Hindawi
Advances in Human-Computer Interaction
Volume 2018, Article ID 1389208, 17 pages
https://doi.org/10.1155/2018/1389208

Research Article

Incorporating Accessibility Elements to the Software Engineering Process

Wesley Tessaro Andrade, Rodrigo Gonçalves de Branco, Maria Istela Cagnin, and Débora Maria Barroso Paiva

Federal University of Mato Grosso do Sul (UFMS), Av. Costa e Silva, s/n, Cidade Universitária, Campo Grande, MS, Brazil

Correspondence should be addressed to Débora Maria Barroso Paiva; dmbpaiva@gmail.com

Received 30 May 2018; Accepted 17 October 2018; Published 2 December 2018

Academic Editor: Hideyuki Nakanishi

The expansion of web is a phenomenon that brings several challenges in different segments of the society. Accessibility is one of these challenges and it is related to the digital inclusion and social welfare of the population. Thus, making accessible software available can contribute to solution of problems that currently exist in relation to access to information and services by all citizens. The purpose of this article is to present an approach that integrates accessibility to the Software Engineering process. We also present the Acero tool, which provides computational support to the proposed approach. Results were evaluated and we concluded that the use of the proposal reached the objectives, supporting different stages of the development process and contributing to obtain accessible software products.

1. Introduction

The availability of computers, electronic equipment, and Internet has made a decisive contribution to major changes in modern society. According to the International Telecommunication Union (ITU (www.itu.int)), subordinate organ of the United Nations Organization (UNO (www.un.org)), in the year 2016 approximately 3 billion people worldwide had access to the Internet, accounting for about 40% of the world's population [1]. Digital inclusion implies being concerned with the needs and demands of all people, including those with special access needs, the elderly, low literacy users, among others. Also, it is important to think about how to make the new interactive applications [2] available to everyone, for example, the smart city and the smart home applications including geopositioning, Internet of Things, and other resources [3, 4].

The World Health Organization (WHO (www.who.int)) estimated in 2014 that approximately 1 billion people in the world had some kind of deficiency ranging from visual and auditory deficits to cognitive and motor difficulties. According to WHO, deficiency is the term used to refer to individuals who have limitations or lack of anatomical, physiological, and/or intellectual structure [5].

In general, web accessibility means that people with disabilities, reduced skills, or situationally induced impairments are able to access, navigate, interact, and contribute to information on the web [6].

In spite of the importance to offer resources that enable digital inclusion, web accessibility has not been a priority [7, 8] and there are some justifications to this, such as the lack of technical knowledge of software engineers and developers (little emphasis is given to the subject during the academic training of students), the lack of tools that support the inclusion of accessibility quickly and simply throughout the software lifecycle, and the predominant software development culture that allocates insufficient resources (time, people) to the design and evaluation of graphical interfaces [9, 10]. However, companies and professionals are now noticing that those who neglect the website usability and the development of accessible products may lose an expressive number of users of their software systems [11, 12].

Software Engineering plays a fundamental role in the development of accessible applications, since it can promote the integration between methodologies and specific accessibility techniques and activities at the software development process. According to Sherman [13] and Groves [14] the benefits of incorporating accessibility during the software

development process are greater than the costs involved, as there is an increase in the number of users and value added to the final product. Additionally, maintenance activities, generally expensive, can be avoided for inclusion of accessibility.

Software development encompasses many activities, such as requirements specification, design, coding, and testing. It is possible to use computers to aid the entire software development process by using Computer-Aided Software Engineering (CASE) tools that support the execution of repetitive tasks, reduce the complexity of development, and improve productivity.

Therefore, considering the difficulties inherent to the development of accessible web applications and the possibility of extending and improving the existing CASE tools, this article is proposed with a double objective. Firstly, we present the Acero approach, which integrates accessibility into the software development process. Secondly, we present the Acero tool, which provides the computational support to allow the automation of the Acero approach. The inclusion of accessibility is transparent to software engineers and developers, in other words, when using the Acero tool, professionals will be able to generate accessible applications without being experts in the area.

This article is organized as follows: Section 2 presents an overview about accessibility; Section 3 presents related work; Section 4 discusses the background of the approach and tool proposed in this article; Section 5 presents the Acero approach and tool; and Sections 6 and 7 present the empirical study and conclusions, respectively.

2. Web Accessibility Concepts

In many countries, web has become the main source of access to government, educational, news, and leisure information and services. Consequently, its use is replacing or decreasing the use of resources that were once heavily used, such as newspapers and magazines in print versions. Therefore, it is necessary that web be accessible in order to provide equality, opportunity for access and interaction for all people who want to use it [6, 15].

Considering the need to provide accessible content on web, a number of governments have instituted laws that make accessible information available, even if only within the scope of government sites, with the aim of guaranteeing equality of opportunity in access. It is common for these nations to adopt national or international guidelines to standardize the development of products and the availability of content on web.

The W3C international organization launched an initiative whose main mission is to coordinate international, technical, and human efforts to improve web accessibility [6, 15, 16]. It is responsible for the important set of accessibility guidelines, called web Content Accessibility Guidelines (WCAG) [17].

WCAG is a set of documents that explains, through guidelines and recommendations, how to make web content accessible to people with disabilities. WCAG is intended for front-end developers, but it is also useful for assisting developers of assessment tools, developers of audit tools, and developers of quality assurance and validation tools.

WCAG 1.0 was published as the current W3C standard in May 1999 and consists of 12 guidelines, divided into checkpoints and properties associated with each of them. The document presents three main groups:

(i) Level 1: developers must meet a number of success criteria so that one or more groups of people can access the web content. If all success criteria associated with this level are met, the site will have conformity "A".

(ii) Level 2: developers should take an additional set of testable success criteria because; otherwise, one or more groups will have difficulty accessing the information. Compliance with this level is described as "AA".

(iii) Level 3: developers could take a more complex set of testable success criteria to make it easier for some groups to access the web. Compliance with this level is referred to as "AAA".

Due to the evolution and creation of new web technologies, the W3C needed to make improvements to meet these new tools and enable the scalability of the WCAG 1.0 being proposed, therefore, the standard WCAG 2.0 [18].

The definition of WCAG 2.0 standard was based on WCAG 1.0 but new recommendations were also made: one of the main modifications was that instead of the fact that each guideline has checkpoints or checklists, 61 successful criteria were presented, which are declarations that can be automatically or manually tested in order to check whether the web content is accessible or not. It is by means of these success criteria that the conformity levels "A", "AA", or "AAA" are established [19]. WCAG 2.0 is composed of 12 guidelines organized under four fundamental principles [17]:

(1) Perceivable: data and components of the interfaces must be presented to users in a perceptible way.

(2) Operable: user interface components must be operative, regardless of the user's needs.

(3) Understandable: contents and operations on the interfaces should be understandable.

(4) Robust: contents and information should be reliably interpreted by a wide variety of tools, including assistive technologies.

Although WCAG is a nonmandatory technical guidance, some countries use or are inspired by this standard with the aim of providing accessible content in their governmental portals [20]. Some examples are Canada [21], Japan [22], Ireland [23], Italian [24], and Brazil [25].

WCAG 2.1 was published as a W3C Recommendation June 2018 [26]. It extends WCAG 2.0 and content that conforms to WCAG 2.1 also conforms to WCAG 2.0. Following WCAG 2.1 guidelines developers will make content more accessible to a wider range of people with disabilities, including accommodations for blindness and low vision, deafness and hearing loss, limited movement, speech disabilities, photosensitivity, and combinations of these, and

some accommodation for learning disabilities and cognitive limitations.

2.1. Assessment Methods and Measurement of Accessibility. The legislations and standards discussed above provide technical guidance on what should be offered in the project and reach only high-level objectives [19].

Evaluation methods contribute to identification of specific failures and coding errors [27]. For greater effectiveness, evaluation methods should be used in conjunction with existing accessibility guidelines and standards. A number of methods for accessibility assessment can be used, such as user-based evaluation [28, 29], conformance evaluation methodology [30], and automated evaluations [31]. It is important to note that evaluation methods can be used together or individually in the desired order [19].

In this research, the use of automated evaluations using existing tools, such as Achecker (http://achecker.ca/checker/index.php) and Access Monitor (http://www.acessibilidade.gov.pt/accessmonitor/), was considered to evaluate the quality of the final application obtained with the utilization of the proposed methodology.

3. Related Work

Recent works have been concerned with accessibility requirements from the early stages of the software development process through tasks and tools that help to elicit and properly implement those requirements. In the latter case, several studies have used model-driven development (MDD) to provide a metamodel of interest domain (e.g., embedded systems, e-commerce, industrial automation systems, etc.) and a metamodel of UI layer, with accessibility requirements incorporated.

Krainz et al. [32–34] propose a MDD approach, in that a metamodel for the domain apps is created in Domain Specific Language (DSL) with accessibility requirements included, and use a set of tools to transform the metamodel into app source code. The outcome from this transformation process is an app prototype with accessibility features (for example, content description for integrated screen reader support or active voice output in selected parts of the app).

Other authors [35] present an approach based on user-centered design (UCD) and on MDD for developing web application and industrial automation systems with accessibility. Models have been elaborated to describe particular UI aspects, as structure or behavior, considering accessibility requirements, as well as domain requirements belonging to industrial automation systems (such as ticket vending machines, washing machines, or automated teller machines (ATM) systems).

In addition, González-García et al. [36] used a MDD approach to create an accessible media player. In a similar way, Zouhaier et al. [37] research adaption of accessible UIs based on models and Miñón et al. [38] show the generation of accessible user interfaces in ubiquitous environments from models.

Other works define software development processes that provide activities and practices in order to produce accessible software, but do not offer support tools.

An agile inclusive process model was defined by Bonacin et al. [39]. This process is based on agile principles and values and it is focused on accessibility and usability of the final product. Its main principles are promote the participation of the users and other stakeholders with the universal access and inclusive design values (i.e., UCD); construct a shared vision of the social context; include more than just technical issues in the development of the system; and promote the digital inclusion through participatory activities. To attend these principles, the process proposed includes many activities related to these issues, including experts analysis, low fidelity prototyping, user acceptance analysis, and workshop with the users.

Rossvoll et al. [40] show an iterative approach with user involvement from the beginning to the end of the software project, containing a set of recommendations based on a UCD process which includes user testing with disabled people and based on experiences from projects for inclusive access developed by the authors. The approach contains both high-level recommendations, such as which overall research methodology to apply, as well as detailed low-level guidelines, such as which activities concerned to accessibility to include in the project workflow and when.

Dias et al. [41] extended a classic model of web application development process to incorporate activities related to users' profile with disabilities and their needs, in order to include accessibility and usability nonfunctional requirements during development. To facilitate the elicitation of these nonfunctional requirements, the authors provide a checklist that contains a list of this type of requirement based on accessibility guidelines and usability heuristics, as well as giving the main needs of users with each type of disability.

Sanchez-Gordon et al. [42] define a software development process, fit to small software enterprises to attend their constraints of staff and budget, which includes accessibility-related tasks in following activities: initiation, analysis, design, construction, integration and test, and delivery. The authors discuss briefly how apply each accessibility-related task when using agile development and they also indicate existing tools, checklists, and standard that can be used during the software development.

Our work becomes a differential in relation to the aforementioned works because it offers a process, supported by tools, that is concerned with accessibility requirements during all the phases of software development. The tools are independent of the application domain and can be used by software engineers according to their needs and skills in order to develop software to attend all users, including users with permanent or temporal disabilities.

4. Background

The approach and tool proposed in this article is the result of some researches that were later integrated. The results of the researches presented in this section can be divided

into three parts. In Section 4.1 the methodology for making integration between accessibility elements and software processes feasible is presented. In Section 4.2 the AccTrace tool is presented, a preliminary software tool developed to contemplate the proposed methodology. In Section 4.3, the Homero Framework is presented, which provides support for code generation and webpage creation in accordance with the WCAG 2.0 guidelines.

4.1. A Methodology for Developing Accessible Web Applications. Maia et al. [43] proposed a process for the development of accessible web applications, called MTA, based on ISO/IEC 12207 [44], which suggests the introduction of accessibility tasks in software development subprocesses. The MTA suggests adapting the subprocesses of ISO/IEC 12207 at all stages of development in order to enable the generation of accessible software products. Results of other authors also contributed to the MTA elaboration [45, 46]. The subprocesses and their tasks are as follows:

(1) System requirements elicitation (the software owner and the final users provides information and the accessibility specialist records them)

  (a) Identify the accessibility requirements of the system

    (i) Identify user characteristics: the abilities (and disabilities) of the final users including perceptual, cognitive, motor, etc.

    (ii) Identify domain requirements: the tasks that need to be supported, social and cultural dynamics, environmental factors, and so on

    (iii) Identify technological requirements: the availability of hardware, software, plug-ins and assistive technologies in the context of final users

(2) System requirements analysis (the software owner, the final users, the development team, and the accessibility specialist refine information)

  (a) Specify the accessibility requirements of the system

    (I) Specification is based on answer to the questions

      (i) User characteristics, who is your target audience? What is the level of expertise the target audience have the subject area of the website?

      (ii) Domain requirements, what is the purpose of the website? What sort of tasks do you expect users to be able to perform using the site?

      (iii) Technological requirements, what assumptions can you make about the browsing and assistive technology available to the target audience and their knowledge of that technology? What other ways already exist to provide access to the information or services provided by the website in question?

  (b) Evaluate the accessibility requirements of the system

    (i) Criteria established by ISO/IEC 12207 are considered: system accessibility requirements are analyzed for relevance, correctness, and testability; consistency and traceability are established between the system accessibility requirements and the customer's requirements baseline

(3) System architectural design (the development team and the accessibility specialist are responsible for producing the design)

  (a) Allocate accessibility requirements to system elements

    (i) The software system is decomposed into a set of hardware and software components together with the assignment of responsibilities for each component. Accessibility requirements are allocated on such components

  (b) Evaluate the architectural design of the system in relation to accessibility requirements

    (i) Criteria established by ISO/IEC 12207 are considered: system design is analyzed for consistency and traceability between the system accessibility requirements and system architecture design

(4) Software requirements analysis (the requirements engineer, the accessibility specialist, and the final users collect accessibility requirements)

  (a) Establish the accessibility requirements of the software

    (i) Use of requirements elicitation techniques (e.g., questionnaires, interviews) to obtain accessibility requirements

  (b) Evaluate software accessibility requirements

    (i) Criteria established by ISO/IEC 12207 are considered: software accessibility requirements are analyzed for correctness and testability, the impact of software requirements on the operating environment is understood, consistency and traceability are established between the software accessibility requirements and the system accessibility requirements, and prioritization for implementing the software accessibility requirements is defined

(5) Software design (the web designer and the accessibility specialist are responsible for producing the design)

(a) Design the accessible external interfaces

(i) Provide alternate or multiple views to address trade-offs between different types of user groups and to optimize the user experience of those user groups

(ii) Establish the layout of the accessible interface elements, such as labels, images, and text editing fields

(iii) Define other elements considering abilities and disabilities, for example, color and size for low vision final users

(iv) Define Accessible Navigational Project

(b) Evaluate the accessibility of the software project

(i) Criteria established by ISO/IEC 12207 are considered: software accessibility requirements are analyzed for correctness and testability, consistency, and traceability between accessibility requirements and accessible design

(6) Software construction (the development team and the accessibility specialist implement the accessible external interface)

(a) Specify techniques for accessibility implementation

(i) Identify programming techniques for accessible interface implementation, such as the Advisory Techniques presented in the WCAG document

(b) Codify each software unit according to the accessibility techniques

(c) Plan accessibility tests of each software unit

(i) Prepare accessibility tests for software, identifying what should be tested, how the accessibility test should be run, what data should be used for the tests, and what results are expected

(d) Perform accessibility tests of each software unit

(i) Run the accessibility tests according to the plan and make the necessary corrections

(7) Software integration testing (the testers and the accessibility specialist are responsible for the accessibility testing considering all elements of the software working together)

(a) Plan accessibility tests of the integrated software

(i) Define the procedures, accessibility test data of the integrated software and the expected results

(b) Perform accessibility test of the integrated software

(i) Run the accessibility tests according to the plan and make the necessary corrections

(8) Acceptance testing (the testers, the final users, and the accessibility specialist are responsible for the acceptance testing)

(a) Plan accessibility acceptance test of the software with final users

(i) Define the procedures, accessibility test data for the acceptance test and the expected results considering participation of people with disabilities

(b) Perform accessibility acceptance test considering participation of people with disabilities

(i) Run the accessibility acceptance tests according to the plan and make the necessary corrections

(9) System Testing (the testers and the accessibility specialist are responsible for the accessibility testing considering all elements of the system working together)

(a) Plan accessibility tests in the system

(i) Define procedures to evaluate whether all elements of the system work correctly in the final user environment

(b) Perform accessibility test in the system

(i) Run the accessibility tests according to the plan and make the necessary corrections. Certify compliance with the requirements of the system

MTA was proposed to guide the development process from the initial stages of the project in order to avoid rework in the maintenance phase (in this phase the client usually requests the inclusion of the requirement accessibility, for different reasons, such as the requirement of laws), which can incur high costs. MTA was evaluated by software developers and it was concluded that, considering accessibility tasks integrated to the development process, it was possible to positively influence the final quality of the product in relation to obtain accessible applications.

4.2. Accessibility in the Phases of Requirements Engineering, Design, and Software Coding: A Support Tool. Considering the MTA, Branco et al. [47, 48] developed a plug-in tool for the Eclipse IDE, called AccTrace, which aims to accomplish the following tasks: to associate the accessibility requirements of a software project to the UML models (use case diagram and class diagram) and automatically generate Java classes with the comments of accessibility implementation techniques. It also performed the tracking of the requirements in the different artifacts, generating a traceability matrix.

For the development of tasks proposed by AccTrace, the tool integrates other solutions as follows: the requirements are specified by the Requirement Designer

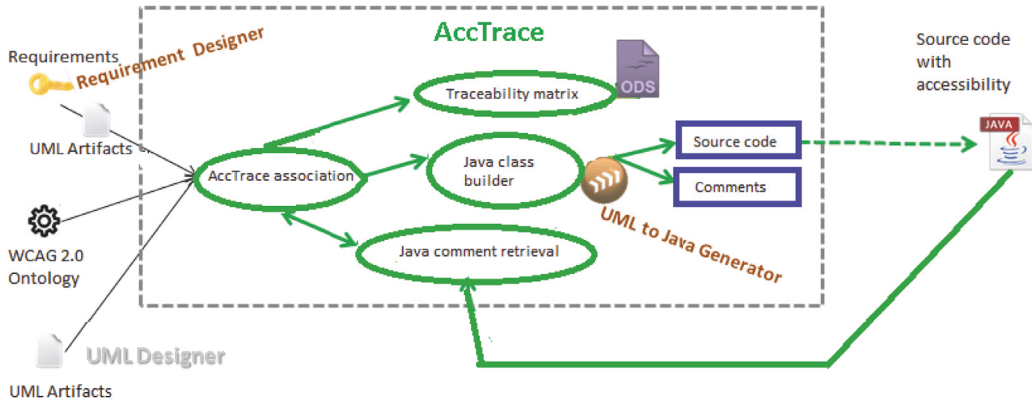

Figure 1: AccTrace task flow to generate the code and the traceability matrix of the requirements [47].

(https://www.obeodesigner.com/en/) plug-in, the UML artifacts are elaborated through the UML Designer (http://www.umldesigner.org/) plug-in and classes are automatically generated using the UML to Java Generator (https://marketplace.eclipse.org/content/uml-java-generator) plug-in. Figure 1 shows the AccTrace task flow to generate the code and the traceability matrix of the requirements.

After defining the accessibility requirements and making the connection between them and the UML artifacts, the AccTrace tool is used to perform the association with WCAG 2.0 Ontology [49]. The ontology is a data model, represented in AccTrace tool as a list of implementation techniques, approaches, success criteria, and tests related to the WCAG accessibility guidelines 2.0. In this way, AccTrace tool allows to create a file (extension .acctrace) that relates the ontology to the artifacts and requirements of the software project.

From the .acctrace association file, it is possible to generate a traceability matrix (extension .ods) that presents, in the form of three tables, the relationships: Requirements x UML Models, Requirements x Ontology, and UML Models x Ontology.

AccTrace tool also supports the creation of Java classes with accessibility comments. The UML to Java Generator plug-in was used as the base, which considers as input the .acctrace association file and the UML artifacts file (.uml extension). Classes are then generated with specific comments that allow the user, directly in his/her code, to retrieve the relevant class relationships, aiding the implementation of accessibility.

The tool has three main views, according to Figure 2 . In the editor (AccTrace Editor View-2) it is possible to generate the associations including the UML models, requirements and implementation techniques. In the requirements view (Requirement Associations View-1) it is possible to visualize which requirements associated with the UML model were selected in the editor. In the Specifications View (Accessibility Specifications View-3) it is possible to visualize the implementation techniques already associated, according to the UML model selected in the editor and the accessibility requirement selected in the view of the requirements. In addition, it is possible to remove the associated implementation techniques.

When selecting the UML model and the requirement, it is possible to associate the accessibility implementation technique (mapped on the Ontology) by right-clicking on the UML model, as shown in Figure 3 . These techniques are linked to the requirements and UML models and are stored. Because the artifacts are described in RDF format (requirements, UML models, and ontology), the links are made from the RDF:ID element. Therefore, any UML model that is described in RDF format can be linked and tracked through the traceability matrix and views in Eclipse.

Considering Figures 2 and 3 it is possible to observe the impact of each user need on the system design. The application "Travel Agency" defined the accessibility requirement "provide text alternatives for any nontext content". The Use Case "Offer Catalog Management" was designed and Guideline 1.1 (described in WCAG 2.0) was associated. The reference to the AEGIS Ontology "G134T3: Load each external or internal style sheet into a CSS validator" may also be observed because it includes the accessibility implementation techniques and other pieces of information that may assist in the implementation of the requirement.

A simple application was modeled to exemplify the impact of including accessibility requirements throughout the software process. The application refers to an Internet search engine using keywords. For this application functional requirements and nonfunctional requirements, especially accessibility requirements were defined: "Make all functionality available from a keyboard", "Make text content readable and understandable", etc. Figure 4 shows the association of accessibility guidelines to use cases and UML classes. Also, Figure 5 illustrates the generated code, highlighting the ViewRenderer class, accessibility comments, and implementation details (guidelines and accessibility techniques).

Once the relationship including requirements, UML models, and accessibility implementation techniques are defined, it is possible to automatically generate the traceability matrix by the AccTrace tool in the Open Document Sheet (ODS) format.

4.3. Homero: A Framework for Supporting the Development of Accessible Web Application Interfaces. The Homero Framework [50, 51] was developed using PHP and aims to simplify

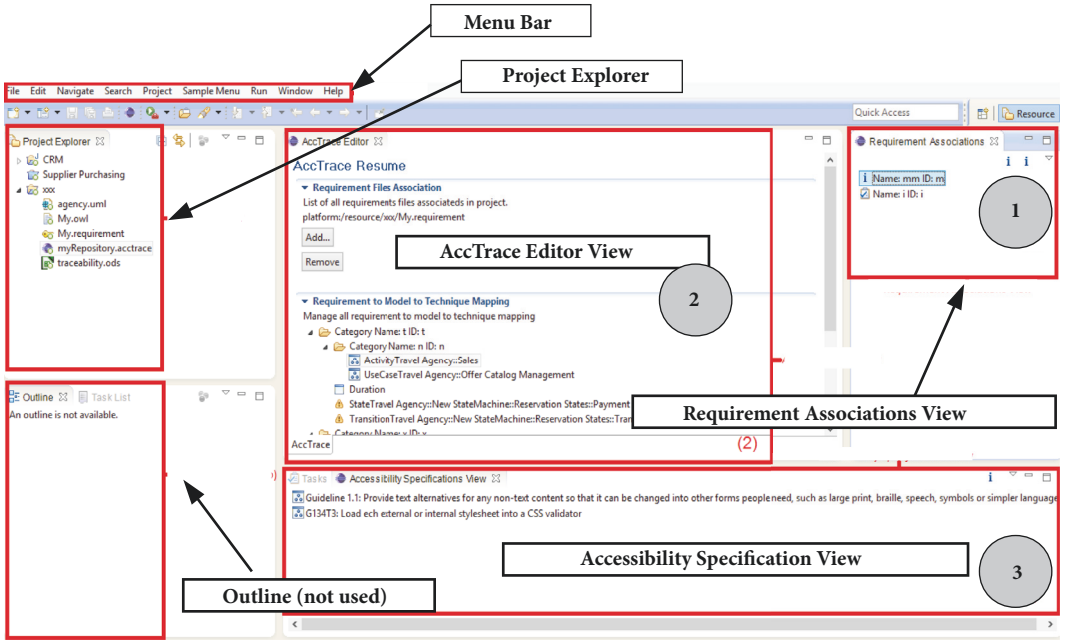

Figure 2: View of the AccTrace tool on the main screen in Eclipse.

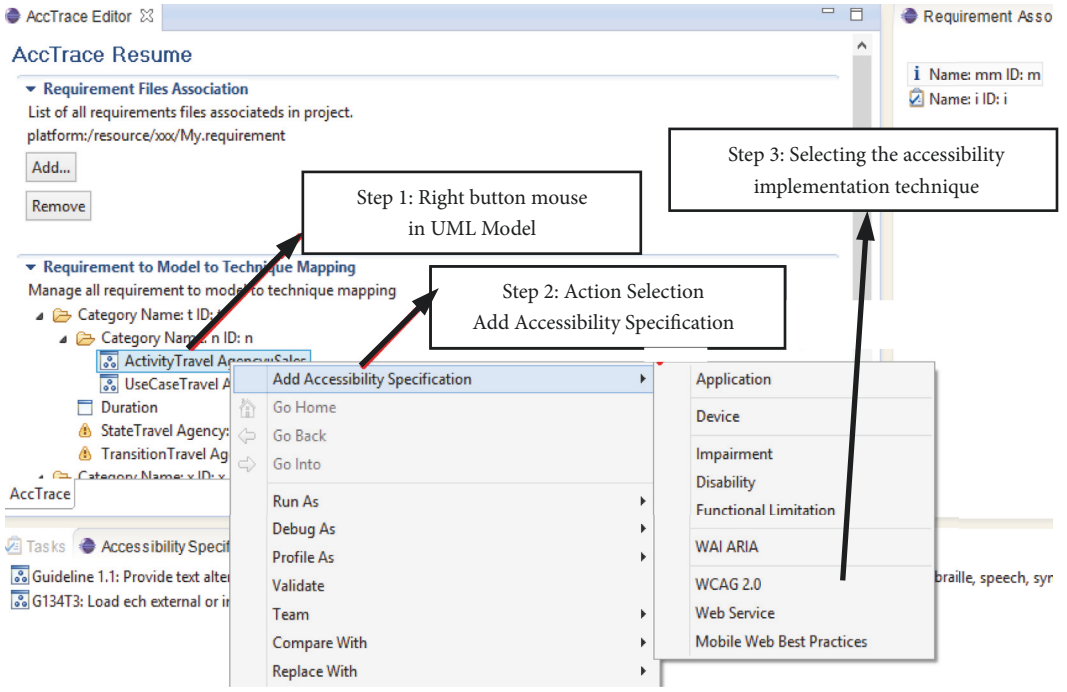

Figure 3: Process to associate the accessibility implementation technique.

the implementation of the WCAG 2.0 accessibility guidelines proposed by W3C. It consists of HTML classes that, when instantiated, provide objects that, when executed, provide an accessible HTML code. Homero provides support for the implementation of various types of HTML elements, such as tables, images, lists, texts, and links.

In Figure 6 is possible analyze a code of an application developed using the Homero Framework. When an object of the image class was created, in line 9, the alternative text was not defined. The second parameter of the constructor was defined with null value, which caused an accessibility error in the final application (Warning-in English: The image assistant text was not specified).

Inclusion of accessibility elements in the phases of the software development process was possible by means of the development of the AccTrace tool and the Homero

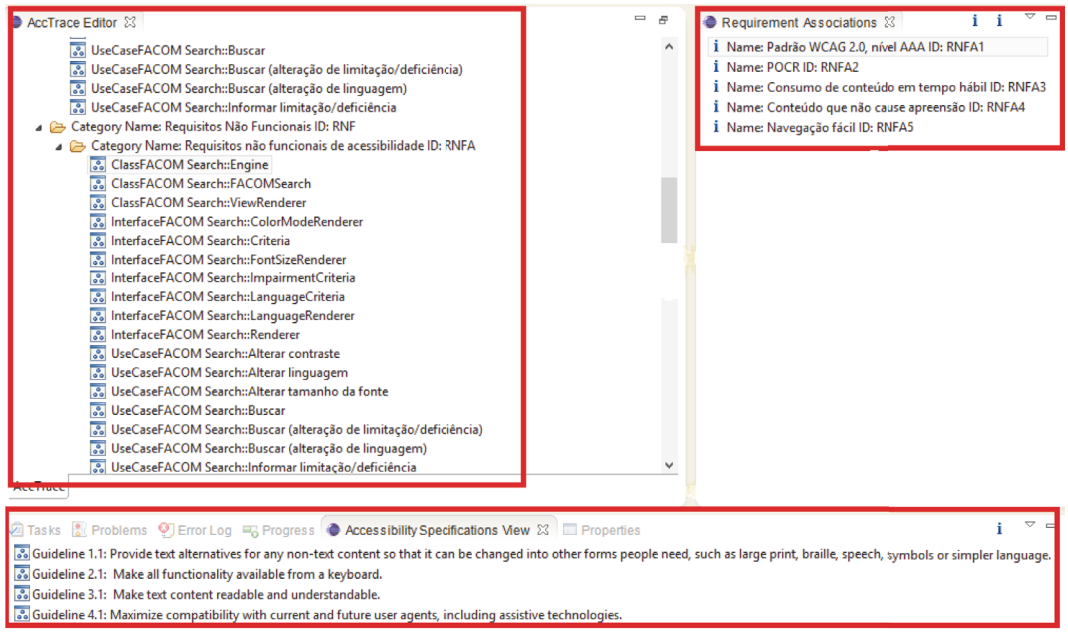

Figure 4: Use cases, UML classes, and accessibility requirements association.

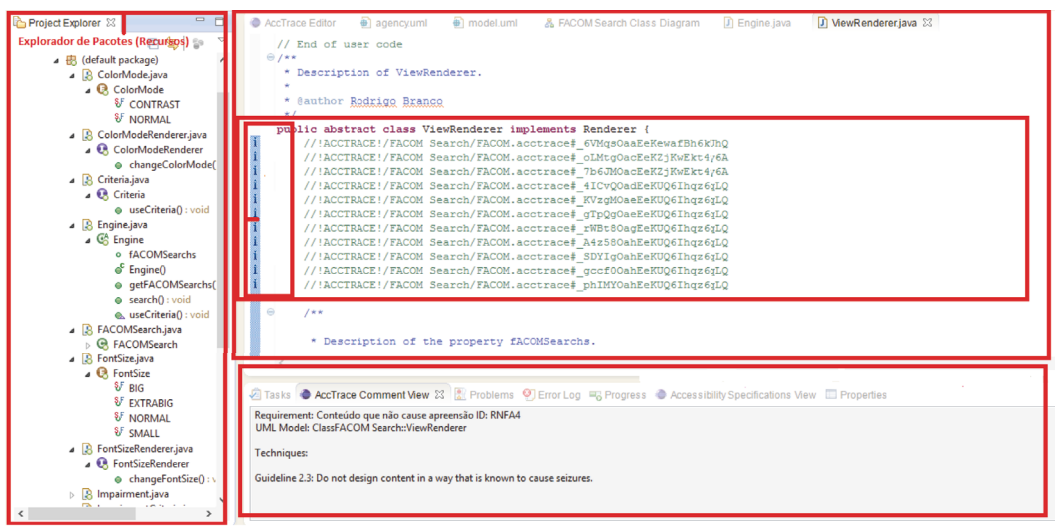

Figure 5: Source code and accessibility comments.

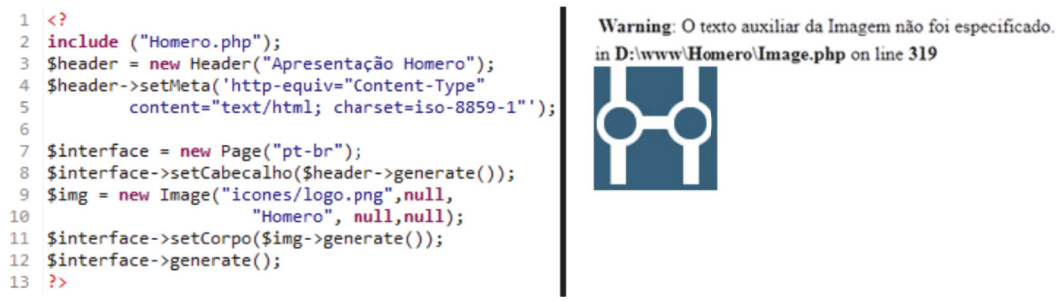

Figure 6: Example of an application developed using the Homero Framework [50].

Framework. However, each of the proposals had different focuses; i.e., the AccTrace tool dealt with the requirements engineering and software design phases and the Homero Framework dealt with the implementation phase. In this way, the integration and creation of a tool that could both support the software development cycle in a unified way and be improved was envisaged.

5. Acero: An Approach and a Tool for Development of Web Accessible Applications

5.1. Acero Approach. The high-level architecture proposed in this project is presented in Figure 7. In the upper layer, the main features designed specifically for the Acero approach and the main features of the AccTrace project are represented. The integration layer is responsible for connecting Acero and AccTrace modules to the infrastructure layer. Such infrastructure layer indicates the required servers and systems.

The Acero approach was implemented and the Acero tool was obtained. Next subsection explains details of the approach emphasizing our practical solution.

5.2. Acero Tool. According to the technical report presented by RebelLabs Tools and Technologies Land scape [52], in 2016 Eclipse IDE was used by 41% of Java developers. It was originally designed for Java development; however, it currently supports several other languages such as PHP, C / C ++, and Python. In addition, it has a public license, which allows the developer to create plug-ins to improve their development environment.

Therefore, Acero was proposed as a plug-in for the Eclipse IDE allowing the integration of the AccTrace tool, developed in Java and the Homero Framework, developed in PHP, and other available tools. In addition, the IDE enables communication with web accessibility analysis tools, such as Achecker and AccessMonitor, which are fundamental in the context of the proposal. Finally, it has public license, which allows developers to create new features.

The Acero tool makes it easier to reuse classes from the Homero Framework because it allows the developer to use it at a higher abstraction level. Through the use of a wizard, it is possible to make the semiautomatic filling of the necessary attributes for instantiation of the Homero Framework classes. The Homero module can contribute to productivity and accessibility because the user does not need to search the documentation of the framework to know the methods and their arguments. In addition, the user becomes aware about which fields may affect the accessibility of the content.

By inserting the data in the wizard fields, such as filename, header, coding, and page language, the Acero tool will enable the use of the Homero Framework in the context of the current project, as shown in Figure 8. The result obtained can be observed in Figure 9. On the left side it is possible to observe the file created within the current project and on the right side it is possible to observe the content of the Homero.php file that has the basic template of the Homero Framework.

Using the correct syntax, it is possible to automatically submit a code to the Homero Framework and obtain as output an accessible code (HTML extension). In addition, it is possible to check which errors the user made and how such errors interfere with the accessibility of the content (for example, if the user forgot to fill in some required field).

Figure 10 summarizes the steps considering the execution of a particular source code. When programming and defining the compilation attributes, the developer presses the *Finish* button and it is generated a file in the HTML format that represents the interface. The user will be able to view the interface in the Eclipse IDE's internal browser and obtain additional information in the Acero Output Window, such as errors in the source code.

Another important feature obtained with the integration of the Homero Framework was the possibility of interpreting code written in the PHP language even when the end user does not have a PHP server with the framework installed in their computer (such functionality is useful in limited environments).

The development environment of the Acero tool offers the possibility for the developer to submit the source code to the tools of automatic assessment of accessibility. This module is very important because it allows the user to evaluate in their development environment the compliance of the code with the accessibility guidelines.

As shown in Figure 11, some fields must be completed by developers so that the evaluation can be performed. Basically, the name of the file that will be evaluated, the name of the output file that will contain the results and the accessibility guidelines are required. The guidelines are available according to each evaluation tool selected by the user.

It is possible to assist the developer by alerting them to possible source code errors. For example, if the user inserted an image, the Acero tool can present the main errors that affect the accessibility of the image element so that they can be avoided.

Figure 12 shows an example of using the predictor of Acero over an HTML code. It is possible to notice that the element imagem.png (line 7) exists (indicated by OK in the figure). Differently, for the header element (line 3) there are three errors that can affect the functioning and the accessibility of the final product. Two of the errors, called E_USER_ERROR, prevent the script from functioning (JS and CSS files not found). Otherwise, errors called E_USER_WARNING impair accessibility. In the case of the prediction of the image element (line 7) if the developer does not add auxiliary text to the image, the script will work; however, the image will not be accessible.

The creation of the traceability matrix in PDF format offers the possibility for the developer to quickly identify links between generated artifacts, especially presenting artifacts that can be affected if a change is made. In addition, Acero allows users to check and correct consistency problems in the file that links requirements, UML artifacts, implementation techniques, and source code comments. Therefore, it will be possible to perform reverse tracing, that is, checking the consistency between the .acctrace association file and the

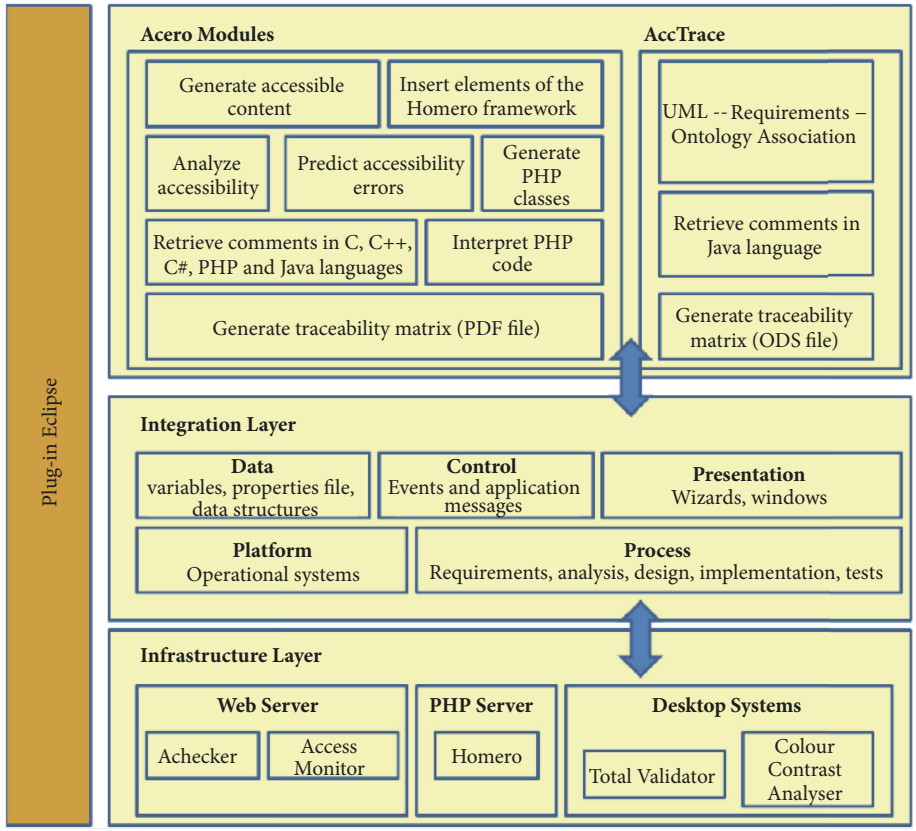

Figure 7: Overview of proposed architecture for the Acero approach.

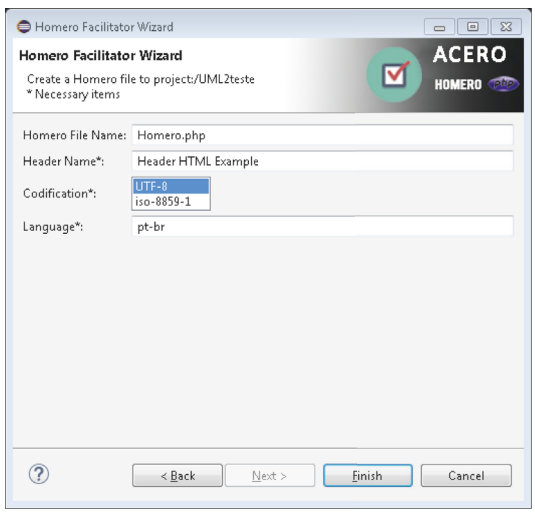

Figure 8: Homero Framework and Acero tool integration.

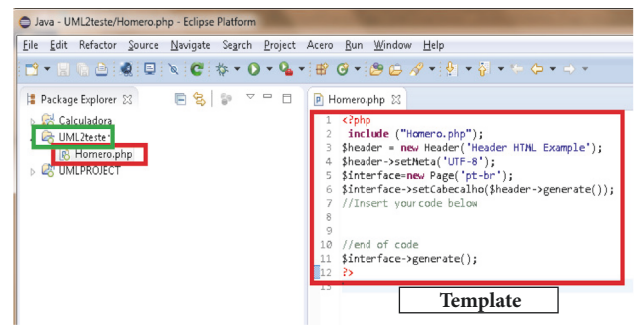

Figure 9: Example of file created in the current project containing the basic template for the use of the Homero Framework.

project files. This functionality is useful when modifications occur in the project, for example, creating and deleting requirements and classes.

The Acero tool allows the automatic creation of PHP classes with comments for implementation of accessibility using the UML artifacts and the traceability matrix. In addition, the association between UML artifacts, requirements, and accessibility ontology may be retrieved. This function is important because the AccTrace tool supports only Java code. With this new feature it is possible to retrieve comments in source code written in PHP, C, C ++, C#, PHP, JavaScript, and Java languages.

It is also important to note that the Acero tool offers the user the possibility of using the Color Contrast Analyzer tool (https://www.paciellogroup.com/resources/contrastanalyser/). It allows the analysis of contrasts of an interface and simulation from the perspective of users with visual impairment, such as cataract and color blindness. The Total Validator Basic tool allows the validation of guidelines

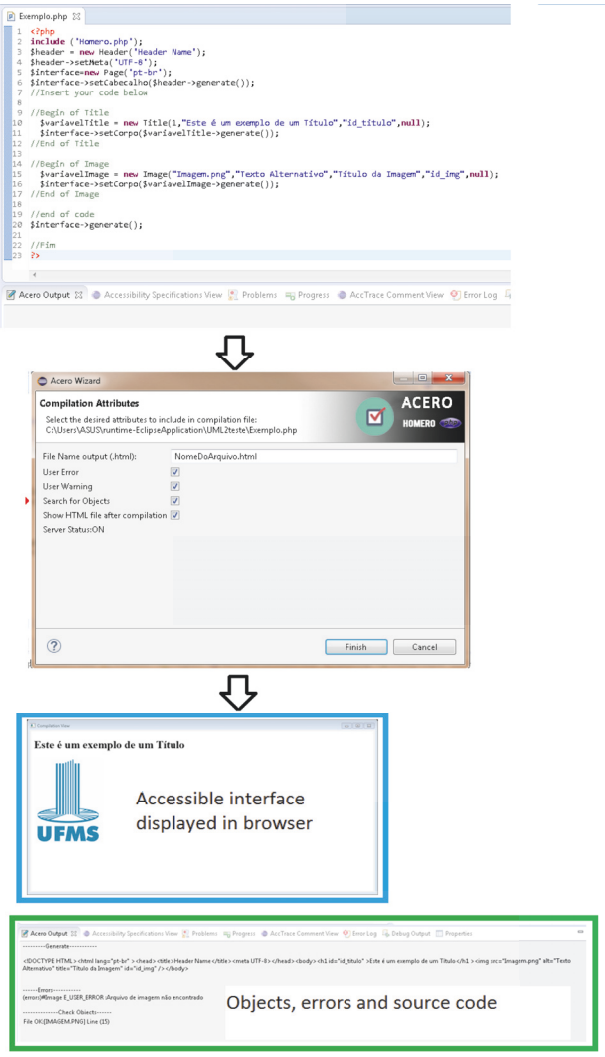

Figure 10: Example of the process to generate an interface accessible from a PHP code and using the Homero Framework.

in the user's source code in offline mode. Both tools are relevant and consolidated in the development of accessible solutions and are indicated by W3C [53]. The integration of these tools in the context of the Acero tool provides the user with a broader development environment as it directly contributes to the automation of accessibility criteria and reduces the possibility of inclusion of errors due to lack of developer knowledge.

6. Evaluation

Two case studies were carried out with the objective of evaluating the effectiveness of the proposed solution, analyzing whether it actually assists in the design and construction of accessible products. Therefore, the evaluation stage took into account Acero approach and tool, including its elements: the MTA process, the AccTrace tool, and the Homero Framework. The main task assigned to the participants was the development of an accessible calculator.

The evaluation of the proposed solution was divided into two stages: initially, as presented in Section 6.1, the MTA process was useful for specifying accessibility elements of the application. Afterwards, the application was developed based on such specification. The second case study, as presented in Section 6.2, used the same specification, however, the focus was to evaluate functionalities of the Acero tool.

6.1. Case Study 1: Focusing on the Acero Approach. The MTA process was used during definition of the main settings for the application, as presented as follows. Accessibility requirements were our main focus.

(1) System requirements elicitation

(a) The accessible calculator will be developed to users with visual impairment. They will use a screen reader as assistive technology. They will use computer to run the application. It is not intended for use on mobile devices

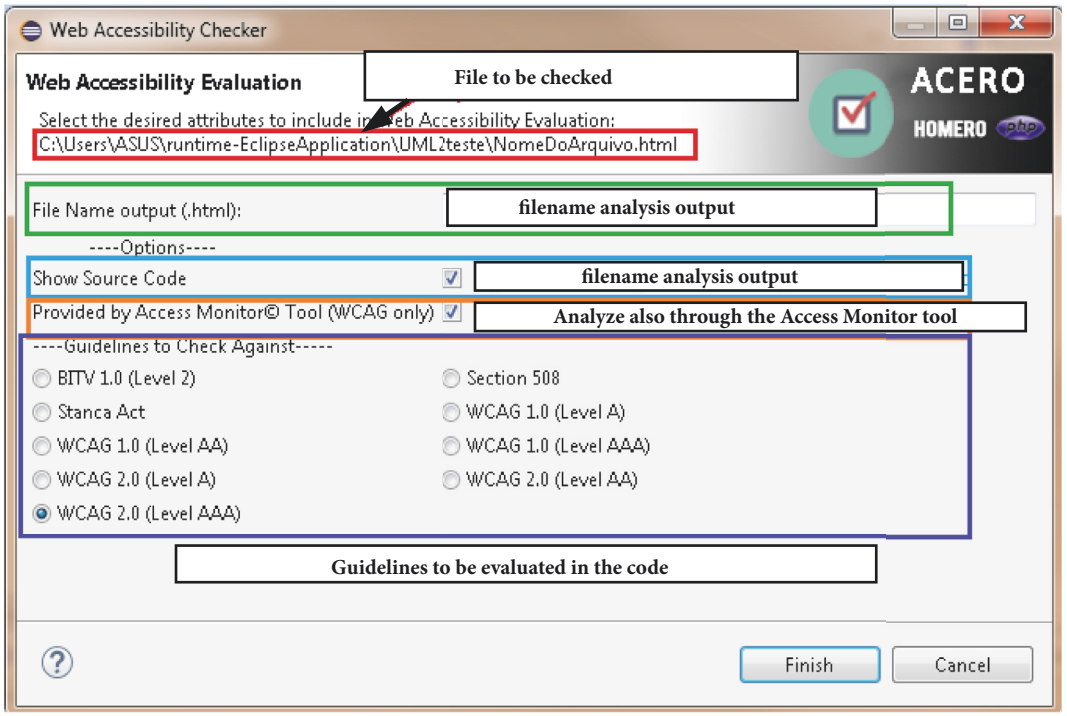

Figure 11: Screen shot of the Acero tool used to evaluate accessibility guidelines in a source code.

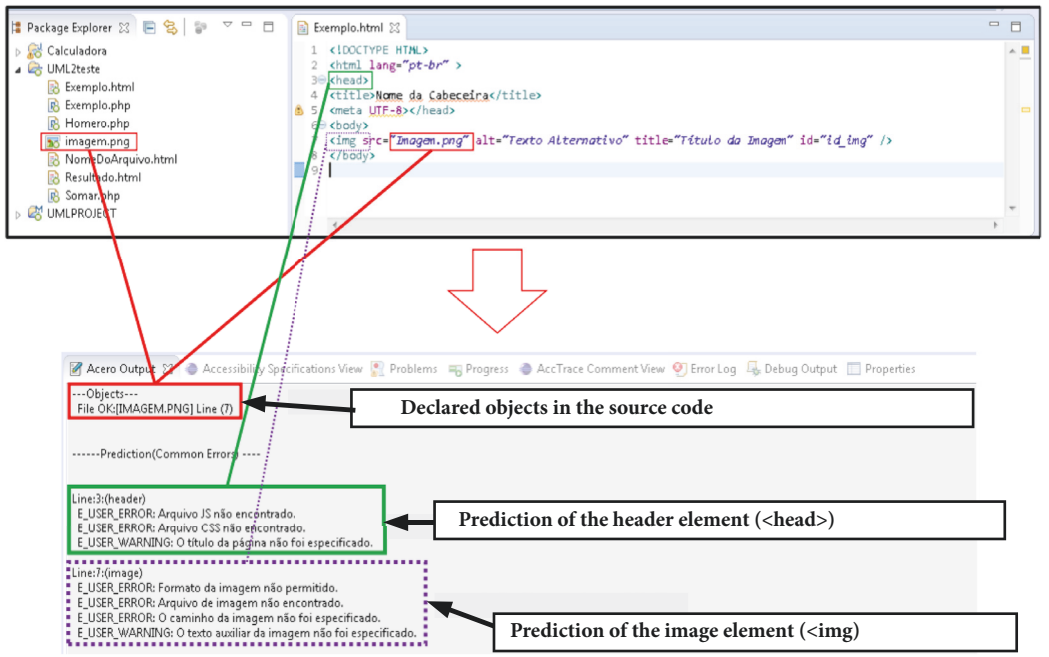

Figure 12: Example prediction of resource use in an HTML page using the Acero tool.

(2) System requirements analysis

(a) The final users use computers frequently and they have completed high school education at least. The application proposal is to allow the execution of the four basic mathematical operations. Final users have previously used similar applications and often use screen readers. Criteria established by ISO/IEC 12207 were considered for accessibility evaluation of system requirements analysis

(3) System architectural design

(a) The application is very simple and this activity was not relevant

(4) Software requirements analysis

(a) The end user is a 64 years old man. His deficiency is low vision. He was interviewed and the following accessibility requirements were identified: (a) provide text alternatives for any nontext content; (b) make all functionality available from a keyboard; (c) do not design content in a way that is known to cause seizures; (d) provide ways to help users navigate, find content, and determine where they are; (e) make text content readable and understandable; and (f) help users avoid and correct mistakes. Criteria established by ISO/IEC 12207 were considered for accessibility evaluation of software requirements analysis

(5) Software design

(a) A prototype of the accessible application was developed prioritizing the use of labels, colors, images, text editing fields and navigational design. Criteria established by ISO/IEC 12207 were considered for accessibility evaluation of the software design

(6) Software construction

(a) The accessible application was developed using the Acero tool. The HTML language was used to implement accessibility requirements. Accessibility test of each unit was carried out using the Total Validator Basic tool (https://www.totalvalidator.com/tools/). The results allowed to evaluate if the established requirements were implemented

(7) Software integration testing

(a) The application is very simple and this activity was not relevant

(8) Acceptance testing

(a) The same user who provided the requirements participated in the acceptance test. In general, the application developed by the students was satisfactory. He suggested changes to the use of labels containing auxiliary texts and navigation design

(9) System testing

(a) The final user used the NVDA screen reader (https://www.nvaccess.org/) to run the application and he considered that the elements of the system worked correctly when together

The case study was carried out with the participation of eight students enrolled in the Faculty of Computer Science of the Federal University of Mato Grosso do Sul (Facom / UFMS), in modalities of graduation, master's degree, and doctorate. All participants contributed in the context of the MTA process, doing interviews with the end user, specifying, and making design decisions. Additionally, they used the Acero tool to the software codification.

Students received training focusing on a general approach to accessibility (approximately 30 minutes) and information about participant's theoretical knowledge was collected. Seven participants did not have knowledge about accessibility guidelines and about WCAG 2.0.

Students defined the functional and nonfunctional accessibility requirements (interviewing the end user) and designed use case diagrams, class diagrams, and a graphical interface prototype (it was presented to the final user). Finally, the student spent an hour and forty minutes for the application codification and tests.

As a result, participants indicated that they were able to easily understand and perform the proposed activities. The received suggestions were incorporated into the planning of the case study 2 and, basically, aimed to increase training time on accessibility and to include new training on the Eclipse IDE. The final application obtained is presented in the Figure 13.

6.2. Case Study 2: Focusing on the Acero Tool. The second case study was conducted with 14 undergraduate students from the last semester of the Computer Engineering course of the Facom / UFMS. The students developed the application individually: 7 students used the Acero approach and tool and 7 students did not use it. In relation to the profile of the participants, 21% indicated they had excellent knowledge about object-oriented programming and 79% indicated that they had good knowledge; 26% indicated that they had excellent knowledge in PHP, Java, and HTML programming languages and 74% indicated that they had the necessary knowledge to develop applications of medium complexity. In addition, only two students indicated that they had minimal theoretical and practical knowledge about web accessibility. Other participants did not have knowledge about this subject. The following hypotheses were considered:

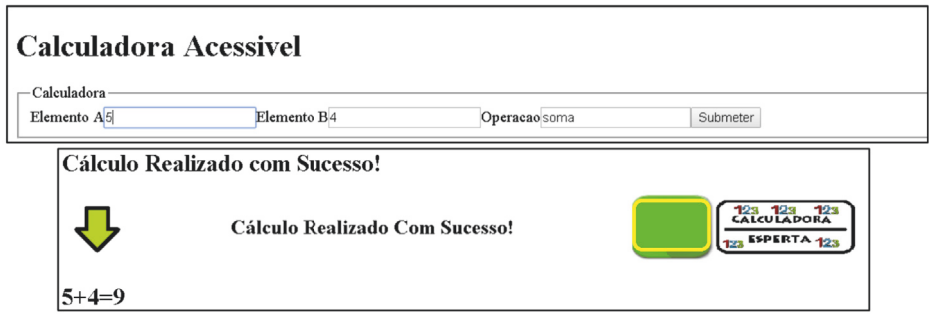

Figure 13: Final application for the case study 1 (in Portuguese).

(i) Time:

(1) H0: the use of the Acero approach and tool does not reduce the time of development of accessible application.

(2) Ha0: the use of the Acero approach and tool reduces the time of development of accessible application.

(ii) Accessibility of the developed application:

(1) H1: the use of the Acero approach and tool does not help in the design of accessible application.

(2) Ha1: the use of the Acero approach and tool assists in the design of accessible application.

In order to mitigate validity threats, training on accessibility (90 minutes) and use of the Eclipse IDE were offered to the participants before the execution of the case study. The same problem of first case study was considered, i.e., development of an accessible calculator. Supporting files and diagrams were provided. The study was divided into five stages:

(1) Step 1: import the Acero tool files into the Eclipse IDE.

(2) Step 2: define the application programming logic.

(3) Step 3: design and implementation of the accessible interface of the application.

(4) Step 4: interface accessibility analysis.

(5) Step 5: host the web Application.

Results of the case study indicated that the two groups have relative equivalence in the elapsed mean time for performing steps (1) and (2) (Figure 14). In these steps no group used the Acero approach and tool because it was external adjustments such as file import, definition programming logic and web code hosting. However, in stages 3, 4, and 5 there was a difference in the meantime of execution of the groups, and it can be inferred that in step (3) the Acero approach and tool contributed to the application development, with the mean time less than the group that did not use the tool. In addition, in step (4), participants who used the Acero tool took less time to complete the step. This difference may have occurred due to the inclusion

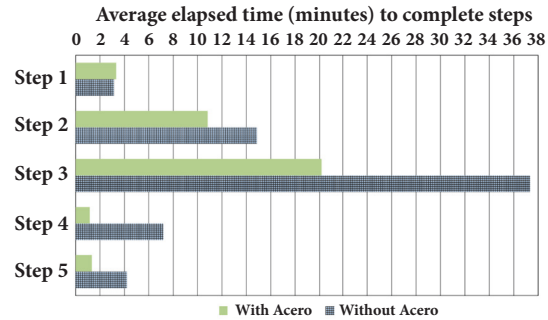

Figure 14: Average time for groups to complete the steps proposed in the study.

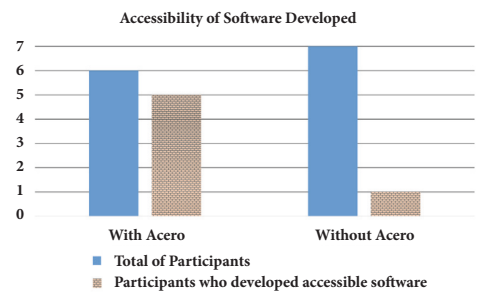

Figure 15: Accessibility of the content.

of the evaluation accessibility module in Acero. Therefore, considering the results obtained, it is possible to reject the null hypothesis H0.

Regarding the second hypothesis, as observed in the Figure 15, it is noticed that only one of the students who used the Acero approach and tool did not develop an AAA level application of WCAG 2.0. Without the use of the Acero tool, only one student developed a level AAA application of WCAG (considering the WCAG 2.0 guidelines that are automatically evaluated). Therefore, we can also reject the null hypothesis H1.

It was observed that the use of Acero reached the proposed objectives, supporting different stages of the development process. In addition, it proved to be compatible and promising in supporting integration with other tools, since all

participants were able to finalize the proposed script for the study.

7. Conclusions and Contributions

This research was started with the objective of designing a methodology to integrate several tools to provide support to the software engineer and the programmer in relation to the development of accessible web applications. Through the case study, it was verified that this objective was reached because the participants were able to go through the entire development process and generate accessible applications.

As the main results achieved, it is noted that the Acero approach and tool contribute to the following:

(i) Ensuring that accessibility is a constant concern throughout the development phase, supporting each stage of the software process directly in the development environment.

(ii) Promoting the familiarization of software engineers and developers with the accessibility and international guidelines proposed to achieve them.

(iii) Providing mechanisms, such as wizards, that facilitate the use of the tool and the implementation of accessibility guidelines during software design and development.

(iv) Automating processes that were once manual, for example, the traceability matrix generation that maintains the relationships including requirements, UML artifacts, and class accessibility implementation techniques.

(v) Integrating and increasing functionalities in the AccTrace and Homero Framework tools. For example, the traceability matrix (previously generated only in the ODS extension) can be generated in the Acero tool in PDF format, a more common format. In addition, the Homero Framework was extended so that it could be used directly within the Eclipse IDE, without the need for the user to be dependent on their documentation or to have knowledge about object-oriented programming.

The choice of the technological tools that could be integrated in order to construct the Acero tool was an important stage of this work. Therefore, a major challenge of this research was to find out which tools were appropriate, the form that each one operated within the application development process, and how to integrate them in a common development environment. The main limitations with respect to the Acero tool are that it supports only Java and PHP programming languages and only considers the WCAG 2.0 guidelines.

The research area that involves the integration between accessibility, development process, and integration of support tools is relatively recent and needs to be improved. Therefore, considering the results obtained in this work, several research proposals can be explored. The following are suggestions for future work:

(1) Include other accessibility standards: designers and developers can choose which standard they would like to consider.

(2) Consider alternatives to the evaluation of guidelines that can not be automated.

(3) Carry out a detailed study of universal design principles and identify possible extension points in the Acero tool that can implement them.

(4) Carry out new case studies considering accessibility specialists and end users.

(5) Provide designers and developers with proposals for architectural models considering different domains (for example, e-gov, e-commerce, etc.). Such architectural models should include accessibility standards and allow the generation of applications in an agile way.

(6) Identify and adopt mechanisms that promote the transfer of knowledge and technology generated for the industry.

In general, it should be noted that this work mainly contributed to the development of an innovative software solution whose focus was to help developers in the design of accessible solutions. The case studies allowed to identify positive points and constraints of the Acero approach and tool and indicated the feasibility of the proposal as an alternative for the development of accessible web applications.

Data Availability

Documentation and source code of the Acero tool are available at https://bitbucket.org/wesley_tessaro/preditor.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors thank CAPES and FUNDECT (T.O. 102/2016) for financial support.

References

[1] International Telecommunication Union, ICT Facts and Figures: The World in 2015, Retrieved July, 2018 from: https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf.

[2] L. E. Janlert and E. Stolterman, "The meaning of interactivity—some proposals for definitions and measures," *Human-Computer Interaction*, vol. 32, no. 3, pp. 103–138, 2017.

[3] M. C. Rodriguez-Sanchez and J. Martinez-Romo, "GAWA – Manager for accessibility Wayfinding apps," *International Journal of Information Management*, vol. 37, no. 6, pp. 505–519, 2017.

[4] A. Iqbal, F. Ullah, H. Anwar et al., "Interoperable Internet-of-Things platform for smart home system using Web-of-Objects

and cloud," *Sustainable Cities and Society*, vol. 38, pp. 636–646, 2018.

[5] World Health Organization, Disability and health, Retrieved January, 2018 from: http://www.who.int/mediacentre/factsheets/fs352/en/.

[6] W3C, Making the Web Accessible, Retrieved December, 2017 from: http://www.w3.org/WAI/.

[7] M. F. Cabrera-Umpiérrez, "3rd Generation accessibility: information and communication technologies towards universal access," *Universal Access in the Information Society*, vol. 15, no. 1, pp. 1–3, 2016.

[8] R. Babu, R. Singh, and J. Ganesh, "AIS transactions on human-computer interaction," *Association for Information Systems*, vol. 2, no. 4, pp. 73–94, 2010.

[9] P. P. Rau, L. Zhou, N. Sun, and R. Zhong, "Evaluation of web accessibility in China: changes from 2009 to 2013," *Universal Access in the Information Society*, vol. 15, no. 2, pp. 297–303, 2016.

[10] H. B. Paulsen, *Finding an Optimal Method for Conducting Accessibility Evaluations of the Norwegian Tax Administration Website, [Master thesis]*, Oslo and Akershus University College of Applied Sciences, 2016, http://hdl.handle.net/10642/3345.

[11] E. Loiacono and S. Djamasbi, "Corporate website accessibility: does legislation matter?" *Universal Access in the Information Society*, vol. 12, no. 1, pp. 115–124, 2013.

[12] S. Giraud, P. Thérouanne, and D. D. Steiner, "Web accessibility: Filtering redundant and irrelevant information improves website usability for blind users," *International Journal of Human-Computer Studies*, vol. 111, pp. 23–35, 2018.

[13] P. Sherman, Cost-Justifying Accessibility, Retrieved October, 2017 from: https://www.ischool.utexas.edu/l385t21/AU_WP_Cost_Justifying_Accessibility.pdf.

[14] K. Groves, "How Expensive is Web Accessibility?" Retrieved September, 2013 from: http://www.karlgroves.com/2011/11/30/how-expensive-is-accessibility/.

[15] I. Management Association, *Assistive Technologies: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*, Information Science Reference, 2014, https://books.google.com.br/books?id=LN6WBQAAQBAJ.

[16] M. Ganslandt and M. Sutinen, "Web Standardization," Center for European Law and Economics, 2009, http://www.talkstandards.com/wp-content/uploads/2009/08/web-standards-090828-final.pdf.

[17] Web Content Accessibility Guidelines (WCAG) 2.0, Retrieved October, 2017 from http://www.w3.org/TR/WCAG20/.

[18] How WCAG 2.0 Differs from WCAG 1.0, Retrieved August, 2017 from: http://www.w3.org/WAI/WCAG20/from10/diff.php.

[19] J. Lazar, D. Goldstein, and A. Taylor, *Ensuring Digital Accessibility through Process and Policy*, Elsevier Science, 2015, https://books.google.com.br/books?id=YepDBAAAQBAJ.

[20] L. Moreno, P. Martínez, J. Muguerza, and J. Abascal, "Support resource based on standards for accessible e-government transactional services," *Computer Standards & Interfaces*, vol. 58, pp. 146–157, 2018.

[21] Web Standards for the Government of Canada, Retrieved August, 2015 from: http://www.tbs-sct.gc.ca/ws-nw/index-eng.asp.

[22] Japanese Industrial Standards and Web Accessibility Infrastructure Commission, Retrieved July, 2018 from: http://waic.jp/docs/jis2010/understanding.html.

[23] Disability Act, Retrieved July, 2018 from http://www.oireachtas.ie/documents/bills28/acts/2005/a1405.pdf.

[24] Legge Stanca, Retrieved July, 2018 from: http://www.webaccessibile.org/.

[25] eMAG - Modelo de Acessibilidade em Governo Eletrónico, Retrieved June, 2018 from: http://emag.governoeletronico.gov.br/.

[26] Web Content Accessibility Guidelines (WCAG) 2.1, Retrieved July, 2018 from: https://www.w3.org/TR/WCAG21/.

[27] G. Brajnik, Y. Yesilada, and S. Harper, "The expertise effect on web accessibility evaluation methods," *Human–Computer Interaction*, vol. 26, no. 3, pp. 246–283, 2011.

[28] A. P. Freire, *Disabled People and the Web: User-based Measurement of Accessibility [Ph.D. thesis]*, University of York, 2012.

[29] A. Aizpurua, S. Harper, and M. Vigo, "Exploring the relationship between web accessibility and user experience," *International Journal of Human-Computer Studies*, vol. 91, pp. 13–23, 2016, http://dx.doi.org/10.1016/j.ijhcs.2016.03.008.

[30] WCAG-EM Overview: Website Accessibility Conformance Evaluation Methodology, Retrieved October, 2017 from: https://www.w3.org/WAI/eval/conformance.html.

[31] M. Vigo, J. Brown, and V. Conway, "Benchmarking web accessibility evaluation tools: measuring the harm of sole reliance on automated tests," in *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, Rio de Janeiro, Brazil, May 2013, http://doi.acm.org/10.1145/2461121.2461124.

[32] E. Krainz, J. Feiner, and M. Fruhmann, *Human-Centered and Error-Resilient Systems Development*, C. Bogdan, J. Gulliksen, S. Sauer et al., Eds., Springer International Publishing, Cham, Switzerland, 2016.

[33] E. Krainz and K. Miesenberger, "Accapto, a generic design and development toolkit for accessible mobile apps," *Studies in Health Technology and Informatics*, vol. 242, pp. 660–664, 2017.

[34] E. Krainz, K. Miesenberger, and J. Feiner, "Can We Improve App Accessibility with Advanced Development Methods?" in *Computers Helping People with Special Needs*, K. Miesenberger and G. Kouroupetroglou, Eds., pp. 64–70, Springer International Publishing, Cham, Switzerland, 2018.

[35] H. Vieritz, F. Yazdi, D. Schilberg, P. Göhner, and S. Jeschke, "User-centered design of accessible web and automation systems," in *Information Quality in e-Health*, A. Holzinger and K. M. Simonic, Eds., pp. 367–378, Springer, Heidelberg, Berlin, Germany, 2011.

[36] M. González-García, L. Moreno, and P. Martínez, "A Model-Based Tool to Develop an Accessible Media Player," in *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, pp. 415-416, New York, NY, USA, 2015.

[37] L. Zouhaier, Y. Hlaoui Bendaly, and L. Jemni Ben, "A MDA-based Approach for Enabling Accessibility Adaptation of User Interface for Disabled People," in *Proceedings of the 16th International Conference on Enterprise Information Systems*, pp. 120–127, SCITEPRESS - Science and Technology Publications, Lda, Portugal, 2014.

[38] R. Miñón, J. Abascal, A. Aizpurua, I. Cearreta, B. Gamecho, and N. Garay, "Model-Based Accessible User Interface Generation in Ubiquitous Environments," in *Proceedings of the Human-Computer Interaction – INTERACT 2011*, P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, and M. Winckler, Eds., pp. 572–575, Springer, Heidelberg, Berlin, Germany, 2011.

[39] R. Bonacin, M. C. C. Baranauskas, and M. A. Rodrigues, *Information Systems*, J. Filipe and J. Cordeiro, Eds., Springer Berlin Heidelberg, Berlin, Germany, 2009.

[40] T. H. Røssvoll and K. S. Fuglerud, *Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion*, C. Stephanidis and M. Antona, Eds., Springer Berlin Heidelberg, Berlin, Germany, 2013.

[41] A. L. Dias, R. P. M. Fortes, and P. C. Masiero, "Increasing the Quality of Web Systems: By Inserting Requirements of Accessibility and Usability," in *Eighth International Conference on the Quality of Information and Communications Technology*, pp. 224–229, IEEE, Lisboa, Portugal, 2012.

[42] S. Sanchez-Gordon, M.-L. Sánchez-Gordón, and S. Luján-Mora, "Towards an engineering process for developing accessible software in small software enterprises," in *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering, ENASE 2016*, pp. 241–246, Rome, Italy, April 2016.

[43] L. S. Maia, *Um Processo para o Desenvolvimento de Aplicações Web AcessÍveis, [Master thesis]*, Universidade Federal de Mato Grosso do Sul, Brazil, 2010.

[44] ISO/IEC, ISO/IEC 12207 - Standard for Informational Technology - Software Lifecycle Processes (ISO/IEC, 1, ch. de la Voie-Creuse - CP 56 - CH-1211 Geneva 20 - Switzerland), 1998.

[45] D. Sloan, A. Heath, F. Hamilton, B. Kelly, H. Petrie, and L. Phipps, "Contextual web accessibility - maximizing the benefit of accessibility guidelines," in *Proceedings of the 2006 International Cross-Disciplinary Workshop on Web Accessibility (W4A): Building the Mobile Web: Rediscovering Accessibility?* pp. 121–131, ACM, New York, NY, USA, May 2006.

[46] D. Hoffman, E. Grivel, and L. Battle, "Designing software architectures to facilitate accessible Web applications," *IBM Systems Journal*, vol. 44, no. 3, pp. 467–483, 2005.

[47] R. G. de Branco, *Acessibilidade nas Fases de Engenharia de Requisitos, Projeto e Codificação de Software: Uma Ferramenta de Apoio [Master thesis]*, Universidade Federal de Mato Grosso do Sul, Brazil, 2013.

[48] R. G. de Branco, M. I. Cagnin, and D. M. B. Paiva, "AccTrace: considerando acessibilidade no processo de desenvolvimento de software," in *Proceedings of the XXI Sessão de Ferramentas do Congresso Brasileiro de Software: Teoria e Prática*, pp. 117–124, 2014.

[49] AEGIS Ontology, Retrieved July, 2013 from: http://www.aegis-project.eu/index.phpoption=com_content&view=article&id=107&Itemid=65.

[50] R. C. Oliveira, *Homero: Um Framework de Apoio ao Desenvolvimento de Interfaces de Aplicações Web Acessíveis, [Master thesis]*, Universidade Federal de Mato Grosso do Sul, Brazil, 2013.

[51] R. C. de Oliveira, A. P. Freire, D. M. Paiva, M. I. Cagnin, and H. Rubinsztejn, "A Framework to Facilitate the Implementation of Technical Aspects of Web Accessibility," in *Proceedings of the 16th International Conference on Human-Computer Interaction*, pp. 3–13, 2014.

[52] S. Maple, "Java tools and technologies landscape," RebelLabs, 2016, https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/.

[53] W3C - Web Accessibility Initiative – Web Accessibility Evaluation Tools List, Retrieved May, 2016 from: https://www.w3.org/WAI/ER/tools/.