Engineering Accessible Web Applications. An Aspect-Oriented Approach

Author: *Adriana E. Martín*
[1]Department of Exact Sciences, Caleta Olivia, University of Patagonia Austral (UNPA-UACO), Santa Cruz, Argentina
[2]GIISCo Research Group, Departamento de Ciencias de la Computación, Universidad Nacional del Comahue, Buenos Aires 1400, Neuquén, Argentina
e-mail: adrianaelba.martin@gmail.com

Supervisor: *Dr. Alejandra Cechich*
GIISCo Research Group, Departamento de Ciencias de la Computación, Universidad Nacional del Comahue, Buenos Aires 1400, Neuquén, Argentina
e-mail: acechich@uncoma.edu.ar

Co-Supervisor: *Dr. Gustavo Rossi*
LIFIA, Facultad de Informática, Universidad Nacional de La Plata and Conicet, La Plata, Argentina
e-mail: gustavo@sol.info.unlp.edu.ar

Universidad Nacional de La Plata
Facultad de Informática

Thesis submitted in fulfillment of the requirements for the
Doctor degree in Informatics Science
Thesis defended on June 5, 2012

ABSTRACT

Every day more and more users with different abilities and/or temporally or permanent disabilities are accessing the Web, and many of them have difficulties in reaching the desired information. However, the development of this kind of software is complicated for several reasons. Though some of them are technological, the majority are related with the need to compose different and, many times, unrelated design concerns which may be functional as in the case of most of the application's requirements, or non-functional such as Accessibility. Even though, there is a huge number of tools and proposals to help developers assess Accessibility of Web applications, looking from the designer perspective, there is no such a similar situation. In this thesis, we present a novel approach to conceive, design and develop Accessible Web applications using concepts from Aspect-Orientation. In order to accomplish our goal, we provide some modeling techniques that we explicitly developed for handling the non-functional, generic and crosscutting characteristics of Accessibility. Specifically, we have enriched the UID technique with integration points to record Accessibility concerns that will be taken into account when designing the user interface. Then, by instantiating the SIG template with association tables, we work on an abstract interface model with Accessibility softgoals to obtain a concrete and accessible interface model for the Web application being developed. We explain deeply our ideas and point out the advantages of a clear separation of concerns throughout the development life-cycle. Thus, our proposal is based on recognized design techniques, which we embedded in a software tool to facilitate the transfer of the approach to the industry.

1. INTRODUCTION

Since 1999, when the W3C[1] introduced the "Web Content Accessibility Guidelines 1.0" (WCAG 1.0) [18] 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. Thus, although there

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

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. 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. 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. As another example, at the meta-level, Accessibility can be specialized according to meta-features like compliance design and content order concerns. 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, where the Accessibility efforts are focalized.

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 Web Engineering (WE) methods, such as UWE [7], OOHDM [14] or OOWS [6], 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. Secondly, since designing accessible Web applications involves the analysis of different interests, we proposed to use Aspect-Oriented Software Development (AOSD[2]) 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"[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 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. Figure 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, 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 text field elements (for student's ID and password respectively), the presence and positioning of their respective label elements must be tested. So, to propitiate an accessible interaction experience on behalf of the student, these layout requirements must "crosscut" the same software artifact (the Web page) more than once, accordingly to the number of text field elements included in the presentation. Clearly this kind of behavior perfectly fits the "scattering" and "tangling" problems[4], which motivate the main AOSD principles.

[2] 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.

[3] "Crosscutting" 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) are examples of this common functionality that 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).

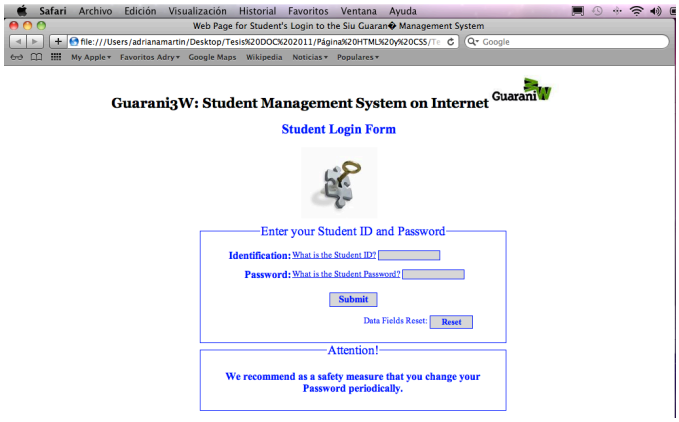

Figure 1: A Student's Login Web page example

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". 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 [14] WE approach.

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

The rest of the work is structured as follows: in Section 2, we offer an overview of our proposal describing the conceptual tools and model we envisage to deal with Accessibility concerns within a WE approach; while in Section 3, we briefly introduce the comparison between related work, focusing on the main contribution of ours. Finally, in Section 4 we conclude, present some further work and some of the contributions related to this work.

2. OUR APPROACH IN A NUTSHELL

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. Our approach, proposes two conceptual tools working together to enable an early capture of the Accessibility concerns. These modeling techniques are the UID [17] with *integration points* and SIG [5] *template* for Accessibility, with which the interaction between OOHDM models links and reinforces Accessibility needs. Following, in Sections 2.1 and 2.2, we introduce these conceptual tools and then, in Section 2.3, we put all the pieces together to give a brief overview of our Aspect-Oriented approach for accessible design.

2.1 Accessibility through UIDs integration points

A User Interaction Diagram (UID) [17] 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 [5] in mind, we introduce the concept of UIDs's *integration points* to model the Accessibility concerns of a user-system interaction. Particularly, we define two kinds of UIDs integration points as follows: (i) User-UID Interaction (U-UI) *integration*

point to propitiate an accessible communication and information exchange between the user and a particular interaction of a UID interaction diagram; and (ii) User-UID Interaction's component (U-UIc) *integration point* 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. For example, Figure 2 shows the resultant UID, corresponding to a 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 propitiate the access for all the students.

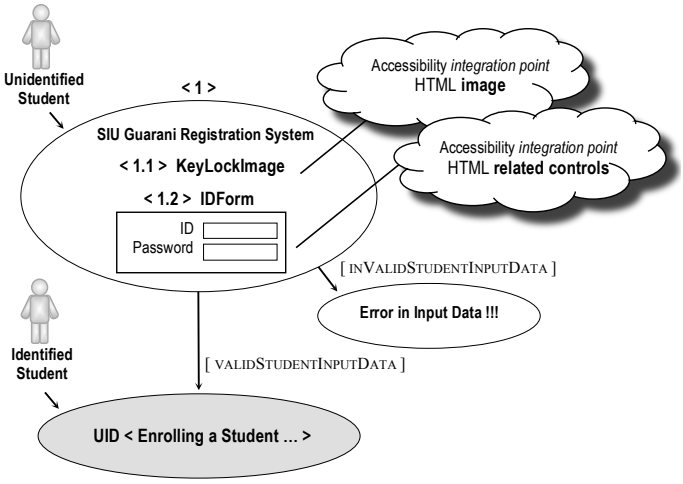

Figure 2: UID with Accessibility integration points: Login a Student given the Student's ID and Password

Basically, the UID with the *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 propitiate that the semantics of the KeyLockImage is correctly transmitted; while the second cloud establishes the <1.2> *integration point* to propitiate an accessible IDForm for user identification.

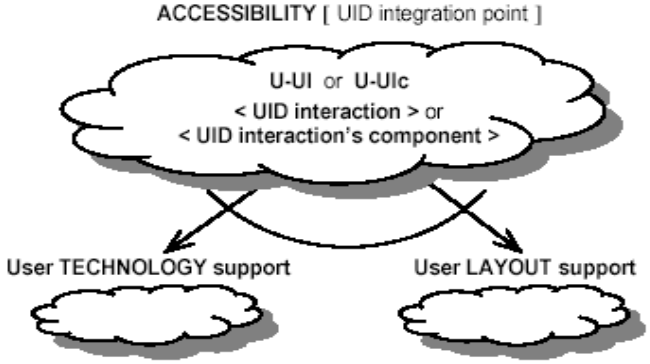

Figure 3: SIG Template for Accessibility

2.2 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 [18] Accessibility requirements. Figure 3 shows our SIG *template* conceptual tool that we introduced taking into consideration proposals from the non-functional requirements [5] and user interface design literature [8] [14]. 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 propitiate 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. For example, returning to Figure 2, we establish the Accessibility softgoal for the interaction's components <1.1> KeyLockImage and <1.2> IDForm to propitiate 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 specifying Accessibility concerns (shown in Figure 3) we work with the W3C-WAI WCAG recommendations (1.0 or 2.0) [18]. To facilitate the instantiation process of the SIG template we establish *association tables* for each of the following five groups of related HTML elements: (i) the HTML *control* elements group; (ii) the HTML *link* and *button* group, (iii) the HTML *text* and *non-text* group, (iv) the HTML *structural* elements group and, (v) the HTML *frame* and *style sheet* elements group, respectively. 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 guidelines [18]. Before proceeding, we must also clarify that we have extended the original Abstract Widget Ontology [14], specifying that an abstract widget can be any of the following: (i) *SimpleActivator*, (ii) *ElementExhibitor*, (iii) *VariableCapture*, (iv) *LogicalStructuring* or (v) *ElementStyling*. We refer the reader to [9] for further details of our *association tables* and their relation with the extension proposed for the Abstract Widget Ontology. Returning to the explanation, the first step to obtain the *association tables* comes from a mapping between abstract interface widgets (ontology concepts from Abstract Widget Ontology) 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. 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.

2.3 An Aspect-Oriented Approach for Accessible Design

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 that require these interface elements. As highlighted in Figure 4(1), we propose a process that 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 Accessibility requirements [18], as shown in Figure 4(2.1) and (2.2) respectively.

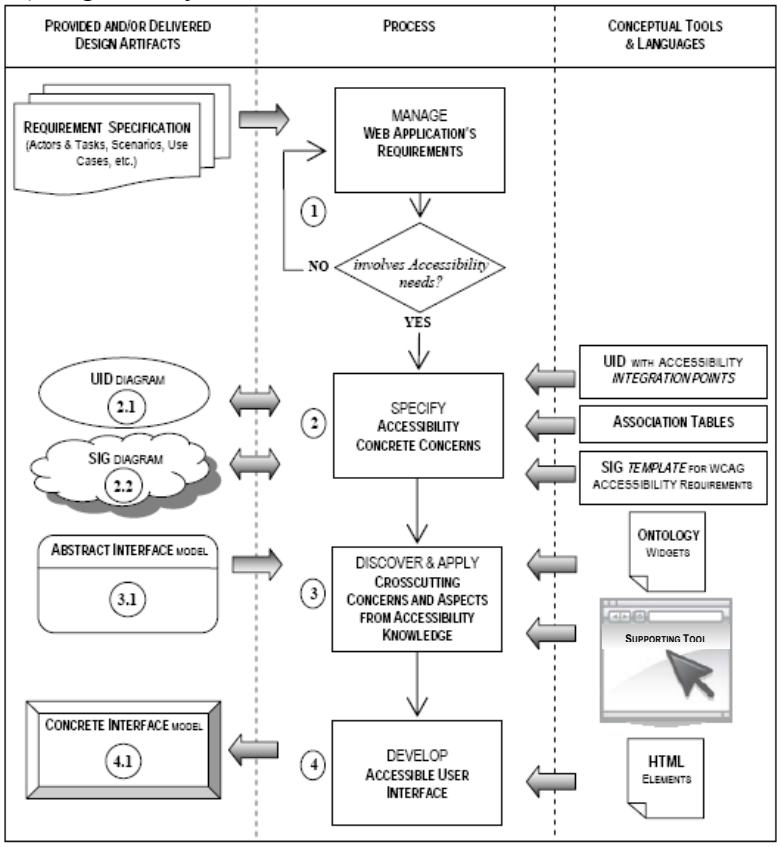

Figure 4: Overview of Our Approach

As we explained previously, we propose these conceptual tools basically to allow the representation of Accessibility requirements while executing a user's task.

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), and then, as shown in Figure 4(4), toward its implementation through the concrete interface model with the desired Accessibility properties (conformance to the WCAG recommendations), as shown in Figure 4(4.1). The purpose here is to find out how WCAG Accessibility requirements "crosscut" interface widgets required for an IDForm. Since, and as we already explain in Section 1, applying the Accessibility concerns to be satisfied at the user interface causes typical crosscutting problems --i.e., "scattering" and "tangling" symptoms, it is clear that Aspect-Orientation is the natural approach to solve these crosscutting problems. 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"[5] to be modeled and instantiated appropriately to avoid "scattering" and "tangling" symptoms. Finally, as highlighted in Figure 4(3), we propose a supporting tool to assist developers to discovering crosscutting concerns and applying aspects from the Accessibility knowledge capture at earlier stages. Basically, the type and the characteristics covered by the tool can be described as those normally provided by a Computer-Aided Software Engineering (CASE) tool.

3. MAIN CONTRIBUTIONS OF OUR WORK

In order to discuss and specify the contributions to the field of accessible design, we have developed an *evaluation framework* to carry out the comparison and evaluation between related work and ours. We highlight that we have studied and applied all related work to the same cases, to which we have applied ours.

[5] An aspect is a module that can localize the implementation of a crosscutting concern; the aspectual decomposition modularizes "scattering" problems --i.e. one concern in many modules, and "tangling problems" --i.e. one module, many concerns.

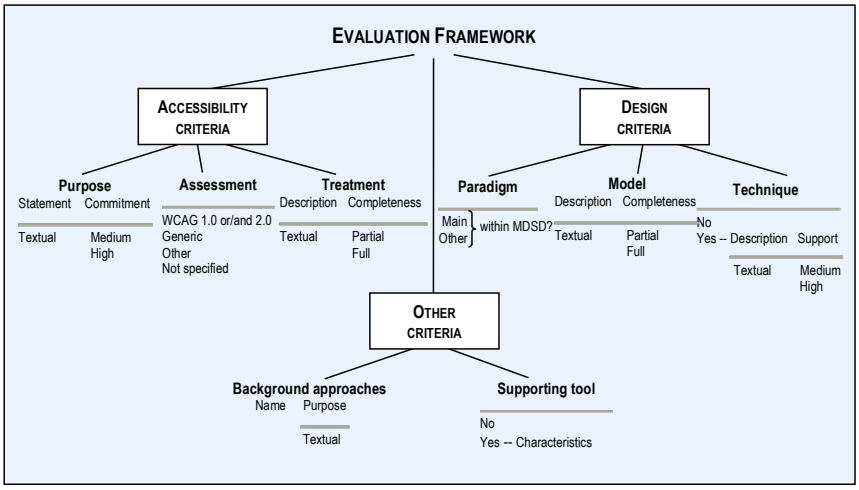

Figure 5: Evaluation Framework

As shown in Figure 5, our *evaluation framework* is divided into the following three main criteria: (i) Accessibility criteria, which assesses the degree of commitment with Accessibility by evaluating three topics: *purpose*, *assessment* and *treatment*, (ii) Design criteria, which evaluates design issues of the approaches under consideration by using three topics: *paradigm*, *model* and *techniques* and, (iii) Other criteria, which considers two additional topics: *background* and *supporting tool.* For brevity reasons, we refer the reader to [9] for further details of our *evaluation framework* and its instantiation for the state-of-the-art to the field of accessible design. Following, in Sections 3.1, we provide a synthesis of the comparison between related work and ours and then, in Section 3.2, we focus on the contributions of our approach.

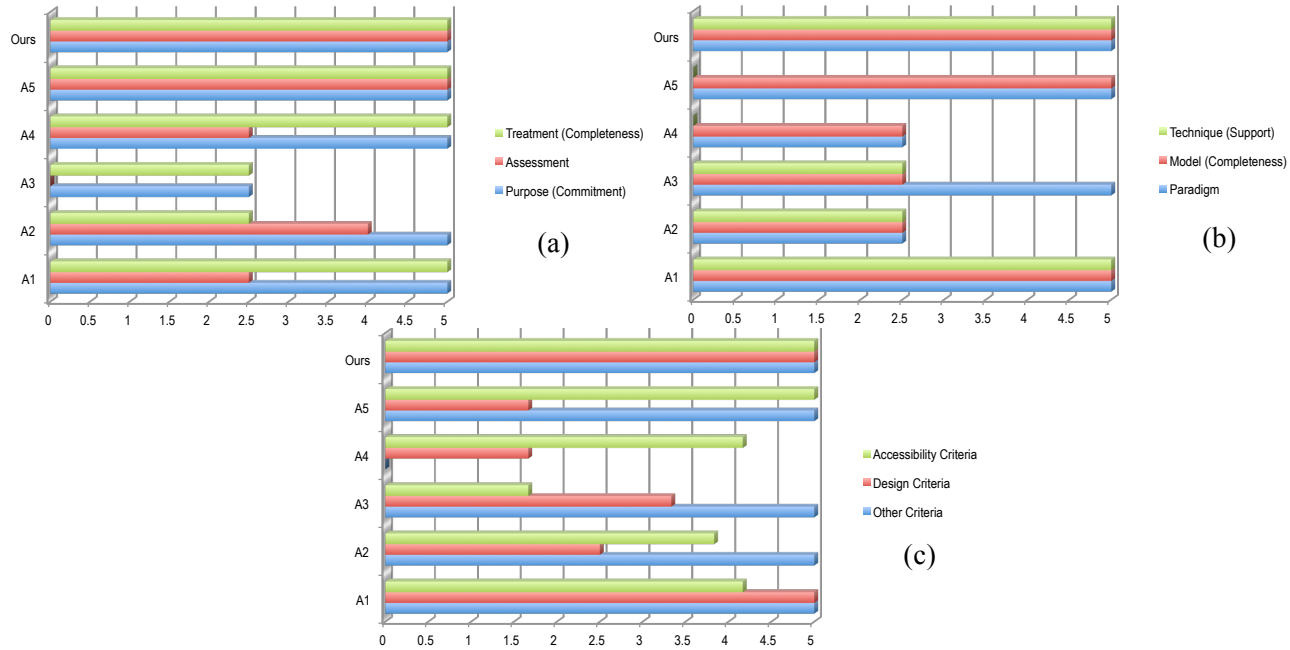

Figure 6: Scoring the Approaches for: (a) Accessibility Criteria, (b) Design Criteria and (c) Average by Approaches/Criteria

3.1 Summary of the Comparison and Evaluation Deliverables

For the purpose of facilitating the evaluation process and understanding the results, we identify each one of the approaches as follow: A1 [13], A2 [4], A3 [1][2][3], A4 [21], A5 [10][11] and Ours [9]. Again, for the sake of brevity, we refer the reader to [9], which provides a comprehensive description and implementation to cases of the approaches that comprise the state-of-the-art. Then, we score the topics related to the Accessibility and Design criteria from 0 to 5, as follows: (i) the scores "high" and "full" match to 5, while the scores "medium" and "partial" match to 2.5; (ii) at

the assessment topic, the option "WCAG 1.0 and 2.0" matches to 5, the option "WCAG 1.0" matches to 4, the option "generic" and "other" match to 2.5, and the option "not specified" matches to 0; and finally (iii) at the paradigm topic, the option "main within MDSD" matches to 5, while the option "other" matches to 2.5. Figures 6.2 and 6.3 show the scoring of the six approaches for the Accessibility and Design criteria, respectively. At this point, we must remember that the scoring process is conducted after a comprehensive study and application to cases of all the approaches; the complete comparison and evaluation. Figures 6(a) and 6(b) show briefly the result of the valuation of the six approaches by Accessibility and Design criteria, respectively. To complete this summary, Figure 6(c) shows the average of scores for the six approaches by Criteria. We should note that for the Other Criteria, we score only the *supporting tool* topic by simply matching the options "yes" and "no" to 5 and 0, respectively.

3.2 Focusing on Ours

As we already said, Ours [9] allows developers to produce accessible interfaces by moving from abstract to concrete architectural views using Aspect-Orientation. This is a main advantage, since allows developers to keep in mind a clear picture of how these architectural views relate each other during the development process, while preserving their own properties: (i) the abstract view ensures clean designs --i.e. free of crosscutting symptoms, which are separated and modeled as aspects for their modularization; while (ii) the concrete view provides the implementation of these designs, but as a consequence of the weaving process that takes place at the code level. Thus, Ours uses Aspect-Orientation to propose a smooth and open transition between models (abstract and concrete views), since this transition allows the independence of the way clean designs will be implemented into accessible code. At this point, we want to state the situation about users having alternatives when browsing; for example, as shown in Figure 1, the Web page offers the student two optional links to look for help for the login process. We highlighted that browsing these pages is optional and therefore, if the student follows these help links, his/her decision will produce a different navigation path. As we said before, we focus on the UI models because, undoubtedly, is at the UI where Accessibility barrier finally show, but notice that this is one of those cases in which navigational issues can affect Accessibility. This is the reason why, to improve the user's experience when browsing to achieve the desired functionality, we have to consider the UI designs for each alternative in the navigation path we have defined as important for the task's functionality. This means that if we provide the user with alternatives in the navigation path, they must be explored and modeled before properly, because they can be relevant to Accessibility and therefore to the success of the user's task. This is an advantage of Ours, because although Ours is focused on UI models, also allows to explore navigational models to avoid unexplored optional browsing that can lead to user interfaces which were not considered initially. Basically, this is possible mainly because of two reasons. In first place, the UID is the conceptual tool used by OOHDM to state transformations between Web application requirements (use case model) and the conceptual, navigational and interface models. As shown in Figure 7, this is the same principle that Ours propitiates between Web applications requirements and accessible UI models. Ours uses two conceptual tools (the UID with *integration points* and SIG *template* for Accessibility), with which the interaction between OOHDM models links and reinforces Accessibility needs.

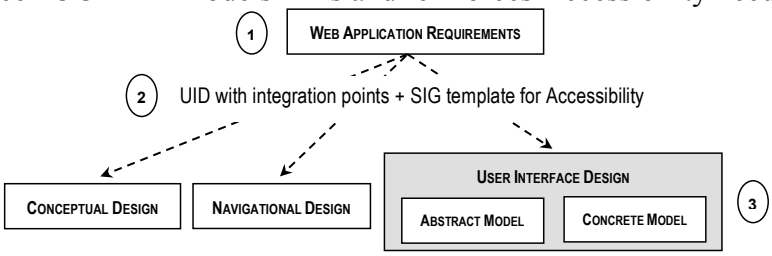

Figure 7: Ours within MDSD paradigm

In second place, since Ours is conceived within the MDSD paradigm, models are related to each other and as a consequence of an iterative and incremental development process. Thus, Ours allows: (i) going back from UI models to navigational models to look for alternatives in the navigation path, (ii) assessing the need and relevance of these alternatives to the functionality under develop, and (iii) going forward from navigational models to UI models to check the Accessibility of the UI related to these alternatives.

4. CONCLUSIONS AND FUTURE WORK: A SYNTHESIS

In this work, we presented a novel WE approach to conceive, design and develop accessible Web applications using Aspect-Oriented concepts, which enabled us to address Accessibility early from requirements and through design to implementation. First, Aspect-Orientation capabilities constitute an important driver to efficiently capturing the orthogonal properties that are typical of the Accessibility's nature. Second, organizing these properties into a model-driven approach gives us better visibility of the components at different levels --i.e. from its conceptualization to its instantiation by particular Accessibility rules. In addition, we provided explicit analysis and design techniques aiming at facilitating the capture of early Accessibility concerns. However, we must take into account that the inclusion of new conceptual tools for treating Accessibility requires an extra effort for developers to get familiar with them. In this sense, we are currently incorporating our ideas into supporting tools to assist developers to design model-driven accessible Web applications. Since our proposal is strongly linked to the model-driven paradigm, we should note 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. 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 WE approach, its application requires a prior knowledge of the WCAG guidelines [18]; (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 without design and documentation as an standard discipline. These two issues demand changes in the development process that must be supported by the organizations. However, our proposal propitiates the reuse of design artifacts, because Accessibility concerns are quite independent from the Web application under development and Accessibility "aspects" could be developed once and be reused in different Web projects.

Finally, we should further validate our proposal and to do so, we are currently following tree different but related paths: (i) generalizing the use of our approach within some of the best known WE approaches to provide accessibility support through Aspect-Orientation techniques (we are already working and have some promising results embedding AO-WAD into UWE and OOWS methods); (ii) improving the supporting tool's functionality to propitiate industry adoption; and (iii) analyzing deeply the impact of applying our proposal on quality attributes of the resulting system, such as reuse, extensibility and modularity, and the developing effort required when using the approach. We are currently carrying out some guided experiments in the area of Web-based systems for academic domains and the petroleum industry.

4.1 Some of the Contributions Related to this Work

- (WWWJ 2010) World Wide Web: Internet and Web Information Systems Journal
  Engineering Accessible Web Applications. An Aspect-Oriented Approach. A. Martín, G. Rossi, A. Cechich, S. Gordillo. World Wide Web: Internet and Web Information Systems Journal ISBN: 978-1-59904-847-5 Vol 13(4); 419-440; DOI: 10.1007/s11280-010-0091-3
- **(**W4A 2011) World Wide Web 8th International Cross-Disciplinary Conference on Web Accessibility

Accessibility at Early Stages: Insights from the Designer Perspective. A. Martín, A. Cechich, G. Rossi. Proceedings of 8th International Cross-Disciplinary Conference on Web Accessibility, Hyderabad, India, 2011; ISBN: 978-1-4503-0476-4; ACM; DOI: 10.1145/1969289.1969302

- (ICSEA 2010) 5th International Conference on Software Engineering Advances
  Supporting an Aspect-Oriented Approach to Web Accessibility Design. A. Martín, R. Mazalú, A. Cechich. In: Proceedings of 5th International Conference on Software Engineering Advances, Nice, France, 2010; ISBN: 978-0-7695-4144-0; IEEE; 20-25; DOI: 10.1109/ICSEA.2010.10

REFERENCES

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

[2] Casteleyn, S., Van Woensel, W., Houben, G-J. A Semantics-based Aspect-Oriented Approach to Adaptation in Web Engineering. In HT (2007) doi.acm.org/10.1145/1286240.1286297

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

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

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

[6] Fons, J., Pelechena, V., Pastor, O., Valderas, P., 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: Modelling and Implementing Web Applications. pp. 65-108. Springer-Verlag, London (2008)

[7] Koch, N., Knapp, A., Zhang, G., Baumeister, H. UML-Based Web Engineering: An Approach Based on Standards. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling 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. Engineering Accessible Web Applications: An Aspect-Oriented Approach. PhD Thesis. http://sedici.unlp.edu.ar/handle/10915/19685 (2012). SeDiCI repository (2012).

[10] Moreno, L., Martinez, P., Ruiz, B. A MDD Approach for Modelling Web Accessibility. IWWOST (2008) doi:10.1.1.163.9478

[11] Moreno, L. AWA: Methodological Framework in the Accessibility Domain for Web Application Development. PhD Thesis. http://www.sigaccess.org/community/theses_repository/phd/lourdes_moreno.php (2010). Accessed April 15th 2010.

[12] Moreno, N., Romero, J., Vallecillo, A. An Overview of Model-Driven Web Engineering and the MDA. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling and Implementing Web Applications. pp. 109-155. Springer-Verlag, London (2008)

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

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

[15] Section 508. Electronic and Information Technology Accessibility Standards http://www.section508.gov/index.cfm?fuseAction=stdsdoc (2000-2010). Accessed 15 April 2010

[16] Stanca Law. Italian Legislation on Accessibility. http://www.pubbliaccesso.it/biblioteca/documentazione/guidelines_study/index.htm (2004). Accessed 25 January 2010.

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

[18] W3C: Web Content Accessibility Guidelines 1.0 and 2.0 (WCAG 1.0 and 2.0). http://www.w3.org/WAI/intro/wcag (1999/2008). Accessed April 15th 2010.

[19] Woods, S. Websites for Visually Impaired Users. Thesis http://wise.vub.ac.be/Downloads/Theses/Woods-thesis.pdf (2006-2007). Accessed April 15th 2009.

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

[21] Zimmermann, G. & 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).