Journal of Universal Computer Science, vol. 19, no. 18 (2013), 2656-2676
submitted: 1/3/13, accepted: 30/10/13, appeared: 1/12/13 © J.UCS

A Model-Based Graphical Editor to Design Accessible Media Players

María González-García, Lourdes Moreno, Paloma Martínez
(Grupo LaBDA, Universidad Carlos III de Madrid, Leganés, Madrid, Spain
{mgonza1, lmoreno, pmf}@inf.uc3m.es)

Raúl Miñon, Julio Abascal
(Informatika Fakultatea, Euskal Herriko Unibertsitatea, Donostia, Spain
{raul.minon, julio.abascal}@ehu.es)

Abstract: The spectacular rise of multimedia Web content, especially video and audio content, makes addressing its accessibility a matter of urgency. All the regulations recognize that this type of content must be accessible to everyone, with or without a disability. To address this issue, this article presents a Model-Based Graphical Editor to design Accessible Media Players. This tool has been created in order to provide support to designers with little background in accessibility. To accomplish this work, a review of Accessibility Standards Regulation has been carried out, a set of accessibility requirements for accessible media players is proposed and a modelling of these requirements has been made.

Keywords: Accessibility, media player, standard, Model-Driven Development, Graphical Editor
Categories: H.1, H.5

1 Introduction

The amount of multimedia content on the Web is increasing at a staggering rate. Unfortunately, audio and video content are frequently inaccessible for many people for a number of reasons: most video clips lack captions; embedded players usually disregard accessibility guidelines; etc. This type of content must be accessible in order to avoid excluding a significant number of users. Therefore, it is necessary to make the effort to provide multimedia content in an accessible way that satisfies the needs of all users. Audio and visual content must satisfy at least the Web Content Accessibility Guidelines (WCAG) [W3C, 94c] of the Web Accessibility Initiative (WAI) [W3C, 94b] in order to be considered accessible.

In addition, it is essential to take into consideration the WAI's User Agent Accessibility Guidelines (UAAG) [W3C, 94] because of the amount of accessibility barriers that appear within the user agent that provides the content, affecting everybody. Nevertheless, most current players show a clear lack of accessibility [González-García, 11][Moreno, 11].

Bearing in mind the aforementioned and the current legislation (such as the Twenty-First Century Communications and Video Accessibility Act of 2010 [FCC, 10]), our motivation and aim is to develop a graphical editor based on the Model-Driven Development (MDD) approach that will allow the design of accessible model-based media players. The reason for using a MDD approach is to separate the platform-independent design from the platform-specific implementation of applications, delaying as much as possible the dependence on specific technologies [Koch, 06]. Moreover, this approach provides technology and platform independency and also facilitates the design of accessible media players by designers new to the area of accessibility.

This work was carried out using a User Interface Description Language (UIDL) [UIDL, 07] called User Interface Extensible Markup Language (UsiXML) [W3C, 09] and two plugins from the Eclipse development framework [Eclipse, 01]: Eclipse Modelling Framework (EMF) [Steinberg, 09] and Graphical Modelling Framework (GMF) [Eclipse, 06].

Before carrying out the graphical editor, a review of different accessibility standards was accomplished in order to obtain a set of accessibility requirements [González-García, 11] and a proposal of modelling of these requirements was drawn [González-García, 13]. This modelling approach has been the starting point for the development of the model-based editor which is presented in this paper.

This paper is organized as follows: Section 2 covers the background of user agents that provide video content, such as standards or regulations, and related work on accessible media players and the model-based approach. Section 3 introduces a set of main guidelines and a set of accessibility requirements that must be fulfilled by a media player for it to be considered accessible. Section 4 presents the proposal for design and development of the graphical editor. With the aim of illustrating and validating the work, Section 5 shows a lab demonstration; which is presented using a case study. Finally, Section 6 provides several conclusions and proposes future steps.

2 Background

This section reviews a number of standards, regulations and related works on accessible media players and their modelling processes. Similar works are also presented, such as the development of a graphical tool based on models using the UsiXML Framework.

2.1 Accessibility Regulation and Standards

The main reference for the standards that regulate multimedia content is WCAG 2.0 [W3C, 08b]. This standard describes how to make Web content more accessible to people with disabilities. Specifically, Guideline 1.2 (Provide alternatives for time-based media) [W3C, 08] of WCAG 2.0 establishes that video content shall be accompanied by media alternatives such as captions (or subtitles for people whose hearing is impaired), audio description, sign language, etc. WCAG 2.0 is the set of accessibility guidelines most referenced in the world and since 2012 is an ISO (ISO/IEC 40500:2012, Information technology - W3C Web Content Accessibility Guidelines (WCAG) 2.0) [ISO, 12]. Following this standard, there are several initiatives in different countries related to Web accessibility such as: Section 508 [U.S. HHS, 86], BITV 2.0 [BMJ, 11], RGAA [DGME, 09], AODA [Ontario E-laws, 05] and UNE 139803 [AENOR, 12]. These initiatives are similar to WCAG 2.0, being in some case a copy of it.

On the other hand, UAAG 2.0 [W3C, 11b] concerns user agents. This standard describes how to make user agents accessible to people with disabilities and how to increase the accessibility of Web content. Another standard (ISO 9241-171, Ergonomics of human-system interaction - Guidance on software accessibility [ISO, 08]), provides guidance on software accessibility, including accessibility requirements related to control of the multimedia content.

The new standard Hyper Text Markup Language 5 (HTML5) [W3C, 13] must also be considered. The current version of HTML5 incorporates support for native audio and video content, avoiding forcing the user to download and install specific third-party plugins to play audio and video. HTML5 did not initially provide support for including subtitles and audio description and although the <track> tag recently sorted out this problem the majority of user agents do not support it completely. Although this tag does not yet work satisfactorily, this standard will be in development until 2014.

2.2 Accessible Media Players

As far as media players are concerned, it is important to highlight a work that establishes the accessibility requirements for a media player to be considered accessible [González-García, 11]. In [Sayago, 12], an ethnological study of YouTube use by older people is conducted, as well as a discussion on the user experience and its design implications. In [Nishimura, 12] two versions of a media player directed at people with disabilities are developed for an educational environment. Another work, [Niederl, 12], presents a video player able to display two different movies. It also provides a guide for producing sign language-based synchronization.

From the review performed, two kinds of media players were identified. On the one hand, there are standalone media players that have more controls and are more accessible than the other type. On the other hand, there are embedded media players, with those based on the Flash technology [Adobe, 09] being among the most widely used. Moreno et al. [Moreno, 11] performed a study on the accessibility of three media players (CCPlayer [NCAM, 09], BBC iPlayer [BBC, 13], and YouTube [YouTube, 13b]) in order to determine whether these media players are compliant with accessibility requirements. This analysis also shows that although YouTube was less accessible than others, its accessibility is improving. For instance: they have developed Easy YouTube [Hiantonia, 09]; they have expanded their automatic captions to six languages [eWEEK, 12]; and they have recently developed Access:YouTube [YouTube, 13], which simplifies the standard YouTube site by allowing the use of assistive technologies and facilitating video clip search and playing.

Other user agents that provide video content with accessibility features are: JW Player [Longtailvideo, 12], which provides captions and audio description; BSPlayer [BSPlayer, 13], which provides captions and the option to change font type and size; and the VideoLan [VLC, 13] media player, which provides keyboard shortcuts and allows users to change the size, font or colour of captions. In addition, some HTML5 players with accessibility features have been found; such as the Acorn Media Player [Ghinda, 13], which provides full keyboard control. It allows the integration of external subtitles in “srt” format to be used to include captions. This player also provides dynamic transcriptions generated from the selected captions. Another example is LeanBack [LeanBack, 10], which supports <video> and <audio> elements, allows subtitles through the use of the <track> element, and provides keyboard shortcuts for desktop browsers. Finally, it is worth mentioning Video JS [VideoJS, 13], which demonstrates a very user-friendly approach for supporting captions.

2.3 Model-based Design Approaches

In this section some works using the UsiXML framework and model-based design of graphical tools are presented. There are numerous works related to model-based approaches using UsiXML, but our interest is focused on the approaches aimed at creating a graphical tool to produce User Interfaces (UIs).

Among the works aimed at the design of graphical tools, FlexiXML [Creissac, 11] can be highlighted. This interpreter tool supports the automatic generation of user interfaces through models expressed in UsiXML. Moreover, in the work presented by Montenegro et al. [Montenegro, 11], a Domain-Specific Model (DSM) for the construction of platform-independent modules of learning management systems (LMS) is designed. This approach aims at building a meta-model for the construction of a domain specific language (DSL), using model-driven engineering (MDE) techniques, and applying the appropriate transformations to achieve a platform-independent model. They use two Eclipse plugins to create the DSM: Eclipse Modelling Framework (EMF) and Graphical Modelling Framework (GMF). These same plugins are used in this work (see Section 4). Another graphical tool, called SPA4USXML, is presented by Miñón et al. [Miñon, 12]. This tool assists the designer of ubiquitous services to create specifications for the Task and the Abstract User Interfaces (AUI) required by the EGOKI adaptive system [Abascal, 11]. SPA4USXML also provides EGOKI with a resource model for selecting the most appropriate type of multimedia resource (text, video, audio, image, etc.) for the user. Finally, the work presented by Kanai et al. [Kanai, 09] proposes a 3D tool for digital prototyping and usability assessment of information appliances. In this work: Firstly, the UsiXML specification is extended; secondly, 3D prototyping and simulation functions are developed; and then automated user test and usability assessment functions are also developed.

3 Accessibility Requirements for Media Players

To accomplish this work a set of accessibility requirements for the development of accessible media players was needed. With this aim, a revision of the standards (in this case UAAG 2.0 and ISO 9241-171) was performed [González-García, 12]. In the following subsections, a set of guidelines that need to be considered for an accessible media player is proposed. Then, based on these guidelines, a set of requirements for accessible media player design is established.

3.1 Main Guidelines

The first step is to decide what type of elements a media player needs to have in order to be considered accessible. According to González-García et al. [González-García, 11] these requirements are:

- To provide alternative content such as captions, audio description, sign language, transcription or extended audio description. Consequently, to provide the control to enable/disable this alternative content.
- To provide complete access to all the features via the mouse, the keyboard or through assistive technologies.
- To provide help and access to documentation describing the accessibility features the media player provides and how to use them.
- To provide a keyboard focus cursor that indicates which element has the focus in the user interface and a text cursor that indicates where the focus is within the text.

Based on these requirements, the following set of elements required in an accessible player has been derived:

- Play, stop and pause.
- Resize.
- Adjust the volume.
- Enable/disable captions.
- Enable/disable audio description.
- Search captions.
- Forward-delay seconds.
- Change the size, colour or font of the captions.
- Access to help documentation.

Firstly, a set of constraints and relationships between the elements should be established based on their functionalities. For example, it does not make sense to allow users to pause, stop, rewind or forward the playback if the play action is not enabled.

3.2 Accessibility Requirements

Table 1 shows the requirements (grouped by category [González-García, 12]) obtained from the previous review of standards and the collection of suitable guidelines. These categories are the result of grouping the requirements according to whether the requirements are native (traditional requirements included in the user agent or media player) or they add new functionality (necessary to satisfy specific accessibility requirements and appearing as "additional" in Table 1). The requirements are also classified into subgroups, depending on whether: they affect playing, viewport (subgroup size), volume or alternative content; or they provide help or search functions.

Every requirement is identified by a code (see Table 1). This code is made up of the first letter of the group, the first letter of the subgroup and two numbers. For instance, the requirement NP01 belongs to the native group, the playback subgroup and it has been assigned the number 01.

There are also other requirements concerning cognitive accessibility, close to usability requisites, that must be taken into account. Users should be able to configure accessibility features saved in previous sessions and to change features on request. In addition, users should be provided with information on accessibility features in order to know, for instance, whether or not the user agent provides keyboard shortcuts. If the agent does provide this, the users should be allowed to navigate through the content without enabling controls or configuring their shortcut preferences.

Code	**Name**	**Description**	**Group**	**Subgroup**	**Source**	
					ISO	**UAAG 2.0**
NP01	Play	Play the video content	Native	Playback	10.8.2	2.11.6
NP02	Stop	Stop the video content	Native	Playback	10.8.2	2.11.6
NP03	Pause	Pause the video content	Native	Playback	10.8.2	2.11.6
NS01	Resize	Resize the viewports	Native	Size	10.5.8	1.8.3
NV01	Mute	Enable or disable the audio content	Native	Volume	10.6.2	1.5.1
NV02	Volume	Adjust the volume	Native	Volume	10.6.2	1.5.1
AP01	Rewind	Delay seconds within a playback	Additional	Playback	10.8.3	2.11.7
AP02	Forward	Forward seconds within a playback	Additional	Playback	10.8.3	2.11.7
AA01	Caption	Enable or disable captions	Additional	Alternatives	10.1.3, 10.7.2	1.1.2
AA02	Audio Description	Enable or disable audio description	Additional	Alternatives	10.1.3	1.1.2
AA03	Size	Change the size of the captions	Additional	Alternatives	10.7.3	1.4.1
AA04	Font	Change the font of the captions	Additional	Alternatives	10.7.3	1.4.1
AA05	Colour	Change the colour of the captions	Additional	Alternatives	10.7.3	1.4.1
AA06	Language Caption	Change the language of the captions	Additional	Alternatives	8.2.1	2.7.1
AA07	Language Audio	Change the language of the audio description	Additional	Alternatives	8.2.1	2.7.1
AH01	Help	Help documentation about accessibility features	Additional	Help	11.1.5	3.3.2
AF01	Find	Search within playback captions	Additional	Find		2.4.5

Table 1: Media player accessibility requirements

4 Graphical Editor Development

In this section we present the methodological framework, which follows a Model-Driven User Interface approach. In addition, the different steps that have been undertaken to develop the graphical editor to offer methodological support for the design of an accessible player are described.

4.1 Methodological Framework

In this work we adopt the UsiXML framework [Limbourg, 05]. UsiXML is a declarative language capturing the essence of what a User Interface is or should be, independently of physical characteristics. This language is structured according to the four levels of abstraction defined by the *Cameleon* reference framework [Calvary, 03]: Task and Concepts, Abstract User Interface (AUI), Concrete User Interface (CUI) and Final User Interface (FUI) (see Figure 1). UsiXML is universally recognized and is used to design and develop interactive systems, due to the richness of the models that provides. It also supports device, platform and modality independence. UsiXML allows User Interfaces to be described for multiple contexts of use: such as Graphical User Interfaces, Character User Interfaces, Multimodal User Interfaces, or Auditory User Interfaces. Therefore, interactive applications with different types of computing platforms, interaction techniques and modalities of use can be described in a way that allows the design to take place independently of the specific characteristics of the physical computing platform. All these aspects increase accessibility as they facilitate different modes of interaction that provide access for a large number of users. For these reasons, the UsiXML framework has been selected as the underlying UIDL for creating our graphical tool.

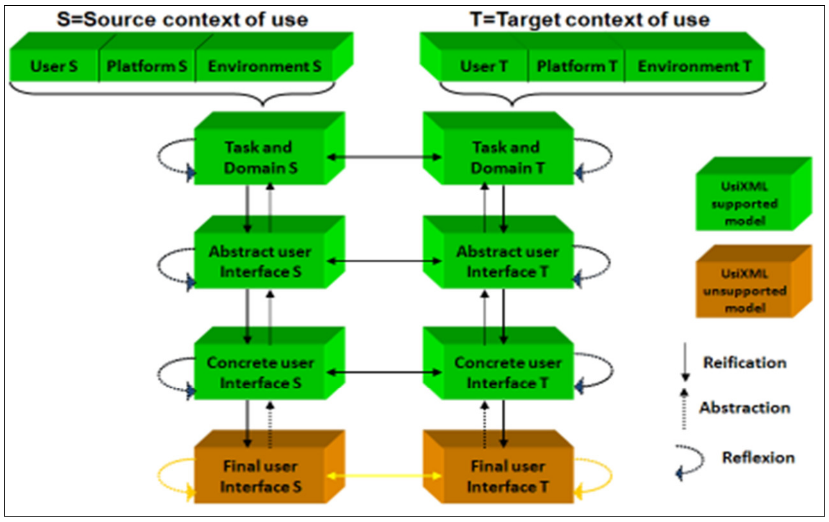


Figure 1: The Cameleon reference framework for multi-target UIs (source: *[W3C, 09])*

On the other hand, the Eclipse framework for developing a graphical editor is used in order to provide support to professionals in the design of accessible players. We use Rich Client Platform (RCP) [Eclipse, 04], which enables Eclipse to be used in a wide range of end-user applications that are not integrated in development environments. This platform also provides features such as plugins, perspectives and views.

4.2 Accessible Player Modelling

Previously to the current approach, some models were created using the Task model and the AUI model of UsiXML [González-García, 12] [González-García, 13]. In that approach knowledge was acquired about how to model the accessibility requirements using design primitives of the UsiXML meta-models.

The Task model included information on the actions that users perform to achieve their goals and the objects involved in these actions. That information also guided the development of the interface; in particular, it influenced the components that would appear in the interface and the ways of manipulating those components [Wilson, 96]. It was thus made up of a set of tasks (abstract task, interaction task, user task, etc.) and a set of relationships (choice, enabling, concurrent, etc.).

An AUI was defined according to the *Cameleon* reference framework; i.e. a user interface supporting an interactive task abstracted from its implementation, independently of any target computing platform or interaction modality. While an AUI could be specified from scratch, it could also be produced from different models (such as a task model, a domain model, or a combination of both) and could be based on information describing the context of use (i.e. the user, the platform, and the environment) [Tran, 12]. An AUI is made up of Abstract Containers (ACs), Abstract Interaction Components (AICs) and Abstract Relationships.

The task model showed the interaction between elements, while the AUI model showed the structure of the interaction elements. In our domain, the Task model (see Figure 2) defined the interaction between a user and an accessible media player and the AUI model defined the structure of this interaction [González-García, 13]. It is worth highlighting that the generation of the AUI model started from the task model.

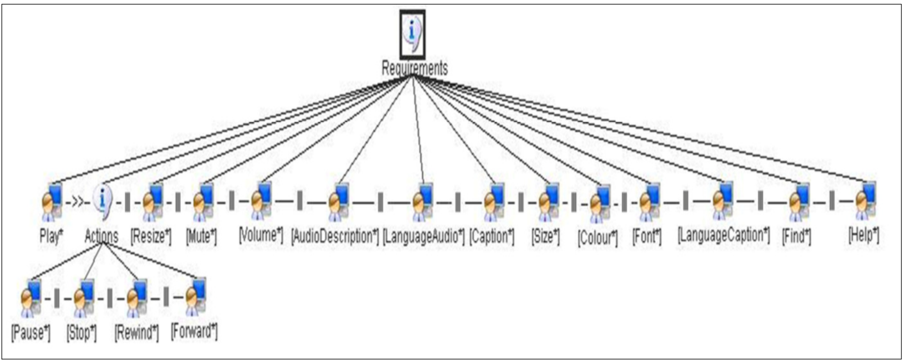


Figure 2: Task model of an accessible user agent that provides accessible video content

Starting from this model, a design solution that specifies the accessibility requirements considered in Section 3.2 was defined. After that work, it was concluded that if anyone wanted to advance to the next levels of the *Cameleon* reference framework (CUI and FUI), it would be necessary to have a detailed knowledge of UsiXML, its meta-models and a set of tools that are not easy to use. Therefore, the provision of a graphical editor could be useful to facilitate the design.

As previously mentioned, it was decided to create a graphical editor to facilitate the task of designing user interfaces describing accessible media players that inherently include accessibility through design primitives. The design and development of the graphical editor, which is the main goal of this work, is introduced in the following section.

4.3 Development of a graphical editor

Due to the domain features, the graphical editor was created based on the CUI Meta-model of UsiXML. The Eclipse GMF plugin was used to develop a graphical editor. GMF provides a runtime infrastructure for developing graphical editors based on the EMF plugin. Consequently, the first step was to create the CUI meta-model with the EMF plugin.

The CUI allows definition of the specification of the appearance and behaviour of a UI for a given context of use [Limbourg, 04]. A CUI is composed of Concrete Interaction Objects (CIO) and Concrete Relationships (CR).

In this work, a restricted CUI meta-model was created with EMF in the domain of accessible media player design. This meta-model includes only the design primitives required for this domain. Specifically, a CUI describes a potential user interface after selecting a particular interaction modality (graphical, vocal, multimodal) [W3C, 09]. It was decided to use the graphical modality for this approach, since it was essential that the user should be able to interact via external devices, such as keyboards or mice.

Based on the graphical modality of the CUI meta-model, the accessibility requirements established in Section 3.2 and a review of some media players to understand their traditional functionality in depth, all the design primitives were selected. Some of them were selected indirectly (for example, the design primitives related to listener), while other primitives were established directly. These primitives include: ToolBar, ToolBarButton, ToolBarSeparator, CommandButton, Slider, ComboBox and ComboItem, which are related to accessibility; and primitives such as VideoComponent, AudioComponent, ProgressionBar, Menu, MenuItem, MenuSeparator, MenuBar and MenuBarItem, which provide the traditional functionality of a media player.

Four abstract concepts were distinguished to define the accessibility requirements using the design primitives of the EMF meta-model. These allow a model-based accessible player to be designed using the graphical editor:

1. First concept: define a requirement type where a user action triggers another action during the playback of the video content that directly affects the playback (e.g. the video content is played, stopped, paused, etc. as a result of this action). The Play ('NP01'), Stop ('NP02'), Pause ('NP03'), Rewind ('AP01'), Forward ('AP02'), Resize ('NS01'), Mute ('NV01'), Caption ('AA01') and AudioDescription ('AA02') requirements can be defined using

this concept. This concept can be defined using the ToolBarButton design primitive; therefore, the ToolBar and ToolBarSeparator design primitives are also needed.

2. Second concept: define a requirement type where a user action triggers an action after stopping the video playback. The Help ('AH01') requirement (which shows additional information), and Find ('AF01') requirement (which searches a caption within the playback), can be defined using this concept. In this case, the CommandButton design primitive is used.
3. Third concept: define a requirement type where a set of options is shown. The Size ('AA03'), Font ('AA04'), Colour ('AA05'), and LanguageCaption ('AA06') requirements (which are related to captions) and the LanguageAudio ('AA07') requirement (which is related to audio description) can be defined using this concept. This concept can be defined using the ComboBox and ComboItem design primitives.
4. Fourth concept: define a requirement type where an element's value is increased or is decreased. Requirement NV02 ('Volume') can be defined using this concept. In this case, a Slider design primitive is used to define the concept.

Table 2 shows all the accessibility requirements mapped to the design primitives. Depending on the types of design primitives, the requirement types and even the designers' (or developers') preferences, the number of design primitives could or could not be determined. For example the ToolBar and Slider design primitives are always the same; however, the ToolBarButton, ToolBarSeparator, CommandButton and ComboBox design primitives depend on the number of requirements, while the ComboItem design primitive depends on the designer or developer preferences.

In conclusion, Table 2 shows a summary of which design primitives of the CUI Meta-model of UsiXML are used to define the accessibility requirements. An EMF Meta-model was thus obtained for the accessible player domain starting from this restricted CUI Meta-model. This EMF was the first step in the development of a Graphic editor with GMF.

4.4 Generation of Final User Interfaces

When the graphical editor has been generated through the GMF plugin, it is possible to modify the auto-generated code to add the new editor extra functionalities. It is particularly worth pointing out that new elements can be created in the menu bar to provide extra functionality to the RCP application. In our approach, this possibility is useful for allocating the functionalities to generate the different FUIs. An example of the transformation from a CUI to a FUI using HTML5 is provided in subsection 5.3.

Requirement code	Requirement Name	Design primitives of CUI model	
NP01	Play	ToolBar	
NP02	Stop		
NP03	Pause		
NS01	Resize		
NV01	Mute		
AP01	Rewind	ToolBarButton	ToolBarSeparator
AP02	Forward		
AA01	Caption		
AA02	AudioDescription		
AH01	Help	CommandButton	
AF01	Find		
AA03	Size	ComboBox	ComboItem
AA04	Font		
AA05	Colour		
AA06	LanguageCaption		
AA07	LanguageAudio		
NV02	Volume	Slider	

Table 2: Mapping between accessibility requirements and design primitives of the CUI meta-model for modelling with the EMF meta-model

4.5 An authoring tool to generate accessible user interfaces

The purpose of the graphical editor is to assist the designers to develop an accessible media player following standards. In order to get this aim and taking into account that not all the designers will be expert in the field, accessibility requirements have been integrated within the editor through the MDD approach. Thus, when designers use the graphic editor they only have to focus on dragging and dropping the graphical elements that will be a part of the final user interface. This authoring tool allows user to design an accessible media player according to their preferences. Furthermore, they will be able to select the desired technology to generate the final user interface.

The graphical editor includes a help wizard in order to face accessibility issues, such as which accessibility requirements need to be incorporated to fulfil a compliance level of UAAG 2.0. Thus, users care about the accessibility levels that are reached and what are the followings steps to comply them.

In order to illustrate the graphical editor and validate the proposal, in the following section a case study is presented.

5 Case Study

As an empirical validation of the research, a lab demonstration is presented here by means of a case study. The demonstration justifies that the methodology can be used in practice and shows the utility of the proposed approach by providing an explanation of the mechanisms used to include accessibility requirements in software development [Wieringa, 10].

5.1 Scenario

The selected scenario is going to be called "Providing control of the audio description". In this scenario, the design of a player for blind students is going to be described. This section is divided in three parts. In the first part, the scenario is going to be described. After the description, the design primitives are going to be presented. And at last, our groups of interest are going to be shown.

5.1.1 Brief description

The Centre for Inclusive Education needs to design a player to deliver multimedia learning content in an accessible way on the Web. Its goal is to offer different types of content depending on the access features and the student preferences. One scenario is "a blind user accessing video content". In this case it is important to provide video content together with alternative content, such as audio description. In order to avoid accessibility barriers created by incompatibility between different technologies (given the wide range of different platforms), the centre wants to use HTML5 to develop the player.

The development team is asked to develop a web interface to integrate the given requirements in a player. One of the designers has to design a player that allows video content and audio description access based on good practice and accessibility standards.

The solution is to propose a model-based design for accessible players. Following, a set of steps is established to carry out the design.

1. The designer has to determine which accessibility requirements must be fulfilled. The "Accessibility Requirements" resource can be used for this purpose (see Section 3.2). By means of this resource, the designer is contacted with requirements such as: allowing users to enable/disable the audio description or captions; to search captions within the playback; etc. Due to the designer is still not an expert in accessibility, he/she decides to use a graphical editor as an authoring tool that will help him/her in the design.
2. The designer uses the graphical editor based on the UsiXML CUI model. To use it, it is crucial to follow the next steps:
    a. Creating a new project.
    b. Selecting the design primitives identified through the palette's graphical elements.

   c. Designing the system.
   d. Validating the design.
   e. Collecting accessibility recommendations provided by the editor to help designers.
   f. Providing the outcome, that is, an accessible player design based on the CUI model.
3. The next step in the development process is transformed the CUI into a FUI model matching the CUI design primitives and software components that are dependent on technology, platform or interaction modality; in this case, in HTML5.

5.1.2 Design primitives

In order to design the scenario, the following primitives are used:

1. ToolBar, ToolBarButton and ToolBarSeparator are used to represent the Play/Pause, Stop, Resize, Mute, Forward, Rewind and AudioDescription elements.
2. CommandButton is used to represent the Help and Find elements.
3. ComboBox and ComboItem are used to represent the audio description language (LanguageAudio).
4. Slider is used to represent the Volume element.

These design primitives are going to be widely explained in Section 5.2.

5.1.3 User groups of interest

The main group of interest are blind students and blind teachers. Apart from this group, our second group of interest are students in general (having or not a disability). Any student can access video content in environments where it can only be accessed by audio channel, for example on the bus, while walking, etc. In this way, a learning resource for all is provided.

5.2 Building of the Concrete User Interface from the graphical editor

For the proposed scenario, the designer would use the following elements of the graphical editor, which are required to provide blind users with accessible user interfaces:

1. A ToolBar element with seven ToolBarButton elements for representing the Play/Pause, Stop, Rewind, Forward, Mute, Resize and Audio Description elements. Additionally, two listeners were respectively associated with the Play and Pause buttons for providing the behaviour of hiding one when the other is active and vice versa.
2. A Video Component together with a ConcreteGraphicalListener element. This ConcreteGraphicalListener allocates one ConcreteEvent typed onGUIClik and a ConcreteAction that is related to the playPauseAction method. This method is a pre-created javaScript function that allows users to play or pause the video only when the video element is clicked.
3. A ComboBox element with two ComboItems that enable English or Spanish audio descriptions.
4. A Slider element to control the volume, with values between 0 and 10.

5. A CommandButton that allows users to enable the Help functionality.
6. The purpose of the Play, Pause, Stop… elements are assigned through the label attribute of each concrete element. It is thus possible to generate an accessible player considering that semantic allowing the designer to select the same element type for different purposes. For example, as previously explained, the Play/Pause, Stop, Rewind, Forward, Mute, Resize and AudioDescription elements are represented by the same concrete element (the ToolBarButton element). Figure 3 illustrates the CUI created for blind users.

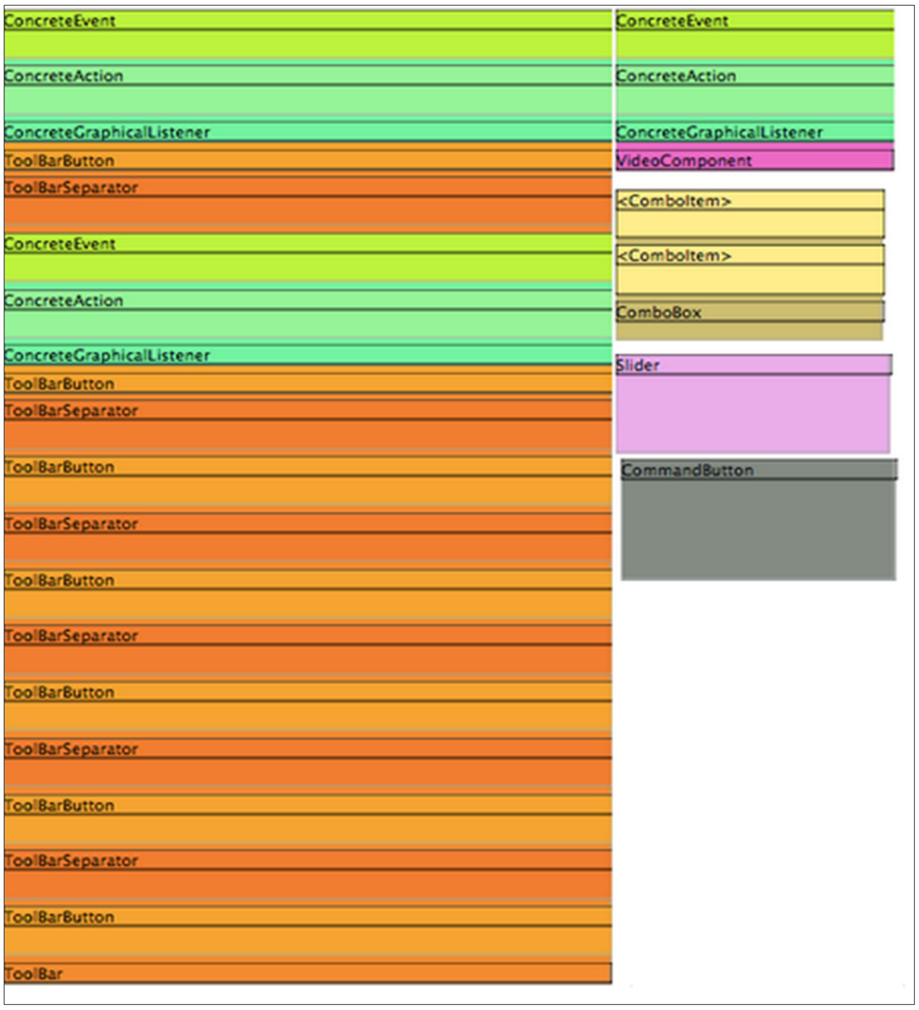
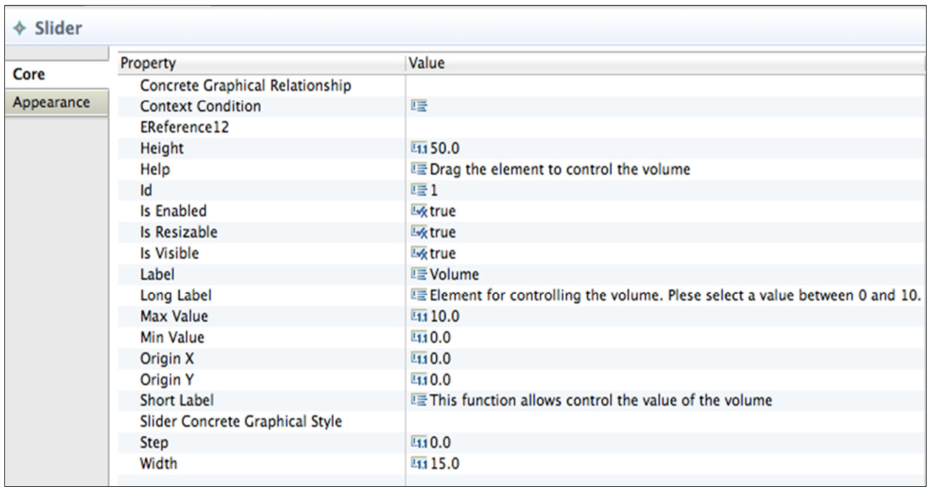


Figure 3: Representation of CUI elements in the graphical editor

This design is the result of dragging and dropping every element that is part of the proposed design, a media player for a blind user. As aforementioned, the graphical editor presents a palette with all the elements that we have modelled previously and a canvas where the elements can be put by means of dragging and dropping them (as it can be seen in Figure 3). This figure represents the accessible media player design based on our CUI model. At first sight, the design is a set of elements whose structure is like a table, but actually, it is the first step to develop the FUI of our media player (section 5.3), because starting from this model a FUI like Figure 6 can be obtained.

In order to understand some of the elements, Figure 4 illustrates the properties of the Slider element that is used to represent the volume of the player.



Slider

Core
Appearance

Property	Value
Concrete Graphical Relationship	
Context Condition	
EReference12	
Height	50.0
Help	Drag the element to control the volume
Id	1
Is Enabled	true
Is Resizable	true
Is Visible	true
Label	Volume
Long Label	Element for controlling the volume. Plese select a value between 0 and 10.
Max Value	10.0
Min Value	0.0
Origin X	0.0
Origin Y	0.0
Short Label	This function allows control the value of the volume
Slider Concrete Graphical Style	
Step	0.0
Width	15.0

Figure 4: Slider element features

5.3 Transformation to accessible final player with HTML5

The generation of the FUI from the CUI model has been implemented via an XSL transformation. An option to perform this process has been integrated in the editor (see

Figure 5). For this work, the HTML5 language was selected for generating the FUI. This language provides native support for embedding video elements without the need for installing third-party plugins and, in addition, it is supported by the most widely used platforms.

As mentioned in the previous subsection, the elements of the CUI have been labelled with their matching semantic, providing the information necessary to perform the transformation. Table 3 illustrates how the CUI elements have been mapped to the HTML5 methods. It must be pointed out that since HTML5 does not provide a specific element for rendering sliders, an accessible slider element [FG, 09] (labelled with WAI-ARIA [W3C, 11]) based on jQuery UI library [jQuery, 13] has been used.

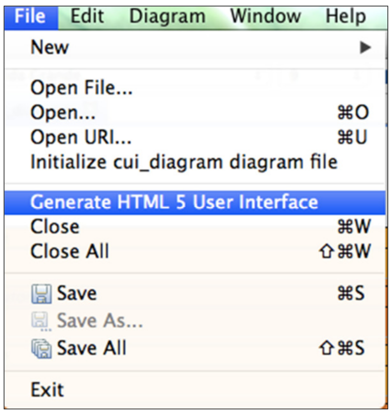
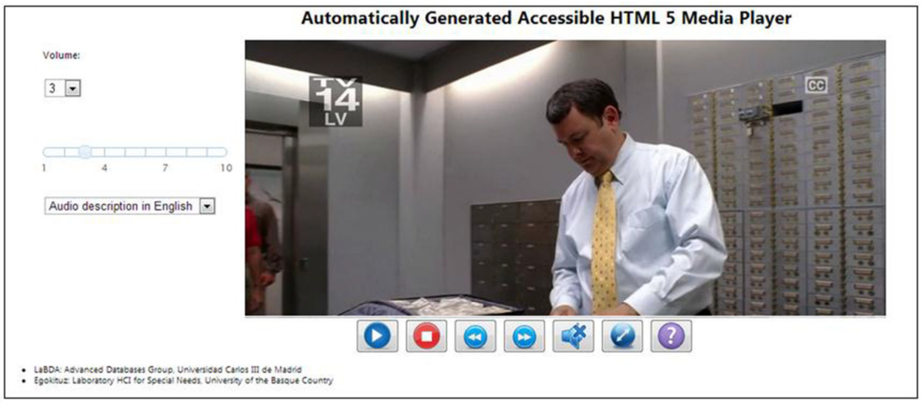

File Edit Diagram Window Help
New
Open File...
Open... ⌘O
Open URI... ⌘U
Initialize cui_diagram diagram file
Generate HTML 5 User Interface
Close ⌘W
Close All ⇧⌘W
Save ⌘S
Save As...
Save All ⇧⌘S
Exit

Figure 5: Generation of HTML5 User Interface

CUI elements	**HTML 5 elements**
ToolBar	Div | Menu + Type attribute
ToolBarButton	Button + Image | Command element (inside menu element)
VideoComponent	Video
ConcreteGraphicalListener	JavaScript function
Slider	jQuery UI slider
ComboBox + ComboItems	Select + Options+track
CommandButton	Button
ToolBarSeparator	CSS Layout | Separator (<hr>) (inside menu element)

Table 3: Mapping between CUI elements and HTML5 methods

Figure 6 shows the result of the generation process. As can be seen, the elements required for an accessible media player have been integrated in the UI. In addition to the controls that have been implemented using non-intrusive JavaScript, the default controls of the video element have been kept. This is because otherwise users with agents that have JavaScript disabled would not be able to interact with the video content. For this work, only basic default CSS rules have been integrated, just for aligning the components in the UI. In future works the editor will be able to attach different types of styles to the elements of the CUI in order to generate more attractive FUIs; following a Responsive Design, using properties such as Progressive Enhancement.


Automatically Generated Accessible HTML 5 Media Player

- LaBDA: Advanced Databases Group, Universidad Carlos III de Madrid
- Egokituz: Laboratory HCI for Special Needs, University of the Basque Country

Figure 6: FUI coded with HTML5

6 Conclusions and future work

It is essential to keep in mind accessibility issues for multimedia content, due to the great amount of this type of content currently available to the users.

In addition to taking into account the accessibility of the content itself, with the inclusion of the alternatives such as captions or subtitles, audio description, transcriptions, etc., it is fundamental to integrate accessibility requirements in the software used to access the content (i.e. the player). In this work we have carried out an exhaustive analysis of related work and the accessibility standards to compile a set of accessibility requirements that must be met in order to create accessible players. Based on these accessibility requirements, a model-based design proposal following the Model-Driven Development approach is provided.

Finally, with the aim of providing support to designers, we presented a model-based graphical editor that facilitates the task of designing accessible players.

The MDD approach has the advantage of making the design independent of the final technology. This approach provides flexibility to the proposal presented, which is an important factor due to the diversity of existing technologies, platforms, interaction modalities and devices.

An added value of this work is that the graphical editor allows the integration of accessibility regardless if the designer is an expert in accessibility.

The design proposal, including the abstract levels (tasks and AUI model), has been developed using the theoretical *Cameleon* framework. An initial prototype of the editor is presented. We are currently working to complete the integration of all constraints for every accessibility requirement in order to complete the validation of the model-based design. Besides illustrating the proposal to the reader, this prototype allowed us to carry out empirical research validation through a lab demonstration.

On the other hand, the fact that this work is based on standards, it does not guarantee the accessibility of the FUI, unless designers follow a user-centred design (UCD). Likewise, users must validate the design.

To sum up the work which has been carried out, we have been able to develop a prototype of an accessible media player taking into account a set of accessibility requirements after analysing accessibility standards and after following a model-based design proposal. The importance of this development is not only the media player itself, but it is the possibility to develop this type of user agent in an accessible way without being any expertise in the accessibility field. Likewise, it is important to highlight that this prototype allows designer to develop media players adapted to users with special needs or not depending on the final user.

We are currently working on a help wizard for the editor. The aim of the wizard is to guide the designer on which requirements have to be included and which do not (according to different standards), and which user groups are affected. The designer is thus informed of the accessibility level achieved and what remains to be done.

Acknowledgements

This research work is supported by the Research Network MAVIR (S2009/TIC-1542) and MULTIMEDICA project (TIN2010-20644-C03-01). This research work has also been partly funded by the Spanish Ministry of Science and Innovation ModelAcces project (grant TIN2010-15549), the Department of Education, Universities and Research of the Basque Government (grant IT395-10). In addition, Raúl Miñón holds a Ph.D. scholarship from the Research Staff Training Programme of the Department of Education, Universities and Research of the Basque Government.

References

[Abascal, 11] Abascal, J., Aizpurua, A., Cearreta, I., Gamecho, B., Garay-Vitoria, N., Miñón, R. (2011). Automatically generating tailored accessible user interfaces for ubiquitous services. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '11). ACM, New York, NY, USA, Pages: 187-194.

[Adobe, 09] Adobe Flash runtime/Statistics http://www.adobe.com/products/flashruntimes/statistics.html.

[AENOR, 12] AENOR, Asociación Española de Normalización y Certificación (2012) UNE 139803:2012, Web content accessibility requirements,, Norma UNE 139803:2012, 2012, http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0049614.

[BBC, 13] BBC iPlayer, 2013, http://www.bbc.co.uk/iplayer/tv.

[BMJ, 11] BITV, Bundesministerium der Justiz,Barrierefreie-Informationstechnik-Verordnung 2.0, 2011, http://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html.

[BSPlayer, 13] BSPlayer, 2013, http://www.bsplayer.com.

[Calvary, 03] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. (2003). A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers. Vol. 15, No. 3, June 2003, pp. 289-308.

[Creissac, 11] Creissac Campos, J., Alves Mendes, S.,(2011). FlexiXML A portable user interface rendering engine for UsiXML. Proceedings of UIDL'2011 - Software Support for User Interface Description Language - Interact'2011 Workshop, Lisboa, Portugal, 2011, (Eds.). ISBN 978-2-9536757-1-9.

[DGME, 09] RGAA , République française, Référentiel Général d'Accessibilité pour les Administrations, 2009, http://www.references.modernisation.gouv.fr/rgaa-accessibilite.

[Eclipse, 01] Eclipse, 2001, www.eclipse.org.

[Eclipse, 04] RCP, Rich Client Platform, 2004, http://www.eclipse.org/home/categories/rcp.php.

[Eclipse, 06] Eclipse, Graphical Modeling Framework, 2006, http://www.eclipse.org/modeling/gmp/.

[eWEEK, 12] eWEEK, 2012, http://www.eweek.com/cloud/youtube-expands-video-captioning-in-6-more-languages/.

[FCC, 10] FCC; Federal Communications Commission, Twenty-First Century Communications and Video Accessibility Act (CVAA), 2010, http://www.fcc.gov/guides/21st-century-communications-and-video-accessibility-act-2010.

[FG, 09] Filament Group, 2009, http://filamentgroup.com/lab/update_jquery_ui_slider_from_a_select_element_now_with_aria_support/

[Ghinda, 13] AcornMedia Player, 2013, http://ghinda.net/acornmediaplayer/.

[González-García, 11] González-García, M., Moreno, L., Martínez, P., Iglesias, A. (2011). Web accessibility requirements for media players. 13th IFIP TC13 Conference on Human-Computer Interaction (INTERACT 2011), Lisboa, Portugal, September, 2011, LNCS, Springer, Volumen: 6949, Pages: 669-674.

[González-García, 12] González-García, M., Moreno, L., Martínez, P., (2012). An approach to User Interface Design of an accessible user agent. Proceeding of the 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI'2012), Douro Region, Portugal, Volume 14, Pages: 254-262.

[González-García, 13] González-García, M., Moreno, L., Martínez, P., (2013). Approach design of an accessible media player. Universal Access in the Information Society (UAIS), 2013, in press.

[Hiantonia, 09] Easy Youtube player, 2009, http://hiantonia.com/journal/2009/06/17/easy-youtube-player-making-it-easier/.

[ISO, 08] ISO, International Organization for Standardization, ISO 9241-171:2008, 2008, Ergonomics of human-system interaction (Guidance on software accessibility), 2008, http://www.iso.org/iso/catalogue_detail.htm?csnumber=39080.

[ISO, 12] ISO, International Organization for Standardization, ISO 40500:2012, 2012, Information technology -- W3C Web Content Accessibility Guidelines (WCAG) 2.0), 2012, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=58625

[jQuery, 13] jQuery UI,2013, http://jqueryui.com.

[Kanai, 09] Kanai, S., Higuchi, T., Kikuta, Y., (2009). 3D digital prototyping and usability enhancement of information appliances based on UsiXML. International Journal on Interactive Design and Manufacturing (IJIDeM), 2009, Volume 3, Issue 3, Pages: 201-222.

[Koch, 06] Koch N. (2006). Transformation Techniques in the Model-Driven Development Process of UWE, Workshop proceedings of the sixth international conference on Web engineering (ICWE'06), Palo Alto, California, July, 2006, Article nº 3.

[LeanBack, 10] LeanBack Player, 2010, http://www.leanbackplayer.com/.

[Limbourg, 04] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., Trevisan, D. (2004). UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces. Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages", AVI, 2004, Pages: 55-62.

[Limbourg, 05] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López, V. (2005). UsiXML: a Language Supporting Multi-Path Development of User Interfaces, in: Bastide, R., Palanque, P., Roth, J. (eds.) Engineering Human Computer Interaction and Interactive Systems. Springer, Heidelberg, LNCS, vol. 3425, Page: 200-220.

[Longtailvideo, 12] JWPlayer, 2012, http://www.longtailvideo.com/jw-player/.

[Miñon, 12] Miñón, R., Moreno, L., Abascal, J., (2012). A graphical tool to create user interface models for ubiquitous interaction satisfying accessibility requirements. Universal Access in the Information Society (UAIS), 2012, in press.

[Montenegro, 11] Montenegro Marín, C.E., Gaona García, P.A., Cueva Lovelle, J.M., Sanjuan Martínez, O., (2011). Aplicación de Ingeniería Dirigida por Modelos (MDA), para la construcción de una herramienta de Modelado de Dominio Específico (DSM) y la creación de módulos en Sistemas de Gestión de Aprendizaje (LMS) independientes de la plataforma. Dyna, 2011, Nro. 169, Pages: 43-52, ISSN; 0012-7353.

[Moreno, 11] Moreno, L., González-García, M., Martínez, P., Iglesias, A.,(2011).A study of accessibility requirements for media players on the Web, 14th International Conference on Human-Computer Interaction (HCII 2011), Florida, USA, July, 2011, Vol: LNCS 6765, Pages: 249-257.

[NCAM, 09] NCAM, CCPlayer, 2009, http://ncam.wgbh.org/invent_build/web_multimedia/tools-guidelines/ccplayer.

[Niederl, 12] Niederl, F., Bußwald, P., Tschare, G., Hackl, J., Philipp, J. (2012). Dubbing of Videos for Deaf People – A sign Language Approach. Proceedings of the 13th International Conference on Computers Helping People with Special Needs (ICCHP 2012), Linz, Austria, Pages: 225-228.

[Nishimura, 12] Nishimura, K., Cohen, M. Media Players for Accessibility. Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments (HCCE'12), New York, USA, 2012, ISBN: 978-1-4503-1191-5, Pages: 184-189.

[Ontario E-laws, 05] AODA, Ontario, Accessibility for Ontarians with Disabilities Act, 2005, http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_05a11_e.htm.

[Sayago, 12] Sayago, S., Forbes, P., Blat, J. (2012). Older people´s social sharing practices in Youtube through and ethnographical lens. Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers (BCS-HCI '12), Birmingham, UK, Pages: 185-194.

[Steinberg, 09] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E. EMF: Eclipse Modeling Framework 2.0. Second Edition, Addison-Wesley Professional, 2009, ISBN: 0321331885.

[Tran, 12] Tran, V., Vanderdonckt, J., Tesoriero, R., Beuvens, F. (2012). Systematic generation of abstract user interfaces. Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems (EICS '12), New York, USA, 2012, ISBN: 978-1-4503-1168-7, Pages: 101-110.

[UIDL, 07] UIDL, User Interface Description Languages, 2007, http://www.uidl.net.

[U.S. HHS, 86] Section 508, 1986, http://www.hhs.gov/web/508/index.html.

[VideoJS, 13] VideoJS, 2013, http://videojs.com/.

[VLC, 13] VLC, VideoLan Organization, 2013, http://www.videolan.org/vlc/index.html.

[Wieringa, 10] Wieringa R.J. (2010). Relevance and problem choice in design science. In: Global Perspectives on Design Science Research (DESRIST). 5th International Conference, 4-5 June, 2010, St. Gallen. pp. 61-76. LNCS 6105. Springer Verlag. 2010. ISBN 978-3-642-13334-3.

[Wilson, 96] Wilson, S., Johnson, P. (1996). Bridging the generation gap: From work tasks to user interface designs. In Computer-Aided Design of User Interfaces (CADUI), Namur University, 1996, Pages: 77-94.

[W3C, 94] UAAG, User Agent Accessibility Guidelines, 1994, http://www.w3.org/WAI/intro/uaag.php.

[W3C, 94b] WAI, Web Accessibility Initiative, 1994, http://www.w3.org/WAI/.

[W3C, 94c] WCAG, Web Content Accessibility Guidelines, 1994, http://www.w3.org/WAI/intro/wcag.php.

[W3C, 08] Guideline 1.2: Provide alternatives for time-based media, 2008, http://www.w3.org/TR/UNDERSTANDING-WCAG20/media-equiv.html.

[W3C, 08b] WCAG 2.0, Web Content Accessibility Guidelines 2.0, 2008, http://www.w3.org/TR/WCAG20/.

[W3C, 09] UsiXML, 2009, http://www.w3.org/2005/Incubator/model-based-ui/wiki/UsiXML.

[W3C, 11] WAI_ARIA, 2011, http://www.w3.org/WAI/intro/aria.

[W3C, 11b] UAAG 2.0, User Agent Accessibility Guidelines 2.0, 2011, http://www.w3.org/TR/UAAG20/.

[W3C, 13] HTML5 Nightly, A vocabulary and associated APIs for HTML and XHTML, 2013, http://www.w3.org/html/wg/drafts/html/master/Overview.html.

[Youtube, 13] ACCESS: Youtube, 2013, http://accessyoutube.org.uk/.

[Youtube, 13b] Youtube, 2013, www.youtube.com.