International Conference on Software Engineering Advances. International Academy, Research and Industry Association (IARIA), Nice, 2010.

Supporting an Aspect-Oriented Approach to Web Accessibility Design.

Martin, Adriana, Mazalu, Rafaela y Cechich, Alejandra.

Supporting an Aspect-Oriented Approach to Web Accessibility Design

Adriana Martín[1,2]
[1]Unidad Académica Caleta Olivia
Universidad Nacional de la Patagonia Austral
Caleta Olivia, Santa Cruz, Argentina
[2]GIISCo, Departamento de Ciencias de la Computación
Universidad Nacional del Comahue
Neuquén, Argentina
e-mail: adrianaelba.martin@gmail.com

Rafaela Mazalu[2], Alejandra Cechich[1,2]
[1]Unidad Académica Caleta Olivia
Universidad Nacional de la Patagonia Austral
Caleta Olivia, Santa Cruz, Argentina
[2]GIISCo, Departamento de Ciencias de la Computación
Universidad Nacional del Comahue
Neuquén, Argentina
e-mail: {rafaelamazalu@gmail.com,
acechich@gmail.com}

***Abstract*—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. Developing accessible Web applications is usually hard for several reasons. Firstly, there is a significant knowledge gap between developers and Accessibility specialists. Secondly, 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. 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. In this paper, we present an aspect-oriented approach to handle the non-functional, generic and crosscutting characteristics of the Accessibility concerns. Our approach is supported by several techniques, which are embedded in a software tool aiming at facilitating transferring the approach to industry.**

Keywords-Web Accessibility; User Interface Models; Web Engineering; Aspect-Oriented Design.

I. INTRODUCTION

Web Accessibility means universal access on the Web, regardless the kind of hardware, software, network platform, language, culture, geographic location and users' capabilities. The Word Wide Web Consortium (W3C) is the main referent of Web Accessibility and has worked for more than ten years in the development of guidelines called Web Content Accessibility Guidelines 1.0 (WCAG 1.0) [16], which are considered as references 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 assist Web developers in assisting the evaluation of Accessibility of existing Web applications. In contrast, there are not so many similar efforts for the early design with the principles of Accessibility in mind [10]. For example, the main goal in Plessers et al. [13] is to generate the annotations for visually impaired users automatically from the explicit conceptual knowledge existing during the design process. The approach integrates the Dante [18] annotation process into the Web Site Design Method (WSDM) [5] that allows Web applications to be developed in a systematic way. The work by Centeno et al. [2] presents the set of rules that, in a Web composition process, a design tool must follow in order to create accessible Web pages. The accessible design proposed by Zimmermann & Vanderheiden [19] is based on existing "best practices of software engineering" (basically uses cases and scenarios), which were designed from its conception to meet functional requirements. Finally, Casteleyn et al. [1], focus on how to extend an application 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.

The objective of our approach is different from the formers because it is focused on providing specific modelling techniques to include Accessibility systematically in a methodology for Web applications development. To find out how the Accessibility concerns should be introduced in the development life cycle, we analyzed how mature model-driven Web Engineering (WE) approaches such as UWE [7], OOHDM [14], OOWS [6] or WSDM [5] face this cycle. All of them comprise several activities focusing on some specific design concerns; however, OOHDM fulfils many of our expectations, so we decided to join our approach to this particular WE approach.

Since designing accessible Web applications involves the analysis of different interests, our approach proposes using Aspect-Oriented Software Development (AOSD) design principles and Web Content Accessibility Guidelines 1.0 (WCAG 1.0) to support the construction of accessible user interfaces. The fact that we choose the aspect orientation to develop our proposal ensures handling naturally the non-functional, generic and crosscutting characteristics of the Accessibility concern. As a motivating example, suppose a Web page whose purpose is the student's login for his/her identification in his/her university system, like the one shown in Figure 1 corresponding to the SIU Guaraní registration system, which is used by many public universities in Argentina. As shown in Figure 1, the page for the student's login provides a user interface composed of HyperText Markup Language (HTML) elements, like labels

and textFields. To ensure an accessible interaction, these HTML elements must fulfill some Accessibility requirements and, it is not surprising that all of these requirements crosscut the same software artifact (the Web page for student's login). For example, and as we see in detail later, an HTML label element is, at the presentation, a basic layout Accessibility requirement for many others HTML elements.

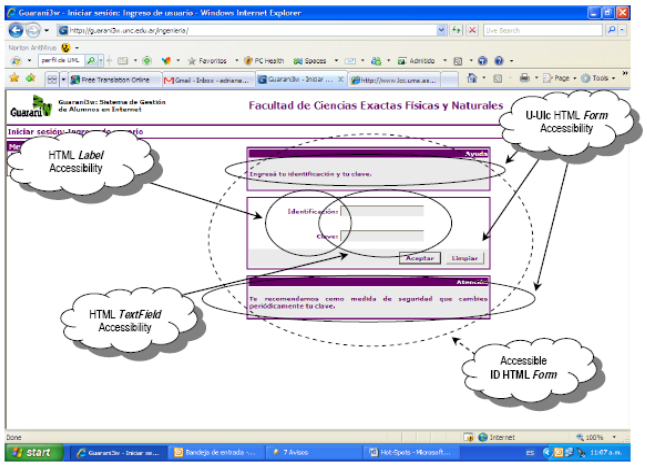

Figure 1. The student's login Web page at the SIU Guaraní registration system

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 respectively label elements must be tested. So, to ensure an accessible interaction 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" [17] also crosscuts the Web page. Clearly this kind of behavior perfectly fits the "scattering" and "tangling" problems which motivate the main AOSD principles.

The rest of the paper is structured as follows: in Section II, we discuss some background issues needed to understand our approach. In Section 3, we offer an overview of our approach using a real application example to illustrate our ideas. Section 4 briefly introduces the architecture of a supporting tool; and in Section 5 we conclude and present some further work.

II. BUILDING BLOCKS: 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 of design. They are: (A.) User Interaction Diagrams (UIDs) with integration points and, (B.) Softgoal Interdependency Graphs (SIGs) template for Accessibility.

A. *Accessibility through UIDs with Integration Points*

A User Interaction Diagram (UID) [15] 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 [3][4] in mind, we introduce the concept of UIDs's integration points [9] 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 ensure the access for all the students.

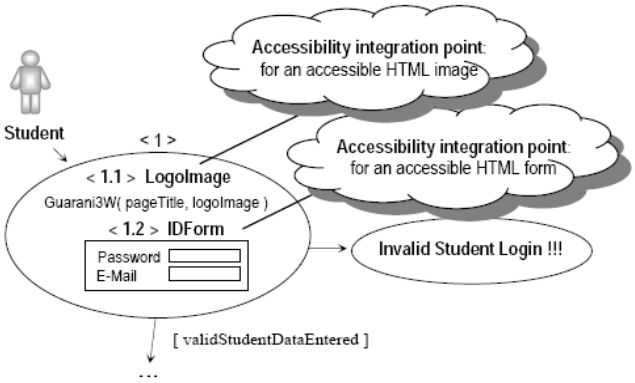

Figure 2. UID with Accessibility integration points: Login a student given the student's ID and password

Basically, the UID with integration points notation prescribes the inclusion of a cloud for every UID interaction or UID interaction's component, which Accessibility is

essential to the user's task completeness. The first cloud establishes the <1.1> integration point to ensure that the semantics of the logo image is correctly transmitted; while the second cloud establishes the <1.2> integration point to ensure an accessible form for user identification.

B. Applying the 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 [9]. Figure 3 shows our SIG *template* conceptual tool which we introduced taking into consideration proposals from the user interface design literature [8].

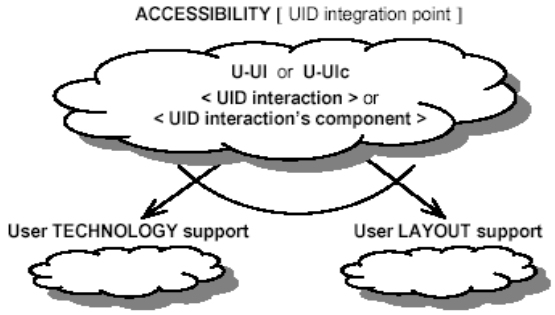

Figure 3. SIG template for Accessibility

Figure 3 shows our SIG template where the Accessibility softgoal 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 ensure 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 --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:

- 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 UAAG 1.0 [17]; 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 [16]. 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 [14]) with their respective concrete HTML elements, and with the Accessibility concerns prescribed for those elements by the WCAG 1.0 checkpoints. Before proceeding, we must clarify that the Abstract Widget Ontology 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 [14] 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 [14]) 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.

III. AN ASPECT-ORIENTED APPROACH TO DEVELOPING ACCESIBLE WEB APPLICATIONS

In the spirit of modern Web Engineering approaches, we propose a model-driven development process in which the construction of a Web application consists of the specification of a set of conceptual models, each addressing a different concern (such as navigation or interface). 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 there are situations in which these artifacts could be developed once and then reused in different Web projects. For example, the 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.

As highlighted in Figure 4(1), this process manages Web application requirements looking for those that involve Accessibility needs. This is because it is at the user's interface level where Accessibility barriers finally show, so we are particularly interested in discovering Accessibility requirements at the user interface design. Then, as shown in Figure 4(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 shown in Figure 4(2.1) and (2.2) respectively. As we explained previously, we propose these conceptual tools basically to allow the representation of Accessibility requirements while executing a user's task.

For example, Figure 5(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 at 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 5(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.

As indicated in 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 5(b), it is clear that aspect-orientation is the natural approach to solve these crosscutting symptoms. The SIG diagrams not only provide Accessibility technology and layout support respectively for any of the HTML form components at the user interface, but also allow Aspects I and II to be modeled and instantiated appropriately to avoid "scattering" and "tangling" problems. Then, as shown in Figure 4(4), Aspects I and II can be seamless injected by aspect "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.

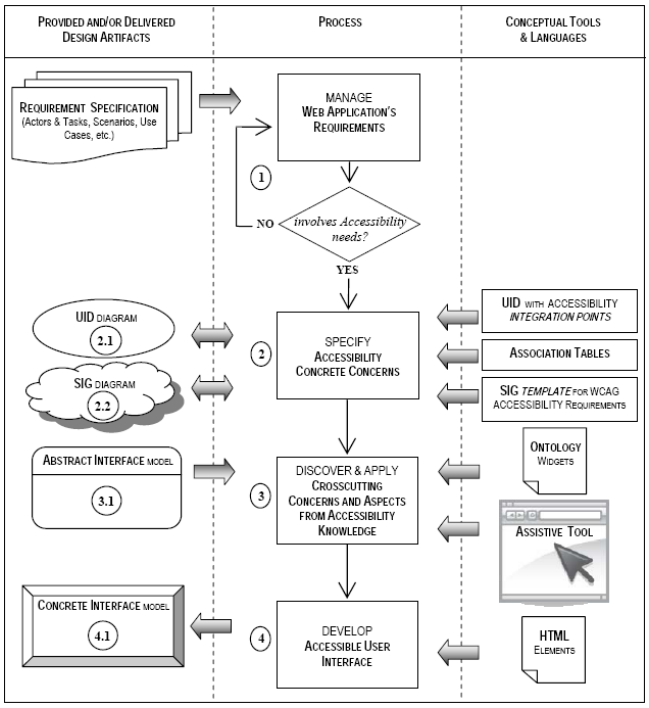

Figure 4. Overview of our approach

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 ensure an accessible user experience.

IV. A SUPPORTING TOOL

Finally, we briefly describe the architecture of a supporting tool that assists the process of discovering and applying crosscutting concerns and aspects from knowledge about Accessibility, as shown in Figure 4. To clarify this point, Figure 6 shows the main components of the tool arranged into three layers: (1) Object Storage, (2) Domain, and (3) Presentation.

(1) Object Storage Layer. Here data elements –i.e concrete interface widgets, are stored in a standardized format. To do so, we have selected the eXtensible Markup Language (XML) that allows us to express WCAG 1.0 checkpoints in a universally understandable language. In Addition, this layer also implements procedures and schemes necessary to correctly transforming the abstract interface model into a concrete HTML-expressed model.

(2) Domain Layer. It is composed of two modules: controller and transformer. The former manages requests coming from the presentation layer and manipulates them by considering Accessibility aspects; i.e. controller is in charge of deciding which information is relevant respect to a particular aspect. It also decided which objects are involved in each treatment. On the other hand, the transformer eliminates the crosscutting nature of SIGs' Accessibility aspects by replacing their checkpoints requirements with fragments of well-formed and accessible HTML code. Thus, the transformer creates the concrete interface implemented in HTML in concordance to the aspects' checkpoints of each SIG. In this way, abstract widgets are mapped into accessible HTML widgets.

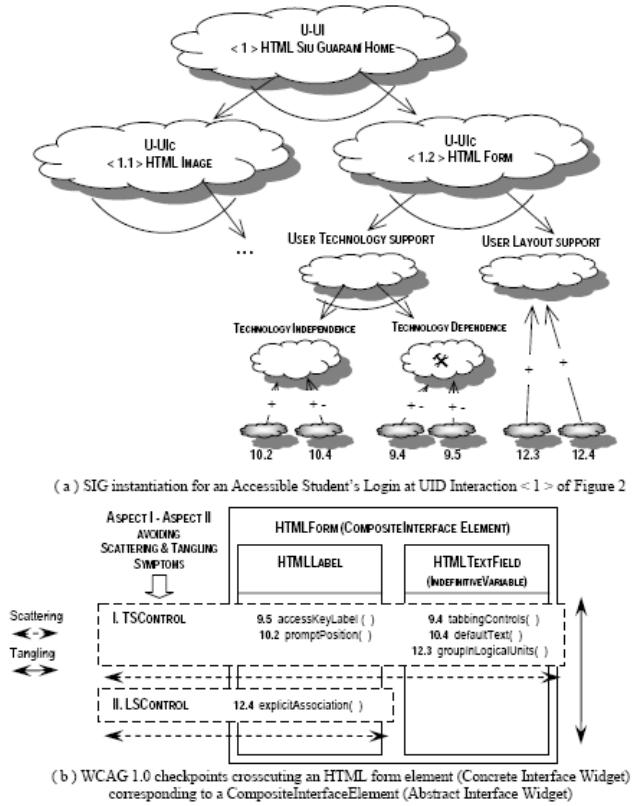

Figure 5. Managing crosscutting symptoms in an Aspect-Oriented manner

(3) Presentation Layer. It is in charge of capturing the abstract interface model and the WCAG 1.0 principles expressed by the SIG for Accessibility. This layer separates treatments between those supporting layout and pragmatic (technology) aspects.

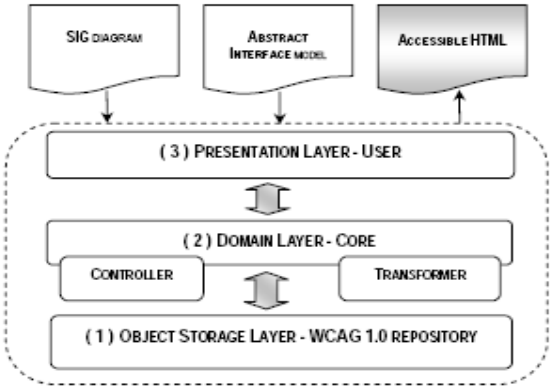

Figure 6. Overview of Tool's Achitecture

A developer can interact with the tool to activate services such as capture the abstract interface model, capturing the Accessibility SIG, access to a description of checkpoints to apply, display the tree view of the Accessibility SIG content, and build an accessible concrete model expressed in HTML, then it is also possible to export the resulting model and the report generated during the construction process.

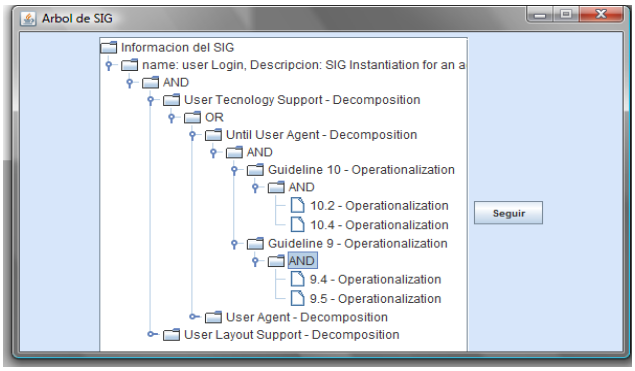

Figure 7. Interface for the Tree of the Accessibility SIG

Figure 7 shows a screenshot of tree structure of the SIG. On this structure, we can recognize the various checkpoints to be implemented grouped by areas. The system is responsible for checking and correcting, if necessary, each of the elements of the abstract interface model. The verification and correction process is guided by checkpoints that are applied to them. The system informs the developer of the missing attributes or attributes in the wrong state. Also it provides the necessary functionality to perform the correction of errors; that is, to add missing elements or labels or modify existing elements. It is up to the developer to make the suggested corrections, or ignore according to circumstances. Some promissory results on the use of the tool have shown that making Accessibility of components visible from early stages allowed better understanding of architectural constraints at the interface design level. However, our case studies are a starting point; so further research is needed to generalize those results.

V. CONCLUSION

Considering the extensibility of our approach, it is important to highlight, that although we have focused on visual disabilities, the proposal can be extended to all kinds of disabilities as the conceptual tools we provided (the UID with *integration points* and SIG *template* for Accessibility) are generic enough to capture Accessibility requirements for all types of impairments.

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 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 [12] 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 [11] for a complete case study and evaluation.

Finally, we will further validate our proposal beyond the example used to illustrate our work. To do so, we are currently analyzing the impact on quality attributes of the resulting system, such as *extensibility*, *reuse*, and *modularity* when using the approach. We are currently carrying out some guided experiments in the area of Web-based systems for academic domains.

ACKNOWLEDGMENTS

This work is partially supported by the UNComa project 04E/072 (Identificación, Evaluación y Uso de Composiciones Software).

REFERENCES

[1] Casteleyn, S., Fiala, Z., Houben, G-J., and van der Sluijs, K. Considering Additional Adaptation Concerns in the Design of Web Applications. AH (2006) doi:10.1007/11768012_28

[2] Centeno, V., Kloos, C., Gaedke, M., and Nussbaumer, M. Web Composition with WCAG in Mind. W4A (2005) doi:10.1145/1061811.1061819

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

[4] Chung, L. and Supakkul, S. Representing FRs and NFRs: A Goal-oriented and Use Case Driven Approach. SERA (2004) doi:10.1007/11668855_3

[5] 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. pp. 303-351. Springer-Verlag, London (2008)

[6] 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. pp. 65-108. Springer-Verlag, London (2008)

[7] 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. pp. 157-191. Springer-Verlag, London (2008)

[8] Larson, J.: Interactive Software: Tools for Building Interactive User Interfaces. Prentice Hall, NJ (1992)

[9] Martín, A., Cechich, A., Gordillo, S., and Rossi, G. A Three-Layered Approach to Model Web Accessibility for Blind Users. LA-WEB (2007) doi:10.1109/LA-WEB.2007.56

[10] 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, pp. 181-205. Information Science Reference, Hershey NY (2008)

[11] Martín, A., Rossi, G., Cechich, A., and Gordillo, S. Engineering Accessible Web Applications. An Aspect-Oriented Approach. WWW Journal (2010) doi 10.1007/s11280-010-0091-3

[12] Offutt, J. Quality Attributtes of Web Software Applications. IEEE Software (2002) doi 10.1002/stvr.425

[13] Plessers, P., Casteleyn, S., Yesilada, Y., De Troyer, O., Stevens, R., Harper, S., and Goble, C. Accessibility: A Web Engineering Approach. WWW (2005) doi:10.1145/1060745.1060799

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

[15] Vilain, P., Schwabe, D., and Sieckenius de Souza, C. A Diagrammatic Tool for Representing User Interaction in UML. UML (2000) doi:10.1007/3-540-40011-7_10

[16] W3C: Web Content Accessibility Guidelines 1.0. (WCAG 1.0). http://www.w3.org/TR/WAI-WEBCONTENT/ (1999). Accessed 05.24.2010

[17] W3C: User Agent Accessibility Guidelines 1.0 (UUAG 1.0). http://www.w3.org/TR/WAI-USERAGENT/ (2002). Accessed 05.24.2010

[18] Yesilada, Y., Harper, S., Goble, G., and Stevens, R. DANTE: Annotation and Transformation of Web Pages for Visually Impaired Users. WWW (2004) doi.acm.org/10.1145/1013367.1013540

[19] 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), 117-128 (2008)