2015-10

A support for developers implement the accessibility guidelines regarding to web menus

Brazilian Symposium on Multimedia and the Web, 21th, 2015, Manaus.
http://www.producao.usp.br/handle/BDPI/49578

A Support for Developers Implement the Accessibility Guidelines Regarding to Web Menus

Humberto Lidio Antonelli
University of São Paulo
São Carlos, SP, Brazil
humbertoantonelli@usp.br

Renata Pontin de Mattos Fortes
University of São Paulo
São Carlos, SP, Brazil
renata@icmc.usp.br

ABSTRACT

Web pages are composed of elements, such as menus that are responsible for assisting navigation on the website. However, many of the menus are not developed properly, which creates accessibility barriers and hinders access to the contents. This paper aims to present a method for creating accessible menus. Initially, we studied the different types of menus and the accessibility guidelines involving the creation of accessible menus. From the studies, we developed a meta-model that gave rise to AMenu language, where we included all the technical details regarding to accessibility. Then, we developed the AMeG tool to facilitate the use of language. Finally, we conducted a case study with developers, in order to verify the feasibility of the approach, arguing its efficiency and limitations. The results indicate a reduction in efforts to develop accessible web menus, since developers do not have to deal with technical details of accessibility.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Interaction Styles; H.5.4 [**Hypertext / Hypermedia**]: User Issues

General Terms

Human-centered computing, Human Factors, Design

Keywords

Accessibility, Navigability, Menu, Web interaction

1. INTRODUCTION

With the increasing number of users and resources offered on the web, accessibility has become a very important factor in view of the essential character of promoting digital inclusion of people with disabilities or temporary mobility restrictions. Although many documents[2, 1, 3, 4] suggest guidelines for creating accessible web content, most developers do not still adopted these guidelines in their projects, resulting in many accessibility problems. One of these problems is the difficulty in relation to navigate the content available on a website, which in some cases it may be impossible for some users. A typical case is unstructured navigation menus that do not have any accessibility.

In general, the development team needs to know the technical details of the accessibility guidelines (such as WCAG 2.0 and ARIA 1.0), in order to develop a fully accessible web application. Thus, it would take time and resource considerable for development team training. However, by using an model-driven approach, developers would only need to know the domain and the syntax of the language, which would decrease the cost of development.

In this context, we developed the Accessible Menu Generator (AMeG), which assists developers in creating accessible web menus in accordance with accessibility guidelines. We carefully studied all accessibility requirements for this web element and we created a domain specific language (DSL), so that developers do not need to worry about the source code.

The main idea behind our 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 accessibility guidelines. Among inherent benefits of model-driven approaches, it can reduce the repetitive programming tasks and protect developers from the complexity of guidelines.

2. RELATED WORK

Several authors have recently described strategies for dealing with the accessibility guidelines, by developing tools or techniques to apply technological resources to help disable people being able to access the diversity of information and services available in the Internet nowadays.

According to [12], for web developers to create functionally accessible web resources they need more than general guidelines and tools that provide them with lists of manual accessibility checks. Web developers need specific web accessibility techniques and tools that help them verify they have correctly implemented the techniques. The techniques also need to support the wider concepts of the web of interoperability and device independence.

Gunderson et. al [12] used a Functional Accessibility Evaluation (FAE) Tool that provides a means to estimate the functional accessibility of web resources by analyzing web pages and estimating their use of the CITES/DRES (Disability Resources and Educational Services), web accessibility best practices. The tool does not determine if a resource

WebMedia'15, October 27–30, 2015, Manaus, Brazil.

DOI: http://dx.doi.org/10.1145/2820426.2820445.

or a collection of resources is accessible or not, but provide summary and detailed reports on the use of accessible markup categorized by the best practices principles.

They argue that for functional accessibility to be achieved more automated tools need to be available to help web developers to review and redesign their web resources. Without automation most web developers will only be checking for a limited scope of web accessibility requirements. Although, they admit the automated tools cannot solve all accessibility problems, tools like FAE and the Mozilla/Firefox Accessibility Extension allow evaluation resources to be devoted to usability issues, rather than the more basic requirements of having accessible markup.

In this paper we propose a support for developers implement the accessibility guidelines regarding to web menus. Such a support aims to contribute as an aid for developers to build web applications focusing on the information matters, instead of leading with implementation details.

In order to disseminate and teach the future developers, [16] firstly conducted a survey of student attitudes toward these issues at the start and end of a usability engineering course that included a group project with an accessibility component. Results of the survey indicate that students' awareness of issues related to usability and accessibility are increased after taking the course and completing the project. The results, potentially valuable to CS educators, could be seen in three ways: (1) They validate the usefulness of the survey instrument in assessing pedagogy in usability engineering, (2) They provide useful insights into the attitudes of CS majors relative to the important topics of usability and accessibility, and (3) They point to possible benefits of including usability and accessibility topics into CS curricula.

The survey results suggest that the experience of an HCI course did not lead to increased rated importance on the factor that includes the accessibility element (AE). Then, it suggests that a course with greater emphasis on AE may lead to students placing greater importance in issues related to usability, particularly with respect to non-typical user interfaces.

The approach behind the developed support proposed in this paper takes into account that for novice developers, an incremental knowledge of implementation details of accessibility guidelines could stimulate students interests on how to improve their knowledge about these technological resources. Thus, the support adopted the MDD (Model-Driven Development) method to help developers to focus on their high-level aspects of a web application under construction, in contrast of being worried about the implementation details.

3. WEB ACCESSIBILITY GUIDELINES

Accessibility can be broadly interpreted as the possibility of using a resource universally, without barriers or with alternative ways to access and use. In web context, the contents of the pages are the resource. So this concept is related to every user, using every agent can understand and interact with the offered content [23].

There is a significant gap of knowledge from developers and experts in accessibility. Most programmers have no necessary knowledge or experience to ensure that their code attends the accessibility requirements [11]. Many design decisions could be based on the practices documented by official institutions of the Web, like the document Web Content Accessibility Guidelines (WCAG) or in works that teach step by step how to get an accessible website [22].

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 [6]. 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.

The navigation menus on web pages have three functional roles of great importance [27]: navigation through links, structural support and contextual suggestions, and support for the search for information. In a survey conducted by Santos and Fortes[21], 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, menu web navigation is the focus of this work. Issues about the organization and navigation on the websites are addressed mainly in the WCAG and WAI-ARIA guidelines.

According to [14], the reference document in web accessibility 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[18]. Furthermore, WCAG is supplemented with a non-normative section that describes specific details of how the technologies should be used[13, 7].

Websites have increasingly complex interactions and advanced interfaces such as tree controls, drop-down menus, among others. Therefore, assistive technologies must be able to interact with these controls in order to provide a richer experience of interaction for users with disabilities. However, most current web technologies, such as HTML and Ajax, do not provide enough context information to assistive technologies. Toward overcoming these challenges in accessibility, WAI-ARIA defines a set of attributes that aid improving the semantics of the web content. Thus, assistive technologies, such as screen readers, can understand the context and the behavior of complex element of the interface that are not native from the HTML language [4]. Currently, WAI-ARIA is at version 1.0 and introduces a suite of technical documents that aid developers increase the accessibility of dynamic contents in interface components.

Accessibility guidelines require huge efforts for understanding them, as well as they offer solutions that cannot be directly applied to specific problems, such as creating menus [10]. Moreover, developers are not always aware of the design solutions and recommendations provided by accessibility specifications [24]. It is worth mentioning that, according to [17], the guidelines have a large number of details, which require careful and thorough reading to be properly under-

stood and used. 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.

4. METHODOLOGY

The methodology used for the development of our 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.

From the concepts previously studied, we created a domain meta-model that 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, in order to facilitate the implementation of accessibility guidelines by the developers, since they do not need to deal with technical details of accessibility directly in the source code.

The DSL is the basis for building web menus models, which are used as input to a set of transformations that will generate source code for different types of accessible menus. These transformations encapsulate the specific knowledge for creating accessible menus. Thus, it is possible to generate the source 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 typing and syntax errors, as well as the conceptual errors can be identified at higher level of abstraction.

After creating the DSL, we also developed a web application, that is called **Accessible Menu Generator (AMeG)**, in order to facilitate the use of the AMenu by the developers. The AMeG has a simple interface and is derived from other tools previously developed in the research group (AWMo[1] and AgendAloca[2]). Thus, the AMeG used a reference architecture that ensured a solid foundation following accessibility criteria [9].

Finally, we conducted a case study to investigate the accessibility of web menus that were created using traditional methods of development regarding to that were generated using AMeG. The details of this case study are presented in the section 6. We have analyzed both *drop-down* and *flyout* menu, because this types of menus are widely used on 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 all patterns that are proposed by Welie and Veer[25], as well as specific patterns for mobile devices that are proposed by Ribeiro[19].

From the data collected, we performed a data triangulation to check and establish validity of the results by analyzing from multiple sources. This analysis technique is essential in qualitative studies because the data collected provide a richer and deeper description, but less accurate than the data collected in quantitative studies [20]. The section 7 shows the analysis of data collected in the case study.

[1]http://garapa.intermidia.icmc.usp.br:3000/awmo/
[2]http://garapa.intermidia.icmc.usp.br:3000/agendaloca/

5. AMEG

The AMeG is a prototype tool that we developed to enable the creation of accessible web menus by any developer, regardless of the level of knowledge they have about the accessibility guidelines. The prototype is restricted to drop-down and flyout menu in order to evaluate the approach.

The key concept behind AMeG approach is to hide the technical details of accessibility. Thus, developers can focus on high-level aspects of a web application under construction rather than be worried about the implementation details. Furthermore, it is possible to reduce the time and cost of training developers on the accessibility guidelines.

AMeG is a web application that was developed using JavaServer Faces (JSF) 2.1. The current version of AMeG is restricted only to the textual editor. The development of textual interface has been prioritized in order to investigate the main purpose of the AMeG that is to enable the generation of web menus with maximum accessibility. However, a graphical interface can be easily implemented and integrated in the tool, since the AMeG has a central data structure to process the information provided by the user.

Figure 1 shows the interface of the AMeG. We can observe that there is a accessibility toolbar on the top, which allows the user to enable high contrast, control the font size of the application as a whole and also select the language for presentation of the application. On the left side of the main content area there is a menu that display the options “Home” “About the AMeG”, and “Help”. The main content has a simple textual editor that the developers can use to create menus using the AMenu language. After entering source code in accordance with the AMenu language, developers can generate the source code of accessible web menu by clicking on the ”Generate menu” button that is below the textual editor. If the source code is not in accordance with the structure required by the AMenu, the AMeG will return an error message and the line number where the problem was located. Thus, the developer can correct the error and generate the menu again. If the structure is correct, a success message is displayed at the top and the result of the transformation will be displayed just below the textual editor.

We developed the AMeG interface based on a reference architecture [13], including the visual style, menus and accessibility features such as high contrast, font size control and also the ability to be translated to multiple languages. The fact that both architecture and interface were inherited from the reference architecture and had good accessibility and usability features gave the AMeG tool development a great start up speed. Moreover, it allowed the development be focused on the inner workings and business logic of the proposed modeling environment.

We have developed AMeG specifically for creating accessible web menus. However, the structure of AMeG can enable the development of other accessible elements, such as forms, tabpanels, wizards, dialogues, among others. Since it is only necessary to improve the language and to define new models of transformation for these elements.

The textual language AMenu, which is used in AMeG, was developed to enable modeling of the web menus domain at high level of abstraction. This modeling is based

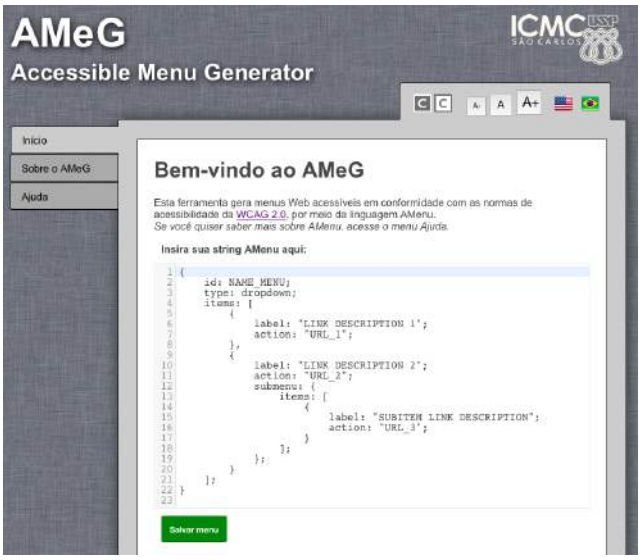

Figure 1: Screen capture of AMeG interface displaying the initial page

on studies of the concepts of web navigation elements, focusing on structures and specific and necessary features for creating menus. Moreover, this modeling includes studies on accessibility guidelines WCAG and WAI-ARIA in order to understand the accessibility requirements regarding to the navigation context as well as how to apply these guidelines in the development of web menus. The information for composition of models is relatively simple, which comprise the basic data needed to create a web menu. The defined DSL has this focus and the meta-model is shown in Figure 2.

According to the 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 and flyout menus.

We used Xtext[8] to implement the AMenu language. 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 available are independent of the Eclipse environment. Also, by making available the Ecore meta-model and the EMF classes for the designed language, it allows programmatic access to both models and meta-models. For the set of transformations, we use the generation tool Xtend[3]. Through the Xtend is possible to create different transformations that are used to generate source code. In practical terms, all the infrastructure of AMenu language is contained within a Java library that can be easily used in other projects in order to facilitate the development of accessible web 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. More details about the technologies behind the AMeG and AMenu are described in [5]

6. CASE STUDY

The case study developed in this work aimed to evaluate our approach comparing with a traditional way for creation of accessible web menus. To analyze differences between the codes and difficulties faced by participants for completing tasks, we recorded three dependent variables: task completion rate, time spent to performed each task, number of requirements achieved per task.

Each participant undertook two tasks on different websites. These tasks were to make the main menu accessible using different approaches. The first task used a traditional way and the second used our approach. Order of presentation of the combinations was counterbalanced between participants.

6.1 Administration

The case study was administered at the Institute of Mathematics and Computer Sciences (ICMC), University of São Paulo (USP). Graduate students were involved in the study with voluntary participation. All our participants took part in the study in their daily environment and using their own computers and programming tools.

6.2 Participants

Since the case study is a qualitative research, our main goal was to involve discerning the various perspectives of the researcher, the case/participant, and others, which may or may not converge [26, 15][4]. We have conducted the study with five participants with age between 25 and 32. All the participants were male. Everyone had initial knowledge in web programming languages, regarding undergraduate courses and without professional experience, but with low level of knowledge in web accessibility. It is important to note that 67% of participants did not know how to create an accessible web menu, in other words, they did not know to use WCAG and WAI-ARIA guidelines. However, these participants mentioned some basic requirements for web menus can be accessible.

6.3 Materials

For this study, we created two websites: the first one was a tourism website with a drop-down menu and the other one was a food recipes website with a flyout menu. Each menu consisted of three level of items in order to evaluate the compliance of all accessibility requirements for web menus. Purposely, both menus had few accessibility concepts.

Figure 3 shows the *drop-down* menu, which comprises 23 items that are divided into 5 items on the first level, 13 items on the second level, and 5 items on the third level.

[3]http://www.eclipse.org/xtend/

[4]See more at: https://goo.gl/ZrwaoL

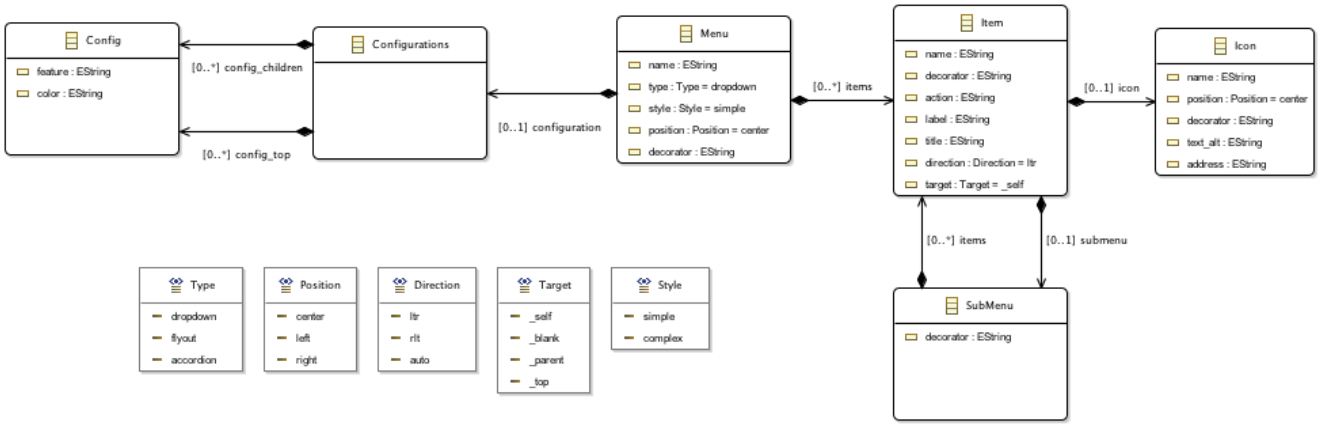

Figure 2: AMenu meta-model

We developed this menu without regard to any aspect of accessibility. Besides, the website was developed with a simple layout without using any CSS framework, as well as we provided interaction with submenus using no Javascript code.

Figure 4 shows the *flyout* menu, which comprises 34 items that are divided into 7 items on the first level, 5 items on the second level, and 22 items on the third level. The HTML code was developed using some correct semantic elements, such as `ul` and `li` tags. In addition, we used a CSS framework to create the website interface and avoided the use of Javascript code to provide interaction with submenus.

6.4 Procedure

Our study had four parts. In the first part, participants were briefed about the study and we took their consent. At this stage we did not tell them that they would perform different tasks. We thought this would affect the results as they might be biased by trying to please the person conducting the study. At this stage, we just told them that we are doing a study about creation of accessible menu and asked them to participate. In the second part, we collected their ethnographic data, and in the third part participants were asked to complete the tasks and their performance was observed. In the last part, we applied a questionnaire to know details about what they felt to perform the tasks.

6.5 Research Questions

The main goal of this study was to compare the developers' experience to create of accessible web menus using AMeG against traditional methods, and to understand the parameters that could affect this. We have tried to identify the main difficulties for creating accessible web menus. In order to do that we mainly investigated the following research questions:

Q1 *Are participants, who performed tasks using AMeG, more satisfied with their performance, in terms of task completion, number of achieved requirements, and level of accessibility of the web menus?* This question aims to investigate our major goal and it aims to quantify this in terms of task completion time, number of achieved requirements in each task, and level of accessibility of the web menus that were developed.

Q2 *Are web menus developed using AMeG more accessible than those developed using traditional methods?* We guess the AMeG can generate web menus with all accessibility requirements. Therefore, this question aims to investigate the quality and accessibility of each web menu through the source code generated by the participants.

Q3 *Does the AMenu language itself as a barrier, any kind of problem, that prevents the use of the AMeG?* This question aims to investigate whether the participants, who used the AMeG, had some kind of difficulty to complete the tasks, such as increased in development time and high learning curve.

7. RESULTS AND DISCUSSION

This section presents the results from the case study. The data collected from the ethnographic information questionnaire show that 4 of 5 participants have little or no experience in developing accessible web systems. The tasks took an average of two hours to complete. Figure 5 shows the time spent by each participant to complete the tasks. The participants took longer time to perform tasks with AMeG, however, this time includes the learning process of the AMenu language, because the first contact with the language occurred during the case study. We guess this time may decrease with the increasing the frequency of use of AMeG. Another point that could improve performance in the tasks is the implementation of a graphical interface for creating menus, which would reduce the learning curve of developers through a more intuitive interaction. Nevertheless, it is important to note that all accessibility requirements required in the case study were achieved.

Table 1 shows the results of each task using the traditional method. We analyze the artifacts produced by the participants in order to find out what percentage has been fulfilled for each of the following accessibility requirements:

Req. 1 Supporting keyboard navigation by using TAB, Enter, Space, and Arrow keys. The functionality of the keys should follow the recommendations of the WAI-ARIA;

(a) Screen capture of menu

```
<div class="sub-nav">
  <div class="menu-item">
    <a href="../e">Antes de Viajar</a>
    <div class="sub-nav">
            <div class="menu-item"><a href="../e">Dicas</a></div>
            <div class="menu-item"><a href="../e">Documentos</a></div>
            <div class="menu-item"><a href="../e">Saúde</a></div>
            <div class="menu-item"><a href="../e">Viajantes especiais</a>
      </div>
            <div class="menu-item"><a href="../e">Serviços</a></div>
        </div>
  </div>
  <div class="menu-item"><a href="../e">Ecoturismo</a></div>
  <div class="menu-item"><a href="../e">Centros Urbanos</a></div>
  <div class="menu-item"><a href="../a">Lugares Românticos</a></div>
  <div class="menu-item"><a href="../e">Mochileiros</a></div>
  <div class="menu-item"><a href="../e">Praia e Descanso</a></div>
  <div class="menu-item"><a href="../e">Turismo Individual</a></div>
</div>
```

(b) Snippet of the source code

Figure 3: Drop-down menu

(a) Screen capture of the menu

```
<li class="menuparent"><a href="/receitas/carnes">Carnes</a>
    <ul>
      <li><a href="/receitas/aves/" title="Receitas de Aves">Aves</a></li>
      <li><a href="/receitas/carnes/" title="Receitas de Carnes">Carnes</a
      ></li>
      <li><a href="/receitas/peixes/" title="Receitas de Peixes">Peixes</a
      ></li>
    </ul>
</li>
<li class="bmenuparent"><a href="/receitas/pratos-rapidos">Pratos Rápidos
      </a>
    <ul>
      <li><a href="/receitas/microondas/" title="Receitas de Microondas">
      Microondas</a></li>
      <li><a href="/receitas/paes-e-lanches/" title="Receitas de Pães e
      Lanches">Pães e Lanches</a></li>
    </ul>
</li>
```

(b) Snippet of the source code

Figure 4: Flyout menu

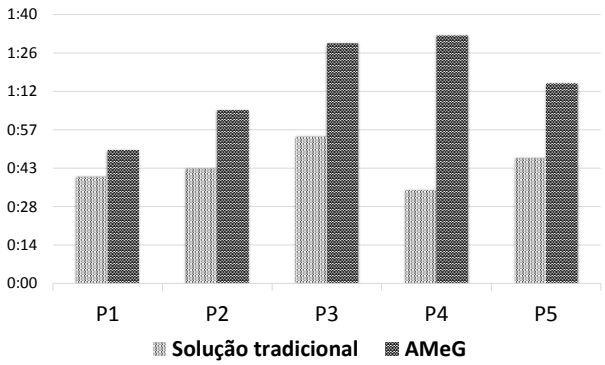

Figure 5: Time spent to complete tasks

Req. 2 Defining a semantic structure using the correct HTML elements. The menu structure should have at least the tags `nav`, `ul`, and `li`;

Req. 3 Using focus indicator. Keyboard navigation through menus should be clearly visible when the focus moves. The behavior should be the same when using the keyboard.

Req. 4 Providing all functionalities by both mouse and keyboard. Users must be able to access all information on the menu, regardless of what input it is using.

Req. 5 Using correctly the ARIA attributes. The ARIA provides an ontology of roles, states and properties that allow the accessibility and interoperability of the interactive elements. Therefore, the menu structure should have at least the following properties: `aria-hidden`, `aria-expanded`, `role`, `aria-haspopup`, `aria-orientation`, and `aria-lablledby`;

As can be seen in Table 1, only 55.3% of the requirements have been fulfilled. This result reflects the level of knowledge of the accessibility guidelines that each participant has. It is also possible note that the most difficult requirement to be fulfilled is the Req. 5, since the WAI-ARIA provides general context information, which hinders the understanding of how to apply the solutions directly in the source code. It is important to properly use these solutions, since assistive technologies use these attributes to inform the user about the status and behavior of a particular menu item.

Although the development time was higher than the traditional solution (Figure 5), the AMeG complied with 100% of the basic requirements, an additional 44.7% compared with the traditional method (Table 1). In addition, the solutions provided by AMeG were developed based on accessibility guidelines (WCAG e WAI-ARIA). Thus, all generated menus are in conformity with these guidelines. Therefore, web menus developed using AMeG are more accessible than those developed using traditional methods. It answers the research question **Q2** defined in Section 6.5.

However, we do not test the web menus with end users in order to verify the completeness of all the accessibility

Table 1: Analysis of artifacts produced from a traditional method

Participants	Type of menu	Req. 1	Req. 2	Req. 3	Req. 4	Req. 5	Req. mean
P1	flyout	21.4%	66.7%	0%	50%	37.5%	35.1%
P2	flyout	85.7%	66.7%	50%	100%	37.5%	68%
P3	drop-down	75%	66.7%	75%	100%	37.5%	70.9%
P4	drop-down	0%	66.7%	50%	50%	0%	33.3%
P5	drop-down	78.6%	66.7%	100%	100%	0%	69.1%
	Total mean	**52.1%**	**66.7%**	**55%**	**80%**	**22.5%**	**55.3%**

requirements. Thereby, for purposes of data analysis, we considered only the compliance with accessibility guidelines.

Finally, Figure 6 shows a comparison between the traditional method and the AMeG of the average time to perform the tasks. The bars indicate the average time it took developers to create the menus using the AMeG, while the line indicates the estimation of time that they should have spent to comply with 100% of the requirements using a traditional method.

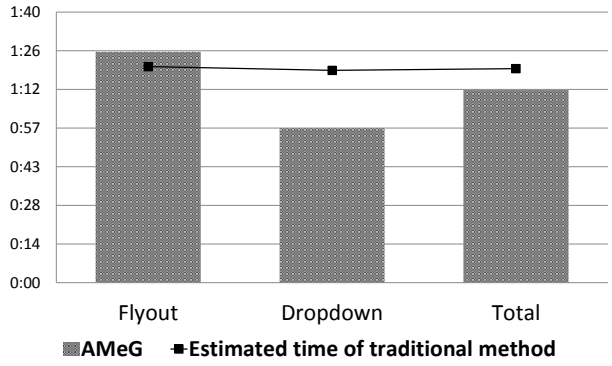

Figure 6: Time comparison between the traditional method and the AMeG

According to Figure 6, the flyout menu spent an average of 1h26min to be developed using the AMeG, supposedly representing an increase of 6.8% (5min) in the time it would take using a traditional solution. We believe that the traditional approach was better at this task due to the initial structure of the menu that we provide, since the developers did not restructure the HTML code to make it accessible. However, it is important to note that using the traditional approach only complied with 51.6% of the accessibility requirements that were specified in the case study. On the other hand, when comparing the development of the drop-down menu, which had a totally inaccessible structure (Figure 4), the AMeG had a performance gain of 37.6%, i.e. a supposed reduction of 22 minutes in development time.

By analyzing the overall mean shown in Figure 6, the AMeG supposedly represents a reduction of 11.2% (8 minutes) in development time and complies with all accessibility requirements, while the traditional method took around 4 minutes extra to develop only 54.6% of all requirements.

Although the tasks took more time using AMeG, 4 of 5 participants preferred to use it for creating web menus, considering they have had more difficulty using a traditional method, as shown in Figure 7. This preference indicates that participants were more satisfied with their performance when used the AMeG. In addition, by analyzing the Figure 7, there was a reduction of 50% in the average level of difficulty for creating accessible web menus using the AMeG. Therefore, the AMenu language is not itself a barrier or has any kind of problem that prevents the use of the AMeG. It answers the research questions **Q1** defined in Section 6.5.

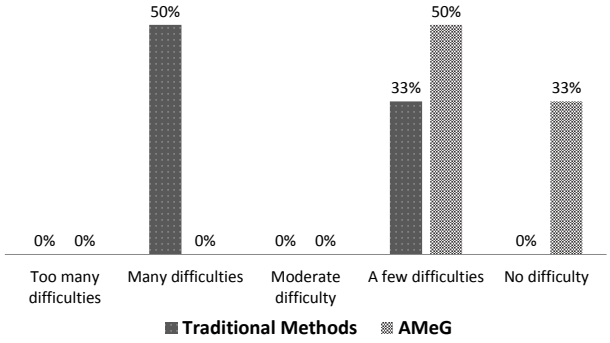

Figure 7: Difficulty of the participants to perform the tasks

Despite the preference of the participants to use the AMeG, they pointed out some specific difficulties that hindered the performance of tasks, such as lack tabbing in the editor and lack of graphic examples for stylization of menus. However, all participants, who preferred to use AMeG, indicated that AMenu language is easy to use and they would return to use the AMeG for creating accessible web menus. It answers the research questions **Q3** defined in Section 6.5.

8. CONCLUDING REMARKS

This paper presented a tool to help developers create accessible web menus. The use of our approach ensures that the menus are generated in accordance with the accessibility guidelines, as well as increases productivity in the development, since developers do not need to spend time dealing with the technical details of accessibility. In addition, the AMenu language is an essential part of our approach, since it encapsulates most of the required accessibility requirements.

From the case study to the developers, it was possible to evaluate the feasibility of our approach, which showed an 50% reduction in the level of difficulty for creating accessible menus. In addition, we identified potential improvements to the tool that can contribute to wide acceptance of AMeG by web development community.

We have developed AMeG specifically to create accessible web menus. However, their structure allows the generation of other accessible elements. In other words, it is need only to define new models of transformation and to update the

AMenu language. The next steps of this study are: develop a graphical interface for creating menus in AMeG, include new types of menus, and conduct more testing with developers, especially with accessibility experts.

9. ACKNOWLEDGMENTS

We thank the participants that contributed to this study, and the FAPESP (2013/23966-8) for having funded this research.

10. REFERENCES

[1] eMAG, Modelo de acessibilidade do Governo Eletrônico, Versão 2.0, 2005. http://goo.gl/8cOQ4y.

[2] Mobile web best practice, aug. 2008. http://www.w3.org/TR/mobile-bp.

[3] Web content accessibility guidelines 2.0, dez. 2008. http://www.w3.org/TR/WCAG20/.

[4] Accessible rich internet applications (WAI-ARIA), mar. 2014. http://www.w3.org/TR/WCAG20/.

[5] H. L. Antonelli and R. P. M. Fortes. AMeG: Accessible Menu Generator - Manual técnico e operacional. Technical Report 403, São Carlos - SP, jan. 2015.

[6] A. Burrell and A. Sodan. Web Interface Navigation Design: Which Style of Navigation-Link Menus Do Users Prefer? In *Data Engineering Workshops, 2006. Proc.. 22nd International Conf. on*, pages 42–42, 2006.

[7] M. Cooper, A. Kirkpatrick, and J. O. Connor. Techniques for WCAG 2.0, 2009. http://www.w3.org/TR/WCAG20-TECHS/.

[8] M. Eysholdt and H. Behrens. Xtext: Implement your language faster than the quick and dirty way. In *Proc. of the ACM International case study Companion on Object Oriented Programming Systems Languages and Applications Companion*, SPLASH '10, pages 307–309, 2010.

[9] R. P. Fortes, A. L. Dias, F. D. Grillo, and P. C. Masiero. Creating a project towards universal access: is it possible? In *INTERACT 2013 Workshop on Rethinking Universal Accessibility: A Broader Approach Considering the Digital Gap*, Cape Town, South Africa, 2013.

[10] A. P. Freire. Disabled people and the Web: User-based measurement of accessibility. Doctor of philosophy, Departament of Computer Science, University of York, United Kingdom, 2012.

[11] A. P. Freire, C. M. Russo, and R. P. M. Fortes. A survey on the accessibility awareness of people involved in web development projects in brazil. In *Proc. of the 2008 International Cross-disciplinary case study on Web Accessibility (W4A)*, W4A '08, pages 87–96, 2008.

[12] J. Gunderson, H. B. Rangin, and N. Hoyt. Functional Web Accessibility Techniques and Tools from the University of Illinois. In *Proc. of the 8th International ACM SIGACCESS case study on Computers and Accessibility*, Assets '06, pages 269–270, 2006.

[13] B. Kelly, D. Sloan, S. Brown, J. Seale, H. Petrie, P. Lauke, and S. Ball. Accessibility 2.0: People, Policies and Processes. In *Proc. of the 2007 International Cross-disciplinary Conf. on Web Accessibility (W4A)*, W4A '07, pages 138–147, 2007.

[14] B. Kelly, D. Sloan, L. Phipps, H. Petrie, and F. Hamilton. Forcing Standardization or Accommodating Diversity?: A Framework for Applying the WCAG in the Real World. In *Proc. of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, W4A '05, pages 46–54, 2005.

[15] J. Lazar, J. H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley Publishing, United Kingdom, 2010.

[16] G. M. Poor, L. M. Leventhal, J. Barnes, D. R. Hutchings, P. Albee, and L. Campbell. No User Left Behind: Including Accessibility in Student Projects and the Impact on CS Students' Attitudes. *Trans. Comput. Educ.*, 12(2):5:1–5:22, Apr. 2012.

[17] C. Power, A. P. Freire, H. Petrie, and D. Swallow. Guidelines are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the web. In *Proc. of CHI'12*, CHI'12, pages 5–10, New York, NY, USA, 2012. ACM.

[18] L. G. Reid and A. Snow-Weaver. WCAG 2.0: A Web Accessibility Standard for the Evolving Web. In *Proc. of the 2008 International Cross-disciplinary Conf. on Web Accessibility (W4A)*, W4A '08, pages 109–115, 2008.

[19] J. M. N. Ribeiro. Web design patterns for mobile device. Mestrado em multimédia, Faculdade de Engenharia da Universidade do Porto, Portugal, 2011.

[20] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 2009.

[21] E. P. B. Santos and R. P. Fortes. 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*, pages 127–129, 2010.

[22] J. Thatcher, W. Cynthia, S. Henry, S. Swierenga, M. Urban, M. Burks, and P. Bohman. *Constructing Accessible Web Sites.*, volume 1. Glasshaus, 2002.

[23] S. Trewin, B. Cragun, C. Swart, J. Brezin, and J. Richards. Accessibility challenges and tool features: An ibm web developer perspective. In *Proc. of the 2010 International Cross Disciplinary case study on Web Accessibility (W4A)*, W4A '10, pages 32:1–32:10, 2010.

[24] W. M. Watanabe, R. P. M. Fortes, and A. L. Dias. Using acceptance tests to validate accessibility requirements in ria. In *Proc. of the International Cross-Disciplinary case study on Web Accessibility*, W4A '12, pages 15:1–15:10, 2012.

[25] M. V. Welie and G. C. V. D. Veer. Pattern languages in interaction design: Structure and organization. In *Proc. Interact '03, M. Rauterberg, Wesson, Ed(s). IOS*, pages 527–534. IOS Press, 2003.

[26] R. Yin. *Case Study Research: Design and Methods*. SAGE Publications, Thousand Oaks, CA, 2nd. edition, 1994.

[27] B.-M. Yu and S.-Z. Roh. 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*, 53(11):923–933, 2002.