Accessibility at Early Stages: Insights from the Designer Perspective

Adriana Martín[1,2] Alejandra Cechich[1]

[1]GIISCo, Dpto. de Ciencias de la Computación, Universidad Nacional del Comahue, Neuquén, Argentina
Tel: (+54) 299 4490 312

[2]Unidad Académica Caleta Olivia, Universidad Nacional de la Patagonia Austral, Caleta Olivia, Santa Cruz, Argentina

{adrianaelba.martin};{acechich}@gmail.com

Gustavo Rossi
LIFIA Research Group, Facultad de Informática, Universidad Nacional de La Plata and Conicet, Calle 115 e/ 49 y 50, (1900) La Plata, Argentina
Tel: (+54) 0221 4228 252
gustavo@sol.info.unlp.edu.ar

ABSTRACT

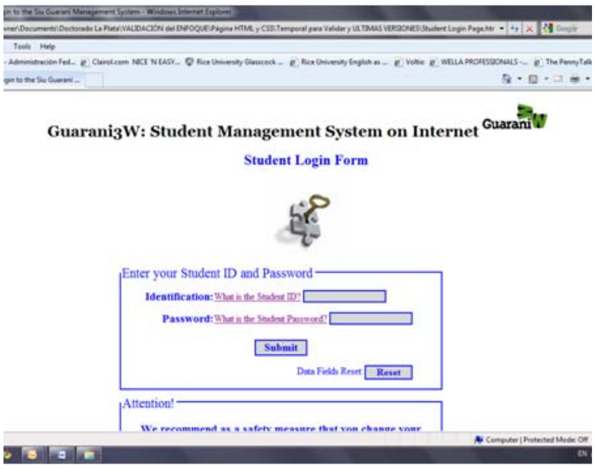
Usually, a huge number of tools and proposals help developers assess Accessibility of Web applications; however, looking from the designer perspective, there is no such a similar situation. It seems that creating accessible Web sites is more expensive and complicated than creating Web sites and then assessing/modifying them. Although this feeling may be largely true, the benefits of modeling Accessibility at early design stages outweigh the needs of a developer to implement that Accessibility. A designer can learn the basics of Web Accessibility and then he/she should be able to incorporate this knowledge into his/her software architecture. The point is to have an idea of how to do so from the beginning. In this paper, we briefly introduce our proposal to model Web Accessibility by moving from abstract to concrete architectural views using aspect-orientation. Our approach takes advantages of modeling Accessibility as an aspect-oriented concern, which is independently treated but related to architectural pieces. We illustrate the approach with a case study and elaborate some insights from the designer perspective.

Keywords

User Interface Models; Web Engineering; Aspect-Oriented design; Accessibility early design.

1. INTRODUCTION

The Word Wide Web Consortium (W3C) is one of the main referents of Web Accessibility and has worked for more than ten years in the development of a standard called Web Content Accessibility Guidelines (WCAG) [27], which is considered a benchmark for most of the laws on Information Technology and Communication worldwide. Based on these recommendations, a number of tools and approaches have emerged in recent years and are available to support Web developers evaluating Accessibility of existing Web applications. However, as we shall see next in the related work section, there are not so many similar efforts for early design with the principles of Accessibility in mind [17]. In most cases Accessibility is considered as a programming issue or dealt with when the Web application is already fully developed, and in consequence the process of making this application accessible involves significant redesign and recoding, which may be considered outside the project's scope and budget [12]. Although, Accessibility is a vital quality attribute for people with disabilities, it has not yet gained enough recognition as a crucial non-functional requirement and success factor for Web applications such as security, performance, accuracy and usability.



Guarani3W: Student Management System on Internet

Student Login Form

Enter your Student ID and Password

Identification: What is the Student ID?

Password: What is the Student Password?

Submit

Data Fields Reset Reset

Attention!

We recommend as a safety measure that you change your

Figure 1. A Student's Login Web Page.

Our modeling approach [18] proposes to include Accessibility concerns systematically within a methodology for Web applications development. Firstly, to find out how Accessibility concerns should be introduced in the development life cycle, we analyzed how mature model-driven[1] Web Engineering (WE[2])

[1] Model-driven (MD) engineering is a software development methodology which 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.

[2] Web Engineering (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.

methods such as UWE [14], OOHDM [24], OOWS [10] or WSDM [9] 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 from the widget ontology [24], to concrete interface elements --e.g. HTML elements, and link both levels of abstraction from a UI design perspective [15] to WCAG checkpoints. Secondly, since designing accessible Web applications involves the analysis of different interests, we proposed to use Aspect-Oriented Software Development (AOSD) design principles and WCAG 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[3] characteristics of the Accessibility concern.

As a motivating example and to introduce properly the ideas behind our modeling approach, let us suppose a typical Web page whose purpose is a student's login aiming at his/her identification at his/her Argentine university system. Figure 1shows the page for the student's login which provides a user interface composed of HyperText Markup Language (HTML) elements, such as labels and textfFields. 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. Since a Web page for student's login requires at least two textField 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, according to the number of textField elements included in the presentation. Additionally, it is highly important to consider the positioning of the label element with respect to a textField element; this technological requirement for "until user agents" [28] --i.e. earlier "user agents" [30], also crosscuts the Web page. Clearly this kind of behavior perfectly fits the "scattering" and "tangling" problems[4] which motivate the main AOSD principles.

[3] "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 which is usually described as "crosscutting concerns" and should be centralized in one location in the code where possible.

[4] "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).

We have developed a supporting tool [19] to assist our proposal for developing accessible user interfaces (UI) for Web applications. Thus, based on our modeling approach [18] assisted by a supporting tool and using the case study bellow, this work is focalized on providing insights from the designer perspective when developing with the Accessibility concern in mind.

Since Accessibility is a critical quality factor to the success of Web applications, this paper is focused at the following:

- We briefly introduce our aspect-oriented approach, whose evolution can be tracked through [16][18][19], to show the advantages of modeling Accessibility concerns from the beginning and within a systematic development.
- We convey the experience gathered during the study and comparative application of ours and other approaches in the field of Accessibility design.
- We provide some insights trying to be critical enough to encourage towards the adoption of design principles and to call for Accessibility awareness within the developer community.

The rest of the paper is structured as follows: in Section 2, we briefly introduce five related work. In Section, 3 we discuss some background issues needed to understand our approach. In Section 4, we offer an overview of our approach using a real application example as a case study to illustrate our ideas. Section 5 continues with the case study and discusses some insights from the Accessibility design field. Finally, in Section 6, we conclude and present future work.

2. RELATED WORK

Our approach makes possible to treat Accessibility as an independent AOSD concern at early stages of the development's life cycle, therefore eliminating crosscutting and as a consequence allowing more modular system development, and the reuse of Accessibility aspects. Following, we briefly introduce five similar approaches that consider modeling the Accessibility concerns in at least, some of the stages of the development life-cycle.

For example, the main goal in Plessers et al. [23] is to produce annotations for visually impaired users automatically from the explicit conceptual knowledge existing during the design process. The approach integrates the Dante [31] annotation process into the Web Site Design Method (WSDM) [9] that allows Web applications to be developed in a systematic way.

The work by Centeno et al. [6] presents a set of rules which a design tool must follow in order to create accessible Web pages in a Web composition process. These rules are formalized with W3C standards like XPath[5] and XQuery[6] expressions, defining conditions to be met in order to guarantee that Accessible chunks of Web pages are safely compound into a page that also results Accessible. The authors also propose using the "Web Composition Service Linking System" (WSLS) [11] as Accessibility enabled authoring tool that makes this task feasible.

[5] W3C XML Path Language at www.w3.org/TR/xpath

[6] W3C XML Query Language at www.w3.org/TR/xquery

The accessible design proposed by Zimmermann & Vanderheiden [33] is based on existing "best practices of software engineering" as uses cases and scenarios, which were designed from its conception to meet functional requirements. The approach defines a new way of using proven tools of software engineering, such as use cases, scenarios, test cases, guidelines and checkpoints, for Accessibility purposes; and to relate them to each other to provide with a process model for accessible design and testing.

The work by Casteleyn et al. [3], focuses on how to extend a Hera-based Web application [13] with new functionality without having to redesign the entire application. To add new functionality, the authors propose to separate additional design concerns and describe them independently. Casteleyn et al. latest implementation [4][5] proposes a Semantic-based Aspect-oriented adaptation approach materialized in the form of a domain specific language, which the authors baptized Semantic-based Aspect-oriented Adaptation Language (SEAL)[7]. To demonstrate the practicality of their proposal, they apply and integrate SEAL in HydraGen engine[8] (an implementation generation tool for Hera-S developed externally by the University of Eindhoven).

Finally, the work by Moreno et al. [20] proposes a domain methodological framework for the development of accessible Web applications, which is called Accessibility for Web Applications (AWA). AWA proposes an Accessibility process and support for modeling by using techniques provided by model-driven development (MDD). The strategy in AWA is based on a Computational Independent Model (CIM), called domain specific AWA-Metamodel, which can be used to build Platform Independent Models (PIMs) and Platform Specific Models (PSMs) for accessible applications within WE methods.

3. UI DESIGN: INTERACTION DIAGRAMS AND SOFTGOAL INTERDEPENDENCY GRAPHS

In this section we introduce briefly two conceptual tools that working together allow to record Accessibility concerns early and as a reminder for design. They are: (2.1) User Interaction Diagrams (UIDs) with integration points, and (2.2) Softgoal Interdependency Graphs (SIGs) template for Accessibility.

3.1 Gathering Accessibility through UIDs with Integration Points

A User Interaction Diagram (UID) [26] is a diagrammatic modeling technique focusing exclusively on the information exchange between the application and the user. UIDs can be used to enrich the use cases models but they are also key graphical tools for linking requirements at later stages of a WE development process to obtain conceptual, navigational and user interface diagrams. With the traditional perspective given by techniques like [7][8] in mind, we introduce the concept of UIDs's integration points [16] to model the Accessibility concerns of a user-system interaction. Particularly, we define two kinds of UIDs integration points as follows:

- User-UID Interaction (U-UI) integration point. This is an integration point for Accessibility at UID interaction level --i.e., to propitiate an accessible communication and information exchange between the user and a particular interaction of a UID interaction diagram.
- User-UID Interaction's component (U-UIc) integration point. This is an integration point for Accessibility at UID interaction's component level --i.e., to propitiate an accessible communication and information exchange between the user and a particular UID interaction's component of an UID interaction.

These integration points with different granularity provide two alternatives for evaluating Accessibility during the interaction between the user and the system. Figure 2 shows the resultant UID, corresponding to the use case "Login a student given the student's ID and password" (introduced in Section 1 by Figure 1), by applying our integration points technique. Notice that all the students (including those with disabilities) will need to interact with this online login Web page. As we can see in the example shown in Figure 2, we define two integration points at UID interaction <1> representing the student's login user-system interaction to consider, from the beginning, the Accessibility requirements that make easy the access for all the students.

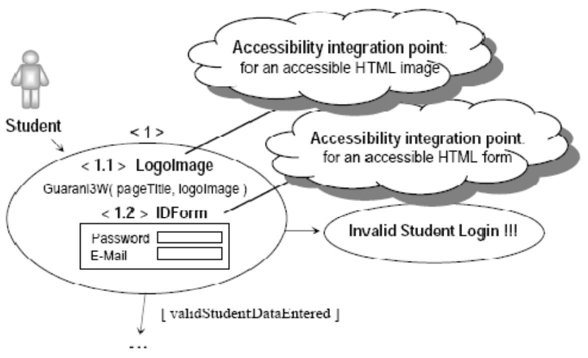

Accessibility integration point:
for an accessible HTML image
Student
< 1 >
Accessibility integration point:
for an accessible HTML form
< 1.1 > LogoImage
Guarani3W(pageTitle, logoImage)
< 1.2 > IDForm
Password
E-Mail
Invalid Student Login !!!
[validStudentDataEntered]
...

Figure 2. UID with Accessibility Integration Points: Login a Student given the Student's ID and Password.

Basically, the UID with integration points notation prescribes the inclusion of a cloud for every UID interaction or UID interaction's component, where Accessibility is essential to the user's task completeness. The first cloud establishes the <1.1> integration point to help convey the right semantics of the logo image; while the second cloud establishes the <1.2> integration point to propitiate an accessible form for user identification.

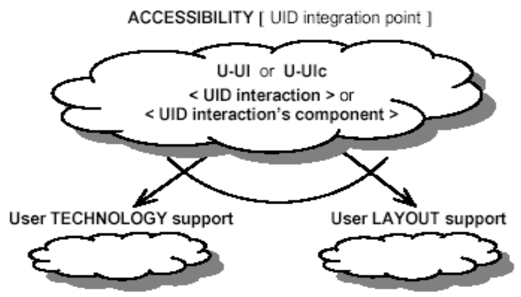


Figure 3. SIG Template for Accessibility.

[7] http://wise.vub.ac.be/downloads/research/seal/SEALBNF.pdf

[8] http://wwwis.win.tue.nl/~ksluijs/material/Singh-Master-Thesis-2007.pdf

3.2 Applying SIG Template for Accessibility

After specifying the Accessibility integration points of the UIDs diagrams, we propose to develop a SIG diagram for WCAG 1.0 Accessibility requirements [16]. Figure 3 shows our SIG *template* conceptual tool which we introduced taking into consideration proposals from the user interface design literature [15].

Figure 3 shows our SIG *template* where the Accessibility softgoal[9] denoted with the nomenclature Accessibility [UID integration point] is the root of the tree. The kind of the UID integration point is highlighted into the root light cloud and related to a particular UID interaction or UID interaction's component number. From the root node we identify two initial branches: (i) the user technology support, and (ii) the user layout support. The user technology support represents the Accessibility softgoal concerns helping to help user's browsing and interaction by improving the Accessibility of user's current and earlier assistive devices and technologies (PDAs, telephones, screen readers, etc.); meanwhile, the user layout support represents the Accessibility softgoal concerns explicitly improving user's browsing and interaction focus on user's interface issues. The Accessibility softgoal concerns supply to their respective supports, prescribing on how to present and/or to logically organize the content we wish to convey to the user. They also warn about the Accessibility barriers as a consequence of an inappropriate choice of presentation and/or structural objects to user's interaction with the content. Now, with this statement in mind, in order to associate the three design decision classes related to user interaction from Larson's user interface design decision framework [15] --i.e., dialogue, presentation and pragmatic, with the Accessibility softgoal concerns at some of the SIG's branches, we take into account the following considerations (Rationale behind this decision can be found in [18]):

- The concerns at the user layout support are associated with the dialogue and/or the presentation classes.
- The concerns at the user technology support are associated with the dialogue and/or the presentation classes if they help achieving device independence, especially focused on supporting the constraints of earlier assistive devices --i.e., "until user agents" as defined by the W3C's WCAG 1.0 [28]; meanwhile, they are associated with the three classes (dialogue, presentation and pragmatic) if they are hardware-dependent.

For example, returning to Figure 2, we establish the Accessibility softgoal for the interaction's components <1.1> LogoImage and <1.2> IDForm to guarantee accessible image and text input fields for all the students by defining two User-UID Interaction's components (U-UIc) integration points for the login process at UID interaction <1>. Finally, to instantiate the SIG template for ensuring Accessibility concerns (shown in Figure 3) we work with the W3C-WAI WCAG 1.0 guidelines [28][10]. To facilitate the instantiation process of the SIG template we establish an association table for groups of related HTML elements. Basically, these association tables have the tasks of linking each abstract interface element present at a user interface model (ontology concepts from an Abstract Widget Ontology [24]) with their respective concrete HTML elements, and with the Accessibility concerns prescribed for those elements by the WCAG 1.0 checkpoints.

It is important to highlight that our approach provides five association tables for groups of related HTML elements: (i) the HTML control group; (ii) the HTML link and button group; (iii) the HTML text and non-text group; (iv) the HTML structural group; and (v) the HTML frame and style sheet group. We called them association tables because of two strong reasons. On one hand, they bind the WACG 1.0 checkpoints required for ensuring Accessibility of the interface widgets present at each HTML group --i.e. they identify the required checkpoint for interface widgets present in a given Web page. On the other hand, they help to classify these WCAG 1.0 checkpoints into the two initial branches of our SIG template for Accessibility --i.e. they provide for each HTML element present in a group, two generic aspects working for the user's layout and technology Accessibility supports respectively.

Before proceeding, we must clarify that the Abstract Widget Ontology [24] provides the vocabulary used to define the abstract interface model specifying that an abstract widget can be any of the following: (i) SimpleActivator, (ii) ElementExhibitor, or (iii) VariableCapture. We refer the reader to [24] for further details of the ontology. Returning to the explanation, the first step to obtain these association tables comes from a mapping between abstract interface widgets (ontology concepts from Abstract Widget Ontology [24]) and concrete interface widgets (HTML elements). While the reason for HTML elements at the concrete interface model is completely clear, the purpose of the widget ontology is to provide an abstract interface vocabulary to represent the various types of functionality that can be played by interface widgets with respect to the activity carried out, or the information exchanged between the user and the application. Thus, the ontology can be thought of as a set of classes whose instances will comprise a given interface. Given these conceptual tools, the instantiation process of the SIG template is conducted as a refinement process over the SIG tree using the abstract interface model and the association tables as a reference.

4. AN ASPECT-ORIENTED APPROACH TO DEVELOP ACCESSIBLE UI FOR WEB APPLICATIONS

We propose an iterative and incremental process, which uses, as input, a set of Web application's requirements as provided by any WE approach (e.g., a set of use cases, goals, etc). The model we envisage to deal with Accessibility concerns within a Web engineering approach is illustrated in Figure 4, whose columns indicate: (i) the overall process with their main activities (in the middle), (ii) the conceptual tools and languages used (on the right) along with relations to the stage of the process where they are required, and (iii) the artifacts provided as input by the WE approach and / or delivered as output by our process (on the left). In order to ease reading, we need to recall here some previous explanations. In Figure 4, most arrows indicate an input or output, except for the UID and SIG diagrams as shown in Figure 4(2.1) and 4(2.2), where the arrows are input/output. This is because

[9] We use the "softgoal" concept as described in [7][8]; from this point of view a "softgoal" is the representation of a non-functional requirement (NFR) that must be satisfied to improve some quality factor, which in our case is Accessibility, of a software under development.

[10] We must note that although in this work we apply WCAG 1.0, we are almost ready to migrate to WCAG 2.0 [23]; we turn to this issue in Section 4.1

there are situations in which these artifacts could be developed once and then reused in different Web projects.

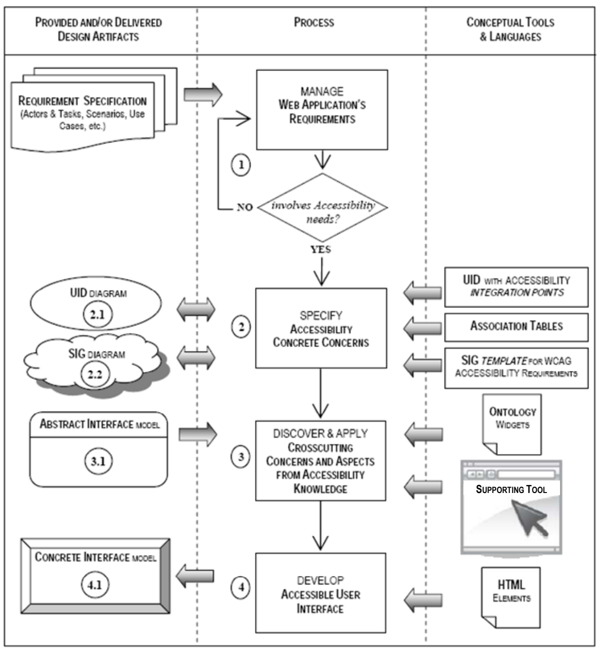

PROVIDED AND/OR DELIVERED DESIGN ARTIFACTS
PROCESS
CONCEPTUAL TOOLS & LANGUAGES
REQUIREMENT SPECIFICATION (Actors & Tasks, Scenarios, Use Cases, etc.)
MANAGE WEB APPLICATION'S REQUIREMENTS
1
NO
involves Accessibility needs?
YES
UID DIAGRAM
2.1
SIG DIAGRAM
2.2
2
SPECIFY ACCESSIBILITY CONCRETE CONCERNS
UID WITH ACCESSIBILITY INTEGRATION POINTS
ASSOCIATION TABLES
SIG TEMPLATE FOR WCAG ACCESSIBILITY REQUIREMENTS
ABSTRACT INTERFACE MODEL
3.1
3
DISCOVER & APPLY CROSSCUTTING CONCERNS AND ASPECTS FROM ACCESSIBILITY KNOWLEDGE
ONTOLOGY WIDGETS
SUPPORTING TOOL
CONCRETE INTERFACE MODEL
4.1
4
DEVELOP ACCESSIBLE USER INTERFACE
HTML ELEMENTS

Figure 4. Overview of Our Approach.

For example, Accessibility requirements of an image or a basic data entry form can be modeled once, and later reuse in new projects which require these interface elements. It is important to highlight, that almost all WE approaches have an explicit development activity for user interface design and, normally, a user interface is specified by the abstract interface and the concrete interface models, providing respectively the type of functionality offered to the user by the interface elements and the actual implementation of those elements in a given runtime environment. So, given a user's task, the SIG model provides the WCAG 1.0 Accessibility checkpoints that crosscut the interface widgets (both, abstract and concrete ones, as shown in Figure 4(3.1) and 4(4.1) respectively), to propitiate an accessible user experience.

At stage 3 and as shown by Figure 4(3), our supporting tool assists developers to discover and apply crosscutting concerns and aspects from knowledge about Accessibility. Figure 5 shows the class diagram of the tool's architecture which has three main components or layers: (1) Object Storage, (2) Core, and (3) Presentation. Particularly, in Figure 5, we focus on the Presentation layer which is isolated from the other layers and it is only related to the Core layer by a dotted line, meaning that there is no straight interaction between these two layers. Thus, the interaction between these two layers, which includes reading and analyzing the abstract interface model under treatment, takes place in a transparent manner due to the proposed tool which implements the typical behavior of aspect-orientation and fulfills conformance to Accessibility Aspects I and II. As shown by Figure 6(a) and we explain in Section 5, Aspect I and Aspect II are specified for avoiding "scattering" and "tangling" symptoms at the user layout and user technology supports for Accessibility.

To reproduce this behavior, the tool uses the Observer pattern and their classes Subject and Observer; each instance of the Subject class maintains a list of instances of the Observer class which are notified of the changes that occur in their respective instance of the Subject class. Applying these design concepts, at the Presentation layer in Figure 5, the AccessibilityTool class is of the class Subject, while at the Core layer in Figure 5, the InterfaceAnalizer class is of the class Observer. Then, the list and updates notification is implemented by the aspects environment (AspectJ).

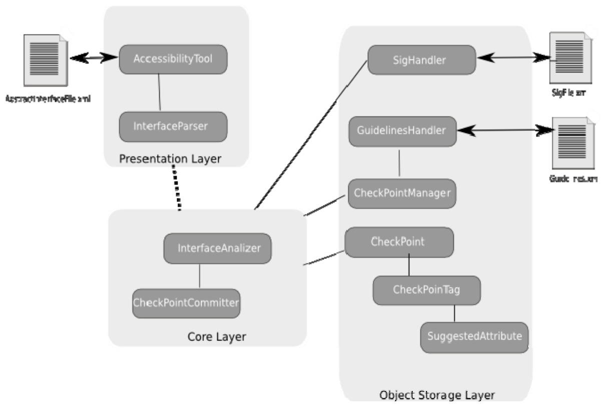


Figure 5. Class Diagram of Our Supporting Tool's Architecture.

Thus, when the developer saves the document edited for the abstract interface model, this automatically triggers this aspect-oriented functionality which is not explicitly invoked by some element of the Presentation layer. Then, as shown in Figure 4(4) and Figure 5(b), Aspects I and II can be seamless injected by the "weaving" mechanism into the core --i.e., user interface models, to achieve the Accessibility softgoal. As shown in Figure 4(4.1), the consequence is an HTML code with the desired conformance to the WCAG 1.0.

We refer the reader to [18][19] for a detailed description of our method and supporting tool.

5. A MOTIVATING CASE

As shown in Figure 4(2.1) and (2.2), we propose an early capture of Accessibility concrete concerns by developing two kinds of diagrams: the UID with Accessibility integration points and the Softgoal Interdependency Graph (SIG) template for WCAG 1.0 Accessibility requirements. As we explained previously in Sections 2 and 3, we propose these conceptual tools basically to allow the representation of Accessibility requirements while executing a user's task.

For example, Figure 6(a) shows the SIG diagram, as a result of an instantiation process of the SIG's template, for the Accessibility integration points outlined by the UID in Figure 2 to identify WCAG 1.0 Accessibility requirements. For brevity reasons, we develop in this diagram only the branch for the UID interaction's component <1.2> IDForm to guarantee accessible text input fields for all the students, including those with disabilities. Applying the SIG's template and using the SIG's notation and vocabulary, the HTML form in the concrete interface model, corresponding to a

CompositeInterfaceElement of the abstract interface model, is the focus of the Accessibility softgoal highlighted into the root light cloud. As shown in Figure 6(a), the user technology support and the user layout support branches are specified into light clouds and dark clouds respectively. The light clouds represent the refined Accessibility softgoal, --i.e., the required WCAG 1.0 guidelines; while the dark clouds represent operationalizing goals --i.e., the required checkpoints to be satisfied. In this case, we use for SIG's instantiation the association table for the HTML control elements, since the Accessibility softgoal is defined for an IDForm.

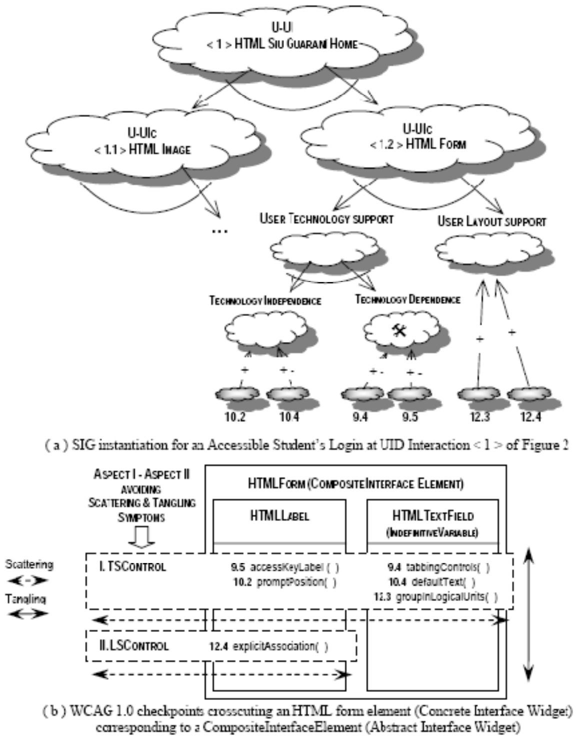
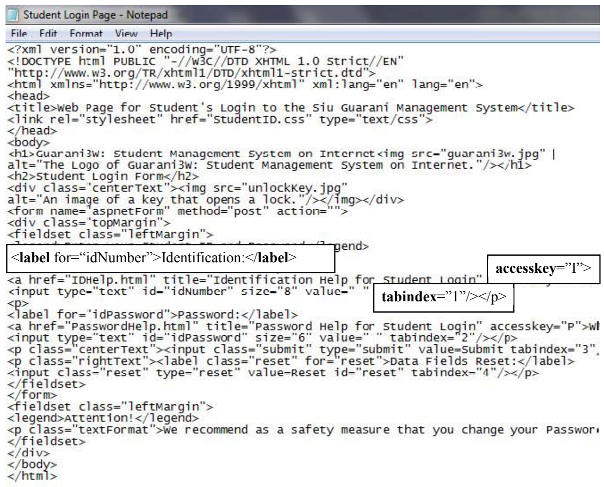


Figure 6. Managing Crosscutting Symptoms in an Aspect-Oriented Manner.

Returning to Figure 4(3), the Accessibility knowledge captured and organized by SIG diagrams at early stages aids designers making decisions through the abstract interface model, as shown in Figure 4(3.1). The purpose here is to find out how WCAG 1.0 Accessibility requirements "crosscut" interface widgets required for an IDForm. Since applying the required WCAG 1.0 checkpoints to be satisfied at the user interface causes typical crosscutting symptoms --i.e., "scattering" and "tangling" problems as shown by Figure 6(b), it is clear that aspect-orientation is the natural approach to solve these crosscutting symptoms. As we already explained above in the previous paragraph and Figure 6(a) shows, the SIG diagrams provide through the instantiation of their two branches, Accessibility technology and layout support respectively for any of the HTML form components at the user interface. Also, and as Figure 6(b) shows, the SIG diagrams branches allow Aspects I and II to be modeled and instantiated appropriately to avoid "scattering" and "tangling" problems.

Finally, Figure 7 shows the HTML document delivered by our supporting tool according to the concrete interface model. Figure 7 shows three boxes to highlight an example where HTML code is improved with Accessibility properties as follow:

- A label with the text "Identification" is inserted and explicitly associated with the texField "idNumber" to make clear what kind of data is expected to be introduced in this textField.
- A keyboard shortcut is defined using the HTML accesskey attribute for the textField "idNumber" to provide direct access through the keyboard key "I".
- A logical tab order "1" is provided using the HTML tabindex attribute for the textField "idNumber" to support a logical page design.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Web Page for Student's Login to the Siu Guaraní Management System</title>
<link rel="stylesheet" href="StudentID.css" type="text/css">
</head>
<body>
<h1>Guarani3W: Student Management System on Internet<img src="guarani3w.jpg" |
alt="The Logo of Guarani3W: Student Management System on Internet."/></h1>
<h2>Student Login Form</h2>
<div class='centerText"><img src="unlockKey.jpg"
alt="An image of a key that opens a lock."/></img></div>
<form name='aspnetForm" method="post" action="">
<div class='topMargin">
<fieldset class="leftMargin">
gend>
<label for="idNumber">Identification:</label>
<a href="IDHelp.html" title="Identification Help for Student Login"   accesskey="I">
<input type="text" id="idNumber" size="8" value=" "   tabindex="1"/></p>
<p>
<label for='idPassword">Password:</label>
<a href="PasswordHelp.html" title="Password Help for Student Login" accesskey="P">Wl
<input type="text" id="idPassword" size="6" value=" " tabindex="2"/></p>
<p class="centerText"><input class="submit" type="submit" value=Submit tabindex="3".
<p class="rightText"><label class="reset" for="reset">Data Fields Reset:</label>
<input class="reset" type="reset" value=Reset id="reset" tabindex="4"/></p>
</fieldset>
</form>
<fieldset class="leftMargin">
<legend>Attention!</legend>
<p class="textFormat">We recommend as a safety measure that you change your Passwor
</fieldset>
</div>
</body>
</html>
```

Figure7. Accessible and Well-Formed HTML Document.

5.1 Discussion from the Accessibility Design Field

We have been working for a while on Accessibility [16][17] and particularly on Accessibility design at early stages of Web applications development process [18] [2] [19]. Particularly, we have been applying aspect-orientation associated with the WCAG 1.0 as the reference guideline, and in this Section we present an outline of the experience gathered on the field. Since the WCAG has two documents (1.0 and 2.0), it is important to make clear at this point that we based our work on the WCAG 1.0, which since 1999 is keeping its value as the benchmark for other valuable Accessibility standards [22][25], while the ongoing migration process to WCAG 2.0 [29] is completed worldwide.

However, as we are concerned with Web Accessibility and the W3C as their main reference, we have already finished the migration of our design approach from WCAG 1.0 to WCAG 2.0 and we are currently working on the migration of our supporting tool as well. We highlight that to realize this upgrade we use the comparison provided by W3C-WAI in [31], since there are still some discrepancies at the Accessibility community11 when

[11] See http://www.w3.org/WAI/WCAG20/from10/comparison/; http://wipa.org.au/papers/wcag-migration; http://www.usability.com.au/resources/wcag2./

providing mappings between the WCAG 1.0 [28] checkpoints onto the WCAG 2.0 [29]success criteria. A complete analysis of this upgrade is outside the scope of the paper.

In order to share Accessibility experience gathered at early stages of the development process for the Web, we indicate some issues which give us the basis for the discussion from a designer perspective. Firstly, we consider that Accessibility concern requires a special treatment during Web development and as a consequence must be handled independently from the rest of the quality concerns. We believe that this is a hot-spot to the success of a proposal which seeks for the prioritization of Accessibility as a main quality concern in Web development. In this sense, our approach shows a high degree of commitment to the Accessibility concern by providing specific techniques developed to "isolate" Accessibility requirements and to ensure their separately treatment from the beginning in the Web development process.

Secondly, 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. Taking into account this fact and supported by our experience gathered from the design field, we are convinced that the AOSD paradigm is the most appropriated to deal with the nature of Accessibility in Web development. Our approach fully applies the AOSD paradigm to deal properly with the non-functional, generic and crosscutting features of Accessibility concern and to ensure its treatment as a first-class citizen early since requirement elicitation are weaved together using specialized techniques (for a thorough discussion on AOSD principles see [1]).

Thirdly, we are interested in considering as additional issues the significance to a design approach of having background and supporting tool. On one hand, the background underlying a design proposal is relevant to its strength. In this sense, our approach goes further because its background includes not only our previous work but also is supported by other's mature and recognized work concerned to field of the approach's purpose. On the other hand, the supporting tool and the kind of support given and features covered by the tool is also relevant, and especially to a design proposal. Related to this issue, our approach provides a supporting tool to assist developers in the implementation of cases, and on the creation of their corresponding models by using reusable components. Currently, our tool provides assistance for applying the Accessibility aspects (prescribed by the SIGs diagrams) to user interface models (abstract and concrete).

Finally, at this point, we would like to reflect on the advantages/disadvantages of model-driven approaches and how this issue benefits/affects our proposal. It is a fact that applying "unified", model-driven approaches brings the benefit of having full documentation and automatic application generation at the expense of introducing some bureaucracy into the development process. Since our proposal suggests the early treatment of the Accessibility concerns through models, we may still be influenced by this reality and its disadvantages --i.e., time and cost consuming, complexity, learning effort, etc.

Related to the project team and development environment, we believe it is important to highlight the following issues: (i) although our approach is completely documented and self-contained within a well-kwon Web engineering approach, its application requires a prior knowledge of the WCAG 1.0 (or 2.0) guidelines and their specific terminology; (ii) although our approach helps to transfer Accessibility requirements, the engineering staff members should not be ruled by ad hoc practices, or used to apply approaches, which have not incorporated the design and documentation of the application under development as an standard discipline. These two issues demand changes in the development process that must be supported by the organizations. In this sense, for Web development, quality is often considered as higher priority than time-to-market with the mantra later-and-better [21] even though they mean extra time and cost consuming. However, since the Accessibility guidelines are quite independent from the Web application under development, there are many cases to which the same Accessibility solution can be applied. Then, recording such recurrent situations (e.g., using patterns) might contribute to reuse them, which supplies to reduce the development effort when implementing our proposal. This is possible because aspects, as we have already explained, could be developed once and be reused in different Web projects. We refer the reader to [18] for a complete case study and evaluation.

6. CONCLUSIONS AND FUTURE WORK

A main factor for the lack of Accessibility at the Web is the major knowledge gap that normally exists between developers and Accessibility specialists. On one hand, most Web programmers do not have the knowledge and experience required to ensure that its code meets the Accessibility requirements. On the order hand, Accessibility specialists have little experience in development and normally even less in programming, not being able to provide examples of pieces of code that can be used by developers to make "accessible" their Web applications. Moreover, it is a common practice to consider Accessibility at the very last stages of the development process, or when applications are already coded. At this point "make these applications accessible" can mean a great deal of redesign and reprogramming effort usually outside the scope of the project --i.e not previously planned and/or budgeted from the beginning.

In this paper, we briefly introduce our proposal to model Web Accessibility by moving from abstract to concrete architectural views using aspect-orientation. Our approach takes advantages of modeling Accessibility as an aspect-oriented concern, which is independently treated as a first-class citizen to avoid barriers from the beginning of the design. We used a case study to illustrate our ideas and point out the advantages of a clear separation of concerns throughout the development life-cycle. Then, based on our experience from the Accessibility field, we aim to provide some insights that Web developers can apply when designing for the Web with the Accessibility concern in mind.

Since we are aware that the new W3C-WAI guidelines and the move to technological neutrality are undoubtedly good, we are almost ready to migrate from WCAG 1.0 [28] to WCAG 2.0 [29]; we have already finished the migration of our aspect-oriented design approach and we are currently working on the migration of our supporting tool as well. We didn't find major inconveniences to upgrade our approach to WCAG 2.0 because as we discussed before, our approach is based on the use of UIDs with integration points and the SIG template for Accessibility linked by association tables. These association tables are the key conceptual tools which enable the migration and support the success criteria from WCAG 2.0 instead of checkpoints from WCAG 1.0 applying some straightforward redefinitions and adjustments. We highlight that to realize this upgrade we use the comparison provided by W3C-WAI, since there are still some discrepancies at the Accessibility community[12] when providing mappings between the WCAG 1.0 checkpoints onto the WCAG 2.0 success criteria.

Finally, we will further validate our proposal working with WCAG 2.0 beyond the example used to illustrate our work and make some comparisons between case studies that we have been applied during the validating process.

ACKNOWLEDGMENTS

This work is partially supported by UNPA-UACO project 21/B107 (Mejora del Proceso de Selección de Componentes para Sistemas de Información Geográficos). Also it is supported by UNComa project 04E/072 (Identificación, Evaluación y Uso de Composiciones Software).

REFERENCES

[1] Baniassad, E. L. A., Clements, P. C., Araújo, J., Moreira, A., Rashid, A., Tekinerdoga, B.: Discovering Early Aspects. *IEEE Software* 23(1), 2006, 61-70 doi.ieeecomputersociety.org/10.1109/MS.2006.8

[2] Bustos, B., Martín, A. and Cechich, A. Diseño de Interfaces Guiado por Restricciones de Accesibilidad Web. in *XIII Congreso Americano en "Software Engineering"*, (Cuenca, Ecuador, 2010), Universidad del Azuay, 229-242

[3] Casteleyn, S., Fiala, Z., Houben, G-J., and van der Sluijs, K. Considering Additional Adaptation Concerns in the Design of Web Applications. in *4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, (Dublin, Ireland, 2006), Springer, 254-258 doi:10.1007/11768012_28

[4] Casteleyn, S., Van Woensel, W., and Houben, G-J. A Semantics-based Aspect-Oriented Approach to Adaptation in Web Engineering. in *18th ACM Conference on Hypertext and Hypermedia*, (Manchester, UK, 2007), ACM, 2007, 189-198 doi.acm.org/10.1145/1286240.1286297

[5] Casteleyn, S., Van Woensel, W., van der Sluijs, K., and Houben, G.J. Aspect-Oriented Adaptation Specification in Web Information Systems: a Semantics-based Approach. *The New Review of Hypermedia and Multimedia*, 15(1), 2009, 39-91 10.1080/13614560902818297

[6] Centeno, V., Kloos, C., Gaedke, M., and Nussbaumer, M. Web Composition with WCAG in Mind. in *5th International Conference on Cross-Disciplinary Workshop on Web Accessibility,*(Sydney, Australia, 2005), Springer, 615-617 doi:10.1145/1061811.1061819

[7] Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. *Non-Functional Requirements in Software Engineering.* Kluwer Academic Publishers, Boston, 2000

[8] Chung, L. and Supakkul, S. Representing NFRs and FRs: A Goal-oriented and Use Case Driven Approach. in *2nd International Conference on Software Engineering Research*, (Los Angeles, USA, 2004), Springer, 29-41 doi:10.1007/11668855_3

[9] De Troyer O., Casteleyn, S., and Plessers, P. WSDM: Web Semantics Design Method. in: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) *Web Engineering: Modeling and Implementing Web Applications*. Springer-Verlag, London, 2008, 303-351

[10] Fons, J., Pelechena, V., Pastor, O., Valderas, P., and Torres, V. Applying the OOWS Model-Driven Approach for Developing Web Applications. The Internet Movie Database Case Study. in: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) *Web Engineering: Modeling and Implementing Web Applications*. Springer-Verlag, London, 2008, 65-108

[11] Gaedke M., Nussbaumer, M., and Meinecke, J.: WSLS: A Service-Based System for Reuse-Oriented Web Engineering. in: Matera, M., Comai, S. (eds.) *Engineering Advanced Web Applications*, Rinton Press, NJ, 2004, 26-37

[12] Hoffman, D., Grivel, E., and Battle, L.: Designing Software Architectures to Facilitate Accessible Web Applications. *IBM Systems Journal*, 44(3), 2005, 467-484 doi 10.1147/sj.443.0467

[13] Houben, G-J., van der Sluijs, K., Barna, P., Broekstra, J., Casteleyn, S., Fiala, Z., and Fransincar, F. Hera. in: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) *Web Engineering: Modeling and Implementing Web Applications*. Springer-Verlag, London, 2008, 163-302

[14] Koch, N., Knapp, A., Zhang, G., and Baumeister, H. UML-Based Web Engineering: An Approach Based on Standards. in: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) *Web Engineering: Modeling and Implementing Web Applications*. Springer-Verlag, London, 2008, 157-191

[15] Larson, J. *Interactive Software: Tools for Building Interactive User Interfaces*. Prentice Hall, NJ, 1992

[16] Martín, A., Cechich, A., Gordillo, S., and Rossi, G. A Three-Layered Approach to Model Web Accessibility for Blind Users. in *5th Latin American Web Congress*, (Santiago de Chile, Chile, 2007), IEEE Computer Society, 2007, 76-83 doi:10.1109/LA-WEB.2007.56

[17] Martín, A., Cechich, A., and Rossi, G.: Comparing Approaches to Web Accessibility Assessment. in: Calero, C., Moraga, Mª Á., Piattini, M. (eds.) *Handbook of Research on Web Information Systems Quality*, IGI Global Information Science Reference, Hershey NY, 2008, 181-205

[18] Martín, A., Rossi, G., Cechich, A., and Gordillo, S. Engineering Accessible Web Applications. An Aspect-Oriented Approach. *World Wide Web Journal*, 13(4), 2010, 419-440 doi:10.1007/s11280-010-0091-3

[19] Martín, A., Mazalú, R., and Cechich, A. Supporting an Aspect-Oriented Approach to Web Accessibility Design. in *5th International Conference on Software Engineering Advances,* (Nice, France, 2010), IEEE, 20-25 doi:10.1109/ICSEA.2010.10

[20] Moreno, L., Martinez, P., Ruiz, B. A MDD Approach for Modeling Web Accessibility. in *7th Int. Workshop on Web-*

[12] See http://www.w3.org/WAI/WCAG20/from10/comparison/; http://wipa.org.au/papers/wcag-migration.htm; http://www.usability.com.au/resources/wcag2./

Oriented Software Technologies, (New York, USA, 2008), CEUR Workshop Proceedings, 1-6 doi:10.1.1.163.9478

[21] Offutt, J. Quality Attributtes of Web Software Applications. *IEEE Software*, 19(2), 2002, 25-32 doi:10.1002/stvr.425

[22] PAS 78. Publicly Available Specification: A Guide to Good Practice in Commissioning Accessible Websites. Retrieved January 1, 2011 from: http://www.hobo-web.co.uk/seo-blog/pas-78/

[23] Plessers, P., Casteleyn, S., Yesilada, Y., De Troyer, O., Stevens, R., Harper, S., and Goble, C. Accessibility: A Web Engineering Approach. in *14th International Conference on World Wide Web*, (Chiba, Japan, 2005), ACM, 353-362 doi:10.1145/1060745.1060799

[24] Rossi, G. and Schwabe, D. Modeling and Implementing Web Applications with OOHDM. in: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) *Web Engineering: Modeling and Implementing Web Applications*. Springer-Verlag, London, 2008, 109-155

[25] Section 508. Electronic and Information Technology Accessibility Standards. Retrieved January 1, 2011 from: http://www.section508.gov/

[26] Vilain, P., Schwabe, D., and Sieckenius de Souza, C. A Diagrammatic Tool for Representing User Interaction in UML. in *3rd International Conference on UML* (York, UK, 2000), Springer, 133-147 doi:10.1007/3-540-40011-7_10

[27] W3C: Web Content Accessibility Guidelines (WCAG) Overview. Retrieved December, 15, 2010, from: http://www.w3.org/WAI/intro/wcag.php

[28] W3C: Web Content Accessibility Guidelines 1.0. (WCAG 1.0). Retrieved December, 15, 2010, from: http://www.w3.org/WAI/intro/wcag.php

[29] W3C: Web Content Accessibility Guidelines 2.0 (WCAG 2.0). Retrieved December, 15, 2010, from: http://www.w3.org/TR/WCAG20/

[30] W3C: User Agent Accessibility Guidelines 1.0 (UAAG 1.0). Retrieved December 15, 2010 from: 05.24.2010http://www.w3.org/TR/WAI-USERAGENT/

[31] W3C-WAI: Comparison of WCAG 1.0 Checkpoints to WCAG 2.0. Retrieved December 15, 2010 from: http://www.w3.org/WAI/WCAG20/from10/comparison/

[32] Yesilada, Y., Harper, S., Goble, G., and Stevens, R. DANTE: Annotation and Transformation of Web Pages for Visually Impaired Users. in *13th International Conference on World Wide Web*, (New York, USA, 2004), ACM, 490-491 doi.acm.org/10.1145/1013367.1013540

[33] Zimmermann, G. and Vanderheiden, G. Accessible Design and Testing in the Application Development Process: Considerations for an Integrated Approach. *Universal Access in the Information Society*, 7(1-2), 2008, 117-128 doi.org/10.1007/s10209-007-0108-6