitling on desktop environments, smart mobiles and smart TV. The ubiquity achieved by web standards enables the deployment of smarter applications for AmI environments that can also be accessed by everyone.

ACKNOWLEDGMENT

This article is supported by the Catalan Government Grant Agency Ref. 2009SGR700.

REFERENCES

[1] HTML5, W3C Working Draft, 24 June 2010, Ian Hickson, http://www.w3c.org/TR/html5/

[2] Web Content Accessibility Guidelines (WCAG) 2.0, W3C Recommendation, 11 December 2008, http://www.w3.org/TR/WCAG/

[3] Accessible Rich Internet Applications (WAI-ARIA) 1.0, W3C Working Draft, 15 December 2009, http://www.w3.org/TR/wai-aria/

[4] "SRT Subtitles", Matroska, http://www.matroska.org/technical/specs/subtitles/srt.html

[5] SMIL 3.0, W3C Recommendation, 1 December 2008, http://www.w3.org/TR/SMIL/

[6] S. Pfeiffer and C. Parker.: Accessibility for the HTML5 <video> element. In 6th Int. cross-disciplinary conference on Web accessibility (W4A'09), pages 98-100, Madrid, Spain, 2009.

[7] "Jquery.srt.js", Jan Gerber, http://v2v.cc/~j/jquery.srt/

[8] "SRT text tags with languages and categories", Michael Dale, Metavid.org, http://metavid.org/w/extensions/MetavidWiki/skins/mv_embed/example_usage/sample_timed_text.php

[9] "HTML5 DFXP Player prototipe", Philippe Le Hegaret, W3C, http://www.w3.org/2008/12/dfxp-testsuite/web-framework/START.html

[10] Timed Text (TT) Authoring Format 1.0 – Distribution Format Exchange Profile (DFXP), W3C Candidate Recommendation, 16 November 2006, http://www.w3.org/TR/2006/CR-ttaf1-dfxp-20061116/

[11] A.Danilo: Lights, Camera, Action. In the 7th International Conference on Scalable Vector Graphics (2009), http://svgopen.org/2009/papers/3-Lights_Camera_Action/#d4e69

[12] Release 2 Specification – Volume 5, Declarative Application, v2.0, Open IPTV Forum, Setember 2010, http://www.oipf.tv/Release_2.html.

[13] Porteiro, Minia (forthcoming) "The Use of Subtitles to Treat Speech-Language Disorders". *MONTI*.

[14] Pereira, Ana, Criteria for elaborating subtitles for deaf and hard of hearing adults in Spain: Description of a case study. In Anna Matamala and Pilar Orero. Listening to Subtitles. Subtitles for the Deaf and Hard of Hearing. (2010) pages 87-102. Bern: Peter Lang.

[15] Lorenzo, Lourdes. Criteria for elaborating subtitles for deaf and hard of hearing children in Spain: A guide of good practice. In Anna Matamala and Pilar Orero. Listening to Subtitles. Subtitles for the Deaf and Hard of Hearing. (2010) pages 139-148. Bern: Peter Lang

[16] Bartoll, Eduard & Anjana Martínez Tejerina. The positioning of subtitles for deaf and hard of hearing. In Anna Matamala and Pilar Orero. Listening to Subtitles. Subtitles for the Deaf and Hard of Hearing. (2010) pages 69-87. Bern: Peter Lang

[17] César, P., Bulterman, C. A., Jansen, A. J. Usages of Secondary Screen in an Interactive Television Environment: Control, Enrich, Share and Transfer television Content. In Proceedings of 6th European Conference EuroITV 2008, Springer, 168-177.