1. INTRODUCTION

1.1 Context and Motivation

Since 1999, when the W3C[1]-WAI[2] introduced the "Web Content Accessibility Guidelines 1.0" (WCAG 1.0) [45] as a set of guiding principles, the fact that Accessibility is a main topic in Web design upon which the success of a Web application depends, has become a landmark statement. However, developing accessible Web applications is usually hard for several reasons.

Firstly, there is a significant knowledge gap between developers and Accessibility specialists. Most developers do not have the necessary skills or training in designing and coding for Accessibility, and most Accessibility specialists have, in turn, limited developing practice [22]. Thus, although there are many available tools and published sources of information on Web Application Accessibility, existing Web Accessibility guidelines and principles (and therefore, experts on these guidelines) do not address additional design issues that may typically arise when developing complex Web applications. To make matters worse, there is little evidence of design approaches dealing with Accessibility from the beginning of the design process. In most cases, Accessibility is regarded as a programming issue or even dealt with when the Web application is already fully developed and, consequently, the process of making this application accessible involves significant redesign and recoding, which might be out of the scope of the project and/or hardly affordable [22]. As we will show next, the main problem with Accessibility is that it is a non-functional software concern, which affects (crosscuts) other application concerns. Generally speaking, a non-functional requirement is a software requirement which does not describe *what* the system will do (functional requirement), but *how* the system will do it; for example, performance requirements, modularity requirements, or quality attributes, which represent constraints on the services or functions offered by a system [39].

[1] The World Wide Web Comsortium at http://www.w3.org/

[2] The Web Accessibility Initiative at http://www.w3.org/WAI/

Although Accessibility has not yet gained much recognition as a crucial non-functional requirement like security, performance, accuracy and usability; it is a vital attribute for people with disabilities. Moreover, Accessibility is a generic concern that may comprise dozens of specialized concerns and, therefore, many requirements associated with these.

For example, at the application-level, Accessibility can be specialized according to the kind of Accessibility support given to the user, where specific requirements related to the user's layout and the user's technology supports are considered. While the former ensures an accessible user's interaction, the user's technology support guarantees browsing regardless of the user's assistive device and further, new requirements related to current and earlier assistive devices characteristics are associated separately --i.e. "user agents" and "until user agents" respectively as the distinction made by the W3C's UAAG 1.0 [48]. The term "user agent" is used by the W3C as a generic description for any software that retrieves and renders Web content for users, such as browsers, mobile phones, screen readers, etc. On the other hand, the term "until user agent" is used by the W3C referring to "user agents" that require developers to provide additional support for Accessibility.

As another example, at the meta-level, Accessibility can be specialized according to meta-features like compliance design and content order concerns. The first one means conformance to some Web Accessibility design principles that are articulated by guidelines, regulations, standards or laws, while the second one refers to how to organize the Web pages content based on research reports and studies like quality in use surveys, conducted experiences, patterns catalogues, etc. In both cases, these specialized concerns have their associated requirements.

Finally, and as an example of the model-level, Accessibility can also comprise different concerns according to the methodological phase for the development of the Web application. Normally, these efforts are focalized on the interface model by applying some conformance assessment criteria, which establish associated requirements for abstract and concrete interface widgets.

In this work we introduce our design approach, which proposes to include Accessibility concerns systematically within a methodology for Web application development. Firstly, to find out how Accessibility concerns should be introduced in the development

life cycle, we analyzed how mature Model-Driven[3] Web Engineering (WE[4]) methods[5], such as UWE [24], OOHDM [36], OOWS [18] or WSDM [13], face this cycle. We realized that all of them comprise several activities to focus on some specific design concerns; however, since OOHDM fulfill many of our expectations, we decided to join our modeling approach to this particular WE method. As an example of the rational of choosing OOHDM as our host WE approach, we have to mention the different views provided by OOHDM at the user interface (UI) model. This fine-grained treatment allows us to move from abstract interface elements, which are those from the widget ontology [36], to concrete interface elements --e.g. HTML elements, and link both levels of abstraction from a UI design perspective [27] to WCAG checkpoints. Secondly, since designing accessible Web applications involves the analysis of different interests, we proposed to use Aspect-Oriented Software Development (AOSD[6]) design principles to support the construction of accessible user interfaces. The fact that we choose aspect orientation to develop our proposal ensures handling naturally the non-functional, generic and "crosscutting"[7] characteristics of the Accessibility concern.

[3] Model-Driven Software Development (MDSD) is a software engineering methodology that focuses on creating and exploiting domain models –i.e. abstract representations of the knowledge and activities that govern a particular application domain, rather than on the computing (or algorithmic) concepts.

[4] Web Engineering (WE) is a specific domain in which MDSD can be successfully applied to implement systems that exploit the Web paradigm. WE is the application of systematic and quantifiable approaches, such as concepts, methods, techniques, tools, to cost-effective requirements analysis, design, implementation, testing, operation, and maintenance of high-quality Web applications.

[5] These development proposals are also known as Model-Driven Web Development (MDWD) approaches because they are concerned to provide methodologies and tools for the design and development of most kinds of Web applications.

[6] Aspect-Oriented Software Development (AOSD) focuses on the identification, specification and representation of "crosscutting" concerns and their modularization into separate functional units as well as their automated composition into a working system.

[7] "Croscutting" is a term used for certain type of functionality whose behavior causes code spreading and intermixing through layer and tiers of an application which is affected in a loss of modularity in their classes. Quality requirements (such as Accessibility), exception handling, validation and login managements are all examples of this common functionality that is usually described as "crosscutting concerns" and should be centralized in one location in the code where possible.

As a motivating example and to introduce properly the ideas behind our modeling approach, let us suppose a typical login Web page whose purpose is aiming a student's identification at his/her university system, such as the SIU Guarani student registration system that is used by a number of Argentine universities[8]. Figure 1.1 shows the page for the student's login that provides a user interface composed of HyperText Markup Language (HTML) elements, such as labels and text fields. To help to an accessible interaction experience these HTML elements must fulfill some Accessibility requirements, which crosscut the same software artifact (the Web page for student's login). For example, and as we will see in detail later, at the presentation level an HTML label element is a basic layout Accessibility requirement for many other HTML elements.

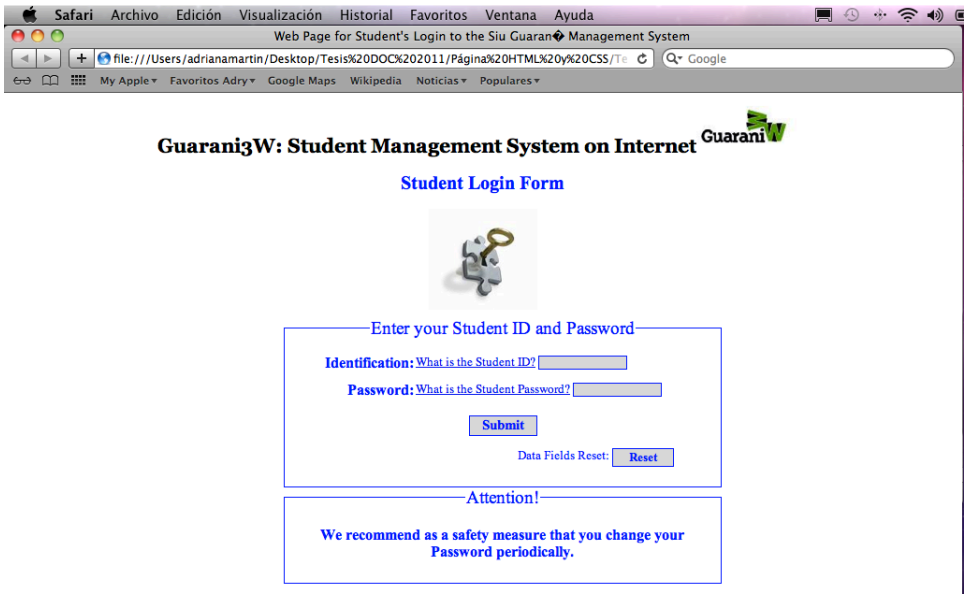

Figure 1.1: A Student's Login Web page example

Since a Web page for student's login requires at least two text field elements (for student's ID and password respectively), the presence of their respective label elements must be tested. So, to propitiate an accessible interaction experience on behalf of the student, this layout requirement must crosscut the same software artifact (the Web page) more than once, accordingly to the number of text field elements included in the presentation. Additionally, it is highly important to consider the positioning of the label

[8] For example the SIU Guarani registration system, as used by the National University of Córdoba at http://www.psi.unc.edu.ar/sistemas/sistemas-de-informacion-academica/siu-guarani

element with respect to a text field element; this technological requirement for "until user agents" [48] --i.e. earlier "user agents", also crosscuts the Web page. Clearly this kind of behavior perfectly fits the "scattering" and "tangling" problems[9], which motivate the main AOSD principles. Since these two Accessibility requirements (presence and positioning of the label elements), are "scattered" in the Web page with a pair of label-text field HTML elements, the Web page is "tangled" with these Accessibility requirements. It seems natural therefore to address Accessibility using the Aspect-Oriented Software Development (AOSD) approach and, it is not just a coincidence that during this work we refer to Accessibility as a "concern". Besides the fact that Accessibility has become a basic quality attribute to any Web application and to improve the evolution of the Web in general, the term "concern" from the AOSD perspective describes accurately the Accessibility features related to its nature. By using the AOSD paradigm we can avoid typical problems of "crosscutting" concerns, such as those shown in the previous Web page example. Our proposal applies these concepts by treating Accessibility as a first-class concern in the context of the OOHDM [36] WE approach. Specifically, we propose the early capture of specific Accessibility concerns, which involve user interactions and activities with the application's interface by introducing some additional extensions to the User Interaction Diagram (UID) [44] technique. As we see in Section 5.3, we also propose a supporting tool to assist our approach.

Thus, looking for a comprehensive response to the problem of developing accessible user interfaces (UI) for Web applications since the early stages of design, we propose the following objectives.

1.2 Objectives

The main objective of this work is ***to define a WE approach (process and techniques) to conceive, design and develop accessible Web applications using Aspect-Oriented***

[9] "Scattering" and "Tangling" symptoms are typical cases of "crosscutting concerns" and they often go together, even though they are different concepts. A concern is "scattered" over a class if it is spread out rather than localized while a concern is "tangled" when there is code pertaining to the two concerns intermixed in the same class (usually in a same method).

concepts, which enable to address Accessibility early from requirements and through design to implementation.

As secondary goals we state:

1. Studying the state-of-art of Accessibility proposals in general, and in particular, focalizing on those proposals for designing Web applications with the Accessibility concern in mind.

2. Studying deeply and applying some relevant related work, selected as a result of the previous goal, to a proposed case study.

3. Defining a process for designing Web applications with Accessibility and providing specific techniques that take advantages of Aspect-Oriented concepts to address Accessibility properly and from early stages of design.

4. Applying our proposal to a case study.

5. Proposing a supporting tool to help developers in applying our proposal.

6. Comparing and discussing the main characteristics of our proposal and the relevant related work selected as a result of previous goals.

1.3 Research Context

This thesis has been ***developed and partially supported by the following research projects:***

- ***UNComa project 04E/072.*** Title: *Identificación, Evaluación y Uso de Composiciones Software*. Period: 2008-2011. Director: Dr. Alejandra Cechich.

- ***UNPA-UACO project 21/B107.*** Title: *Mejora de Proceso de Selección de Componentes para Sistemas de Información Geográficos*. Period: 2010-2011. Director: Dr. Alejandra Cechich.

- ***UNLP project PICT-PAE 2187.*** Title: *Desarrollo de Familias de Aplicaciones Web*

y Context Aware. Period: 2009-2011. Director: Dr. Gustavo Rossi.

- ***UNComa project 04/E059.*** Title: *Mejora del Proceso de Desarrollo de Software Basado en Componentes*. Period: 2005-2007. Director: Dr. Alejandra Cechich.

1.4 Structure

The structure of this thesis is organized as follow:

- In Chapter 2, ***Accessibility within WE approaches***, we firstly introduce Web Accessibility, mainly focusing on those features that are relevant for our work. Then, we concentrate on introducing properly some selected related work and applying them to a proposed case study.

- In Chapter 3, ***Background of our Proposal***, we introduce four key topics that we will use throughout the rest of the work, since they are the conceptual basis of our proposal.

- In Chapter 4, ***An Approach for Engineering Accessible Web Applications***, we first provide a general overview of the model we envisage to deal with Accessibility concerns within a Web engineering approach. Then, we conduct a detailed description of the proposed process and techniques for implementing our proposal step-by-step.

- In Chapter 5, ***Applying our Proposal***, we carry out clearly the implementation of our approach following the step-by-step process as we described in Chapter 4. To do so, we propose a complete case study composed of 3 (three) level-deep navigation and 2 (two) optional help anchors. We also introduce a supporting tool that we specially develop to help developers on the design process when applying our proposal.

- In Chapter 6, ***Comparing our Proposal***, we first introduce an evaluation framework that we develop to provide proper comparison criteria for the approaches. Then, we carry out the comparison and develop a discussion about the main characteristics of the related work and our proposal.

- In Chapter 7, ***Conclusions and Future Work***, we conclude summarizing issues from the designer perspective and as a result of our experience gathered at early stages of the Web development process. Then, we state some open questions that lead to future research.