Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 67 (2015) 95 – 104

Procedia Computer Science

6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Infoexclusion (DSAI 2015)

A model-driven development for creating accessible web menus

Humberto Lidio Antonelli[a], Elias Adriano N. da Silva[a], Renata Pontin M. Fortes[a,*]

[a] *University of São Paulo (USP), São Carlos, SP, Brasil*

Abstract

Web pages are composed by several elements that allow users to interact with available content. Some of these elements, such as menus, are responsible for assisting the navigation in the website, helping in the localization and access of the information requested by the user. However, many of menus available in Internet are not developed in an accessible manner, which creates accessibility barriers for many users, especially users with disabilities. Non-accessible menus hinder users find the information that they are looking for and therefore they can give up accessing the web site. This paper aims at addressing the problem by helping developers to create accessible web menus based on model-driven approach. Adopting this approach can help developers build accessible menus without knowing the technical details about accessibility. In direction to this objective, we initially studied the different types of menus and their structures, as well as the accessibility guidelines that involve creating accessible menus. From the study, we developed a meta-model that originated the DSL (Domain Specific Language) called AMenu (Accessible Menu). Then we included all the technical details about accessibility in the transformations. Finally, we presented an exploratory evaluation of DSL, discussing their efficiency and limitations. The results indicate a reduction in efforts for developing accessible web menus, since developers do not have to deal with technical details of accessibility. Our approach provides an alternative to traditional approaches to development for creating accessible web menus. The principles used in this study can be included in CASE tools and IDEs with great acceptance in the industry and therefore facilitate and disseminate the development of accessible systems.

Keywords: Accessibility; Navigability; Menu; Domain-specific Language

1. Introduction

With the increasing number of users and resources offered in the web, accessibility has become a very important research trend that considers promoting digital inclusion of people with disabilities or temporary mobility restrictions. On the other hand, government agencies in many countries have been committed in regulations for providing accessible content in the Internet[1,2]. However, despite all research, and governmental efforts for providing laws that reinforce and promote the creation of accessible content for the web, many systems are still not developed according to provided

* Tel.: +55-16-3373-9658.
E-mail address: humbertoantonelli@usp.br (Humberto Lidio Antonelli). eliasnog@usp.br (Elias Adriano N. Silva). renata@icmc.usp.br (Renata Pontin M. Fortes).

doi:10.1016/j.procs.2015.09.253

legislation, as noted in the survey conducted by Bittar et al.[3]. One of the initiatives that aim at promoting accessibility is held by Web Accessibility Initiative (WAI) through Web Content Accessibility Guidelines (WCAG). Although WCAG consist of guidelines for creating accessible web content, most developers still not adopting them in their projects. As a result, most systems are not accessible or present many accessibility problems. Thus, approaches that facilitate the adoption of the accessibility recommendations need to be investigated in order to improve the web with the production of accessible content and, therefore, allowing most users as possible to access web contents without difficulty or at least with low barriers.

Among accessibility problems is the difficulty on navigating in the contents available on a website, which in some cases, it may be impossible for some users. A typical case of navigation problems is the unstructured web menus that have no accessibility. This paper presents a model-driven approach for creating accessible web menus, as a support tool, for developers. We developed a DSL and a set of code transformations that help the development of accessible web menus. The main idea behind this approach is to embed the specific knowledge on guidelines in the code transformations and allows developers to focus on domain concerns while the automated mechanisms generates code according WCAG guidelines. Among inherent benefits of model-driven approaches, this approach can reduce the repetitive programming tasks and protect developers from the complexity of guidelines.

This article is organized as follows: Section 2 presents an overview of the concepts necessary for the reasoning of the objectives and characterization of the problem; Section 3 describes the methodology used to perform this work; Section 4 presents the domain modeling concepts and the creation of AMenu; Section 5 describes a case study in order to provide an analysis of the feasibility of our approach; Section 6 presents related work. Finally, Section 7 presents the conclusions and future work.

2. Menus accessibility

Web accessibility refers to enabling perception, understanding, navigation and interaction with the web by everyone, who free of barriers (or with low barriers, though counting on assistive technologies) may also use or contribute to content generation (web 2.0) and development of web systems[4,5]. Therefore, in general terms, the web accessibility definition aims to ensure the use of the web for people with disabilities.

According to WAI, responsible for setting standards on the web, web accessibility includes issues related to how all disabilities, whether visual, auditory, cognitive, speech, physical or neurological affect the access to the web[6]. Ideas and concepts related to the topic also extend up to the inclusion of older people, that even not classified as people with disabilities in general, have limitations that hinder their interaction with the web. Therefore, web accessibility can benefit all people. Creating accessible websites can also increase usability, allowing everyone to access the contents available according to their preferences, such as access to web systems using any browser or platforms (desktop or mobile)[4]. Despite the accessibility be a fundamental requirement that must be fulfilled so that everyone can access the available contents on the web, many developers, especially those who are beginning to work with web projects, usually do not have enough knowledge about accessibility, or they still have encountered difficulties to design and implement accessible web interfaces[7,8].

The inclusion of accessibility in websites requires initially awareness and training of developers team about the importance of inserting this requirement during the development process. In addition, the specific details, that should be taken in account during the development of an accessible solution, require also that developers know specific technologies, such as access platform, size screen, features of devices, and others.

In this sense, designing mechanisms to support the development of accessible elements is necessary and the issue needs to be further explored. A well-designed interface allows quick access to the main contents of a website, an efficient structure of navigation, among other features that enable users to interact with the available contents. Common users, who have no difficulty on accessing the web, may not realize the importance of accessible navigation, for instance, they do not need the semantic structure of the contents. On the other hand, visually impaired users need to use the keyboard and screen-reader to interact with websites and, therefore, one of the accessibility requirements that must be attended is the semantic structure of the contents.

In order to guide the designers about the development of accessible artifacts, many documents propose accessibility guidelines. According to Kelly et al.[9], the reference document in this area is the WCAG elaborated by WAI. This document defines a set of guidelines that help developers creating web contents, explaining how to make it more accessible to people with disabilities. Currently, WCAG is at version 2.0 and consists of principles and general guidelines. Although WCAG does not describe technological conditions, the guidelines provide information about known development methods that are consistent with WCAG conformity[10]. Furthermore, WCAG is supplemented with a non-normative section that describes specific details of how the technologies should be used[11,2].

According to Yu and Roh[12], the navigation menus on web pages have three functional roles of great importance: navigation through links, structural support and contextual suggestions, and support for the search for information. In a survey conducted by Santos and Fortes[13], it was found that among the elements highlighted in the WCAG, which caused the greatest impact on development of accessible content is the navigation element. Therefore, navigation is the focus of this work. Issues about the organization and navigation on the websites are addressed mainly in the following WCAG 2.0 guidelines: **i)** *Guideline 1.3* - Create content that can be presented in different ways without losing information or structure; **ii)** *Guideline 2.1* - Making all the features become available from a keyboard, and; **iii)** *Guideline 3.2* - Making the web pages appear and operate in predictable ways.

Accessibility guidelines comprised of recommendations that provide guidelines for the developers for creating accessible menus. However, these guidelines require huge efforts for understanding them and offer accessibility solutions that cannot be directly applied to specific problems for creating menus, such as consistency in the contents organization, location of information within the web page, absence of functionality expected by users, etc[14]. Therefore, it is important that developers can understand the accessibility guidelines to apply them in their projects in order to improve development of web menus. It is worth mentioning that, according to Power et al.[15], the guidelines have a large number of details, which require careful and thorough reading to be properly understood and used.

According to Burrel and Sodan[16], the contents need to be structured in a way that the navigation become more intuitive in order to locate the information more effectively. However, developers aim to develop menus with complex interaction features and nice layout, but without regarding, nevertheless, factors such as the internal structure of the code. Thus, assistive technologies that use this structure, as well as users who have difficulties interacting with most elaborate elements, would be not able to access the information available.

Following the accessibility guidelines (WCAG[17] and WAI-ARIA[18]), as well as considering the studies about web menus[12,16,19], we found that the development of accessible menus should have a semantic structure, with the use of HTML tags indicating a navigation block structured that comprises a list of items (`<nav>`, `<ul>` and `<li>`). In addition, the HTML tags need to contain attributes that provide a more accessible navigation, such as the `aria`, `role`, and `title` attributes. The Figure 1a shows a traditional menu, while Figure 1b shows the same menu described in an accessible manner.

```
 1 <div id="menu">
 2   <div id="mainmenu1">
 3     <a href="#" onclick="toggle('submenu1')">Category 1</a>
 4   </div>
 5   <div id="submenu1" style="display:none">
 6     <div id="submenu">
 7       <a href="#">Submenu 1a</a>
 8     </div>
 9     <div id="submenu">
10       <a href="#">Submenu 1b</a>
11     </div>
12   </div>
13   <div id="mainmenu2">
14     <a href="#">Category 2</a>
15   </div>
16   <div id="mainmenu3">
17     <a href="#">Category 3</a>
18   </div>
19 </div>
```

(a) An example of a not accessible menu.

```
 1 <nav role="navigation" class="menu" >
 2   <ul role="menubar" aria-hidden="false">
 3     <li role="menuitem" aria-haspopup="true">
 4       <a href="#">Category 1</a>
 5       <ul role="menu" aria-hidden="true">
 6         <li role="menuitem">
 7           <a href="#" tabindex="-1">Submenu 1a</a>
 8         </li>
 9         <li role="menuitem">
10           <a href="#" tabindex="-1">Submenu 1b</a>
11         </li>
12       </ul>
13     </li>
14     <li role="menuitem">
15       <a href="#">Category 2</a>
16     </li>
17     <li role="menuitem">
18       <a href="#">Category 3</a> </li>
19   </ul>
20 </nav>
```

(b) Same menu example, but accessible one.

Fig. 1: Examples of web menus.

From the analysis of the differences between both menus, it can be concluded that a specific knowledge of the structure used for creating accessible menus is required. It is observed on line 1 of Figure 1b, the `<nav>` tag shows semantically that this content block contains navigation elements, which does not occur in Figure 1a. Another point to be emphasized is the hierarchical structure of the links that is shown in lines 2 and 5 of Figure 1b. This structure does not exist in the traditional menu (Figure 1a). However, many other accessibility points need to be considered for an accessible development, such as the use of Javascript to control the interaction, CSS to control the visual presentation, and others. In terms of development, developers need to know specific HTML tags that make up the structure of the menu, as well as the attributes needed to make navigation more accessible. In other words, the development process can become costly and repetitive for these menus.

Thus, regarding to these issues, the following objectives were outlined: (a) to ensure that the accessibility guidelines can be followed, (b) to increase productivity in the development of accessible menus, (c) to reduce the effort of developers with repetitive programming work, (d) to increase portability, (e) to facilitate maintenance, (f) to increase the reuse of knowledge, and (g) to provide the correct development of menus. From these outlined objectives, this study presents a model-driven approach for developing accessible web menus. Instead of following the traditional development path for creating accessible web pages, which requires knowledge of accessibility guidelines by the project team and a manual coding process, this paper proposes the use of Model-Driven Engineering (MDE)[1]. The use of MDE will allow to raise the level of abstraction in the development of menus and consequently eliminate problems related to the lack of technical knowledge of the guidelines by developers.

In this way, since the development of accessible web menus is easy, navigation becomes more efficient and help to improve interaction by all users. As previously mentioned, the use of accessible web menus might increase the usability of the web pages. Therefore, we followed this approach in order to address the issues of usability, ensuring satisfaction, efficiency, and quality in use by users.

The MDE approaches indicate that modeling and mechanisms of transformation used to generate code from models are the best way to develop software systems, rather than coding exclusively. Advantages and objectives previously presented are inherent to MDE technologies. A developer does not need to manually interact with all the source code.

3. Methodology

The methodology used for the development of the approach began with a domain analysis of menus. We studied the concepts of web navigation elements with focus on structures and characteristics necessary for creating menus. Next, we studied the documents about the web accessibility, such as WCAG and WAI-ARIA, in order to understand the accessibility requirements that are specific of navigation context. Finally, we studied how to apply the accessibility guidelines in developing web navigation menus.

The domain analysis was the basis for creation of a meta-model, which included the concepts previously studied. Along with the specifications of textual syntax, this meta-model originated the textual domain-specific language (DSL): **AMenu – Accessible Menu**. This DSL is intended to model the domain of web navigation menus at a high level of abstraction.

The DSL is the basis for building web menus models, which are used as input to a set of transformations that will generate accessible code for different types of menus. These transformations encapsulate the specific knowledge for creating accessible menus. Thus, it is possible to generate the code of a web menu without requiring technical knowledge of accessibility by developers. In addition, transformation tools prevent the introduction of accidental errors, such as typos and syntax, as well as the conceptual errors can be identified at higher level of abstraction.

Finally, we conducted a case study about the development of menus with the DSL and transformations that we created. This case study comprised the development of a web page using the proposed approach to generate accessible menus. By the time of writing this article, we have analyzed the drop-down menu due to its wide use in websites in general. The idea is following an incremental model, through the study of a different type of navigation menu on each iteration. Then, adding these new concepts and models while other types are investigated. We intend to study the flyout menu, accordion menu, and others, as the proposed standards by Welie and Veer[22].

We used Xtext[23] to implement the DSL. Xtext provides a set of tools that enables the definition of a grammar in EBNF (Extended Backus-Naur Form) notation, as well as creating of features of textual editions such as a parser for the language, validators, and code generators. Furthermore, one of the advantages of Xtext is that the all resources

[1] MDE is also known as MDD (Model-Driven Development), MDSD (Model-Driven Software Development), or MD *. All these acronyms refer to the same concept, whose main idea is to recognize the importance of models in the software process, not only as a "guide" for development and maintenance tasks, but as part of the software[20,21].

available are independent of the Eclipse environment. For the set of transformations, we use the generation tool Xtend[2]. Through the Xtend is possible to create different transformations that are used to generate source code.

4. DSL for menus modeling

A DSL can be textual (allowing to specify programs) or visual (allowing to specify models or diagrams)[24]. The definition of the language typically requires a meta-model that can capture the common points and domain variables. Meta-model is a structure similar to a class diagram with elements such as classes, attributes, associations, and aggregations. The existence of a meta-model, as conceptual schema, generally indicates that a greater number of instances can be expressed, regardless of whether the final representation is textual one or a visual diagram.

In this sense, we defined a DSL for domain of web navigation menus in order to implement the approach proposed in this work. This domain has been selected, among the various elements of web interaction, because of the importance that menus exert in communication between users and the system[25]. However, our approach can be extended to for developing of other interaction elements, such as those presented by Wile e Veer[22].

The information for composition of models is relatively simple, which comprise the basic data needed to create a navigation menu. The defined DSL has this focus and the meta-model is shown in Figure 2.

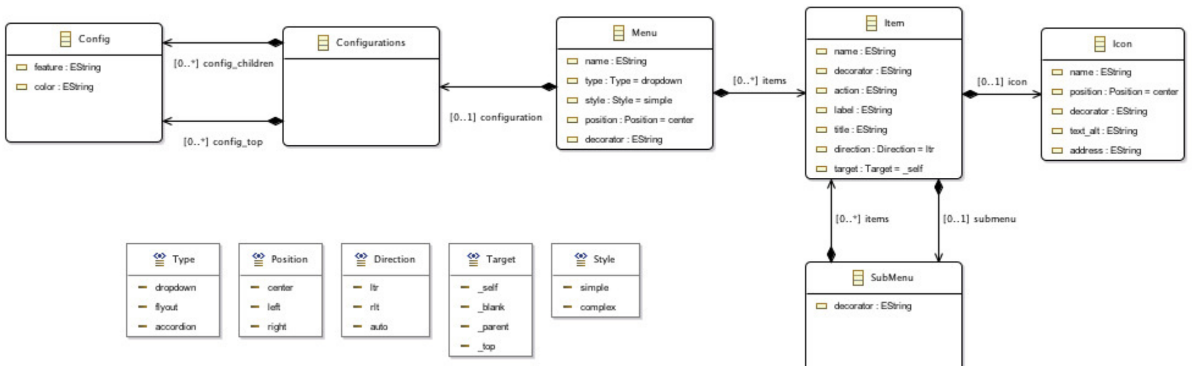

Fig. 2: AMenu meta-model

According to the Class diagram shown in Figure 2, the meta-model comprises the definition of a single menu that contains one or more items, as well as other optional attributes. Each menu item must contain the action to be performed (for example, redirect to another page) and the text (label) that informs the user about the action. These attributes are the basic information necessary for generating the navigation structure. Other attributes such as name, title, and icon (or image) are optional, because they are complementary information that may be required or not. In addition, a menu might representing tree data structure when a menu item is composed of other menu. This characteristic is related to the pattern of drop-down menus.

We defined the AMenu language as textual form due to the representation be simple to use, ease of learning, and flexibility of the tools used in development. In addition, the terms used to define the AMenu are very close to the used by developers, becoming easier the learning curve. However, the meta-model designed can also be represented graphically. In other words, from the same elements created in the representation of meta-model, it is possible to define a graphical approach for modeling menus through the GMF[3]. The graphical approach can be complementary to the textual representation facilitating the menu display, since the menu elements can be manipulated graphically. Figure 3a shows the main elements of grammar AMenu language.

[2] http://www.eclipse.org/xtend/
[3] http://www.eclipse.org/modeling/gmp/

`Menu` rule (line 1) is the root node of the menu structure which gives rise to the first level of tree. `Item` rule (line 12) requires only the necessary information (action and label, lines 17 and 18 respectively) for composition of the navigational structure. Figure 3b shows an example of a model created according to our AMenu DSL.

```
 1  Menu:
 2    (
 3        ( 'id''=' name=ID ';' )?
 4      & ( 'type''=' type=Type ';' )
 5      & ( 'style''=' style=Style ';' )?
 6      & ( 'position''=' position=Position ';' )?
 7      & ( 'class''=' decorator=STRING ';' )?
 8      & ( 'config''=''[' configuration=Configurations ']'';' )?
 9      & ( 'items''=''[' (items+=Item) (',' items+=Item)* ']'';' )
10    )
11  ;
12  Item:
13    '{'
14       (
15          ( 'id''=' name=ID ';' )?
16        & ( 'class''=' decorator=STRING ';' )?
17        & ( 'action''=' action=STRING ';' )?
18        & ( 'label''=' label=STRING ';' )
19        & ( 'title''=' title=STRING ';' )?
20        & ( 'dir''=' direction=Direction ';' )?
21        & ( 'target''=' target=Target ';' )?
22        & ( 'icon''=' icon=Icon ';')?
23        & ( 'submenu''=' submenu=SubMenu ';')?
24       )
25    '}'
26  ;
```

(a) Main elements of grammar.

```
id = menu;
items = [
  {
    action = 'home.html';
    label  = 'Inicio';
    title  = 'Pagina Inicial';
  },
  {
    action = 'guias-mapas.html';
    label  = 'Guias e mapas';
  },
  {
    action = 'dicas-viagem.html';
    label  = 'Dicas de Viagem';
  }
];
```

(b) A simple model of menu using the grammar.

```
def final CharSequence genHtml(){
  var num_id = 1
  '''
  <nav role="navigation" id="«this.
      _id_menu»" «IF this._menu.
      decorator != null» class="«
      this._menu.decorator»"«ENDIF»>
    <ul role="menubar" class="nav-
      menu">
      «FOR i : this._menu.items»
        «genItem(i, "-", num_id++)»
      «ENDFOR»
    </ul>
  </nav>
  '''
}
```

(c) Rule for code generation of menu items

Fig. 3: Definition of the language AMenu

Despite the AMenu has a simple structure, it gives rise to the complete code of an accessible web menu navigation because all knowledge related to accessibility details are embedded in the transformations. The code generators have a key role because they are responsible for translating the high level specifications (models) in executable code snippets. These generators can also add benefits to software development process because they always produce the same code, in a predictable and controlled way, and with fewer errors. For instance, a generator can automatically apply code patterns in a systematic way with lower error rate[26].

In this sense, we defined transformations using the Xtend technology in order to support the code generation of accessible web menus. These transformations convert the models defined by AMenu on HTML, CSS and Javascript source codes. The codes produced by AMenu can be easily integrated with the rest of a web application. Figure 3c shows a code snippet of the transformation to generate drop-down menus accessible.

5. Case study

In an exploratory manner, the case study developed in this work aimed to compare the solution generated by our approach with a traditional development approach. The main focus was to analyze differences between codes concerning to the implementation of web accessibility requirements.

The development team needs knowledge about the accessibility guidelines for developing a accessible web application. Therefore, it would take considerable time and resource to the team's training about the accessibility concepts and how they should apply these concepts in the project. All the effort would be spent just to create a simple web page. On the other hand, the developers would need only domain knowledge and syntax used when MDE is adopted in the project. Let's take the following scenario to illustrate the operation of our approach:

> "A website consists of several components, which include a navigation menu that allows the user to navigate between the various pages of the website. Thus, a travel agency has a website with a main navigation menu of type dropdown. This menu comprises five basic options: *home*, *exchange*, *sea cruise*, *ecological roadmap*, *contact*".

Figure 4 shows a menu snippet of the scenario that was developed without accessibility requirements. The menu was created by a developer that has no knowledge of the accessibility guidelines. In general, it can be seen that the code does not contain a semantic framework to assistive technologies navigate efficiently. The developer used the `<div>` element for the presentation of items (lines 1, 3, 4, 5, 6, and 7). However, for developing an accessible menu it is required the use of specific elements (`<nav>`, `<ul>`, and `<li>`), which indicate the composition of a list of elements (in this case, a list of links).

```
 1 <div class="item">
 2   <a href="ecological-roadmap.html">Ecological Roadmap</a>
 3     <div class="submenu">
 4       <div class="item"><a href="amazonia.html">Amazônia</a></div>
 5       <div class="item"><a href="bonito.html">Bonito</a></div>
 6       <div class="item"><a href="pantanal.html">Pantanal</a></div>
 7       <div class="last">
 8         <img class="corner_left" src="corner_left.png">
 9         <img class="middle" src="dot.gif">
10         <img class="corner_right" src="corner_right.png">
11       </div>
12   </div>
13 </div>
```

Fig. 4: Snippet code of non-accessible menu used in the case study.

In order to let the menu accessible using our approach, we considered the same problem and performed the domain modeling using the AMenu. Figure 5a shows the domain modeling concerning to the code snippet shown in Figure 4. Figure 5b shows the HTML code of the accessible menu generated from domain modeling (Figure 5a).

```
action  = 'ecological-roadmap.html';
label   = 'Ecological Roadmap';
submenu = {
    items   = [
        {
          action = 'amazonia.html';
          label  = 'Amazônia';
        },
        {
          action = 'bonito.html';
          label  = 'Bonito';
        },
        {
        label   = 'Pantanal';
        action = 'pantanal.html';
        }
    ];
};
```

(a) DSL code used to generate accessible menu

```
 1 <li role="menuitem" class="nav-item has-sub">
 2   <a id="turistm-1" href="ecological-roadmap.html" target="_self" dir="ltr" tabindex="0"
       aria-haspopup="true" aria-owns="turistm-1-0" aria-controls="turistm-1-0" aria-expanded="false">
 3     Ecological Roadmap
 4   </a>
 5   <ul id="turistm-1-0" aria-hidden="true" role="menu" class="sub-nav-group sub-nav" aria-labelledby="
       turistm-1-">
 6     <li role="menuitem" class="nav-item" aria-describedby="turistm-1">
 7       <a id="turistm-1-1" href="amazonia.html" target="_self" dir="ltr" tabindex="-1">
 8         Amazônia
 9       </a>
10     </li>
11     <li role="menuitem" class="nav-item" aria-describedby="turistm-1">
12       <a id="turistm-1-2" href="bonito.html" target="_self" dir="ltr" tabindex="-1">
13         Bonito
14       </a>
15     </li>
16     <li role="menuitem" class="nav-item" aria-describedby="turistm-1">
17       <a id="turistm-1-3" href="pantanal.html" target="_self" dir="ltr" tabindex="-1">
18         Pantanal
19       </a>
20     </li>
21   </ul>
22 </li>
```

(b) Accessible menu code generated by our approach.

Fig. 5: Snippet code of accessible menu used in the case study.

The source codes (HTML, CSS, and Javascript) of accessible web menus were generated by using our approach from the modeling of problem. In other words, the developer did not need to implement the accessibility requirements, since these requirements have been already implemented in the transformations. It is possible to see that some menu elements are very similar when we analyze the generated HTML code (Figure 5) and compare it with the traditional code (Figure 4). Although both menus have equivalent functionality and layout, there are elements that differ, especially their accessibility details. As previously stated, an accessible web menu requires the implementation of specific details. Therefore, the web menu, that our approach generated, has a semantic structure by using correct HTML tags (`<ul>` and `<li>` on lines 1, 5, 6, 11 and 16 of Figure 5). Furthermore, the `role` attribute indicates the semantic feature for each item, as well as others specific ARIA attributes related to this domain. Many other accessibility requirements are encapsulated in the Javascript and CSS code that are not presented here.

Therefore, it is possible to see that the greatest benefit when using our approach is code generation in accordance with the accessibility guidelines (such as WCAG and WAI-ARIA) without requiring that developers know the technical details about these guidelines. Moreover, the effort on the developer to encode each item is reduced, since the entire structure is generated by the approach. Moreover, our approach reduce the developer effort for creating each item, since all structure is generated by the approach. In this way, our approach also ensures that errors are not introduced in encoding, as the appropriate closing tags.

In terms of productivity, we calculated the ratio between the specification and the code (RSC). This metric determines the relationship between a specified element and the corresponding generated code[20]. For example, if the specification of a 10-line entity produces 1000 lines of code, there is a relation of 1 to 100. This means that each domain element generates around 100 lines and from then on we can check the gain of productivity in terms of lines of code. This metric is calculated as follows:

$$RSC = \sum LOC(code\ gerated) / \sum NEE(models) \tag{1}$$

where: NEE(models) is the number of model specification elements.

To calculate the metric, the *LOC* value of the generated code was 273; we define that an element of specification is a DSL code line. The solution has 40 lines of implementation. However, we consider only 21 by discounting lines consist of opening and closing braces ({, }) or brackets ([,]). Therefore, the calculation was made as follows:

$$RSC = (273)\ /21 \ \therefore RSC \cong 13 \tag{2}$$

The result of the Equation 2 shows that our approach has generated 13 source lines of code for each line of code written. In this case, the gain was not very significant because the selected domain for presentation of the prototype is small. Although, in the usual context of development systems, approaches using DSLs or DSMLs[4] have been quite productive in terms of code generation[28,29]. The use of our approach ensures that all accessibility requirements of the web menus that are proposed by the WCAG and WAI-ARIA guidelines are attended. However, we have not included the lines relating to Javascript and CSS codes in the calculus above. Despite being part of our approach, we consider that these codes are a external element to the code generation process. Moreover, the approach presented in this work can be extended for developing accessible web systems as a whole, so that all the benefits of MDE can be achieved.

In the context of code generation, it can be argued that using the amount of code lines to measure the productivity may not be too efficient because generators typically produce a very dense code. However, it is worth mentioning that the construction of generators is made from a reference implementation in our approach, which means that the generated code follows the same standards and format used by developers.

6. Related works

Jeschke et al.[30] proposed an approach using MDE to include accessibility requirements in the early stages of the development process. In their work they used the extent of UWE (UML-based Web Engineering) for project modeling, addressing only the accessibility standards in navigation, guidance and overview of the site. However, level of encryption solutions are not addressed, since the proposal is based on the production of UML models. In addition, the authors refer to the structuring of the general contents, not worrying about the specifics of the elements. The approach proposed in this study is different because it deals especially automatic code generation with focus on composition for the construction of accessible elements structure.

Bittar et al.[31] investigated an approach for constructing a navigation structure accessible in wiki, by using MDE to include accessibility requirements on the code generated by the wiki. Therefore, the authors proposed that the information architecture is developed from the creation of namespaces[5] as they allow the code of wikis can be generated automatically. Thus, the inclusion of accessibility and usability practice in software development becomes possible. The approach presented by the authors is facing the context of wikis, while the approach proposed in this paper can be used on any web context.

Moreno et al.[32] discussed the implementation of accessibility for Web Applications (AWA) approach to engineering method Object Oriented Web Solutions (OOWS), to produce accessible web applications focusing on navigation

[4] Domain Specific Modeling Language[27]

[5] Namespaces are textual expressions used to rank pages and other media files. Thus, a namespace is the key to access the resources available, the concept involved in its use is similar to the file system[31].

requirements. The proof of concept is presented in order to show the practical applicability and usefulness of the approach. When using the AWA approach in the MDE, the previously defined OOWS models were extended with accessibility criteria, providing the generation of source code with an affordable navigation. The approach proposed in this paper deals with the automatic generation of final code for web applications, while the work of Moreno et al. propose a solution that focused on context frames. Moreover, it is argued that the language set for the menu templates (AMenu), is simple and produces artifacts and syntax familiar to web developers.

7. Concluding Remarks

This paper presented a model-driven approach that assists developers creating accessible web menus. The proposed approach is based on both web accessibility and MDE, that offer many benefits to system developers. From the case studied, it is possible to note some of these benefits, which proves the feasibility of our approach.

We conclude therefore that the use of our approach ensures that the accessibility guidelines are attended, since accessibility requirements for creating accessible web menus are implemented in the transformations. In addition, our approach increases productivity in the development of accessible web menus, since the time spent for training on accessibility guidelines could be redirected to other project activities. Another point is the reduction of effort required for repetitive programming work, because all the source codes of the menus are generated by the transformations, as well as ease of maintenance, since codes are automatically generated from transformations that can be easily changed.

Our approach can further enhance the portability, since the same domain models can be used to generate menus of different patterns with adding of new transformations. Moreover, increasing the reuse of knowledge, since models are easier to comprehend than code. Thus, developing high level of abstraction facilitates communication between team members, which allows these models to be reused more easily. Our approach can also be extended for developing accessible web systems in general.

However, it is important to conduct an experiment with developers to evaluate the feasibility of our approach and improve it in this aspect. Moreover, we need to evaluate how to insert the AMenu in the development processes. Thus, the next step of this work is to carry out further evaluation of the approach.

In future works new transformations will be created, as well as updating the DSL grammar so that our approach can generate other types of accessible menus, as those proposed by Welie e Veer[22]. In addition, we intend to expand the field of approach, to cover the generation of accessible code to other elements patterns for mobile devices proposed by Ribeiro (2011)[33], such as tabpanel, accordion, among others. We also aim to extend the approach to creating accessible web menus through graphic interface, enabling the integration of AMenu with AWMo tool[5]. If automated mechanisms like used in this approach were used in well accepted IDEs like Eclipse and others, they could make even easier the development of accessible systems.

8. Acknowledgments

The authors thank FAPESP, processes number 2012/24487-3 and 2013/23966-8, for having funded this research.

References

1. eMAG, Modelo de acessibilidade do Governo Eletrônico, Versão 2.0. 2005. Http://goo.gl/8cOQ4y.
2. Cooper, M., Kirkpatrick, A., Connor, J.O.. Techniques for WCAG 2.0. 2009. Http://www.w3.org/TR/WCAG20-TECHS/.
3. Bittar, T., Faria, F., Amaral, L., Fortes, R.. An assessment of accessibility in contact forms of brazilian public universities. In: *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*. 2012, p. 1–6.
4. Rutter, R., Lauke, P., Waddell, C., Thatcher, J., Henry, S., Lawson, B., et al. *Web Accessibility: Web Standards and Regulatory Compliance*. Apresspod Series. Apress; 2006. ISBN 9781590596388. URL: `http://books.google.com.br/books?id=dlJ94KZqwqcC`.
5. Grillo, F.D.N., Fortes, R.P.M.. Tests with Blind Programmers Using AWMo: An Accessible Web Modeling Tool. In: Stephanidis, C., Antona, M., editors. *Universal Access in Human-Computer Interaction. Design and Development Methods for Universal Access*. Springer; 2014, p. 104–113. doi:10.1007/978-3-319-07437-5_11.
6. Henry, S.L.. Introduction to Web Accessibility. 2005. Http://www.w3.org/WAI/intro/accessibility.php.
7. Freire, A.P., Russo, C.M., Fortes, R.P.. A survey on the accessibility awareness of people involved in web development projects in brazil. In: *Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A)*. ACM; 2008, p. 87–96.

8. Miñón, R., Moreno, L., Martínez, P., Abascal, J.. An approach to the integration of accessibility requirements into a user interface development method. *Science of Computer Programming* 2014;**86**(0):58 – 73. URL: `http://www.sciencedirect.com/science/article/pii/S0167642313001032`. doi:http://dx.doi.org/10.1016/j.scico.2013.04.005; special issue on Software Support for User Interface Description Languages (UIDL 2011).
9. Kelly, B., Sloan, D., Phipps, L., Petrie, H., Hamilton, F.. Forcing Standardization or Accommodating Diversity?: A Framework for Applying the WCAG in the Real World. In: *Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*; W4A '05. New York, NY, USA: ACM. ISBN 1-59593-219-4; 2005, p. 46–54. URL: `http://doi.acm.org/10.1145/1061811.1061820`. doi:10.1145/1061811.1061820.
10. Reid, L.G., Snow-Weaver, A.. WCAG 2.0: A Web Accessibility Standard for the Evolving Web. In: *Proceedings of the 2008 International Cross-disciplinary Conference on Web Accessibility (W4A)*; W4A '08. New York, NY, USA: ACM. ISBN 978-1-60558-153-8; 2008, p. 109–115. URL: `http://doi.acm.org/10.1145/1368044.1368069`. doi:10.1145/1368044.1368069.
11. Kelly, B., Sloan, D., Brown, S., Seale, J., Petrie, H., Lauke, P., et al. Accessibility 2.0: People, Policies and Processes. In: *Proceedings of the 2007 International Cross-disciplinary Conference on Web Accessibility (W4A)*; W4A '07. New York, NY, USA: ACM. ISBN 1-59593-590-8; 2007, p. 138–147. URL: `http://doi.acm.org/10.1145/1243441.1243471`. doi:10.1145/1243441.1243471.
12. Yu, B.M., Roh, S.Z.. The effects of menu design on information-seeking performance and user's attitude on the World Wide Web. *Journal of the American Society for Information Science and Technology* 2002;**53**(11):923–933. URL: `http://dx.doi.org/10.1002/asi.10117`. doi:10.1002/asi.10117.
13. Santos, E.P.B., Fortes, R.P.. CMS instance of a tool to support the scheduling of undergraduate final year project on the Web. *Simpósio Brasileiro de Sistemas Multimídia e Web - WebMedia* 2010;:127–129.
14. Freire, A.P.. *Disabled people and the Web: User-based measurement of accessibility*. Doctor of philosophy; Departament of Computer Science, University of York; United Kingdom; 2012. URL: `http://etheses.whiterose.ac.uk/id/eprint/3873`.
15. Power, C., Freire, A.P., Petrie, H., Swallow, D.. Guidelines are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the web. In: *Proceedings of CHI'12*; CHI'12. New York, NY, USA: ACM; 2012, p. 5–10.
16. Burrell, A., Sodan, A.. Web Interface Navigation Design: Which Style of Navigation-Link Menus Do Users Prefer? In: *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*. 2006, p. 42–42. doi:10.1109/ICDEW.2006.163.
17. Web content accessibility guidelines 2.0. 2008. URL: `http://www.w3.org/TR/WCAG20/`.
18. Accessible rich internet applications (wai-aria). 2014. URL: `http://www.w3.org/TR/WCAG20/`.
19. dos Santos, E.P., de Lara, S., MA Watanabe, W.M., Filho, M.C., Fortes, R.P.. Avaliação de usabilidade de horizontal barra de navegação com menus drop-down por adultos de meia idade. In: *Anais da 29 ^{a} Conferência Internacional de ACM em Design of Communication*. New York, NY, EUA: ACM. ISBN 978-1-4503-0936-3; 2011, URL: `http://doi.acm.org/10.1145/2038476.2038504`. doi:10.1145/2038476.2038504.
20. Lucrédio, D.. *Uma abordagem orientada a modelos para reutilização de software*. Doutorado em Ciências de Computação e Matemática Computacional; ICMC/USP; São Carlos - SP; 2009. URL: `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-01072008-143726/`.
21. Lucrédio, D., Bittar, T.J., Fortes, R.P.M.. vol. III; chap. 3 - Desenvolvimento Web orientado a modelos: conceitos, ferramentas e técnicas. Mário Meireles Teixeira, César Augusto Camillo Teixeira, Fernando Antonio Mota Trinta, Pedro Porfirio Muniz Farias. (Org.). Minicursos XV Webmedia. Fortaleza - CE: SBC Editora; 1 ed.; 2009, p. 1–44.
22. Welie, M.V., Veer, G.C.V.D.. Pattern languages in interaction design: Structure and organization. In: *Proc. Interact '03, M. Rauterberg, Wesson, Ed(s). IOS*. IOS Press; 2003, p. 527–534.
23. Eysholdt, M., Behrens, H.. Xtext: Implement your language faster than the quick and dirty way. In: *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*; SPLASH '10. New York, NY, USA: ACM. ISBN 978-1-4503-0240-1; 2010, p. 307–309. URL: `http://doi.acm.org/10.1145/1869542.1869625`. doi:10.1145/1869542.1869625.
24. van Deursen, A., Klint, P., Visser, J.. Domain-specific Languages: An Annotated Bibliography. *SIGPLAN Not* 2000;**35**(6):26–36. URL: `http://doi.acm.org/10.1145/352029.352035`. doi:10.1145/352029.352035.
25. Lai, Y.R., Waugh, M.L.. From Information Searching to Learning: A Comparison of Contrasting Hypertextual Menu Designs for Computer-Based Instructional Documents. 1994;.
26. Cleaveland, J.C.. Building application generators. *IEEE Software* 1988;**5**(4):25–33.
27. Brambilla, M., Cabot, J., Wimmer, M.. Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering* 2012; **1**(1):1–182.
28. da Silva, E.A.N., Fortes, R.P.M., Lucredio, D.. A Model-Driven Approach for Promoting Cloud PaaS Portability. In: *Anual International Conference on Software Engineering-CASCON*. 2013, .
29. Papotti, P.E., do Prado, A.F., de Souza, W.L.. Reducing time and effort in legacy systems reengineering to MDD using metaprogramming. In: *Proceedings of the 2012 ACM Research in Applied Computation Symposium*. ACM; 2012, p. 348–355.
30. Jeschke, S., Pfeiffer, O., Vieritz, H.. Using Web Accessibility Patterns for Web Application Development. In: *Proceedings of the 2009 ACM Symposium on Applied Computing*; SAC '09. New York, NY, USA: ACM. ISBN 978-1-60558-166-8; 2009, p. 129–135. URL: `http://doi.acm.org/10.1145/1529282.1529308`. doi:10.1145/1529282.1529308.
31. Bittar, T.J., Lobato, L.L., Fortes, R.P.M., Neto, D.F.. Accessible Organizational Elements in Wikis with Model-driven Development. In: *Proceedings of the 28th ACM International Conference on Design of Communication*; SIGDOC '10. New York, NY, USA: ACM. ISBN 978-1-4503-0403-0; 2010, p. 49–56. URL: `http://doi.acm.org/10.1145/1878450.1878459`. doi:10.1145/1878450.1878459.
32. Moreno, L., Valverde, F., Martínez, P., Pastor, O.. Supporting navigation accessibility requirements in web engineering methods. *J Web Eng* 2013;**12**(3-4):181–202. URL: `http://dl.acm.org/citation.cfm?id=2535629.2535630`.
33. Ribeiro, J.M.N.. *Web Design Patterns for Mobile Device*. Mestrado em multimédia; Faculdade de Engenharia da Universidade do Porto; Portugal; 2011. URL: `http://hdl.handle.net/10216/64860`.