An approach to the integration of accessibility requirements into a user interface development method

Raúl Miñón [a,*], Lourdes Moreno [b,1], Paloma Martínez [b], Julio Abascal [a]

[a] *Laboratory of HCI for Special Needs, University of the Basque Country/Euskal Herriko Unibertsitatea, Manuel Lardizabal 1, 20018 Donostia, Spain*

[b] *Computer Department, Universidad Carlos III de Madrid, Av. Universidad 30, 28911 Leganes, Spain*

Abstract: It is a legal requirement in many countries to ensure accessibility for Web applications. Although there are numerous regulations and standards regarding accessibility in the de-velopment of user interfaces, accessibility is nevertheless quite difficult to achieve, let alone to maintain at a high level of quality throughout the development process. This difficulty is due to diverse factors including, but not limited to, the lack of proper development meth-ods, authoring tools and accessibility training for user interface development professionals. In an attempt to offer a solution to these difficulties, this paper proposes a methodological approach for the integration of accessibility requirements into a user interface develop-ment method based on User Interface Description Language (UIDL) or, more specifically, on the USer Interface eXtensible Markup Language (UsiXML) framework. The proposed strat-egy involves the integration of accessibility requirements into design primitives of the user interface development method. This paper focuses on accessibility requirements related to navigation in the Task Model, Abstract User Interface Model and Transformation Model. The application of the approach shown for the SPA4USXML tool also includes a proof of concept and validation of the proposal. The study shows that accessibility requirements included at the design stage in the modelling of user interfaces can be systematized through mech-anisms such as new transformation rules and the use of support tools such as SPA4USXML.

Keywords: Accessibility, User interface development method Model driven development, User Interface Description Language.

Abbreviations: AODA, Accessibility for Ontarians with Disabilities Act; ARIA, Accessible Rich Internet Applications; AUI, Abstract User Interface; AWA, Accessibility Web Applications; BITV 2, Barrierefreie Informationstechnik-Verordnung; ISO, International Organization for Standardization; MDE, Model Driven Engineering; RGAA, Référentiel Général d'Accessibilité pour les Administrations; SPA4USXML, Service Provider Annotations for Ubiquitous Services through UsiXML; UIDL, User Interface Description Language; UsiXML, User Interface eXtensible Markup Language; W3C, World Wide Web Consortium; WAI, Web Accessibility Initiative; WCAG, Web Content Accessibility Guidelines; WAfA, Web Authoring for Accessibility.

* Corresponding author. Tel.: +34 943015113.
E-mail addresses: raul.minon@ehu.es (R. Miñón), lmoreno@inf.uc3m.es (L. Moreno), paloma.martinez@inf.uc3m.es (P. Martínez), julio.abascal@ehu.es (J. Abascal).
1 Tel.: +34 91 6249988.

1. Introduction

In the Information society, people access services and products through user interfaces. However, not all people can access them, due to the fact that there are accessibility barriers. It is no longer the case that individuals with disabilities are the only users affected by accessibility barriers. Indeed, nowadays, numerous users experience problems when using the Web, mobile phones, etc., due to different types of disabilities, functional limitations, technological limitations or environmental factors. It can be said that the digital gap is, in fact, growing. One important demographic trend to study when considering the importance of accessibility is the ageing of the general population. For years, it has been predicted that we will be an ageing society, however with an active involvement of users in the use of ICT, which is confirmed by research data [1]. As the population gets older, there is an increased likelihood of people suffering from some disability, whether temporary or permanent.

There are major initiatives, at various levels, aimed at the legislation, standardization and regulation of accessibility guidelines and standards for the design of universally accessible interfaces. However, these directives are often not followed [2,3].

In addition, accessibility is quite difficult to achieve, let alone to maintain at a high level of quality. This difficulty is due to diverse factors, including the lack of proper development methods, authoring tools and accessibility training for professionals [4,5]. In an attempt to offer a solution to these difficulties, this paper proposes a methodological approach to integrate accessibility requirements for User Interface Description Languages (UIDLs) [6] to assist and guide professionals in the development of accessible user interfaces.

In this approach, accessibility is understood as conformance to the WCAG guidelines and other accessibility-related standards, with the ultimate goal being the definition of specific accessibility requirements to be included in the different steps of a user interface development method to satisfy these accessibility guidelines. This paper also exemplifies the method with the UIDL language called UsiXML [7] and shows an extension made to the SPA4USXML tool [8] to achieve our objectives.

The rest of this paper is organized as follows: Section 2 discusses related work; the methodological approach is presented in Section 3; Section 4 describes the approach for UsiXML; the extension made to SPA4USXML is described and exemplified in Section 5; Section 6 offers a discussion and, finally, Section 7 presents the conclusions and proposes areas for further research.

2. Related works

There are numerous initiatives, directives, pieces of legislation and standards aimed at achieving a high standard of accessibility, which identify problems and suggest new, accessible designs. As regards accessibility standards, special mention must be made of the World Wide Web Consortium (W3C) and the Web Accessibility Initiative (WAI) [9]. The Web Content Accessibility Guidelines (WCAG) [10] are the most important component of the WAI, are considered the official standard in the European Union and are referenced in most legislation worldwide. There are also a number of other important initiatives, such as BITV 2 [11], RGAA [12], AODA [13] and the Section 508 (29 U.S. Code §794d) [14] technical standards, some of which are specifically related to Web accessibility. Although less extensive, these standards are nevertheless very similar to WCAG and studies can be found that compare the former and latter [15].

WCAG 2.0, the current version of the W3C accessibility guidelines [16], was developed to be applied not only to existing W3C technologies but also to other current and emerging technologies. Furthermore, additional guidelines and techniques such as WAI-ARIA have been created to address accessibility in currently used rich Internet applications (RIAs) such as Flash and AJAX [17].

Although WCAG 2.0 guidelines – which are on their way to becoming an ISO standard [18] – are focused on Web content, they nevertheless include accessibility requirements that are applicable to user interfaces. In the area of interactive systems, one must also consider standards such as ISO 9241: Ergonomics of human–system interaction – focusing in particular on parts 129 (Guidance on software individualization), 143 (Form-based dialogues), 151 (Guidance on World Wide Web user interfaces) and 171 (Guidance on software accessibility) [19] – among others. The present study takes into consideration both WCAG 2.0 and ISO 9241-171:2008.

Concerning adaptability and access as a characteristic of a user interface, it is important to mention the series of ISO/IEC 24751 standards generated from the Learner Information Package Accessibility for LIP Specification (ACCLIP) for e-learning systems environments [20].

The majority of relevant studies based on WCAG 1.0 and 2.0 deal with the evaluation of accessibility, whether it is the quantitative measurement of accessibility [21] or qualitative evaluation with the necessary tools [22,23]. Although some authors provide instructions for the use of these guidelines [24] or propose simple frameworks in an attempt to maximize the benefits gained from the implementation of these guidelines [25], very few address the question of how to incorporate accessibility requirements in the software development process [26]. Where this does occur, individual suggestions are generally offered as opposed to comprehensive solutions [27]. There has also been a lack of technological support for authors in the different activities of the development process [28] taking accessibility into account.

Some studies have been found that propose the application of WCAG to Semantic Web technologies such as ontologies, XML, XMLSchema and OWL [29,30]. Among the relevant studies found in the area of Web engineering methods, some propose the use of patterns on a method to include certain accessibility requirements in user interfaces [31] and others focus on a user-friendly approach that uses well-designed access patterns to provide efficient access in the navigation and retrieval of information for a website [32].

Nevertheless, very few articles have been found that directly address the question of how to model accessibility according to WCAG, one of which has an aspect-oriented approach [26]. Thus, none of the previous approaches fully satisfy the accessibility standards. However, one interesting attempt meriting particular mention is the Dante approach, integrating the Web Authoring for Accessibility (WAfA) ontology [33,34] for the visually impaired into the WSDM method [30]. In this study, the Accessibility Web Applications (AWA) approach has been adopted. This approach offers methodological support for the development of accessible Web applications from the perspective of Web engineering and includes conceptual elements capable of abstracting the accessibility requirements proposed in WCAG 2.0 [35].

There are a number of works on user models for designing inclusive products. EU-funded projects include VICON [36], VERITAS [37], GUIDE [38] and MyUI [39]. These projects present generic interoperable user models that describe the relevant characteristics of users interacting with products and user interfaces with physical, cognitive and sensory attributes, habits, preferences and accessibility capabilities.

With regard to the use of User Interface Description Languages (UIDLs) to support the development of accessible user interfaces, there are several tools that generate graphically diverse types of UIDLs; e.g. GUMMY [40] is a generic multi-platform GUI builder designed to create user interfaces (using an incremental design process) for a wide range of computing platforms and Liquid Apps [41] facilitates the graphical editing of abstract user interfaces. Both of these are compliant with the UIML syntax [42]. Some tools have also been developed to generate code compliant with the UsiXML syntax [7]: GUILayout++ [43] is a tool for designing user-centric interfaces through an iterative process based on prototyping and evaluation; GRAFIXML [44] enables the simultaneous design and generation of UIs for different contexts of use; and IDEALXML [45] is a tool that allows the editing of task models, abstract user interface models, domain models, and mapping models. However, none of the works found explicitly considers accessibility requirements, which are understood as conformance to the WCAG guidelines. Therefore, in this approach the SPA4USXML tool [8] is extended to systematize the inclusion of these accessibility requirements. This tool was enriched with additional resources that follow a model with an Access For All approach [20] to ensure accessibility and adaptability. In this work the approach is different in that the integration of the accessibility requirements involves the integration of design primitives into meta-models and models.

3. Integrating accessibility requirements into a user interface development method

In this section, the accessibility requirements obtained from accessibility standards are presented. As mentioned earlier, the present work focuses on describing those requirements related to structure and navigation. An abstraction is made, taking into account the official standard documentation, in order to determine the accessibility requirements to be integrated into a user interface development method. The objective is to identify which requirements should be present in the design primitives of the user interface method. If these cannot be located, mechanisms should be established in a strategy to extend the semantic richness of the method until the desired accessibility is achieved. Such mechanisms can be diverse and there are different views on how they should be constructed. This section offers a general view of how this integration into the User Interface Description Language (UIDL) can be carried out and later sections describe it in greater detail for the UsiXML framework.

3.1. Requirements from accessibility standards

WCAG 2.0 is a widely accepted set of standards for the definition of Web accessibility. This Web standard can be extrapolated to other software standards, since many of its requirements apply to user interfaces in interactive systems software; e.g. ISO 9241-171:2008 (Guidance on software accessibility) provides specifications for the design of accessible software and has been taken into account here due to the possibility of using it for the specification, design, development, evaluation and procurement of software platforms and software applications. Therefore, in this paper, "accessibility" should be understood as conformance to both WCAG 2.0 and ISO 9241-171:2008. Indeed, all decisions regarding the type of accessibility requirements to be considered in this study have been made with respect to both of these documents.

WCAG 2.0 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also often make your Web content more useable to users in general [16].

WCAG 2.0 is organized into separate layers differentiated by the specificity of the guidance offered for the design and implementation of accessible websites. The most general of these layers is represented by four fundamental principles (perceivable, operable, understandable, and robust), which, in turn, are articulated and supported by a more specific layer including a total of twelve particular guidelines. The guidelines are not testable but they provide success criteria that are testable. For each success criterion in the WCAG 2.0 document, there is documentation with a variety of techniques [46].

The techniques are classified according to the technology used. These include general (identified with the code "G"), followed by HTML and XHTML (identified with the code "H"), CSS (identified with the code "C"), and ARIA (identified with the code "ARIA"). This paper uses a general type of technique to achieve independence from the end user interface technology.

Likewise, ISO 9241-171:2008 covers issues associated with designing accessible software for people with the widest range of physical, sensory and cognitive abilities, including those who are temporarily disabled, and the elderly. It is applicable to the accessibility of interactive systems. While it does not cover the behaviour of, or requirements for, assistive technologies (including assistive software), it addresses the use of assistive technologies as an integrated component of interactive systems. The standard is composed of 93 requirements grouped into 10 categories. Within these categories, each requirement is further grouped according to its priority, such that WCAG 2.0 are grouped according to priority levels 1, 2 and 3. In each requirement, an indication is given regarding whether it is to be applied in the operating system, in applications or in both.

3.2. Abstracting accessibility requirements

As explained in earlier sections, the objective of the approach presented is to provide methodological support to facilitate the development of user interfaces conforming to WCAG and ISO 9241-129:171. This objective is achieved by using a specific strategy — the integration of accessibility requirements into the meta-models and models of the user interface development method.

The first step in this strategy is to distinguish by abstraction which requirements can be integrated for modelling. Conceptual abstractions of WCAG were used to include accessibility. The results of these abstractions were derived from an analysis of the semantics of official WCAG documentation. For this proposal, the Accessibility for Web Applications (AWA) approach was applied [35] to abstract accessibility requirements described in WCAG. The requirement classification obtained distinguishes between navigation, content, presentation and user interaction requirements. These requirements are represented in a meta-model in MOF with constraints in the OCL language (see Fig. 1).

This work describes the integration of WCAG, focusing on requirements related to navigational issues in engineering. The guideline related to accessibility requirements for navigation is Guideline 2.4: "Provide ways to help users navigate, find content and determine where they are" with success criteria including:

- *A mechanism is available to bypass blocks of content that are repeated on multiple Web pages* (Success Criterion 2.4.1); *More than one way is available to locate a Web page within a set of Web pages* (Success Criterion 2.4.5); and *Information about the user's location within a set of Web pages is available* (Success Criterion 2.4.8). These success criteria indicate that different navigation resources must be provided to the user. From these success criteria is obtained the "Navigation resources" requirement to locate the content, provide navigation mechanisms (e.g. breadcrumb trail, site map and navigation bars) and provide the current location in the application to the users.
- *A mechanism is available to allow the purpose of each link to be identified from link text alone or from the link context (Success Criteria 2.4.4 and 2.4.9*). These success criteria indicate that the purpose of each link and its labels must be clear to the user. The "Good navigation" requirement is obtained from the analysis of these success criteria.
- *The information contained in Web pages must be organized into specific components (Information, structure, and relationships) and descriptive labels are available to identify specific components within the content (Success Criteria 1.3.1, 2.4.2, 2.4.6 and 2.4.10).* These success criteria indicate that the contents must be logically organized using the heading and levels elements. In addition, windows must have titles that describe their topic. The "Structural mark-up" requirement is obtained from the analysis of these success criteria.

Therefore, from the analysis of this guideline, the following accessibility requirements must be addressed: *Navigation resources, Good navigation* and *Structural mark-up*. Table 1 shows WCAG 2.0 Success Criteria and some techniques supported with each of these requirements.

These requirements are important to provide accessibility to ICTs; for instance, the requirements included in "Structural mark-up" are essential for navigation of blind users who access ICT through a screen reader. That is because the screen reader accesses content through the headers and their semantic labelling. Besides, if it complies with the requirements included in "Good navigation", the blind users can choose which target to link, because the link text provides them the purpose of the link. Finally, if it fulfils the accessibility requirements included in "Navigation resources" for users with visual impairments they may find it easier to navigate to the correct part of the site by using a search, rather than scrolling through a large navigation bar using a screen magnifier or screen reader. A person with cognitive disabilities may prefer a site map that provides an overview of the site rather than reading and traversing through several Web pages. Individuals with cognitive limitations may find it easier to use search features than to use a hierarchical navigation scheme that may be difficult to understand.

As indicated in Table 1, when considering which navigation-centred accessibility requirements from ISO 9241-129:171 to integrate into the design primitives of a user interface development method, only two – 10.5.1 (*Provide meaningful window titles*) and 10.5.2 (*Provide unique window titles*) – were chosen and have been identified within the "Structural mark-up" requirement. This specification includes software requirements related to input navigation mechanisms such as "Facilitate

Table 1
Mapping between WCAG 2.0, ISO 9241-171:2008 and Techniques for WCAG 2.0.

Abstraction AWA approach	WCAG 2.0 SUCCESS CRITERIA (Level)		ISO 9241-171:2008	General techniques for WCAG 2.0 (used in this approach)
NAVIGATION RESOURCES	2.4.1	Bypass Blocks (A)		G123: Adding a link at the beginning of a block of repeated content to go to the end of the block
	2.4.5	Multiple Ways (AA)		G64: Providing a Table of Contents G63: Providing a site map
	2.4.8	Location (AAA)		G65: Providing a breadcrumb trail
GOOD NAVIGATION	2.4.4	Link Purpose (In Context) (A)		G91: Providing link text that describes the purpose of a link
	2.4.9	Link Purpose (Link Only) (AAA)		G91: Providing link text that describes the purpose of a link
STRUCTURAL MARK-UP	2.4.2.	Page Titled (A)	10.5.1 10.5.2	G88: Providing descriptive titles for Web pages
	1.3.1	Info and Relationships (A)		G115: Using semantic elements to mark up structure
	2.4.6	Headings and Labels (AA)	10.5.1 10.5.2	G130: Providing descriptive headings G131: Providing descriptive labels
	2.4.10	Section Headings (AAA)		G141: Organizing a page using headings

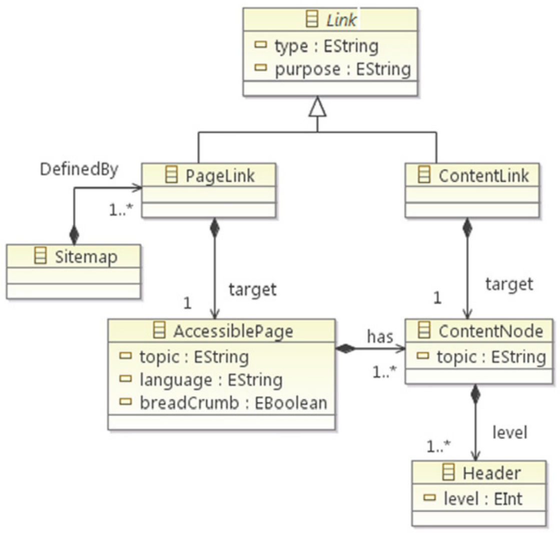
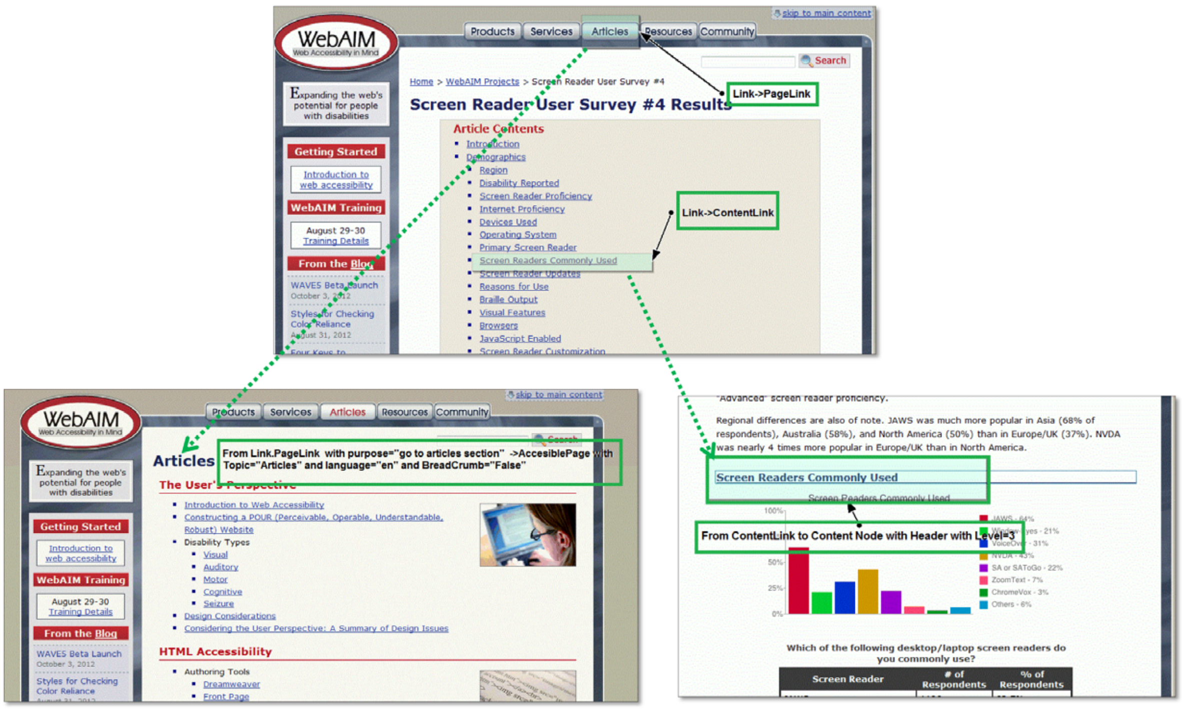


Fig. 1. AWA_Meta-model corresponding to navigation requirements.

long list and menu navigation" and "Enable non-pointer to navigation windows". These were not considered since their integration should be carried out in specific, technology-dependent development phases, rather than with design primitives of the method. Moreover, other requirements found concerning how design tasks are carried out, such as "*Optimize the number of steps required for any task*" and "*Provide Undo and/or Confirm functionality*", were not considered due to their being domain-dependent and guidelines to be followed to design tasks.

Fig. 1 shows the proposed meta-model for the collection of the navigation-related requirements according to WCAG in the Web environment. In order to support the "Structural mark-up" requirement, the AccessiblePage entity represents a Web page. The topic attribute describes the main content of the page having a correspondence with the text to include in the Title element in the final user interface (Web page, windows), required to meet WCAG 2.0 Success Criterion 2.4.2.

An AccessiblePage is made up of several ContentNodes, which are also described using the Topic attribute and used to describe the header elements to meet WCAG 2.0 Success Criterion 1.3.1. Additionally, if the Topic attribute provides descriptive headings technique G130 can be applied and WCAG 2.0 Success Criterion 2.4.6 can be met. The ContentNodes are structured into different levels using the Header entity. WCAG 2.0 Success Criterion 2.4.10 indicates that the headers organize the content according to technique G141. The level association between ContentNode and Header establishes the


Link->PageLink

Link->ContentLink

From Link.PageLink with purpose="go to articles section" ->AccesiblePage with Topic="Articles" and language="en" and BreadCrumb="False"

From ContentLink to Content Node with Header with Level=3

WebAIM
Products
Services
Articles
Resources
Community
skip to main content
Home > WebAIM Projects > Screen Reader User Survey #4
Screen Reader User Survey #4 Results
Article Contents
Screen Readers Commonly Used
Articles
The User's Perspective
HTML Accessibility
Screen Readers Commonly Used
Which of the following desktop/laptop screen readers do you commonly use?

Fig. 2. Screenshots of example to illustrate meta-model with accessibility requirements.

ordering sequence between different headers. The "Good navigation" requirement is supported by the Purpose attribute of the Link modelling entity. This attribute has a correspondence with the text of the link purpose in the final Web code (as described in 13.1 of WCAG 1.0 and 2.4.4 and 2.4.9 of WCAG 2.0) using technique G91, indicating that the link text must describe the purpose of a link.

Finally, as regards the "Navigation resources" requirement for the location of Web content, WCAG 2.0 Success Criterion 2.4.5 indicates that more than one way is required to locate a Web page, using two or more of the many techniques proposed by WCAG 2.0: (a) providing a site map, according to technique G63, represented by the Sitemap entity that is made up of the different PageLinks; and (b) the Boolean breadcrumb attribute of the AccessiblePage entity has a correspondence with the breadcrumb trail navigation indicated in technique G65 to meet WCAG 2.0 Success Criterion 2.4.8. If this attribute is set to true, the current location of the user must be shown in the Web page.

The following example is provided to illustrate the meta-model and its accessibility requirements related to navigation. This example consists of an accessible user interface for Web environments, such as the "WebAIM"[2] web site. This web site is very well known in the field of accessibility, it provides useful and relevant accessibility resources. As shown in Fig. 2, there are links in a web page. These links have a purpose and can be of two types: the ContentLink type, where the target leads to an area of the same Web page (ContentNode); and other target, where the link goes to another Web page (PageLink). In Fig. 2, these links would be, correspondingly: "Commonly Used Screen Readers" and "Articles". The target Web page has to be an accessible Web page (AccessiblePage), which must comply with the following accessibility requirements: provide navigation resources as breadcrumb trail, Web document language and a topic representing the title of the page, as indicated in Fig. 2. When the target leads to an area of the same Web page (ContentNode), this area has a topic and a header. This header has a level (Header.level) as indicated in Fig. 2.

Having distinguished the accessibility requirements to be integrated into the design primitives of a method, the next section describes which approaches can be followed to perform the integration.

3.3. An approach to integrate accessibility requirements into a user interface development method

In order to guarantee accessibility and satisfy accessibility requirements, the requirements must be incorporated at different points of the user interface development method. This goal is referred to here as a quality requirement through the systematization of accessibility for each requirement. Therefore, the systematization strategy to be followed is the integration of specific accessibility requirements into the design primitives of the user interface development method. The present paper focuses on the study methods that use User Interface Description Languages (UIDLs). As Guerrero-García et al.

[2] WebAIM: Web Accessibility In Mind: http://webaim.org/.

claim [6], *A UI Description Language (UIDL) consists of a high-level computer language for describing the characteristics of interest of a UI with respect to the rest of an interactive application in order for these to be used during some stages of the UI development life cycle*.

The quality of accessibility is ensured through the integration of requirements in the early stages and their subsequent monitoring during the rest of the development process. Although different possible approaches have been studied for this work, all the approaches nonetheless share the common aim that this conformance should be guaranteed as early as the modelling undertaken in the abstract design phase.

A UIDL can include various meta-models to define the semantics of such a UIDL. These meta-models adhere to the principle of separation of concerns. Some examples of the meta-models are: context of use, task, domain, abstract user interface, concrete user interface and transformation. These meta-models are not used concurrently and are manipulated during the different steps of a user interface development method. In order to support this type of development method, software is required throughout the user interface development life cycle to create, edit, and check models that are compliant with these meta-models and to produce user interfaces using these methods.

The first of the two main approaches found involves the extension of the UIDL meta-models. As a result of this extension, conceptual models can be extended with the semantic expressivity required for accessibility, such that the result conforms both to the accessibility requirements, and to UIDL tools. The disadvantage of this approach is that it is impossible to use the resources and tools offered by UIDL frameworks, since these fall out of compliance with the new extended meta-models.

The second approach would be more viable, since it does not involve the extension of the meta-model. Instead, with this approach an attempt is made to extract the semantic expressivity required for accessibility from the UIDL meta-models. This semantic expressivity required for accessibility corresponds to the inclusion of accessibility requirements related to the navigation view of the previous section. The strategy to follow is shown below (see Fig. 3). In general terms, it can be described as follows. Initially, it is necessary to check whether the accessibility requirement is applicable in this step of the user interface generation process; if not, the requirement must be evaluated in the following step. If the expressivity of the accessibility required is found in the meta-model, we must ensure that it is continued through the subsequent steps in the development process. In order to achieve this, the transformation model, including its model-to-model rules (these rules are denominated *core rules*; see Section 4.1 for greater detail), should be revised. If the expressivity of the accessibility required is not included in the meta-model through the *core rules*, other mechanisms must be implemented to include it. These mechanisms are to edit the transformation model with the inclusion of new rules that incorporate these requirements (using native design primitives included in the method meta-model). Finally, if the creation of new rules is not sufficient to obtain all the expressivity of the accessibility required, new mechanisms must be defined (such as the addition of an editor tool to insert the semantic accessibility that the method lacks into the development process of the method). In addition, the "RL" variable of this algorithm provides a list of the accessibility requirements that must be implemented through transformation rules and that have to be considered in the subsequent steps of the transformation process. The "RV" variable provides the set of accessibility requirements that must be included in the *core rules* and have to be validated to guarantee accessibility in the final user interface.

In this study the authors have selected the second integration strategy in order to comply with the UIDL requirements. This strategy is exemplified with specific models of the UsiXML framework. Consequently, and as a necessary first step, a preliminary study was undertaken to determine how the UIDL meta-model should conform to the accessibility requirements.

4. Approach for USer Interface eXtensible Markup Language (UsiXML)

This section focuses on describing the approach and strategy previously seen to integrate accessibility requirements into the UsiXML Abstract User Interface (AUI) meta-model. The steps of the second strategy defined in Section 3.3 were followed to carry out this study, extracting expressivity of the accessibility required included in the UsiXML task meta-model and transferring these semantics to the AUI meta-model. This study is limited to the accessibility requirements related to navigation seen in Section 3.2.

In order to provide a better understanding of this approach, the following subsections provide a general view of the UsiXML reference framework; subsequently, the application of the approach to the UsiXML reference framework is presented; and, finally, the transformation model designed, based on the results of the previous step, is illustrated.

4.1. UsiXML in a nutshell

UsiXML 2.0 (which stands for USer Interface eXtensible Markup Language) [7] is an XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces, Graphical User Interfaces, Auditory User Interfaces, and Multimodal User Interfaces. UsiXML is uniformly used throughout the different steps of an MDE-compliant development life cycle to store the models involved in the various processes [47]. The MDE-compliant approach for developing UIs is decomposed into four major steps that result from the Cameleon reference framework [47]: *Task & Domain (Computing Independent Model in MDE), Abstract User Interface (Platform Independent Model in MDE), Concrete User Interface (Platform Specific Model in MDE)* and *Final User Interface (code level in MDE)*. The meta-models (currently) defined

```
$r_{i..n}$: Accessibility requirements set
M: User Interface Development Method
RL: Set of accessibility requirements included in this step that must be
considered in the following steps of the process
RV: Set of accessibility requirements that must be validated in the core
rules

FOR i=0 TO i=n DO
   IF (ri "is applicable in" M_metamodel) THEN
      IF ($r_i$ "is included in" M_metamodel) THEN
         IF NOT ($r_i$ "is transferred in next step of" M_process) THEN
               ("Create new rules that transfer $r_i$" in M_TransfRules)
               RL.add($r_i$)
         ELSE
               ("$r_i$ is implemented by core rules")
               RV.add($r_i$)
         ENDIF
      ELSE
         ("Execute additional mechanism to include semantics of ri and
         the correspondent rule to transfer that semantic")
         RL.add($r_i$)
      ENDIF
   ENDIF
ENDFOR
```

Fig. 3. Strategy to integrate accessibility requirements into the User Interface Development Method.

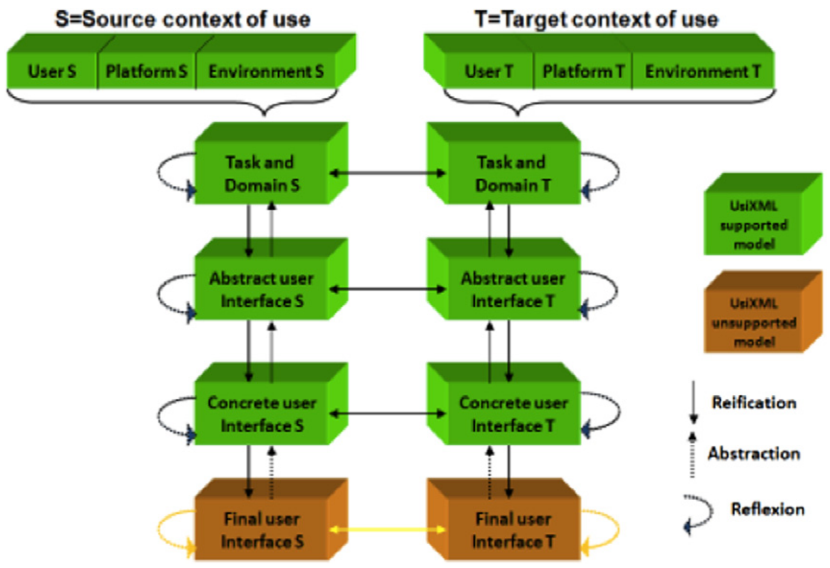


Fig. 4. UsiXML reference framework (source: UsiXML in W3C Incubator Activity [48]).

in the UsiXML project are the Domain, Task, Context, Abstract User Interface, Concrete User Interface, Transformation, and Workflow. UsiXML also provides rules to indicate how to transform source models into target models. These rules are referred to as *core rules* in this article. Different transformation paths are possible to support subsequent transformation processes. Fig. 4 was extracted from the W3C, Model-based user interface incubator group [48] to illustrate the steps of the UsiXML process and the possible transformation paths.

4.2. Applying the methodological approach to UsiXML

In this subsection the algorithm provided in Fig. 3 is applied to the success criteria for the requirements abstracted in Section 3.2, in relation to the transformation between the UsiXML Task meta-model and the UsiXML AUI meta-model. Table 2 shows the results of this process.

Concerning the "Navigation resources" requirement, the semantics to cover the *ByPass Blocks* success criterion (2.4.1) are included in the task meta-model, because this meta-model provides a centrality attribute of the TaskDecoration entity which is useful to identify the main content. However, there are no mechanisms in the *core rules* to access the main content of an AUI. The task meta-model relationships allow the semantics to run a mechanism to provide the user with *multiple*

Table 2
Results of applying the methodological approach to transformation between the UsiXML Task meta-model and the UsiXML AUI meta-model.

ABSTRACTION OF WCAG 2.0 REQUIREMENTS	ri	Included in the meta-model		Not included in the meta-model
		and transferred to the next step	and not transferred to the next step	
NAVIGATION RESOURCES	ByPassBlocks		X	
	Multiple Ways		X	
	Location		X	
GOOD NAVIGATION	Link Purpose (In context)	X		
	Link Purpose (Link only)	X		
STRUCTURAL MARK-UP	Page Titled			X
	Info and Relationships	X		
	Headings and Labels			X
	Section Headings			X

ways (2.4.5) of guiding him/her in a set of windows and enabling him/her to *locate* (2.4.8) his/her position. These semantics are not transferred to the next step.

In order to integrate the "Good navigation" requirement, it is seen that the task meta-model relationships also allow the extraction of semantics to *assign a purpose to each link* (2.4.4) (2.4.9) from the relationships between the TaskDecoration and TaskTemporalization entities. The task meta-model and the corresponding semantics are transferred to the next step in the process by the *core rules*. This is also the situation for the *Info and Relationships* (1.3.1) success criterion of the "Structural mark-up" accessibility requirement, since the elements of the UsiXML syntax provide sufficient semantics to mark up the structure adequately from the composition relationships between the tasks of the TaskComposition of the task meta-model and are also considered in the *core rules*.

The *Page Titled* (2.4.2), *Headings and Labels* (2.4.6) and *Section Headings* (2.4.10) success criteria of the "Structural mark-up" accessibility requirement are not covered by the semantics of the task model, since in this model there was no attribute identified to fill in the label attribute of the AUI model. Therefore, an additional mechanism is necessary to provide these semantics.

In conclusion, the success criteria marked in the second and third data columns of Table 2 require specific and new rules to be included in the AUI model and need to be traced in the subsequent steps of the process. Furthermore, the success criteria marked in the second column require an additional and external mechanism to provide the semantics necessary to fulfil the accessibility requirements called *ri* in Table 2. However, those marked in the first data column are included in the *core rules* and it is necessary to validate the inclusion of these semantics in those rules.

4.3. Transformation model designed

Based on the results in Table 2, a transformation model was designed for incorporating accessibility requirements into the AUI models. This transformation model (illustrated in Fig. 5) was designed by introducing the rules identified as necessary in the second and third columns in Table 2. As previously stated, besides the rules mentioned, there are other rules (*core rules*) that perform the core transformations from the task models to the AUI model. However, these rules are not illustrated because they are not related to accessibility requirements and go beyond the scope of this paper.

As stated in Table 1, the "Navigation resources" requirement is met by satisfying the *ByPass Blocks* (2.4.1), *Multiple Ways* (2.4.5) and *Location* (2.4.8.) success criteria. Regarding the *ByPass Blocks* success criterion, a rule was implemented to generate an Abstract Navigation element targeting the main content of the window. Two different rules were implemented to meet the *Multiple Ways* success criterion: the first generates a table of contents for each window and the second generates a site map of the application. Finally, when a window participates in a process, a sequence of the tasks involved in the process is generated to satisfy the *Location* success criterion.

No rules for the "Good navigation" requirement were introduced in this method because the *core rules* transfer the semantics of the purpose of each link. Nevertheless, the *core rules* should be revised to guarantee that the requirement is satisfied.

In order to cope with the "Structural mark-up" requirement, it is necessary to provide a label for each task as external information, since the task model does not offer the semantics necessary to provide the AUI model with either a page title or headings and labels. The labels provided are stored in the label of each Abstract element according to a specific rule, allowing the *Page Titled* (2.4.2), *Headings and labels* (2.4.6) and *Section Headings* (2.4.10) success criteria to be met in the subsequent steps of the MDE transformation process. As shown in Table 2, the *Info and Relationships* (1.3.1) success criterion is considered in the *core rules*. The semantics of the regarding *Info and Relationships* (1.3.1) success criterion are found through the composition relationship between Abstract Containers and Abstract Elements in the AUI.

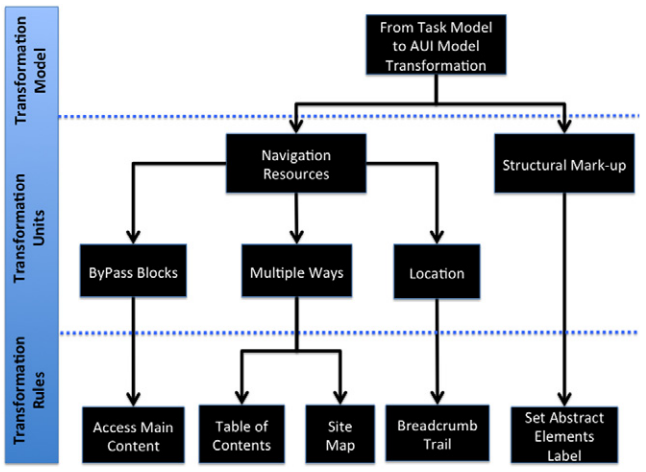


Fig. 5. Transformation Model from Task Model to AUI Model in relation to accessibility navigation requirements.

The following section provides further details of the transformation rules and some examples.

5. Applying the approach to the SPA4USXML tool of the Egoki system

This section describes, briefly: the tool used to generate AUI models; the improvements performed to cope with the accessibility requirements described in Section 3.2; and a proof of concept to illustrate these improvements.

5.1. SPA4USXML in a nutshell

SPA4USXML [8] is an authoring tool providing service designers with the functionality to generate Task and Abstract User Interface (AUI) models that are compliant with the USIXML syntax from a service description. In addition, it makes it possible to relate different interaction resources (texts, images, videos, audios, etc.) to the Abstract Interaction Elements (AIE) of the AUI, generating a Resource Model. This tool was developed to provide the Egoki system [49] with the models necessary to automatically generate adapted user interfaces to allow people with special needs to interact with ubiquitous services.

5.2. Extension of the tool to support accessibility requirements

In this approach, the SPA4USXML tool was extended with the aim of supporting the transformation rules identified in Section 4.3 to enable the service designer to include accessibility without being an accessibility expert. Specifically, the transformation from a Task model to an Abstract User Interface model was enriched by including the accessibility rules (developed with the XSLT technology) illustrated in Fig. 5. These rules and other required improvements are described in detail below.

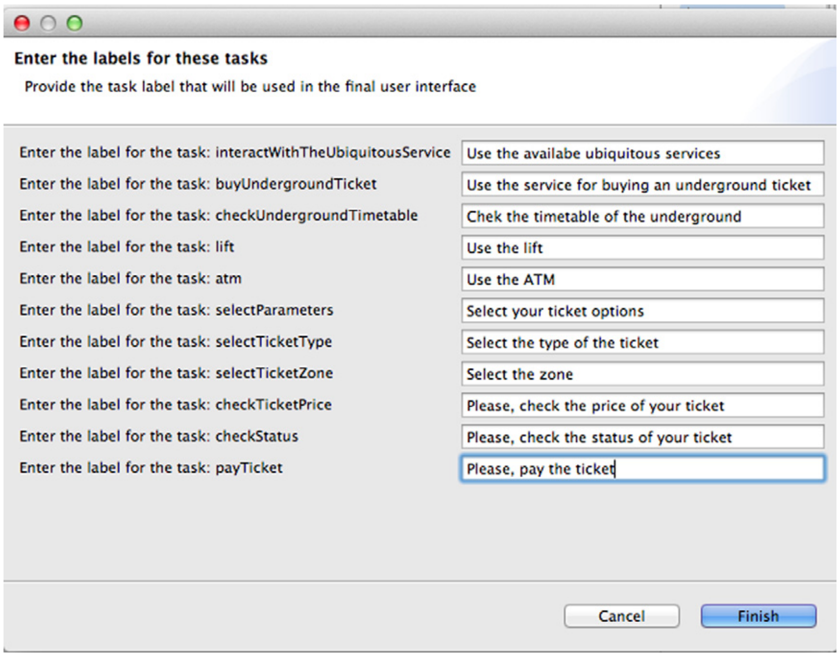
5.2.1. Select level granularity wizard

Although this improvement does not directly cover any specific accessibility requirement, it was essential to perform an extension to allow service designers to select the *task level granularity*. Tasks matching the level selected will allocate the container with the highest level. Therefore, these tasks will be rendered as windows in a desktop application and as web pages in a Web application.

The level of a task is gathered from the *TaskComposition* element, which denotes the relationships between parent nodes and child nodes. In order to select this granularity level a wizard is provided to the user. In this approach all the tasks having a level lower than the granularity level are considered as abstract elements, except the leaf nodes. In addition, due to the nature of the enhancement the generation of multiple Abstract User Interfaces from a task model is permitted.

5.2.2. Access to main content rule

This transformation rule generates, at the beginning of each window, an abstract navigation element targeting the container that allocates the element with the highest centrality. It is supposed that the element with the highest centrality should represent the main content of the window. The window is thus provided with a mechanism to *bypass blocks* and get direct access to the main content.



Enter the labels for these tasks

Provide the task label that will be used in the final user interface

Enter the label for the task: interactWithTheUbiquitousService	Use the availabe ubiquitous services
Enter the label for the task: buyUndergroundTicket	Use the service for buying an underground ticket
Enter the label for the task: checkUndergroundTimetable	Chek the timetable of the underground
Enter the label for the task: lift	Use the lift
Enter the label for the task: atm	Use the ATM
Enter the label for the task: selectParameters	Select your ticket options
Enter the label for the task: selectTicketType	Select the type of the ticket
Enter the label for the task: selectTicketZone	Select the zone
Enter the label for the task: checkTicketPrice	Please, check the price of your ticket
Enter the label for the task: checkStatus	Please, check the status of your ticket
Enter the label for the task: payTicket	Please, pay the ticket

Fig. 6. Wizard implemented for providing the labels for the abstract elements.

5.2.3. *Multiple ways rules*

Two different transformation rules were implemented to provide *Multiple Ways* to the user:

- Site Map Rule. A new abstract user interface is generated that contains abstract navigation elements targeting all the elements that are not part of the current page and are not abstract elements.
- Table of Contents Rule. In the pages generated, a container is generated with abstract navigation elements targeting all the elements available in the current page.

5.2.4. *Breadcrumb trail rule*

The *Breadcrumb Trail* transformation rule checks if there are any temporal relationships between the elements with the granularity level selected. If there are temporal relationships a new abstract container element is created with abstract navigation elements targeting the different tasks taking part in the relationship; except for the current task, which is generated as an Abstract output element.

5.2.5. *Set abstract elements label rule*

The *Page Titled* (2.4.2), *Headings and Labels* (2.4.6) and *Section Headings* (2.4.10) success criteria of the "Structural mark-up" requirement are met by providing external information to fill in the label attributes of each abstract element, since the task model does not provide this information. In the subsequent steps of the MDE process, these labels are necessary for labelling the titles, section headings and structure of the resulting final user interfaces. The rule implemented stores the provided information in the corresponding label attributes. Therefore, to avoid an extension of the elements of the Task model an assistant was implemented such that when the user triggers the transformation process, it asks the service designer for a label for each task available in the task model (see Fig. 6). With this information, the system is able to generate the abstract user interfaces with the necessary values in the label attributes.

5.3. *Proof of concept*

In order to explain how the tool generates the accessible elements according to the requirements detailed in this document, we use a ubiquitous scenario with the following services available: *Buy Underground Ticket service, Check Underground Timetable, Lift service* and *ATM service* (Fig. 7 shows the task decomposition of the available services. Due to the nature of the example, only the *Buy Underground Ticket* service was decomposed in detail).

Imagine that a service designer wants to produce different accessible user interfaces for the services available. The designer should: import the service functionality description to the task editor of SPA4USXML; enrich the model loaded in the editor with the desired relationships between tasks (these relationships have to be modelled manually because the description of the services does not provide the necessary semantics to infer them) and with new tasks if the designer

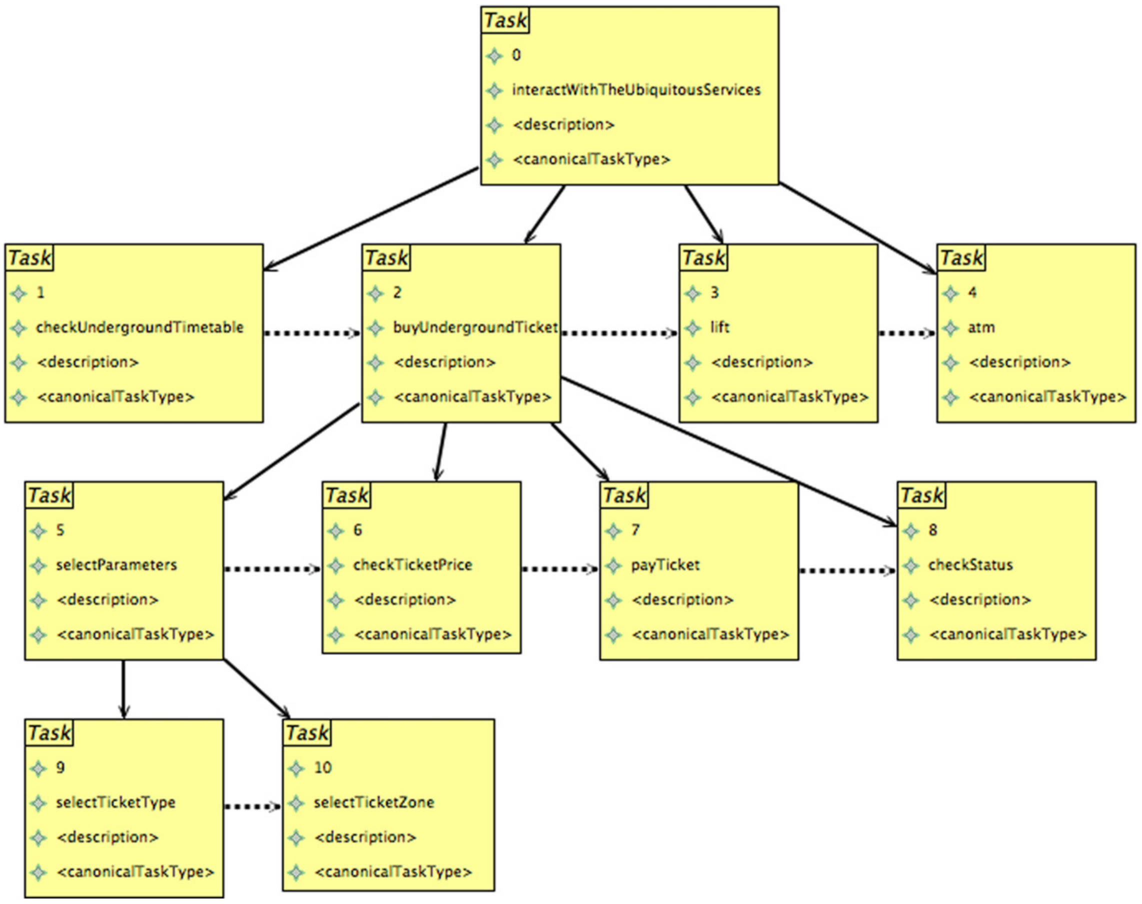


Fig. 7. Task decomposition of the ubiquitous services available.

considers it necessary; generate the necessary abstract user interfaces (AUI); enrich them with additional abstract elements and the necessary abstract listeners (the scope of this article does not consider the domain model and, therefore, the semantics necessary to automatically generate the abstract listeners cannot be inferred); and assign different resources to the resulting AUIs [8]. With this information the Egoki adaptive system is able to generate accessible user interfaces [49].

In this section, we want to illustrate the process of generating the different AUIs from the task model. Below, there are examples of the transformation rules for the accessibility requirements explained in the previous section.

Besides being described with the same palette element, each task temporalization (represented by dashed arrows) can be different depending on the type attribute. The relationships between selectParameters and checkTicketPrice, checkTicketPrice and payTicket and payTicket and checkStatus are of the enabling type; the rest of the temporal relationships are of the order-independence type.

5.3.1. *Select level granularity wizard example*

When generating the Abstract User Interfaces derived from the task model, the designer has to select a granularity level. The purpose of the granularity level is not for limiting the task model. This level is used in the transformation from task meta-models to AUI meta-models (when the task model has already been created without limitations) to provide the possibility of generating more than one AUI model (rather than just one). For example, when the final user interfaces are web pages it would be possible to create a web site rather than just a simple web page.

If the designer selects level 1, an AUI will be generated with all the elements of the task model. If he/she selects level 2, the *Interact with the Ubiquitous Services* task will be considered as an abstract element and four different AUIs will be generated: *Buy Underground Ticket, Check Underground Timetable, Lift* and *ATM*. Consequently, if he/she selects level 3, the previous tasks will be considered as abstract elements and an AUI will be generated for the *Select Parameters, Check Price, Pay for the Ticket* and *Check Ticket Status* tasks (it should be noted that if the other services had also been decomposed the system would have generated the corresponding AUIs). Finally, if level 4 is selected five AUIs will be generated, corresponding to the leaf nodes of the *Buy Underground Ticket Service*.

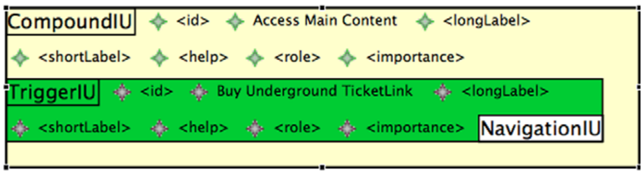


Fig. 8. Example of the container generated by the Access Main Content rule.

5.3.2. *Access to main content rule example*

Let us suppose that the service designer selects the granularity level 1 and all child nodes of the *Buy Underground Ticket* task have a centrality value of 8 and the rest of the tasks a lower value. As explained earlier, since the granularity level is 1 only an AUI with all the elements will be generated.

The first element of the AUI will be an abstract container with a navigation element targeting the parent node of the element with the highest centrality; i.e. the *Buy Underground Ticket service* element. Fig. 8 shows an example of the AUI generated.

5.3.3. *Multiple ways rules example*

Imagine that in order to illustrate the examples of the *Site Map Rule* and *Table of Contents Rule* the granularity level is 2. Four different AUIs will be generated (one for each level 2 task) and an additional AUI will be generated containing the site map. This site map consists of an abstract container allocating abstract navigation elements targeting each level 2 task. If there had been temporal relationships between the level 2 elements, the abstract navigation elements that did not correspond to the first elements of the process would not have been included in the site map. Fig. 9 displays this example.

Regarding the Table of Contents rule, in each of the generated AUIs (except that generated for the site map) a container is created with links to the elements described in the current AUI. In the AUI corresponding to the Buy Underground Ticket service the links generated will target the elements: Select Parameters, Select Ticket Type, Select Zone, Check Price, Pay for the Ticket and Check Ticket Status. Fig. 10 displays this example.

5.3.4. *Breadcrumb trail rule example*

Assuming that the level granularity is 3, four different interfaces will be generated and each will be provided with an abstract container allocating the necessary path to complete the sequence. This path is ordered according to the process and is composed of abstract navigation elements targeting the other elements of the path, except the current element (which will be displayed as an abstract output element). For example, the path for the *Check Price* element will be: *Select Parameters* link $\rightarrow$ *Check Price* output element $\rightarrow$ *Pay for the ticket* link $\rightarrow$ *Check Ticket Status* link.

5.3.5. *Set abstract elements label example*

In this example, the user has to provide a descriptive label for each task described in Fig. 7, through the wizard shown in Fig. 6. These labels will be stored in the label attribute of the corresponding abstract interaction element. For instance, for the task with the attribute name "Check Price" the label could be "Please, check the price of your ticket".

6. Discussion

One of the greatest challenges of the approach is to obtain the necessary semantics from the task models to generate the AUI models with accessibility requirements. Generally, the UsiXML task meta-model includes the necessary semantics. However, as stated in Section 5.2.5, the information necessary for meeting some success criteria is not provided. Therefore, in order to avoid the extension of the task meta-model (see Section 3.3), it is necessary to provide this information externally.

In this work, it was decided to provide the wizard shown in Fig. 6, since the user participates in the transformation process through SPA4USXML. However, for other approaches (where the user does not participate in this process) other alternatives are necessary; for example, the provision of an XML document with the necessary task labels.

7. Conclusions

The study concludes that accessibility requirements from the design phase included in user interfaces modelling can be systematized through mechanisms such as new transformation rules and the use of support tools such as SPA4USXML.

Despite it not having been a trivial process to abstract the accessibility guidelines of the different standards and to follow the strategy proposed to integrate accessibility requirements into the User Interface Development Method, these mechanisms can be included in the initial stages of the user interface development process. This paper also remarks on the importance of considering accessibility in the conceptualization of new UIDLs by applying the first approach proposed in Section 3.3. However, when the UIDL has already been conceptualized and the meta-models do not include accessibility requirements, it is better to apply the second approach proposed in Section 3.3, in order to ensure interoperability with

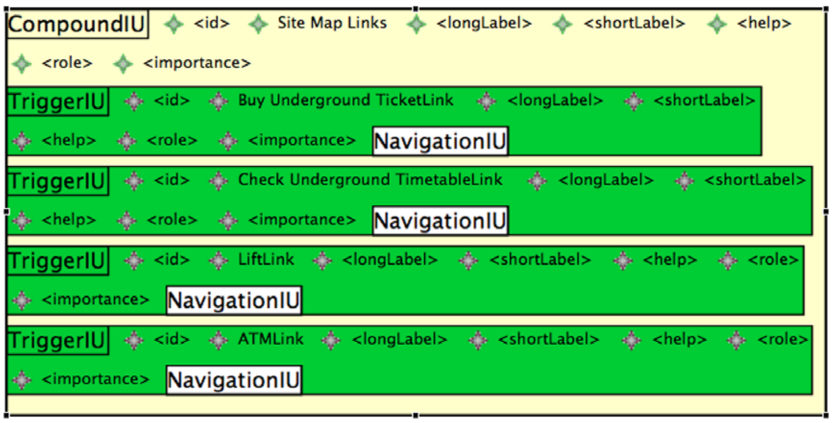

CompoundIU <id> Site Map Links <longLabel> <shortLabel> <help> <role> <importance>
TriggerIU <id> Buy Underground TicketLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> Check Underground TimetableLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> LiftLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> ATMLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU

Fig. 9. Example of the window generated by the Site Map rule.

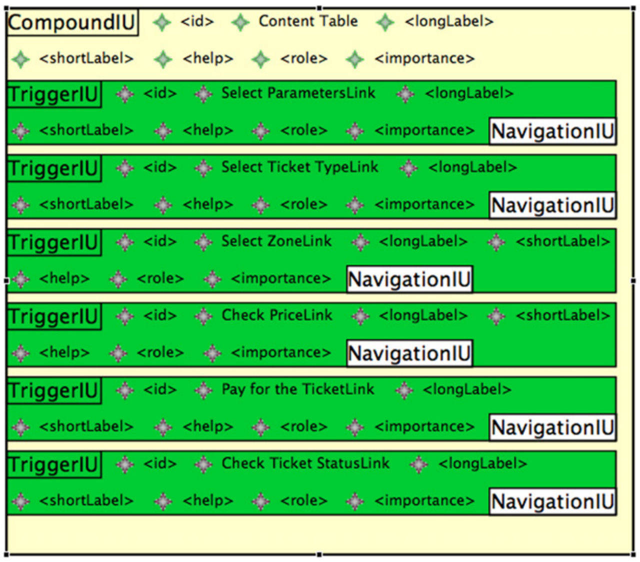

CompoundIU <id> Content Table <longLabel> <shortLabel> <help> <role> <importance>
TriggerIU <id> Select ParametersLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> Select Ticket TypeLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> Select ZoneLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> Check PriceLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> Pay for the TicketLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU
TriggerIU <id> Check Ticket StatusLink <longLabel> <shortLabel> <help> <role> <importance> NavigationIU

Fig. 10. Example of the container generated by the Table of Contents rule.

other resources and tools already built for that language. This approach has been carried out specifically within the UsiXML framework to integrate accessibility requirements related to navigation in the Abstract User Interface (AUI) meta-model. As a result, the quality of accessibility can be guaranteed by taking advantage of developments in technologies based on a User Interface Markup Language such as UsiXML.

Acknowledgements

This research work has been partly funded by the Department of Education, Universities and Research of the Basque Government and by the Research Network MAVIR (S2009/TIC-1542 (www.mavir.net), and MULTIMEDICA Project (TIN2010-20644-C03-01). In addition, RM enjoys a PhD scholarship from the Research Staff Training Programme of the Department of Education, Universities and Research of the Basque Government. We thank the INREDIS project, which has been the foundation for this work. We also want to thank the members of the ITEA Project for providing us with the documents of the UsiXML specification.

References

[1] United Nations (ONU) 2008, World Population Prospects: The 2008 Revision. http://www.un.org/esa/population/publications/wpp2008/wpp2008_text_tables.pdf, Accessed 18 July 2012.
[2] A. Olalere, J. Lazar, Accessibility of U.S. federal government home pages: Section 508 compliance and site accessibility statements, Government Information Quarterly 28 (2011) 303–309.

[3] M.G. Olsen, How Accessible is the Public European Web (2008). http://www.mortengoodwin.net/publicationfiles/how_accessible_is_the_european_web.pdf, Accessed 18 July 2012.
[4] A.P. Freire, R. Goularte, R.P. De Mattos Fortes, Techniques for developing more accessible Web applications: a survey towards a process classification, in: Proceedings of the 25th Annual ACM international Conference on Design of Communication, SIGDOC '07, El Paso, Texas, USA, October 22–24, 2007, ACM, New York, NY, 2007, pp. 162–169.
[5] J. Lazar, A. Dudley-Sponaugle, K. Greenidge, Improving Web accessibility: a study of webmaster perceptions, Computers and Human Behavior 20 (2) (2004) 269–288.
[6] J. Guerrero-García, J.M. González-Calleros, J. Vandedonckt, J. Muñoz-Arteaga, A theoretical survey of user interface description languages: preliminary results, in: Web Congress, 2009, LA-WEB '09. Latin American, pp. 36–43.
[7] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, V. López, UsiXML: a language supporting multi-path development of user interfaces, in: R. Bastide, P. Palanque, J. Roth (Eds.), Engineering Human Computer Interaction and Interactive Systems, in: LNCS, vol. 3425, Springer, Heidelberg, 2005, pp. 200–220.
[8] R. Miñón, L. Moreno, J. Abascal, A graphical tool to create user interface models for ubiquitous interaction satisfying accessibility requirements, Special Issue on Accessibility aspects in UIDLs of J. Universal Access in the Information Society (2012) http://www.sciencedirect.com/science/article/pii/S0167642313001032.
[9] W3C, Web Accessibility Initiative (WAI), 2011. http://www.w3.org/WAI/, Accessed 18 July 2012.
[10] W3C, WAI, Web Content Accessibility Guidelines (WCAG) Overview, 2012. http://www.w3.org/WAI/intro/wcag.php, Accessed 18 July 2012.
[11] Bundesministerium der Justiz (BMJ), Barrierefreie Informationstechnik-Verordnung (BITV) 2.0, (2011). http://www.gesetze-im-internet.de/bitv_2_0/index.html, Accessed 18 July 2012.
[12] Référentiel Général d'Accessibilité pour les Administrations (RGAA), Le portail de la modernisation de l'Etat. http://www.references.modernisation.gouv.fr/rgaa-accessibilite, Accessed 18 July 2012.
[13] Ontario- e-Laws, Integrated Accessibility Standards made under the Ontario Regulation 191/11 (Accessibility for Ontarians With Disabilities Act, 2005), June 7, 2011, Ontario.ca. http://www.e-laws.gov.on.ca/html/source/regs/english/2011/elaws_src_regs_r11191_e.htm, Accessed 18 July 2012.
[14] Section 508 of Rehabilitation Act. http://www.section508.gov/, Accessed 18 July 2012.
[15] Web Accessibility Policy Making: An International Perspective (Revised Edition 2012), white paper jointly researched by G3ict, The Centre for Internet & Society and The Hans Foundation. Editor: Nirmita Narasimhan. http://g3ict.org/resource_center/publications_and_reports/p/productCategory_whitepapers/subCat_0/id_150, Accessed 18 July 2012.
[16] W3C, WAI, Web Content Accessibility Guidelines (WCAG) 2.0, 2008. http://www.w3.org/TR/WCAG/, Accessed 18 July 2012.
[17] L. Moreno, P. Martínez, B. Ruíz, A. Iglesias, Toward an equal opportunity web: applications, standards, and tools that increase accessibility, IEEE Computer 44 (5) (2011) 18–26. http://www.computer.org/portal/web/csdl/doi/10.1109/MC.2010.370.
[18] ISO, ISO/IEC DIS 40500 Information technology, W3C Web Content Accessibility Guidelines (WCAG) 2.0 (TC/SC: JTC 1), 2012.
[19] ISO, ISO 9241-171:2008, Ergonomics of human-system interaction, Part 171: Guidance on software accessibility.
[20] ISO, ISO/IEC 24751-2:2008. Information technology, Individualized adaptability and accessibility in e-learning, education and training – Part 2: "Access for all" personal needs and preferences for digital delivery.
[21] M. Vigo, G. Brajnik, Automatic web accessibility metrics: where we are and where we can go, Interacting with Computers 23 (2) (2011) 137–155.
[22] M. Arrue, M. Vigo, J. Abascal, Supporting the development of accessible web applications, J. Universal Access in the Information Society 14 (16) (2008) 2699–2719.
[23] W3C, WAI, Web Accessibility Evaluation Tools: Overview (2012). http://www.w3.org/WAI/ER/tools/, Accessed 18 July 2012.
[24] D. Sloan, B. Kelly, A. Heath, H. Petrie, F. Hamilton, L. Phipps, Contextual Web accessibility — maximizing the benefit of accessibility guidelines, in: W4A: Proceedings of the 2006, International Cross-Disciplinary Workshop on Web accessibility (W4A), Edinburgh, UK, 23–24 May, ACM Press, New York, NY, USA, 2006, pp. 121–131.
[25] O. Nykänen, I. Kaikuvuo, Adapting Web accessibility guidelines to an application development process, in: International Design for All Conference, Rovaniemi, Finland, 2006.
[26] A. Martín, G. Rossi, A. Cechich, S. Gordillo, Engineering accessible Web applications. An aspect-oriented approach, World Wide Web 13 (2010) 419–440.
[27] P.R. Bohman, S. Anderson, A conceptual framework for accessibility tools to benefit users with cognitive disabilities, in: Proceedings of the 2005 international Cross-Disciplinary Workshop on Web Accessibility (W4a), W4A '05, Chiba, Japan, May 10–10, 2005, ACM, New York, NY, 88, 2005, pp. 85–89.
[28] J. Xiong, M. Winckler, An investigation of tool support for accessibility assessment throughout the development process of web sites, Journal of Web Engineering, Rinton Press 7 (4) (2008) 281–298.
[29] L. Moreno, P. Martínez, J. Contreras, R. Benjamins, Towards Accessible Semantic Web Applications, DC — International Conference on Dublin Core and Metadata Aplications, Madrid Spain, 2005, pp. 87–95.
[30] P. Plessers, S. Casteleyn, Y. Yesilada, O. De Troyer, R. Stevens, S. Harper, C. Goble, Accessibility: a Web engineering approach, in: Proceedings of the 14th international Conference on World Wide Web, WWW '05, Chiba, Japan, May 10–14, 2005, ACM, New York, NY, 2005, pp. 353–362.
[31] S. Jeschke, O. Pfeiffer, H. Vieritz, Developing accessible applications with user-centered architecture, in: International Conference on Computer and Information Science (IEEE), Portland/Oregon, May 2008.
[32] S. Ceri, M. Matera, F. Rizzo, V. Demaldé, Designing data-intensive web applications for content accessibility using web marts, Communications of the ACM 50 (4) (2007) 55–61.
[33] Y. Yesilada, S. Harper, C. Goble, R. Stevens, Dante annotation and transformation of web pages for visually impaired users, in: The Thirteenth International World Wide Web Conference, 2004.
[34] S. Harper, Y. Yesilada, Web Authoring for Accessibility (WAfA), Web Semantics: Science, Services and Agents on the World Wide Web 5 (3) (2007) 175–179. http://dx.doi.org/10.1016/j.websem.2007.05.001.
[35] L. Moreno, Ph.D These: "AWA, Methodological framework in the accessibility domain for Web application development", Advisor: Paloma Martínez Fernández, Universidad Carlos III de Madrid, Computer science department, 2010. http://www.sigaccess.org/community/theses_repository/phd/lourdes_moreno.php, Accessed 18 July 2012.
[36] Vicon project, http://vicon-project.eu/, Accessed 18 July 2012.
[37] Virtual and augmented environments and realistic user interactions to achieve embedded accessibility design, http://veritas-project.eu/, Accessed 18 July 2012.
[38] Gentle user interfaces for elderly people, http://www.guide-project.eu/, Accessed 18 July 2012.
[39] Mainstreaming accessibility through synergistic user modelling and adaptability, http://www.myui.eu/, Accessed 18 July 2012.
[40] J. Meskens, J. Vermeulen, K. Luyten, K. Coninx, Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me, in: Procs. of the working conference on Advanced Visual Interfaces, AVI 2008, Napoli, Italy, May 28–30, 2008, 2008, pp. 233–240.
[41] LiquidApps application, http://liquidapps.harmonia.com/features/, Accessed 18 July 2012.
[42] OASIS User Interface Markup Language (UIML), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uiml, Accessed 18 July 2012.
[43] F. Montero, V. López-Jaquero, Guilayout++: supporting prototype creation and quality evaluation for abstract user interface generation, in: Procs. of the 1st. USer Interface eXtensible Markup Language Workshop (UsiXML-EICS 2010). June 20, 2010, Berlin, Germany, 2010, pp. 39–44.
[44] B. Michotte, J. Vanderdonckt, GrafiXML, a multi-target user interface builder based on UsiXML, in: Procs of 4th Int. Conf. on Autonomic and Autonomous Systems ICAS'2008, IEEE Computer Society Press, Los Alamitos, 2008, pp. 15–22.

[45] F. Montero, V. López-Jaquero, IdealXML: an interaction design tool-a task-based approach to user interfaces design, in: Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006, Bucharest, 6–8 June 2006, Springer-Verlag, Berlin, 2006, pp. 245–252 (Chapter 20).
[46] W3C, Web Accessibility Initiative (WAI), Techniques for WCAG 2.0. 2012. http://www.w3.org/TR/WCAG20-TECHS/, Accessed 18 July 2012.
[47] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, J. Vanderdonckt, A unifying reference framework for multi-target user interfaces, Interacting with Computers 15 (3) (2003) 289–308.
[48] W3C, Model-based user interface incubator group, 2005. http://www.w3.org/2005/Incubator/model-based-ui/wiki/UsiXML, Accessed 18 July 2012.
[49] J. Abascal, A. Aizpurua, I. Cearreta, B. Gamecho, N. Garay-Vitoria, R. Miñón, Automatically generating tailored accessible user interfaces for ubiquitous services, in: Procs. of the 13th Int. ACM SIGACCESS Conf. on Computers and Accessibility, ASSETS 2011, pp. 187–194.