



Universidad Nacional de La Plata

Facultad de Informática

Thesis submitted in fulfillment of the requirements for the degree of
Doctor (Ph.D.) in Informatics Science

Engineering Accessible Web Applications. An Aspect-Oriented Approach

Author: *Mg. Adriana E. Martín*
Supervisor: *Ph.D. Alejandra Cechich*
Co-Supervisor: *Ph.D. Gustavo Rossi*

September 2011

ABSTRACT

Building Accessible Web applications is nowadays a must. Every day more and more users with different abilities and/or temporally or permanent disabilities are accessing the Web, and many of them have special difficulties in reaching the desired information. However, the development of this kind of Web 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 specific application's requirements, or non-functional such as Accessibility itself.

Even though, today 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. 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 modelling Accessibility at early design stages outweigh the needs of a developer to implement that Accessibility.

In this thesis, we present a novel approach to conceive, design and develop Accessible Web applications in an Aspect-Oriented manner. In order to reach our goal, we provide some modeling techniques that we specifically developed for handling the non-functional, generic and crosscutting characteristics of Accessibility as a quality factor concern. Specifically, we have enriched User Interaction Diagrams (UIDs) with *integration points*, which are used to reason and document Accessibility for activity modeling during user interface design. Then by instantiating a Softgoal Interdependency Graph (SIG) *template* with *association tables*, we work on an abstract interface model (composed by ontology widgets) to obtain a concrete and accessible interface model for the Web application being developed. We use a real application example to illustrate our ideas and point out the advantages of a clear separation of concerns throughout the development life-cycle.

RESUMEN

Desarrollar aplicaciones Web Accesibles es en la actualidad una necesidad. Cada día más y más usuarios con capacidades diferentes y/o discapacidades temporales o permanentes acceden a la Web, y muchos de ellos tienen dificultades especiales para obtener la información deseada. Sin embargo, el desarrollo de este tipo de software Web es complicado por varias razones. Si bien algunas de estas son de índole tecnológicas, la mayoría están relacionadas con la necesidad de componer *intereses* de diseño distintos y muchas veces no relacionados entre sí, los cuales a su vez pueden ser funcionales, como lo son la mayoría de los requerimientos específicos de una aplicación, o no-funcionales, como lo es la Accesibilidad.

Aún existiendo hoy en día un gran número de herramientas y propuestas para ayudar a los desarrolladores en la evaluación de la Accesibilidad de las aplicaciones Web, la situación no es la misma al observar desde la perspectiva del diseñador Web. Parece ser que diseñar sitios Web accesibles es más costoso y complejo que crear sitios Web y luego evaluarlos/modificarlos. A pesar de que este sentimiento puede ser ciertamente verdadero, los beneficios al modelar la Accesibilidad en etapas tempranas del diseño superan ampliamente las necesidades de un desarrollador al implementar esa Accesibilidad.

En esta tesis, presentamos un enfoque original para concebir, diseñar y desarrollar aplicaciones Web Accesibles con una modalidad Orientada a Aspectos. Para alcanzar nuestro objetivo, ofrecemos algunas técnicas de modelado que desarrollamos específicamente para manejar las características no-funcionales, genéricas y transversales de la Accesibilidad como un *interés* de factor de calidad. Específicamente, enriquecimos los “User Interaction Diagrams” (UIDs) con *puntos de integración*, los cuales usamos durante el diseño de la interfaz de usuario, para razonar y documentar la Accesibilidad en la actividad de modelado. Luego, instanciando la plantilla del “Softgoal Interdependency Graph” (SIG) con las *tablas de asociación*, trabajamos en el modelo de interfaz abstracta (compuesta por “ontology widgets”) para obtener un modelo de interfaz concreta y accesible de la aplicación Web en desarrollo. Para ilustrar nuestras ideas y señalar las ventajas de una clara separación de *intereses* durante el ciclo de vida de desarrollo, utilizamos un ejemplo de aplicación real.

Index

1.	INTRODUCTION	14
1.1	Context and Motivation	14
1.2	Objectives	18
1.3	Research Context	19
1.4	Structure.....	20
2.	ACCESSIBILITY WITHIN WE APPROACHES	23
2.1	Web Accessibility	23
2.2	Proposals for Developing Accessible Web Applications	27
2.2.1	Providing a Student of his/her Faculty Site	28
2.2.2	Automatic Annotations for Accessibility.....	29
2.2.3	Rules for an Accessible Composition	33
2.2.4	Adaptation to tackle Crosscutting Concerns.....	36
2.2.5	User Needs through Personas	41
2.2.6	Model-Driven Development with AWA.....	43
3.	BACKGROUND OF OUR PROPOSAL	47
3.1	Introducing the Basis	47
3.2	Aspect-Oriented Composition	47
3.2.1	Aspectual Implementations: Advices and Pointcuts	49
3.3	Reference Frameworks and Ontologies	51
3.3.1	Design Decisions within a User Interface Framework	51
3.3.2	An Ontology to share Abstract Interface Vocabulary.....	53
3.4	User Interaction Diagrams.....	55
3.5	Softgoals Interdependency Graphs	57
3.6	Web Content Accessibility Guidelines Documents.....	58

4.	AN APPROACH FOR ENGINEERING ACCESSIBLE WEB APPLICATIONS.....	61
4.1	Our Approach in a Nutshell.....	61
4.2	Identifying Application’s Requirements that Involve Accessibility Needs.....	63
4.3	Specifying Accessibility Concrete Concerns.....	65
4.3.1	Using UUIDs with Integration Points Technique.....	65
4.3.2	Applying the SIG Template.....	67
4.4	Discovering Crosscutting and Applying Aspects.....	70
4.5	Designing with Accessible Interface Widgets.....	72
4.5.1	A Mapping between Ontology Concepts and HTML Elements.....	72
4.5.2	Association between Ontology Concepts-HTML Elements-WCAG Checkpoints.....	75
5.	APPLYING OUR PROPOSAL.....	84
5.1	A Case Study.....	84
5.2	Our Proposal Step-by-Step on the Field.....	86
5.3	A Supporting Tool for Our Proposal.....	104
5.3.1	Architecture’s Overview: Layers and Classes.....	105
5.3.2	Tool’s Resources: XML Schemas and Specifications.....	107
5.3.3	Tool’s User Interfaces.....	117
5.3.4	Some Insights about the Tool.....	120
6.	COMPARING OUR PROPOSAL.....	123
6.1	Comparison Criteria.....	123
6.2	Discussion.....	127
6.3	Focusing on Ours.....	137
6.3.1	Migrating to WCAG 2.0.....	138
7.	CONCLUSIONS AND FUTURE WORK.....	142
7.1	Conclusions.....	142
7.2	Future Work.....	144

7.3	Publications related to this Thesis	145
7.3.1	Journals	145
7.3.2	Book Chapters.....	146
7.3.3	International Conferences	146
7.3.4	National Conferences.....	147
7.4	Other related Publications	148
7.4.1	International Conferences	148
7.4.2	National Conferences.....	149
	APPENDIX I – WCAG 1.0	151
	APPENDIX II – WCAG 2.0	178
	REFERENCES.....	212

1. INTRODUCTION

1.1 Context and Motivation

Since 1999, when the W3C¹-WAI² introduced the “Web Content Accessibility Guidelines 1.0” (WCAG 1.0) [45] as a set of guiding principles, the fact that Accessibility is a main topic in Web design upon which the success of a Web application depends, has become a landmark statement. However, developing accessible Web applications is usually hard for several reasons.

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

¹ The World Wide Web Consortium at <http://www.w3.org/>

² The Web Accessibility Initiative at <http://www.w3.org/WAI/>

Although Accessibility has not yet gained much recognition as a crucial non-functional requirement like security, performance, accuracy and usability; it is a vital attribute for people with disabilities. Moreover, Accessibility is a generic concern that may comprise dozens of specialized concerns and, therefore, many requirements associated with these. For example, at the application-level, Accessibility can be specialized according to the kind of Accessibility support given to the user, where specific requirements related to the user's layout and the user's technology supports are considered. While the former ensures an accessible user's interaction, the user's technology support guarantees browsing regardless of the user's assistive device and further, new requirements related to current and earlier assistive devices characteristics are associated separately --i.e. "user agents" and "until user agents" respectively as the distinction made by the W3C's UAAG 1.0 [48]. The term "user agent" is used by the W3C as a generic description for any software that retrieves and renders Web content for users, such as browsers, mobile phones, screen readers, etc. On the other hand, the term "until user agent" is used by the W3C referring to "user agents" that require developers to provide additional support for Accessibility.

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

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

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

life cycle, we analyzed how mature Model-Driven³ Web Engineering (WE⁴) methods⁵, such as UWE [24], OOHD [36], OOWS [18] or WSDM [13], face this cycle. We realized that all of them comprise several activities to focus on some specific design concerns; however, since OOHD fulfill many of our expectations, we decided to join our modeling approach to this particular WE method. As an example of the rationale of choosing OOHD as our host WE approach, we have to mention the different views provided by OOHD at the user interface (UI) model. This fine-grained treatment allows us to move from abstract interface elements, which are those from the widget ontology [36], to concrete interface elements --e.g. HTML elements, and link both levels of abstraction from a UI design perspective [27] to WCAG checkpoints. Secondly, since designing accessible Web applications involves the analysis of different interests, we proposed to use Aspect-Oriented Software Development (AOSD⁶) design principles to support the construction of accessible user interfaces. The fact that we choose aspect orientation to develop our proposal ensures handling naturally the non-functional, generic and “crosscutting”⁷ characteristics of the Accessibility concern.

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

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

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

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

⁷ “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), exception handling, validation and login managements are all examples of this common functionality that is usually described as “crosscutting concerns” and should be centralized in one location in the code where possible.

As a motivating example and to introduce properly the ideas behind our modeling approach, let us suppose a typical login Web page whose purpose is aiming a student's identification at his/her university system, such as the SIU Guarani student registration system that is used by a number of Argentine universities⁸. Figure 1.1 shows the page for the student's login that provides a user interface composed of HyperText Markup Language (HTML) elements, such as labels and text fields. To help to an accessible interaction experience these HTML elements must fulfill some Accessibility requirements, which crosscut the same software artifact (the Web page for student's login). For example, and as we will see in detail later, at the presentation level an HTML label element is a basic layout Accessibility requirement for many other HTML elements.

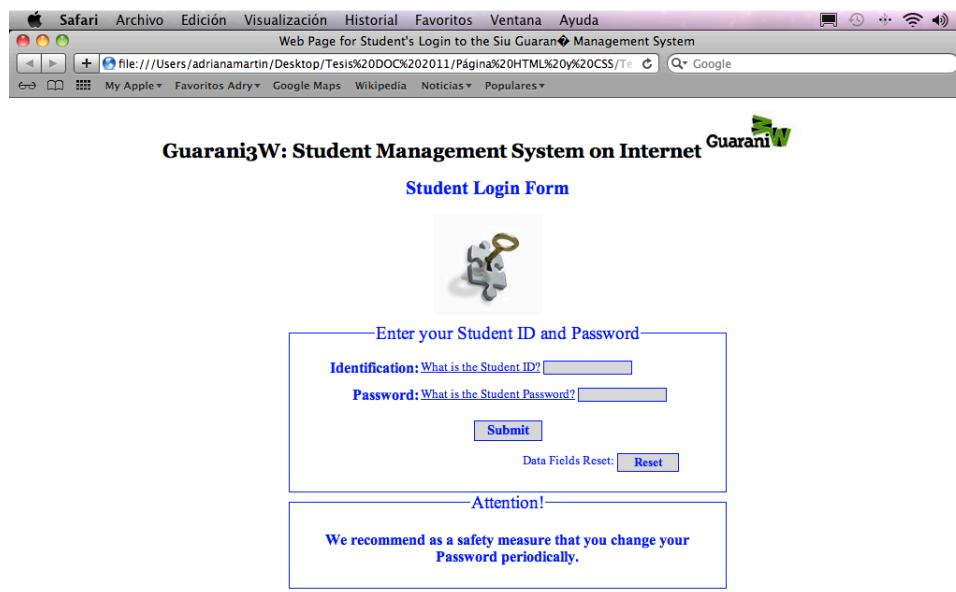


Figure 1.1: A Student's Login Web page example

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

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

element with respect to a text field element; this technological requirement for “until user agents” [48] --i.e. earlier “user agents”, also crosscuts the Web page. Clearly this kind of behavior perfectly fits the “scattering” and “tangling” problems⁹, which motivate the main AOSD principles. Since these two Accessibility requirements (presence and positioning of the label elements), are “scattered” in the Web page with a pair of label-text field HTML elements, the Web page is “tangled” with these Accessibility requirements. It seems natural therefore to address Accessibility using the Aspect-Oriented Software Development (AOSD) approach and, it is not just a coincidence that during this work we refer to Accessibility as a “concern”. Besides the fact that Accessibility has become a basic quality attribute to any Web application and to improve the evolution of the Web in general, the term "concern" from the AOSD perspective describes accurately the Accessibility features related to its nature. By using the AOSD paradigm we can avoid typical problems of “crosscutting” concerns, such as those shown in the previous Web page example. Our proposal applies these concepts by treating Accessibility as a first-class concern in the context of the OOHDM [36] WE approach. Specifically, we propose the early capture of specific Accessibility concerns, which involve user interactions and activities with the application’s interface by introducing some additional extensions to the User Interaction Diagram (UID) [44] technique. As we see in Section 5.3, we also propose a supporting tool to assist our approach.

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

1.2 Objectives

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

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

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

As secondary goals we state:

1. Studying the state-of-art of Accessibility proposals in general, and in particular, focalizing on those proposals for designing Web applications with the Accessibility concern in mind.
2. Studying deeply and applying some relevant related work, selected as a result of the previous goal, to a proposed case study.
3. Defining a process for designing Web applications with Accessibility and providing specific techniques that take advantages of Aspect-Oriented concepts to address Accessibility properly and from early stages of design.
4. Applying our proposal to a case study.
5. Proposing a supporting tool to help developers in applying our proposal.
6. Comparing and discussing the main characteristics of our proposal and the relevant related work selected as a result of previous goals.

1.3 Research Context

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

- **UNComa project 04E/072.** Title: *Identificación, Evaluación y Uso de Composiciones Software.* Period: 2008-2011. Director: Dr. Alejandra Cechich.
- **UNPA-UACO project 21/B107.** Title: *Mejora de Proceso de Selección de Componentes para Sistemas de Información Geográficos.* Period: 2010-2011. Director: Dr. Alejandra Cechich.
- **UNLP project PICT-PAE 2187.** Title: *Desarrollo de Familias de Aplicaciones Web*

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

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

1.4 Structure

The structure of this thesis is organized as follow:

- In Chapter 2, ***Accessibility within WE approaches***, we firstly introduce Web Accessibility, mainly focusing on those features that are relevant for our work. Then, we concentrate on introducing properly some selected related work and applying them to a proposed case study.
- In Chapter 3, ***Background of our Proposal***, we introduce four key topics that we will use throughout the rest of the work, since they are the conceptual basis of our proposal.
- In Chapter 4, ***An Approach for Engineering Accessible Web Applications***, we first provide a general overview of the model we envisage to deal with Accessibility concerns within a Web engineering approach. Then, we conduct a detailed description of the proposed process and techniques for implementing our proposal step-by-step.
- In Chapter 5, ***Applying our Proposal***, we carry out clearly the implementation of our approach following the step-by-step process as we described in Chapter 4. To do so, we propose a complete case study composed of 3 (three) level-deep navigation and 2 (two) optional help anchors. We also introduce a supporting tool that we specially develop to help developers on the design process when applying our proposal.
- In Chapter 6, ***Comparing our Proposal***, we first introduce an evaluation framework that we develop to provide proper comparison criteria for the approaches. Then, we carry out the comparison and develop a discussion about the main characteristics of the related work and our proposal.

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

2. ACCESSIBILITY WITHIN WE APPROACHES

2.1 Web Accessibility

Generally speaking, in the World Wide Web (WWW), where users have the freedom to choose what best meets their expectations, the quality of a user interface (UI) can make the difference between maintaining the Web site competitiveness (or not) within its domain --e.g. e-Business and B2B¹⁰, e-Education (e-Teaching and e-Learning), e-Government, GIS¹¹ (GeoWeb, Web Mapping and Web GIS), etc., and even compromise the Web site survival.

In May 2006 foreword by Molly Holzschlag said [41]:

“...Berners-Lee’s vision has always had to do with the human side of the Web. After all, it’s not machines that use the Web, but people... Accessibility is not about disabilities; rather, it’s about people getting to shared information that the vision of the Web has made manifest...”

Web Accessibility is dedicated to achieving the access to the Web by everyone, regardless of their permanent or temporary disabilities, age-related problems, generational gaps, personal skills and preferences, culture and developed education, etc. While it is true that Web Accessibility emerged initially to help accessing the Web to people with disabilities, currently there is no doubt about the spectrum of benefits that Accessibility provides to the universe of Web users. In this thesis, we have chosen not to provide several definitions of Web Accessibility, as is usually done to describe its

¹⁰ Business to Business (B2B) also known as e-Biz, is the exchange of products, services, or information between businesses rather than between businesses and consumers.

¹¹ A Geographic Information Systems (GIS) is a system of hardware and software used for storage, retrieval, mapping, and analysis of geographic data. GeoWeb consists of location-aware Web technologies usually manifested on the WWW; Web Mapping then refers to those online applications that permit users to view or create maps on a Web platform, usually with limited or no GIS analysis; while Web GIS then refers to GIS that use Web technologies as a method of communication between the elements of a GIS.

scope and contributions (these definitions are all available at the Internet¹²). Instead, we prefer to introduce Table 2.1 that clearly shows how Accessibility can help all users to face accessing the Web at different life situations; after all, we all have different skills and abilities.

Table 2.1: Web Accessibility benefits the entire universe of Web Users

Disability	People with Disability	People “without Disability”
Vision	Blinds	Users who are driving in the dark...
Low vision	Low-vision Users	Users who are using a device with a small display...
Hearing	Deafs	Users who are in forced silence (library) or using music players with headphones...
Low hearing	Low-hearing Users	Users who are in noisy environments...
Motor impaired	Motor impaired Users due to illness or traumatic injuries (permanent or temporary)	Users who are wearing tight clothes, protective clothing, overalls, workware... Users on a moving and/or unstable vehicle --e.g. a train...
Cognitive impaired	Users who are limited in their abilities to process and memorize information, to take decisions, to learn, to perform intellectual tasks.	Users who are tired, fatigued, distracted, worried, sleepy, drunk...
Communicational impaired	Users having difficulties to understand linguistic and textual.	Users who have no knowledge of the language, slogans or symbols...

The World 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 (WCAG¹³), which is considered a benchmark for most of the laws on Information Technology and Communication worldwide. The WCAG has two documents, the WCAG 1.0 [45] and the WCAG 2.0

¹² W3C (2005) definition at <http://www.w3.org/WAI/intro/accessibility.php>; ISO/TS 16071 (2003) definition at http://www.iso.org/iso/catalogue_detail.htm?csnumber=30858; Hull (2004) definition at <http://ausweb.scu.edu.au/aw05/papers/refereed/arora/paper.html>; Fourney and Carter (2006) definition at <http://userlab.usask.ca/papers/IEA06DF-JC.pdf>; etc.

¹³ WCAG overview at <http://www.w3.org/WAI/intro/wcag>

[46], whose stable specifications were released in 1999 and 2008 respectively. Since their longstanding presence in the Accessibility arena, the WCAG 1.0 has provided the basis for the promulgation of other Accessibility standards and legislation in several countries. For example, this is the case for the US Section 508 [38], the UK PAS 78 [34] and the Italian Legislation on Accessibility [40]. Currently, the migration process from WCAG 1.0 to WCAG 2.0 of these standards and legislation is taking place. In Argentina, Web Accessibility is an issue that has been recently included in the State's agenda. The legislation 26.653 called “Guía de Accesibilidad para Sitios Web del Sector Público Nacional¹⁴”, which adheres to WCAG 1.0 document, was approved by Resolution 69/2011 on June 27th 2011. In August 2011, Argentina became a member of the W3C¹⁵. We will return on WCAG and its documents in Section 4.6, and then also in Section 7.3.1 where we will explain how we carry out the migration of our proposal.

Since 1999, when the first W3C Accessibility document was released, a number of tools and approaches have emerged and are available to support Web developers evaluating Accessibility of existing Web applications. However, Accessibility has not yet gained enough recognition as a crucial non-functional requirement such as other quality factors. This situation may be due to several reasons, but probably, it had much to do with the way Accessibility was first introduced to Web developers --i.e. by showing only its side committed with disability. This lack of knowledge within developer's community, prevented them from getting involved with the cause, and as a consequence, the work has been addressed mostly by Accessibility specialists and entities engaged with disability. As we shall see next in Section 2.2, the status is worse from a design perspective, since it is a fact that there are not many efforts considering Accessibility at early stages of the development process.

At this point, we would like to perform some considerations concerning to the relationship between Accessibility and Web development stages. As we already said in Chapter 1, Web Engineering (WE) focuses on stages, which create and exploit domain models, to face the development life cycle of Web applications. Almost every mature

¹⁴ Access to Public Information by Law 26.653 at <http://www.infoleg.gov.ar/infolegInternet/anexos/175000-179999/175694/norma.htm>

¹⁵ Argentina became a member of the W3C at <http://www.puntogov.com/nota.asp?nrc=2641>

WE method proposes the following five stages, each one delivering its respective model: requirements, conceptual, navigation, user interface and implementation. In the best cases, Accessibility is submitted to user interface (UI) codification and implementation stage. In most cases, Accessibility is addressed when the 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 [22].

Finally, when we talk about Web Accessibility, we must specify the target of the Accessibility efforts since to establish the client-server Web relationship, several components are required. This means that Web Accessibility depends on these components working together and improvements in specific components could substantially improve Web Accessibility. Thus, for example, we can evaluate the Accessibility of the following components: (i) User agents, client devices or assistive technologies, such as PCs and notebooks, cell phones, iPods and iPads, screen readers¹⁶, screen magnifiers¹⁷, braille keyboards¹⁸, PDAs, etc., (ii) Web browsers, such as Safari, Mozilla Firefox, Internet Explorer, Opera, etc., (iii) Authoring tools --i.e. software that helps creating Web sites¹⁹, (iv) Web pages --i.e. the content, structure, presentation and layout of Web documents and (v) Web navigation --i.e. how the Web user moves from one Web page to another when traveling through the cyberspace. The W3C-WAI provides valuable standards to improve the Accessibility of these components²⁰ that are

¹⁶ Software for the visually impaired users that reads the contents of a computer screen, converting the text to speech.

¹⁷ A screen magnifier is software that interfaces with a computer's graphical output to present enlarged screen content.

¹⁸ Portable units used to take notes using the Braille system; quite often use chording techniques (key combinations), but some units are designed with a traditional keyboard.

¹⁹ A list of some Authoring tools and their comparison at <http://www.edb.utexas.edu/minliu/multimedia/Compare%20Web%20Authoring%20Tools.pdf>

²⁰ W3C-WAI guidelines and techniques at <http://www.w3.org/WAI/guid-tech.html>

called “Essential Components of Web Accessibility”²¹. As examples of these standards, we already mentioned the WCAG documents [45] [46], which are focused on explaining how to make accessible the Web content component and, the User Agents Accessibility Guidelines (UAAG) [48] document²², which provides guidelines for designing user agents that lower barriers to Web accessibility for people with disabilities. As we are especially interested in developing accessible Web applications, our work focuses its efforts on designing user interfaces (UI) by applying the WCAG recommendations to propitiate better access to content, help navigation and improve the user experience while interacting with the application.

2.2 Proposals for Developing Accessible Web Applications

This section reviews the most relevant proposals that aim to consider the Accessibility concern in at least, some of the stages of the development life-cycle. To provide a more complete description and also to perform a more thorough analysis of these proposals, in Section 2.2.1 we introduce a case study that we use to apply each one of them.



Figure 2.1: A simplified University home page example

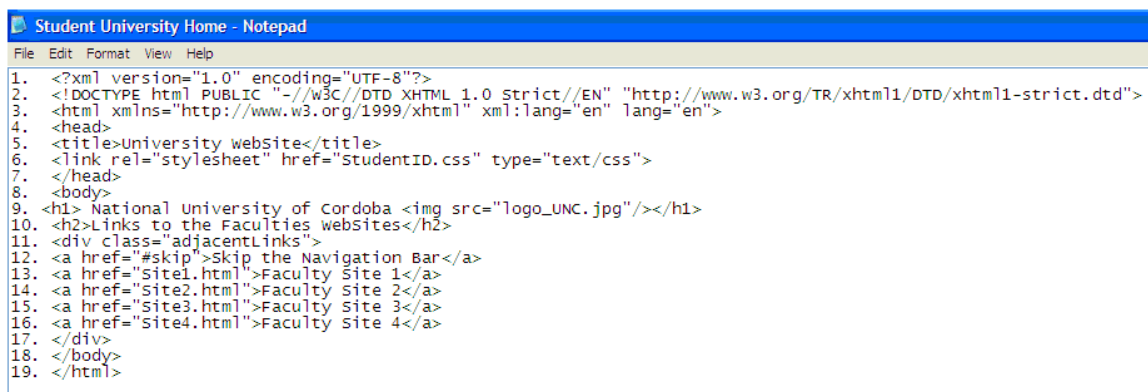
²¹ W3C-WAI: strategies, guidelines, resources to make the Web accessible to people with disabilities at <http://www.w3.org/WAI/intro/components.php>

²² UAAG overview and UAAG 2.0 working draft at <http://www.w3.org/WAI/intro/uaag.php>

2.2.1 Providing a Student of his/her Faculty Site

In this section we present the typical situation faced by a college student when looking for his/her respective Faculty site. Let us assume that the student enters the home page of the University of which depends the desired Faculty and this home page has the appearance illustrated in Figure 2.1.

As we can see in Figure 2.1 the page offers the student a set of related links to the Faculties that make up the University. The name of each Faculty is an anchor the student can use to browse to his/her Faculty site. Since links are navigation mechanisms that create a set of paths a user may take through a site, it is very important to keep a consistent style of presentation for links, as for every interface of components relevant to the interaction interface-functionality. Thus, taking into account Accessibility recommendations for links will allow users to locate and skip navigation mechanisms more easily to find important content. This helps people with learning and reading disabilities but also makes navigation easier for all users. Predictability will increase the likelihood that people will find information at your site, or avoid it when they so desire [45]. Returning to the University home, Figure 2.2 illustrates the corresponding HTML code for this page example.



```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4. <head>
5. <title>University webSite</title>
6. <link rel="stylesheet" href="StudentID.css" type="text/css">
7. </head>
8. <body>
9. <h1> National University of Cordoba </h1>
10. <h2>Links to the Faculties webSites</h2>
11. <div class="adjacentLinks">
12. <a href="#skip">skip the Navigation Bar</a>
13. <a href="site1.html">Faculty Site 1</a>
14. <a href="site2.html">Faculty site 2</a>
15. <a href="site3.html">Faculty site 3</a>
16. <a href="site4.html">Faculty site 4</a>
17. </div>
18. </body>
19. </html>
```

Figure 2.2: The HTML code for the University home page example

As we can see at lines 12, 13, 14, 15 and 16 of Figure 2.2, a set of five HTML *a* elements is defined for a “skip” option and four Faculties, and they are enclosed with an HTML *div* element at lines 11 and 17 of the styling *class* “adjacentLinks”. Following, we use this simple example to discuss the way the five approaches cited at this chapter work for improving more accessible user interface designs.

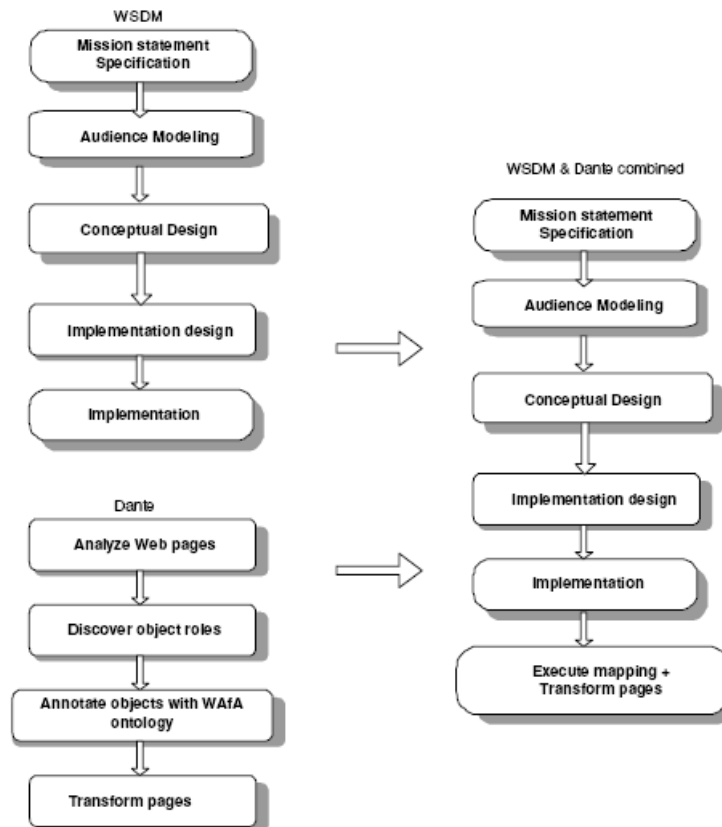


Figure 2.3: The WSDM with Dante from [51]

2.2.2 Automatic Annotations for Accessibility

The main goal in Plessers et al. [35] is to generate annotations for visually impaired users automatically from explicit conceptual knowledge existing during the design process. The approach integrates the Dante [52] annotation process into the Web Site Design Method (WSDM) [13] that allows Web sites and Web applications to be developed in a systematic way. The annotations are generated from explicit conceptual knowledge captured during the design process by means of WSDM's modeling concepts. These WSDM's modeling concepts, used in the different phases, are described using the WSDM OWL²³ ontology. To generate code the authors establish a transformation process that takes the conceptual design models as input and generates a set of annotations as a consequence. The transformation process consists of two annotation steps: authoring and mobility, which resemble the original annotation

²³ OWL Web Ontology Language at <http://www.w3.org/TR/owl-ref/>

process of the Dante approach. The difference is that the authoring annotation in Dante is manual and based on the HTML source code of the Web site. The integration of the Dante [52] annotation process into the Web Site Design Method (WSDM) [13] is graphically illustrated by Figure 2.3 [51].

As we can see in Figure 2.3 the transformation to an accessible design, takes place at the “Execute mapping + Transform pages” step, where a mapping between WSDM and Dante ontologies applies. The WSDM key models where transformation takes place are the WSDM site structure model and the WSDM presentation model, both outputs of the WSDM Implementation Design phase.

that is annotated with concepts from the Dante’s WAFa²⁴ ontology, a relationship between the concepts in the WSDM ontology and the WAFa ontology is established. By using these mapping rules,

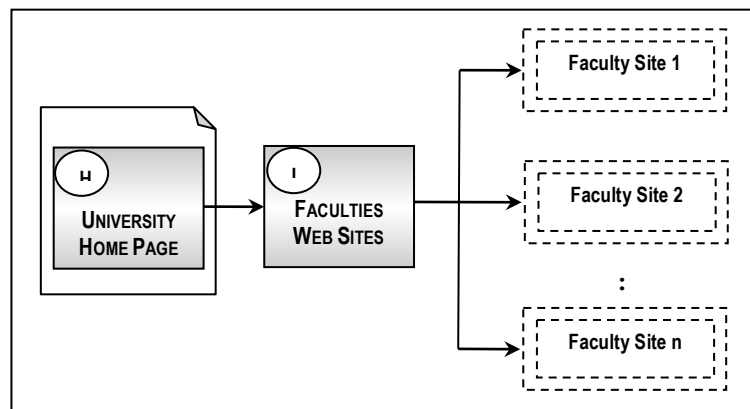


Figure 2.4: Part of the WSDM site structure model for the University home page example

Now, applying this proposal for developing the page example of Section 2.2.1, Figure 2.4 shows part of the WSDM site structure model. As we can see in Figure 2.4, we enrich this model of the University home page with navigational aid links --i.e. the home link and the landmark link components represented by means of the symbols “H” and “L” respectively. From home, the landmark link component offers a list of links that the student may choose when browsing to his/her Faculty Web site.

Figure 2.5 provides the WSDM presentation model as a page template for the University home page example, where the navigational aid links “H” and “L” from

²⁴ Web Authoring for Accessibility (WAFa) at <http://augmented.man.ac.uk/ontologies/wafa.owl>

Figure 2.4 are graphically highlighted in grey. Having these WSDM key models, the transformation process consists of two steps: (1) *Authoring Annotation transformation* which uses the information specified in the WSDM models and the Dante’s WAFa ontology to generate the authoring annotation and, (2) *Mobility Annotation transformation* which uses the output of the previous transformation as well as the WSDM models to extend the authoring annotation with mobility annotation to improve Accessibility.

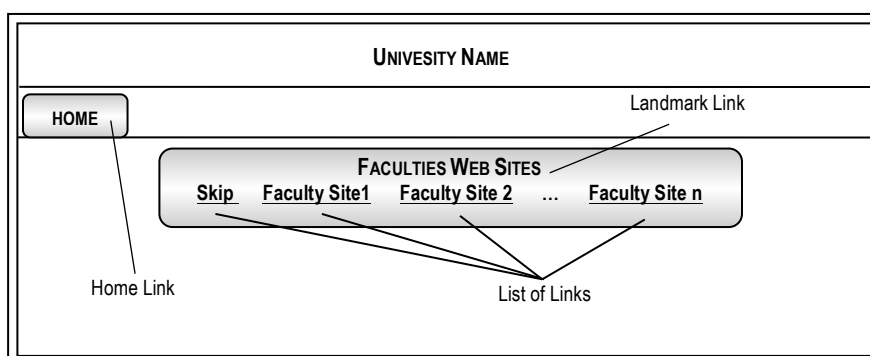


Figure 2.5: The WSDM presentation model for the University home page example

Following, we will explain the transformation process for the University home page example taking into account the WSDM models of Figures 2.4 and 2.5:

(1) *Authoring Annotation transformation*. This process uses the mapping rules between modeling concepts defined in the WSDM ontology and authoring concepts from the WAFa ontology. The “list of text links” at the page example, can be represented by the *List* concept (at WSDM ontology) and by the *NavigationalList* concept (at the WAFa ontology), but this is not a straightforward one-to-one mapping. So, assuming the set **C** as the set of all WSDM modeling concepts and the set **I** as the set of all instances of these modeling concepts, Figure 2.6 shows the corresponding mapping rule for the “list of links” to the Faculties web sites at the University page example. To avoid confusion while applying this rule, the WSDMs concepts are prefixed with “wsdm” and the WAFa concepts with “wafa”. The *NavigationalList* WAFa concept is given in bold, followed by its meaning (in italic), an informal explanation of the mapping rule and finally, a formal definition using first-order predicate logic.

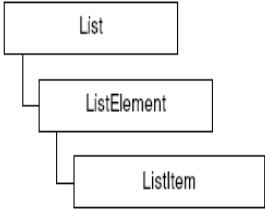
WSDM ONTOLOGY CONCEPT	WAFa ONTOLOGY CONCEPT	MAPPING RULE BETWEEN WSDM AND WAFa ONTOLOGIES
List 	NavigationallList	wafa:NavigationallList : A "list of links". The annotation can be generated for <code>wsdm:List</code> where all list elements have a <code>wsdm:Link</code> defined upon them [35]. $\forall i \in I, \exists y \in I: \text{wsdm:List}(i) \wedge$ $(\forall x \in I: \text{wsdm:hasChild}(i, x) \wedge$ $\text{wsdm:ListItem}(x) \wedge$ $\text{wsdm:hasNavigationReference}(x, y) \wedge$ $\text{wsdm:NavigationReference}(y) \rightarrow$ $\text{wafa:NavigationallList}(i)$

Figure 2.6: Mapping rule for the “list of links” at the University home page example

(2) *Mobility Annotation transformation*. This process re-uses the mapping rules provided by the Dante approach [52], adjusting them to interact with the WSDM models instead of the HTML code of the Web page. Taking the output of the previous transformation as well as the WSDM models, we extend the *NavigationallList* authoring annotation with mobility annotation to improve Accessibility. Figure 2.7 provides the mapping rule [35] for mobility annotation transformation that applies to objects authoring annotated as a *NavigationallList*. All the links in the list are text links corresponding to the Faculties’ names for whose Web sites access are allowed to students. As the mapping rule from Figure 2.7 shows, the *NavigationallList* authoring concept must be annotated with the *DecisionPoint* and *NavigationPoint* mobility concepts, while the *TextLink* authoring concept (required because all the links in the list are text links) must be annotated with *NavigationPoint* and *TravelMemory* mobility concepts. As a consequence, the *NavigationallList*, where all the links in the list are *TextLink*, must be annotated with *DecisionPoint*, *NavigationPoint* and *TravelMemory* mobility concepts.

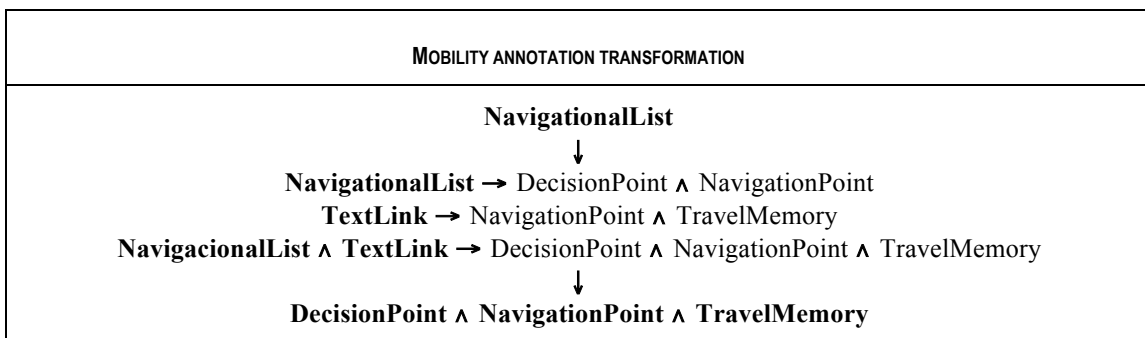


Figure 2.7: Mapping rule for the *NavigationallList* at the University home page example

A *DecisionPoint* is a choice point where alternative paths of browsing are possible; while a *NavigationPoint* provides a possible route and the user exercises some control by choosing to follow or not to follow it; finally, a *TravelMemory* holds information about where the user has been and provides means to get back there. For the particular case of the University home page example, these mobility concepts will offer a student a point from where it is possible to choose a Faculty name, browse to its Web site and also get back from there to the University home page. We must to keep in mind that authoring and mobility concepts are from WafA ontology, so the application of the rule for the Pleasers proposal [35], looks like shows Figure 2.8.

$$\begin{array}{c}
 \forall i \in I: \text{wsdm:String}(i) \vee \\
 (\exists x, y \in C: \\
 \text{wsdm:ObjetcChunkReference}(i) \wedge \text{toProperty}(i, x) \wedge \text{rang}(x,y) \wedge \text{wsdm:String}(y)) \\
 \rightarrow \text{Text}(i) \\
 \\
 \forall i \in I: \text{wafa:NavigationalList}(i) \wedge \\
 (\forall x \in I, \exists y \in I: \\
 \text{wsdm:hasChild}(i,x) \wedge \text{wsdm:ListItem}(x) \wedge \text{wsdm:hasChild}(x,y) \wedge \text{Text}(y)) \\
 \rightarrow \text{wafa:DecisionPoint} \wedge \text{wafa:NavigationPoint} \wedge \text{wafa:TravelMemory}
 \end{array}$$

Figure 2.8: The Pleaser et al. [35] proposal for the University home page example

The botton-half of the rule is a direct translation of the original rule, applied to the objects annotated as a *NavigationalList* where all *wsdm:ListItems* are text elements. The top-half of the rule formally defines a text element [35]. For further details of this proposal, we refer the reader to [35].

2.2.3 Rules for an Accessible Composition

The work by Centeno et al. [9] presents a set of rules that, in a Web composition process, a design tool must follow in order to create accessible Web pages. These rules are formalized with W3C standards like XPath²⁵ and XQuery²⁶ 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

²⁵ W3C XML Path Language at www.w3.org/TR/xpath

²⁶ W3C XML Query Language at www.w3.org/TR/xquery

using the “Web-Composition Service Linking System” (WSLS) [20] as Accessibility enabled authoring tool that makes this task feasible, and focus on how this tool incorporates Accessibility into the process of generating new Web contents. The XPath and XQuery expressions spot HTML nodes and attributes having Accessibility problems. This work proposes to properly manage these spot elements by an authoring tool, so that the author’s attention can be directly brought to these barriers in a semi-automated edition process.

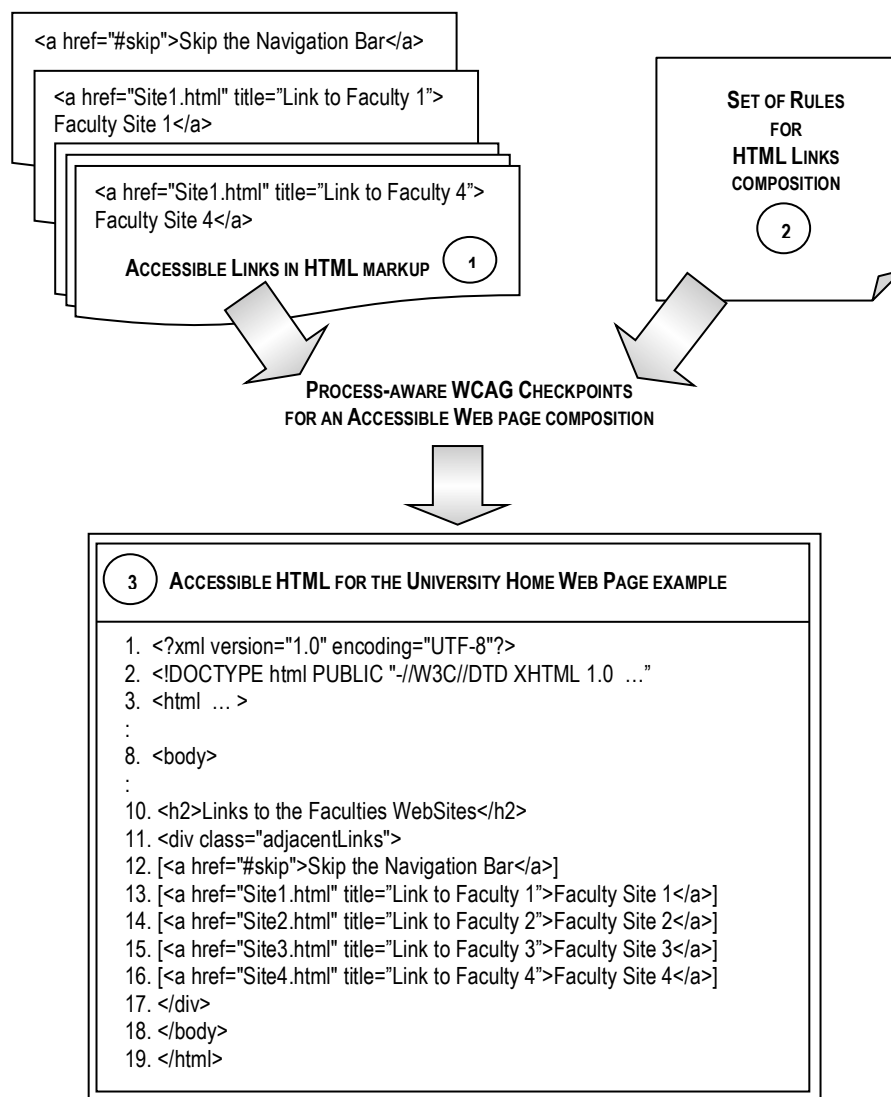


Figure 2.9: The Centeno et al. [9] proposal for the University home page example

The WSLS approach follows the AOSD separation of concerns principle to decompose complexity and control Accessibility over six distinguished categories: Data, Presentation, Navigation, User, Interaction, Process and Communication. The six

elements are mediated by a service control function. Beyond the advantage of the reuse aspect of these components, separation of concerns facilitates also being compliant to the underlying guidelines [9].

Figure 2.9 resumes graphically the proposal at Centeno et al. [9] applied to the page example of Section 2.2.1. As highlighted in Figure 2.9 (1), given $\$S1$ to $\$S5$ compoundable pieces of HTML markup (also called HTML snippets), each one represents an accessible link to a Faculty of the student's University. The composition of these accessible chunks of Web pages, must follow some rules in order to create an accessible "list of links" at the University home page. The proposal provides a set of rules that are focused on formalizing the conditions to be met so that accessible HTML snippets can be safely compound into a page that also results accessible from the WCAG point of view. As shown in Figure 2.9 (2), from the set of rules provided by the proposal, we select for the page example only those rules for HTML links composition. For example, rule 10.5 establishes "provided that all $\$S1$'s and $\$S2$'s links have non-consecutive links (some printable text between links), their composition could have consecutive links without such printable characters if a $\$S2$'s link appears just in front of $\$S1$'s link" [9]. This condition for rule 10.5 ("non-consecutive links") is formalized with a combination of XPath and XPointer as depicted in Figure 2.10 Since this formalization is somewhat difficult for those unfamiliar with XPath and XPointer, the next row of Figure 2.10 summarizes its meaning in simpler terms to facilitate its reading; remember that "a" represents an HTML *a* element that is used to define links.

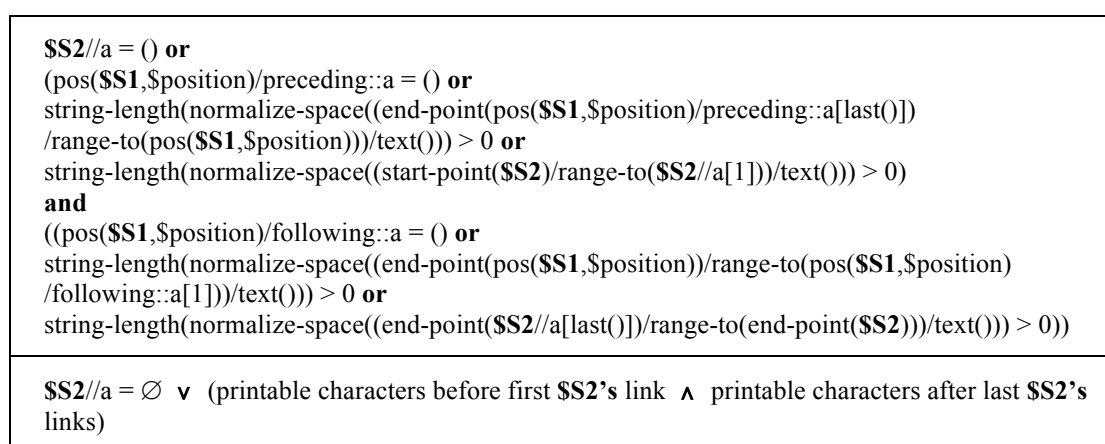


Figure 2.10: XPath + XPointer pre-conditions for avoiding consecutive links without printable non-linkable characters between them [9]

Meanwhile, rule 13.1 establishes “there should be no links sharing both a text and a title but pointing to different targets; provided $SS1$ and $SS2$ have no such ambiguous links there exist a functional dependency such that for every pair of (link’s contents, link’s title) only a single target may be found in both $SS1$ and $SS2$. In that case, we should also make sure that no link in $SS1$ is similarly described in $SS2$ (and pointing to a different target), or vice-versa; if so, an ambiguity would be introduced in the composed result” [9]. This condition for rule 13.1 (“clear links”) is formalized with XPath as depicted in Figure 2.11.

```
(every $a1 in $S1//a satisfies $S2//a[text() = $a1/text() and @title = $a1/@title and @href != $a1/@href] = ()) and  
(every $a2 in $S2//a satisfies $S1//a[text() = $a2/text() and @title = $a2/@title and @href != $a2/@href] = ())
```

Figure 2.11: XPath pre-condition for avoiding ambiguous links [9]

Returning to Figure 2.9, given $SS1$ to $SS5$ HTML snippets corresponding to Faculty links and rules 10.5 and 13.1, a process-aware WCAG checkpoints takes place for Web page composition to deliver an accessible “list of links” at page example. As we can see in Figure 2.9 (3), the “list of links” conform rules 10.5 and 13.1 responding respectively to the statements “non consecutive links” --i.e. printable characters between links where included, and “clear links” --i.e. title’s, target’s and content’s links are properly specified, to avoid students get confuse while browsing his/her University home page example. For further details of this proposal, refer to [9].

2.2.4 Adaptation to tackle Crosscutting Concerns

Casteleyn et al. [6], focus on how to extend an application with new functionality without having to redesign the entire application. The work states that since creating a Web application has become an increasingly complex task, various design issues like device-dependence, privacy, security, Accessibility, localization, personalization, etc. have become extremely relevant to the application performance. To add new functionality, the authors propose to separate additional design concerns and describe them independently. By using a component-based implementation, they show how to extend a Web application to support additional design concerns at the presentation

generation level. Furthermore, they demonstrate how an aspect-oriented approach can support the high-level specification of these (additional) design concerns at a conceptual level.

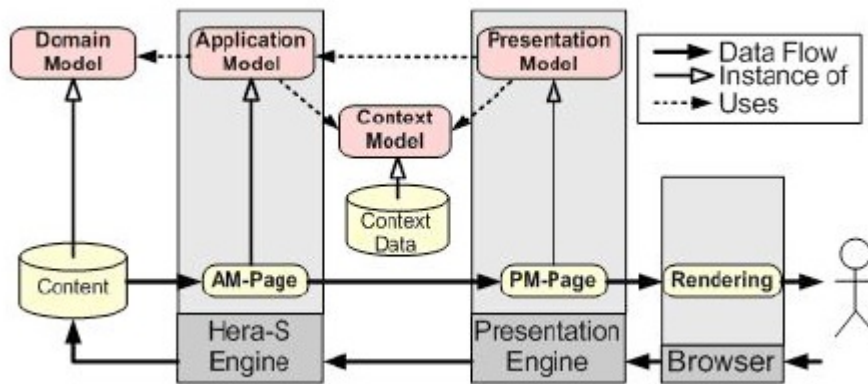


Figure 2.12: Hera-S architecture [8]

The work firstly illustrates how to add adaptation to an existing Hera-based Web application [23], using a component-based implementation. To do so, they apply the Generic Adaptation Components (GAC) approach [16] provided by the AMACONT²⁷ project. Niederhausen et al. introduce further work over this foundation [32] that proposes an aspect-oriented view on adaptation engineering within the AMACONT framework. By separating the specification of adaptation from the underlying application in the form of so-called adaptation aspects, this work proposes to add new or modify existing adaptation concerns on demand. The authors also present an extension of their graphical authoring tool AMACONTBuilder [15]. This extension allows Web engineers to intuitively incorporate adaptation aspects into Web applications. Casteleyn et al. latest implementation [7] [8] proposes a Semantic-based Aspect-oriented adaptation approach materialized in the form of a domain specific language, which the authors called Semantic-based Aspect-oriented Adaptation Language (SEAL)²⁸. It is presented in the context of a Web Information System (WIS) design method, Hera-S, which combines the popular open source Resource Description Framework (RDF)²⁹

²⁷ System Architecture for Multimedia Adaptive WebCONTENT at http://www-mmt.inf.tu-dresden.de/Forschung/Projekte/AMACONT/index_en.xhtml

²⁸ SEAL BNF specification at <http://wise.vub.ac.be/downloads/research/seal/SEALBNF.pdf>

²⁹ W3C RDF/XML syntax specification at <http://www.w3.org/TR/REC-rdf-syntax/>

called Sesame [5] and the rich modeling capabilities of Hera [23], a model-driven approach for engineering Web applications based on semantically structure data. They choose Hera-S because: (i) it naturally builds on Semantic Web data and, (ii) it was conceived with adaptation in mind. An illustrative overview of Hera-S architecture is shown in Figure 2.12 from [8]. Basically, the architecture receives data from the actual source, which conforms to the Domain Model (DM). The Application Model (AM) is instantiated according to the context data provided by the Context Model (CM), resulting in so-called Application Model Pages (AMPs). The authors devised their own custom-made aspect language SEAL to provide adaptation support in the context of Hera-S. By using SEAL's syntax, which is based on BNF notation, they show their adaptation engineering perspective applying pointcuts and advices expressions.

```

:UniversityUnit a ams:NavigationalUnit ;
ams:hasInput [ a ams:Variable ;
ams:varName "U";
ams:varType uncdb:University] ;

ams:hasAttribute [
rdfs:label "UniversityName" ;
ams:hasQuery
"SELECT N1 FROM {$U} rdf:type {uncdb:University};
rdfs:label {N1}"];

ams:hasSetRelationship [
rdfs:label "Faculties" ;
ams:refersTo :FacultyUnit ;
ams:hasQuery
"SELECT F FROM {$U} rdf:type {uncdb:University};
uncdb:unversityFaculty {F}"
].

:FacultyUnit a ams:NavigationalUnit ;
ams:hasInput [ a ams:Variable ;
ams:varName "F";
ams:varType uncdb:Faculty] ;

ams:hasAttribute [
rdfs:label "FacultyName" ;
ams:hasQuery
"SELECT FN FROM {$F} rdf:type {uncdb:Faculty};
rdfs:label {FN}"
].

```

Figure 2.13: The Hera-S AM for the University home page example

To demonstrate the practicality of their proposal, they apply and integrate SEAL in the HydraGen engine³⁰ (an implementation generation tool for Hera-S developed externally by the University of Eindhoven).

Now, applying this proposal for developing our University home page example of Section 2.2.1, a Hera-S Application Model (AM) using Turtle RDF notation³¹ would include the statements shown in Figure 2.13.

An Hera-S Application Model (AM) is specified by means of navigational units (denoted by `ams:NavigationalUnit` and called shorthand: units). A unit can be used to represent a page and it is a primitive that (hierarchically) groups elements (called attributes) that will together be shown to the user. The type of a unit (denoted by `ams:varType`) refers to a domain data and the specification of this type is done by using the namespace-prefix from the Hera-S Domain Model (DM). Our Hera-S AM example bellow, consists of two units, *UniversityUnit* and *FacultyUnit*, which are of the type `uncdb:University` and `uncdb:Faculty` respectively (in this case this namespace-prefix from our Hera-S DM stands for “Universidad Nacional de Córdoba Data Base”). Both units are navigational units of Hera-S AM, each one representing a particularly grouping of information. For example, the *UniversityUnit* contains one attribute (denoted by `ams:hasAttribute`) representing the university’s name and a set of navigational relationships (denoted by `ams:hasSetRelationship`) from *UniversityUnit* to *FacultyUnit*. Note that the `ams:SetRelationship` “refersTo” the *FacultyUnit*, which specifies what exactly to show for every faculty. Since a unit will mostly correspond to (a) specific domain concept(s), one or several content elements are needed in order to instantiate the unit. For example, in the *UniversityUnit* the output of the SeRQL queries (denoted by `ams:hasQuery`) provides a university name and a number of members which will be used respectively to instantiate the *UniversityName* and the *Faculties* of the *UniversityUnit*.

Now, by using the domain specific language SEAL it is possible to apply the Casteleyn et al. proposal [8], to provide aspect-oriented adaptation support in the context of Hera-

³⁰ Hydragen: An implementation of Hera-S at <http://www.wis.win.tue.nl/~ksluijs/material/Singh-Master-Thesis-2007.pdf>

³¹ W3C-Turtle at <http://www.w3.org/TeamSubmission/turtle/>

S for the University home page example of Section 2.2.1. As Figure 2.14 shows, we have instantiated the adaptation requirement to stand for the Accessibility requirements of adjacent links. The *adaptation aspect* is composed of a pointcut and an advice; while pointcut expressions select exactly those elements from the Application Model (AM) where adaptation concerns need to be applied. Advices specify exactly what needs to be done to the element(s) selected in the pointcut [8]. Back to our example of Section 2.2.1, the pointcut in Figure 2.14 selects sets of relationships --i.e. consecutive links, which originate from a(ny) University unit and target a(ny) Faculty unit. The advice is conditioned to users using a “screen-reader” device. As we explained above, in Hera-S the user’s context is captured by the Context Model (CM) and with Hera-S notational conventions, referencing this user’s context is done using a “cm:” -prefix.

Adaptation REQUIREMENT: *for users using a screen-reader avoid consecutive links and clearly identify the target of each one of them.*

Adaptation ASPECT:

POINTCUT: type SetRelationship and from uncdb:University and to uncdb:Faculty

ADVICE: if (cm:userDevice.type = “screen-reader”) {

ADD attribute containing hasLabel “Faculty Name”, hasQuery “SELECT FN FROM {SF} rdf:type {uncdb:Faculty}; rdfs:label {FN}”;

ADD rdf:plainLiteral “[” and “]” surrounding;

};

Figure 2.14: Aspect-oriented adaptation using SEAL for Accessibility requirements of the University home page example

Firstly, the advice adds an AM attribute to the relationships selected in the pointcut showing the faculty name with the label “Faculty Name” and the corresponding query, if the user’s device is a “screen-reader”. Secondly, the advice also uses plain RDF(s)³² to add square brackets surrounding the relationships selected in the pointcut.

Although, this approach is primarily focused on adapting an existing Web application, we include it because the approach proposes to add relevant design concerns, like

³² W3C-RDF:PlainLiteral: A data type for RDF Plain Literals at http://www.w3.org/TR/rdf-plain-literal/#Syntax_for_rdf:PlainLiteral_Literals

Accessibility, in an aspect-oriented manner and, it is representative of other similar works in the adaptation field, like [1] [37]. For further details of this proposal, we refer the reader to [6] [7] [8].

2.2.5 User Needs through Personas

By using existing “best practices of software engineering” for Accessibility purposes, the approach by Zimmermann & Vanderheiden [53] presents a methodology for accessible design and testing to capture functional requirements. The approach defines a new way to use proven tools of software engineering, like use cases, scenarios, test cases, guidelines and checkpoints, for Accessibility purposes; and to relate them to each other, thus facilitating automation as much as possible. The resultant methodology or process model for accessible design and testing consist of: (i) capturing Accessibility requirements in a way that makes them tangible and comprehensible, through use cases and the technique of user profiling “personas” [53], (ii) making Accessibility requirements concrete through scenarios and guidelines for accessible design, (iii) manual and automatic testing based on test cases and Accessibility checkpoints that are derived from guidelines, and (iv) complementary user testing and expert reviews, thus evaluating intermediate and end results, and continuously improving the overall process model.

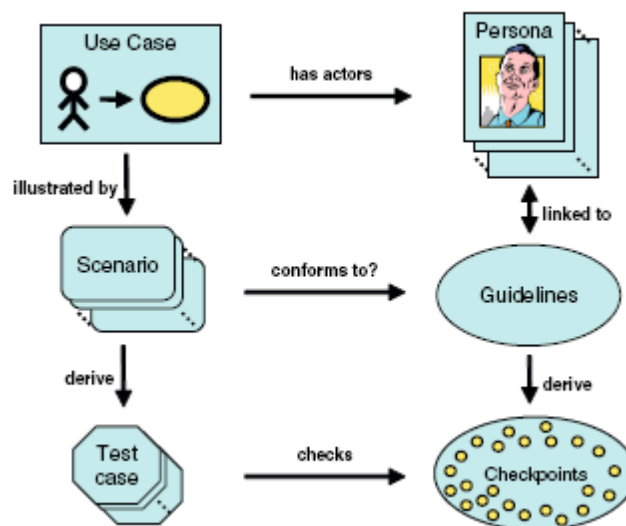


Figure 2.15: Components of the integrated approach and their relationships [53]

In this way for design projects that are employing a use case driven methodology, this approach allows to incorporate accessible design into the existing processes rather than having to add Accessibility as a new process [53]. Figure 2.15 from [53] shows how basic design tools as use cases, scenarios and test cases are linked to personas, guidelines and checkpoints respectively for Accessibility purpose.

Figure 2.16 shows the process model for accessible design and testing by Zimmermann & Vanderheiden [53] applied to our University home page example of Section 2.2.1 and using WCAG 1.0 Accessibility guidelines.

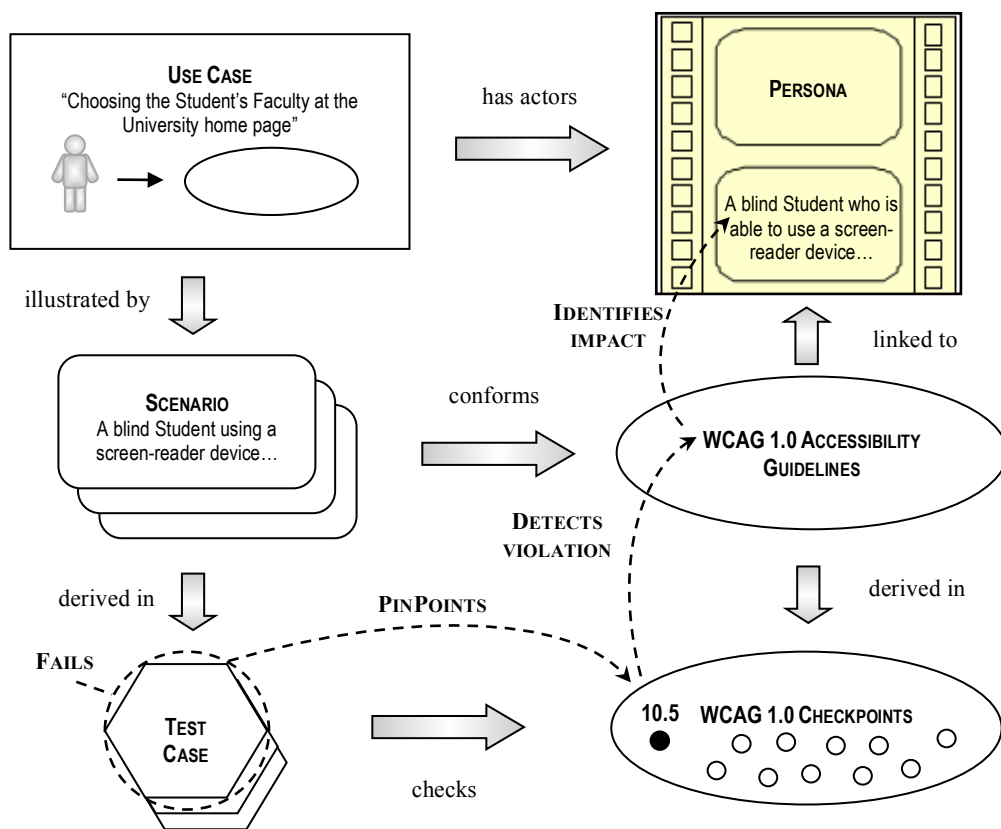


Figure 2.16: The Zimmermann & Vanderheiden [53] proposal for the University home page example

Figure 2.16 shows a situation where a test case is failing because an Accessibility requirement for adjacent links is not met. In this case, the proposed model makes it possible to pinpoint to a particular checkpoint that is causing the failure (10.5 checkpoint), and trace it back to a particular guideline that is violated (guideline 10 from WCAG 1.0). This allows identifying a particular persona (a blind Student) who

despite being able to use a screen-reader will not be able to access the application because of the Accessibility barrier identified by the test case failure. The model presented here is not only useful for fixing the Accessibility problems, but also provides a context to the developers for understanding the consequences of failure [53]. For further details of this proposal, we refer the reader to [53].

2.2.6 Model-Driven Development with AWA

Accessibility for Web Applications (AWA) [29] [30] offers a domain specific methodological framework for the development of accessible Web applications. The AWA framework provides: (i) a specific Accessibility process (which can be adopted by other processes), indicating activities, artifacts and their sequence in the different phases of integrating Accessibility criteria, and (ii) the support for modeling and using techniques provided by Web Engineering (WE) methods as well as Model-Driven Development (MDD), the focus of this work.

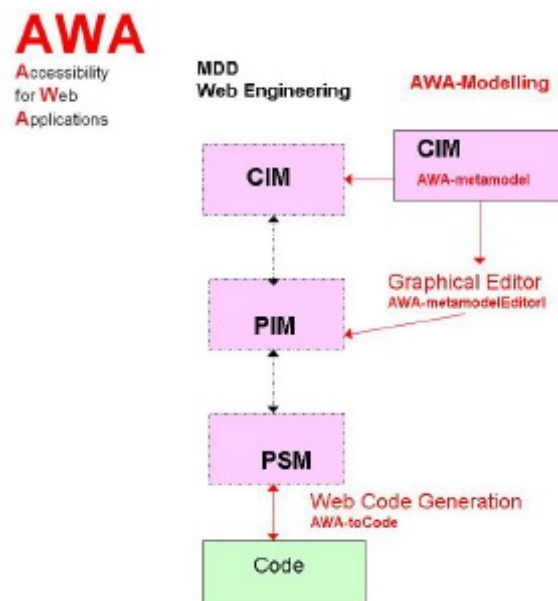


Figure 2.17: AWA for MDA development process [29]

As shown in Figure 2.17, the strategy in AWA consists of providing 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. The authors provide an AWA-

toCode resource and the strategy is based on a transformation Model-to-Text (M2T) to generate code from PSMs. In this work, they also announced that they have developed a CASE support for metamodeling, using the Ecore plugin from the Eclipse Modeling Framework (EMF)³³ [29].

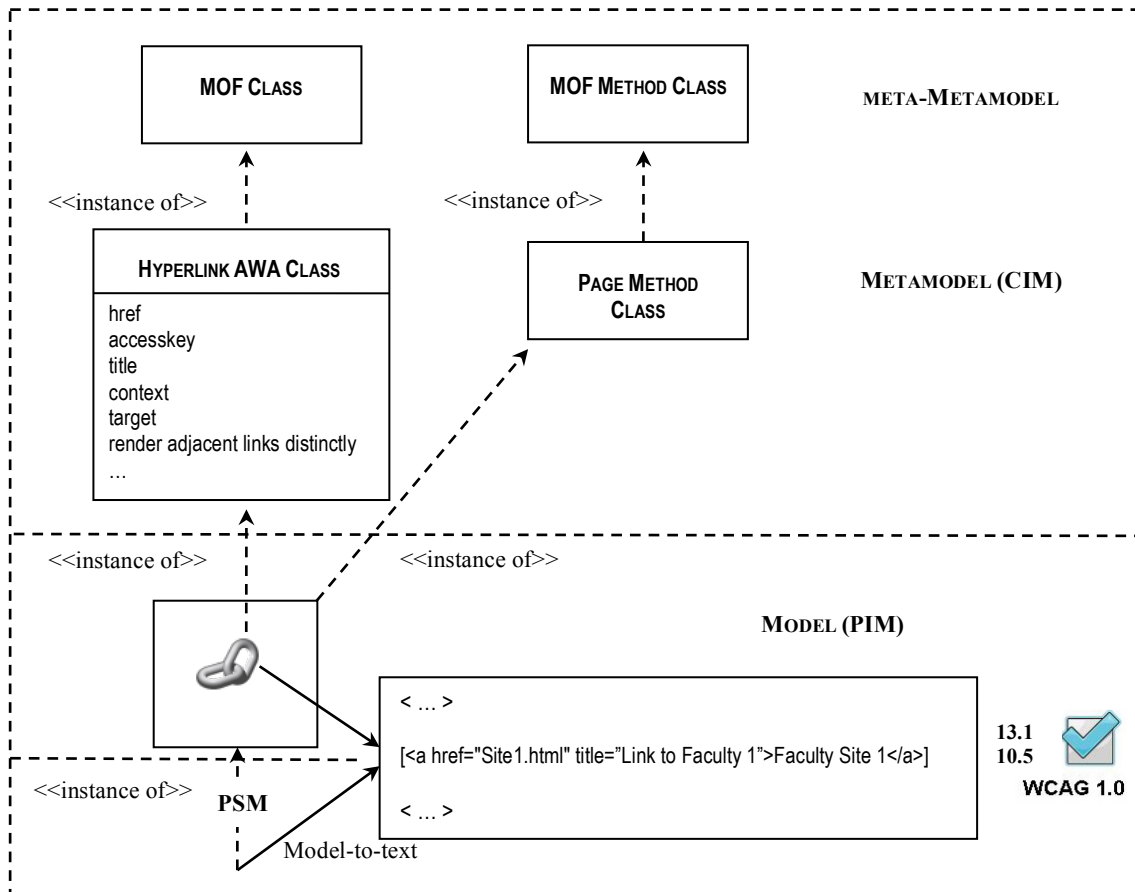


Figure 2.18: The Moreno et al. [29] proposal for the University home page

Figure 2.18 shows AWA for Model Driven Architecture (MDA)³⁴ applied to the Hyperlink concept required by our University home page example of Section 2.2.1. Here several constructors have been defined in the MetaObject Facility (MOF)³⁵ to support the abstraction of Web Accessibility concepts. The diagram develops the concept of hyperlink that includes required attributes to enable the hyperlink to meet the

³³ EMF overview at <http://help.eclipse.org/indigo/index.jsp?topic=/org.eclipse.emf.doc/references>

³⁴ OMG-MDA overview at <http://www.omg.org/mda/>

³⁵ OMG-MOF specification at <http://www.omg.org/mof/>

WCAG standard, such as the title attribute. This attribute contributes to satisfy the 13.1 checkpoint of WCAG 1.0 that establishes “Clearly identify the target of each link”. To continue with the example of Section 2.2.1, Moreno et al. [29] do not consider the 10.5 checkpoint of WCAG 1.0 as a property for the link/hyperlink concept. Although, notice that as we have done at the hyperlink AWA class in Figure 2.10, it is possible to include the “render adjacent links distinctly” attribute, to enable meeting this Accessibility requirement, if the presence of adjacent links makes it necessary.

A graphic element representing a hyperlink (MOF meta-object) has been defined in the AWA-Editor, and may be included in the PIM models, which contain knowledge provided by the AWA-Metamodel necessary for the Web code generation in the final phase [29]. For further details of this proposal, we refer the reader to [29] [30].

In this Chapter we presented Accessibility in the context of some WE approaches. We reviewed and applied in a case study five different proposals [35] [9] [6] [53] [30] that consider this quality factor in the development process of Web applications.

After introducing background (Chapter 3) and our proposal (Chapter 4), we will apply it (Chapter 5) and we will come back to the approaches summarized here to compare them to our proposal (Chapter 6).

2. ACCESSIBILITY WITHIN WE APPROACHES

2.1 Web Accessibility

Generally speaking, in the World Wide Web (WWW), where users have the freedom to choose what best meets their expectations, the quality of a user interface (UI) can make the difference between maintaining the Web site competitiveness (or not) within its domain --e.g. e-Business and B2B¹⁰, e-Education (e-Teaching and e-Learning), e-Government, GIS¹¹ (GeoWeb, Web Mapping and Web GIS), etc., and even compromise the Web site survival.

In May 2006 foreword by Molly Holzschlag said [41]:

“...Berners-Lee’s vision has always had to do with the human side of the Web. After all, it’s not machines that use the Web, but people... Accessibility is not about disabilities; rather, it’s about people getting to shared information that the vision of the Web has made manifest...”

Web Accessibility is dedicated to achieving the access to the Web by everyone, regardless of their permanent or temporary disabilities, age-related problems, generational gaps, personal skills and preferences, culture and developed education, etc. While it is true that Web Accessibility emerged initially to help accessing the Web to people with disabilities, currently there is no doubt about the spectrum of benefits that Accessibility provides to the universe of Web users. In this thesis, we have chosen not to provide several definitions of Web Accessibility, as is usually done to describe its

¹⁰ Business to Business (B2B) also known as e-Biz, is the exchange of products, services, or information between businesses rather than between businesses and consumers.

¹¹ A Geographic Information Systems (GIS) is a system of hardware and software used for storage, retrieval, mapping, and analysis of geographic data. GeoWeb consists of location-aware Web technologies usually manifested on the WWW; Web Mapping then refers to those online applications that permit users to view or create maps on a Web platform, usually with limited or no GIS analysis; while Web GIS then refers to GIS that use Web technologies as a method of communication between the elements of a GIS.

scope and contributions (these definitions are all available at the Internet¹²). Instead, we prefer to introduce Table 2.1 that clearly shows how Accessibility can help all users to face accessing the Web at different life situations; after all, we all have different skills and abilities.

Table 2.1: Web Accessibility benefits the entire universe of Web Users

Disability	People with Disability	People “without Disability”
Vision	Blinds	Users who are driving in the dark...
Low vision	Low-vision Users	Users who are using a device with a small display...
Hearing	Deafs	Users who are in forced silence (library) or using music players with headphones...
Low hearing	Low-hearing Users	Users who are in noisy environments...
Motor impaired	Motor impaired Users due to illness or traumatic injuries (permanent or temporary)	Users who are wearing tight clothes, protective clothing, overalls, workware... Users on a moving and/or unstable vehicle --e.g. a train...
Cognitive impaired	Users who are limited in their abilities to process and memorize information, to take decisions, to learn, to perform intellectual tasks.	Users who are tired, fatigued, distracted, worried, sleepy, drunk...
Communicational impaired	Users having difficulties to understand linguistic and textual.	Users who have no knowledge of the language, slogans or symbols...

The World 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 (WCAG¹³), which is considered a benchmark for most of the laws on Information Technology and Communication worldwide. The WCAG has two documents, the WCAG 1.0 [45] and the WCAG 2.0

¹² W3C (2005) definition at <http://www.w3.org/WAI/intro/accessibility.php>; ISO/TS 16071 (2003) definition at http://www.iso.org/iso/catalogue_detail.htm?csnumber=30858; Hull (2004) definition at <http://ausweb.scu.edu.au/aw05/papers/refereed/arora/paper.html>; Fourney and Carter (2006) definition at <http://userlab.usask.ca/papers/IEA06DF-JC.pdf>; etc.

¹³ WCAG overview at <http://www.w3.org/WAI/intro/wcag>

[46], whose stable specifications were released in 1999 and 2008 respectively. Since their longstanding presence in the Accessibility arena, the WCAG 1.0 has provided the basis for the promulgation of other Accessibility standards and legislation in several countries. For example, this is the case for the US Section 508 [38], the UK PAS 78 [34] and the Italian Legislation on Accessibility [40]. Currently, the migration process from WCAG 1.0 to WCAG 2.0 of these standards and legislation is taking place. In Argentina, Web Accessibility is an issue that has been recently included in the State's agenda. The legislation 26.653 called “Guía de Accesibilidad para Sitios Web del Sector Público Nacional¹⁴”, which adheres to WCAG 1.0 document, was approved by Resolution 69/2011 on June 27th 2011. In August 2011, Argentina became a member of the W3C¹⁵. We will return on WCAG and its documents in Section 4.6, and then also in Section 7.3.1 where we will explain how we carry out the migration of our proposal.

Since 1999, when the first W3C Accessibility document was released, a number of tools and approaches have emerged and are available to support Web developers evaluating Accessibility of existing Web applications. However, Accessibility has not yet gained enough recognition as a crucial non-functional requirement such as other quality factors. This situation may be due to several reasons, but probably, it had much to do with the way Accessibility was first introduced to Web developers --i.e. by showing only its side committed with disability. This lack of knowledge within developer's community, prevented them from getting involved with the cause, and as a consequence, the work has been addressed mostly by Accessibility specialists and entities engaged with disability. As we shall see next in Section 2.2, the status is worse from a design perspective, since it is a fact that there are not many efforts considering Accessibility at early stages of the development process.

At this point, we would like to perform some considerations concerning to the relationship between Accessibility and Web development stages. As we already said in Chapter 1, Web Engineering (WE) focuses on stages, which create and exploit domain models, to face the development life cycle of Web applications. Almost every mature

¹⁴ Access to Public Information by Law 26.653 at <http://www.infoleg.gov.ar/infolegInternet/anexos/175000-179999/175694/norma.htm>

¹⁵ Argentina became a member of the W3C at <http://www.puntogov.com/nota.asp?nrc=2641>

WE method proposes the following five stages, each one delivering its respective model: requirements, conceptual, navigation, user interface and implementation. In the best cases, Accessibility is submitted to user interface (UI) codification and implementation stage. In most cases, Accessibility is addressed when the 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 [22].

Finally, when we talk about Web Accessibility, we must specify the target of the Accessibility efforts since to establish the client-server Web relationship, several components are required. This means that Web Accessibility depends on these components working together and improvements in specific components could substantially improve Web Accessibility. Thus, for example, we can evaluate the Accessibility of the following components: (i) User agents, client devices or assistive technologies, such as PCs and notebooks, cell phones, iPods and iPads, screen readers¹⁶, screen magnifiers¹⁷, braille keyboards¹⁸, PDAs, etc., (ii) Web browsers, such as Safari, Mozilla Firefox, Internet Explorer, Opera, etc., (iii) Authoring tools --i.e. software that helps creating Web sites¹⁹, (iv) Web pages --i.e. the content, structure, presentation and layout of Web documents and (v) Web navigation --i.e. how the Web user moves from one Web page to another when traveling through the cyberspace. The W3C-WAI provides valuable standards to improve the Accessibility of these components²⁰ that are

¹⁶ Software for the visually impaired users that reads the contents of a computer screen, converting the text to speech.

¹⁷ A screen magnifier is software that interfaces with a computer's graphical output to present enlarged screen content.

¹⁸ Portable units used to take notes using the Braille system; quite often use chording techniques (key combinations), but some units are designed with a traditional keyboard.

¹⁹ A list of some Authoring tools and their comparison at <http://www.edb.utexas.edu/minliu/multimedia/Compare%20Web%20Authoring%20Tools.pdf>

²⁰ W3C-WAI guidelines and techniques at <http://www.w3.org/WAI/guid-tech.html>

called “Essential Components of Web Accessibility”²¹. As examples of these standards, we already mentioned the WCAG documents [45] [46], which are focused on explaining how to make accessible the Web content component and, the User Agents Accessibility Guidelines (UAAG) [48] document²², which provides guidelines for designing user agents that lower barriers to Web accessibility for people with disabilities. As we are especially interested in developing accessible Web applications, our work focuses its efforts on designing user interfaces (UI) by applying the WCAG recommendations to propitiate better access to content, help navigation and improve the user experience while interacting with the application.

2.2 Proposals for Developing Accessible Web Applications

This section reviews the most relevant proposals that aim to consider the Accessibility concern in at least, some of the stages of the development life-cycle. To provide a more complete description and also to perform a more thorough analysis of these proposals, in Section 2.2.1 we introduce a case study that we use to apply each one of them.



Figure 2.1: A simplified University home page example

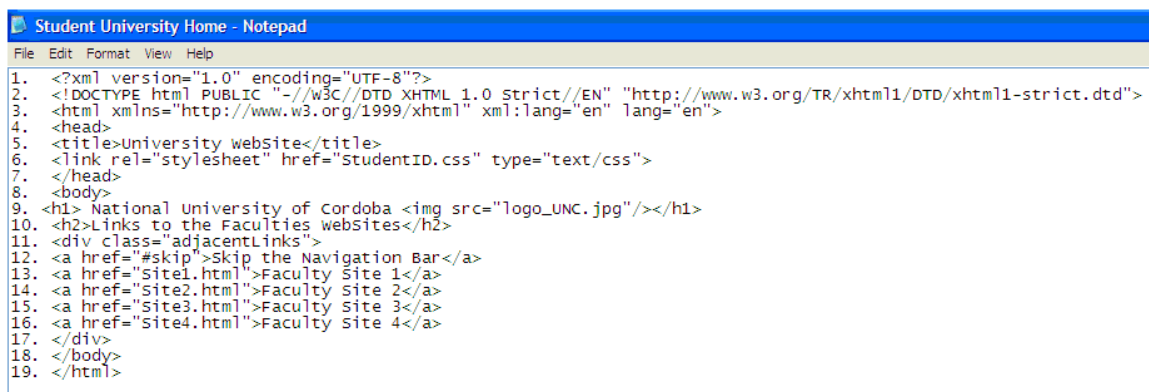
²¹ W3C-WAI: strategies, guidelines, resources to make the Web accessible to people with disabilities at <http://www.w3.org/WAI/intro/components.php>

²² UAAG overview and UAAG 2.0 working draft at <http://www.w3.org/WAI/intro/uaag.php>

2.2.1 Providing a Student of his/her Faculty Site

In this section we present the typical situation faced by a college student when looking for his/her respective Faculty site. Let us assume that the student enters the home page of the University of which depends the desired Faculty and this home page has the appearance illustrated in Figure 2.1.

As we can see in Figure 2.1 the page offers the student a set of related links to the Faculties that make up the University. The name of each Faculty is an anchor the student can use to browse to his/her Faculty site. Since links are navigation mechanisms that create a set of paths a user may take through a site, it is very important to keep a consistent style of presentation for links, as for every interface of components relevant to the interaction interface-functionality. Thus, taking into account Accessibility recommendations for links will allow users to locate and skip navigation mechanisms more easily to find important content. This helps people with learning and reading disabilities but also makes navigation easier for all users. Predictability will increase the likelihood that people will find information at your site, or avoid it when they so desire [45]. Returning to the University home, Figure 2.2 illustrates the corresponding HTML code for this page example.



```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4. <head>
5. <title>University webSite</title>
6. <link rel="stylesheet" href="StudentID.css" type="text/css">
7. </head>
8. <body>
9. <h1> National University of Cordoba </h1>
10. <h2>Links to the Faculties webSites</h2>
11. <div class="adjacentLinks">
12. <a href="#skip">skip the Navigation Bar</a>
13. <a href="site1.html">Faculty Site 1</a>
14. <a href="site2.html">Faculty site 2</a>
15. <a href="site3.html">Faculty site 3</a>
16. <a href="site4.html">Faculty site 4</a>
17. </div>
18. </body>
19. </html>
```

Figure 2.2: The HTML code for the University home page example

As we can see at lines 12, 13, 14, 15 and 16 of Figure 2.2, a set of five HTML *a* elements is defined for a “skip” option and four Faculties, and they are enclosed with an HTML *div* element at lines 11 and 17 of the styling *class* “adjacentLinks”. Following, we use this simple example to discuss the way the five approaches cited at this chapter work for improving more accessible user interface designs.

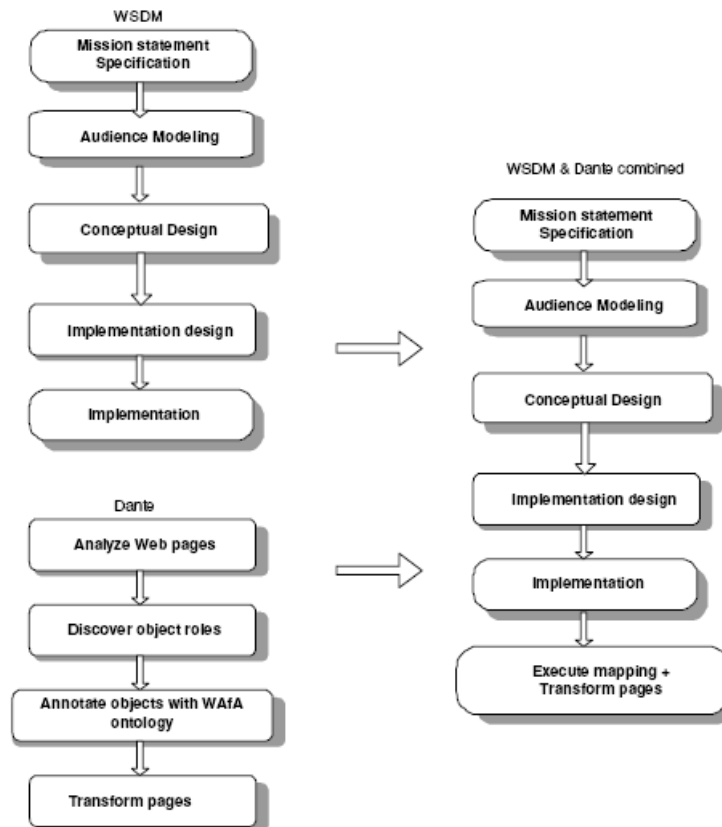


Figure 2.3: The WSDM with Dante from [51]

2.2.2 Automatic Annotations for Accessibility

The main goal in Plessers et al. [35] is to generate annotations for visually impaired users automatically from explicit conceptual knowledge existing during the design process. The approach integrates the Dante [52] annotation process into the Web Site Design Method (WSDM) [13] that allows Web sites and Web applications to be developed in a systematic way. The annotations are generated from explicit conceptual knowledge captured during the design process by means of WSDM's modeling concepts. These WSDM's modeling concepts, used in the different phases, are described using the WSDM OWL²³ ontology. To generate code the authors establish a transformation process that takes the conceptual design models as input and generates a set of annotations as a consequence. The transformation process consists of two annotation steps: authoring and mobility, which resemble the original annotation

²³ OWL Web Ontology Language at <http://www.w3.org/TR/owl-ref/>

process of the Dante approach. The difference is that the authoring annotation in Dante is manual and based on the HTML source code of the Web site. The integration of the Dante [52] annotation process into the Web Site Design Method (WSDM) [13] is graphically illustrated by Figure 2.3 [51].

As we can see in Figure 2.3 the transformation to an accessible design, takes place at the “Execute mapping + Transform pages” step, where a mapping between WSDM and Dante ontologies applies. The WSDM key models where transformation takes place are the WSDM site structure model and the WSDM presentation model, both outputs of the WSDM Implementation Design phase.

that is annotated with concepts from the Dante’s WAFa²⁴ ontology, a relationship between the concepts in the WSDM ontology and the WAFa ontology is established. By using these mapping rules,

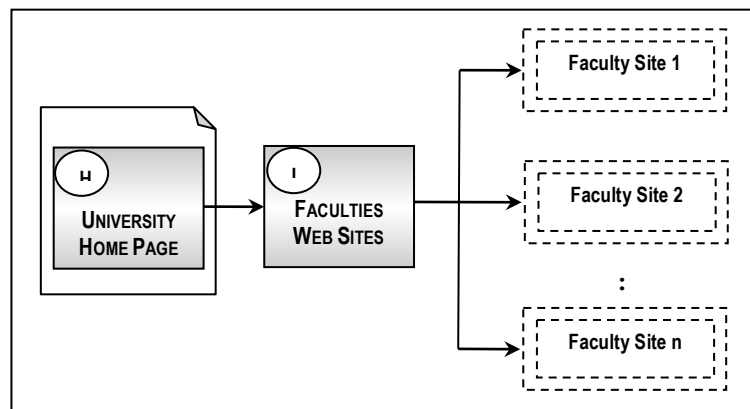


Figure 2.4: Part of the WSDM site structure model for the University home page example

Now, applying this proposal for developing the page example of Section 2.2.1, Figure 2.4 shows part of the WSDM site structure model. As we can see in Figure 2.4, we enrich this model of the University home page with navigational aid links --i.e. the home link and the landmark link components represented by means of the symbols “H” and “L” respectively. From home, the landmark link component offers a list of links that the student may choose when browsing to his/her Faculty Web site.

Figure 2.5 provides the WSDM presentation model as a page template for the University home page example, where the navigational aid links “H” and “L” from

²⁴ Web Authoring for Accessibility (WAFa) at <http://augmented.man.ac.uk/ontologies/wafa.owl>

Figure 2.4 are graphically highlighted in grey. Having these WSDM key models, the transformation process consists of two steps: (1) *Authoring Annotation transformation* which uses the information specified in the WSDM models and the Dante’s WAFa ontology to generate the authoring annotation and, (2) *Mobility Annotation transformation* which uses the output of the previous transformation as well as the WSDM models to extend the authoring annotation with mobility annotation to improve Accessibility.

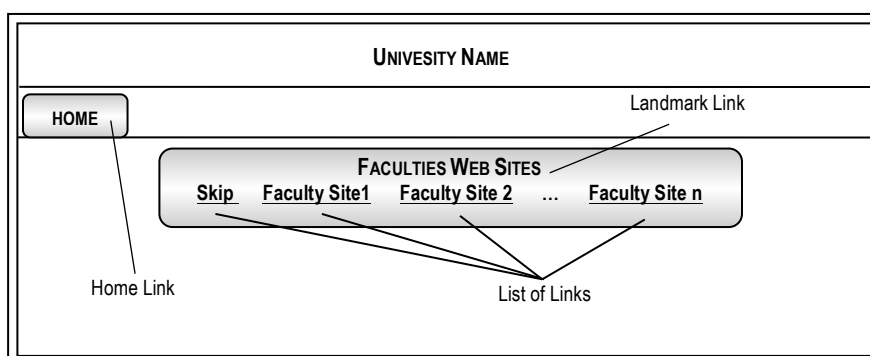


Figure 2.5: The WSDM presentation model for the University home page example

Following, we will explain the transformation process for the University home page example taking into account the WSDM models of Figures 2.4 and 2.5:

(1) *Authoring Annotation transformation*. This process uses the mapping rules between modeling concepts defined in the WSDM ontology and authoring concepts from the WAFa ontology. The “list of text links” at the page example, can be represented by the *List* concept (at WSDM ontology) and by the *NavigationalList* concept (at the WAFa ontology), but this is not a straightforward one-to-one mapping. So, assuming the set **C** as the set of all WSDM modeling concepts and the set **I** as the set of all instances of these modeling concepts, Figure 2.6 shows the corresponding mapping rule for the “list of links” to the Faculties web sites at the University page example. To avoid confusion while applying this rule, the WSDMs concepts are prefixed with “wsdm” and the WAFa concepts with “wafa”. The *NavigationalList* WAFa concept is given in bold, followed by its meaning (in italic), an informal explanation of the mapping rule and finally, a formal definition using first-order predicate logic.

WSDM ONTOLOGY CONCEPT	WAFa ONTOLOGY CONCEPT	MAPPING RULE BETWEEN WSDM AND WAFa ONTOLOGIES
<p>List</p>	<p>NavigationallList</p>	<p>wafa:NavigationallList: A "list of links". The annotation can be generated for <code>wsdm:List</code> where all list elements have a <code>wsdm:Link</code> defined upon them [35].</p> $\forall i \in I, \exists y \in I: \text{wsdm:List}(i) \wedge (\forall x \in I: \text{wsdm:hasChild}(i, x) \wedge \text{wsdm:ListItem}(x) \wedge \text{wsdm:hasNavigationReference}(x, y) \wedge \text{wsdm:NavigationReference}(y)) \rightarrow \text{wafa:NavigationallList}(i)$

Figure 2.6: Mapping rule for the “list of links” at the University home page example

(2) *Mobility Annotation transformation.* This process re-uses the mapping rules provided by the Dante approach [52], adjusting them to interact with the WSDM models instead of the HTML code of the Web page. Taking the output of the previous transformation as well as the WSDM models, we extend the *NavigationallList* authoring annotation with mobility annotation to improve Accessibility. Figure 2.7 provides the mapping rule [35] for mobility annotation transformation that applies to objects authoring annotated as a *NavigationallList*. All the links in the list are text links corresponding to the Faculties’ names for whose Web sites access are allowed to students. As the mapping rule from Figure 2.7 shows, the *NavigationallList* authoring concept must be annotated with the *DecisionPoint* and *NavigationPoint* mobility concepts, while the *TextLink* authoring concept (required because all the links in the list are text links) must be annotated with *NavigationPoint* and *TravelMemory* mobility concepts. As a consequence, the *NavigationallList*, where all the links in the list are *TextLink*, must be annotated with *DecisionPoint*, *NavigationPoint* and *TravelMemory* mobility concepts.

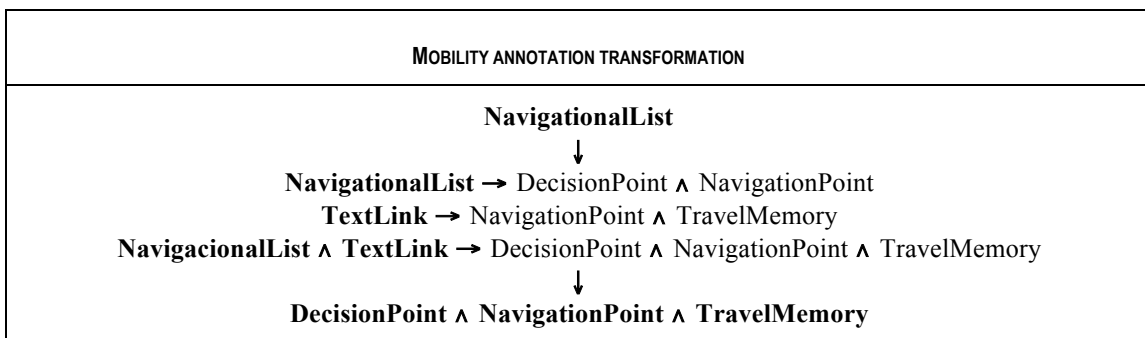


Figure 2.7: Mapping rule for the *NavigationallList* at the University home page example

A *DecisionPoint* is a choice point where alternative paths of browsing are possible; while a *NavigationPoint* provides a possible route and the user exercises some control by choosing to follow or not to follow it; finally, a *TravelMemory* holds information about where the user has been and provides means to get back there. For the particular case of the University home page example, these mobility concepts will offer a student a point from where it is possible to choose a Faculty name, browse to its Web site and also get back from there to the University home page. We must to keep in mind that authoring and mobility concepts are from WafA ontology, so the application of the rule for the Pleasers proposal [35], looks like shows Figure 2.8.

$$\begin{array}{c}
 \forall i \in I: \text{wsdm:String}(i) \vee \\
 (\exists x, y \in C: \\
 \text{wsdm:ObjetcChunkReference}(i) \wedge \text{toProperty}(i, x) \wedge \text{rang}(x,y) \wedge \text{wsdm:String}(y)) \\
 \rightarrow \text{Text}(i) \\
 \\
 \forall i \in I: \text{wafa:NavigationalList}(i) \wedge \\
 (\forall x \in I, \exists y \in I: \\
 \text{wsdm:hasChild}(i,x) \wedge \text{wsdm:ListItem}(x) \wedge \text{wsdm:hasChild}(x,y) \wedge \text{Text}(y)) \\
 \rightarrow \text{wafa:DecisionPoint} \wedge \text{wafa:NavigationPoint} \wedge \text{wafa:TravelMemory}
 \end{array}$$

Figure 2.8: The Pleaser et al. [35] proposal for the University home page example

The botton-half of the rule is a direct translation of the original rule, applied to the objects annotated as a *NavigationalList* where all *wsdm:ListItems* are text elements. The top-half of the rule formally defines a text element [35]. For further details of this proposal, we refer the reader to [35].

2.2.3 Rules for an Accessible Composition

The work by Centeno et al. [9] presents a set of rules that, in a Web composition process, a design tool must follow in order to create accessible Web pages. These rules are formalized with W3C standards like XPath²⁵ and XQuery²⁶ 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

²⁵ W3C XML Path Language at www.w3.org/TR/xpath

²⁶ W3C XML Query Language at www.w3.org/TR/xquery

using the “Web-Composition Service Linking System” (WSLS) [20] as Accessibility enabled authoring tool that makes this task feasible, and focus on how this tool incorporates Accessibility into the process of generating new Web contents. The XPath and XQuery expressions spot HTML nodes and attributes having Accessibility problems. This work proposes to properly manage these spot elements by an authoring tool, so that the author’s attention can be directly brought to these barriers in a semi-automated edition process.

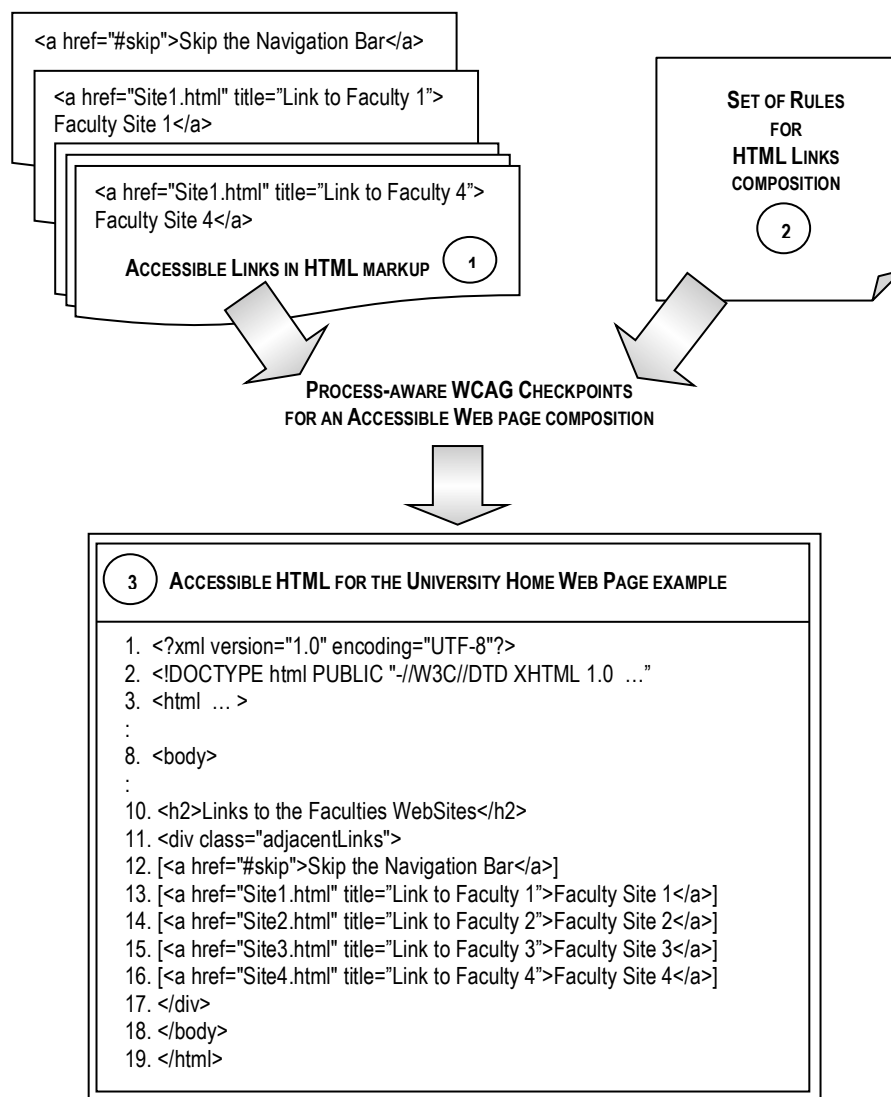


Figure 2.9: The Centeno et al. [9] proposal for the University home page example

The WSLS approach follows the AOSD separation of concerns principle to decompose complexity and control Accessibility over six distinguished categories: Data, Presentation, Navigation, User, Interaction, Process and Communication. The six

elements are mediated by a service control function. Beyond the advantage of the reuse aspect of these components, separation of concerns facilitates also being compliant to the underlying guidelines [9].

Figure 2.9 resumes graphically the proposal at Centeno et al. [9] applied to the page example of Section 2.2.1. As highlighted in Figure 2.9 (1), given $\$S1$ to $\$S5$ compoundable pieces of HTML markup (also called HTML snippets), each one represents an accessible link to a Faculty of the student's University. The composition of these accessible chunks of Web pages, must follow some rules in order to create an accessible "list of links" at the University home page. The proposal provides a set of rules that are focused on formalizing the conditions to be met so that accessible HTML snippets can be safely compound into a page that also results accessible from the WCAG point of view. As shown in Figure 2.9 (2), from the set of rules provided by the proposal, we select for the page example only those rules for HTML links composition. For example, rule 10.5 establishes "provided that all $\$S1$'s and $\$S2$'s links have non-consecutive links (some printable text between links), their composition could have consecutive links without such printable characters if a $\$S2$'s link appears just in front of $\$S1$'s link" [9]. This condition for rule 10.5 ("non-consecutive links") is formalized with a combination of XPath and XPointer as depicted in Figure 2.10 Since this formalization is somewhat difficult for those unfamiliar with XPath and XPointer, the next row of Figure 2.10 summarizes its meaning in simpler terms to facilitate its reading; remember that "a" represents an HTML *a* element that is used to define links.

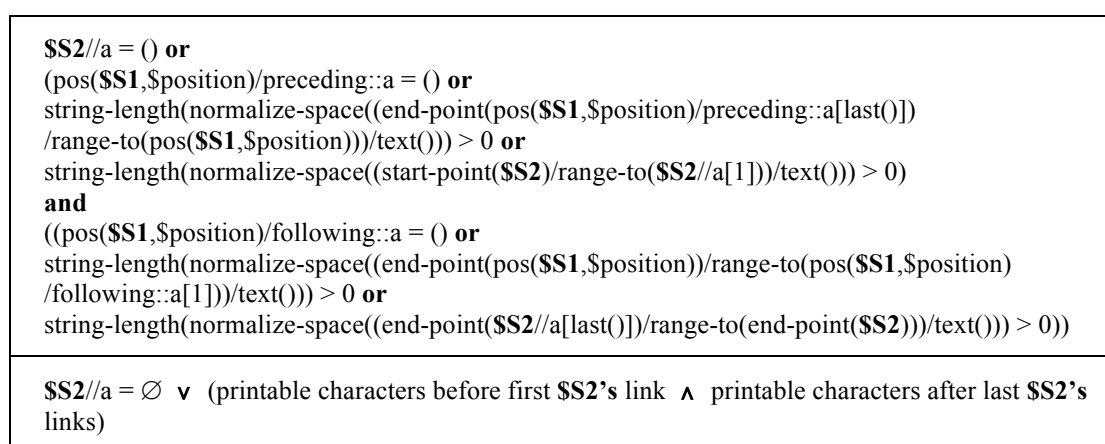


Figure 2.10: XPath + XPointer pre-conditions for avoiding consecutive links without printable non-linkable characters between them [9]

Meanwhile, rule 13.1 establishes “there should be no links sharing both a text and a title but pointing to different targets; provided $\$S1$ and $\$S2$ have no such ambiguous links there exist a functional dependency such that for every pair of (link’s contents, link’s title) only a single target may be found in both $\$S1$ and $\$S2$. In that case, we should also make sure that no link in $\$S1$ is similarly described in $\$S2$ (and pointing to a different target), or vice-versa; if so, an ambiguity would be introduced in the composed result” [9]. This condition for rule 13.1 (“clear links”) is formalized with XPath as depicted in Figure 2.11.

```
(every $a1 in $S1//a satisfies $S2//a[text() = $a1/text() and @title = $a1/@title and @href != $a1/@href] = ()) and  
(every $a2 in $S2//a satisfies $S1//a[text() = $a2/text() and @title = $a2/@title and @href != $a2/@href] = ())
```

Figure 2.11: XPath pre-condition for avoiding ambiguous links [9]

Returning to Figure 2.9, given $\$S1$ to $\$S5$ HTML snippets corresponding to Faculty links and rules 10.5 and 13.1, a process-aware WCAG checkpoints takes place for Web page composition to deliver an accessible “list of links” at page example. As we can see in Figure 2.9 (3), the “list of links” conform rules 10.5 and 13.1 responding respectively to the statements “non consecutive links” --i.e. printable characters between links where included, and “clear links” --i.e. title’s, target’s and content’s links are properly specified, to avoid students get confuse while browsing his/her University home page example. For further details of this proposal, refer to [9].

2.2.4 Adaptation to tackle Crosscutting Concerns

Casteleyn et al. [6], focus on how to extend an application with new functionality without having to redesign the entire application. The work states that since creating a Web application has become an increasingly complex task, various design issues like device-dependence, privacy, security, Accessibility, localization, personalization, etc. have become extremely relevant to the application performance. To add new functionality, the authors propose to separate additional design concerns and describe them independently. By using a component-based implementation, they show how to extend a Web application to support additional design concerns at the presentation

generation level. Furthermore, they demonstrate how an aspect-oriented approach can support the high-level specification of these (additional) design concerns at a conceptual level.

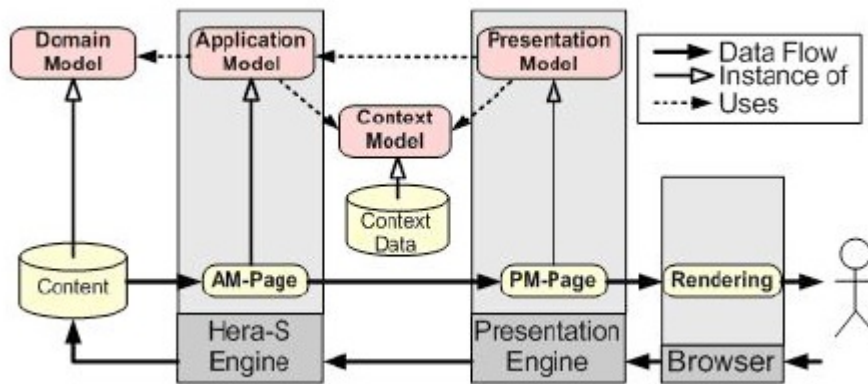


Figure 2.12: Hera-S architecture [8]

The work firstly illustrates how to add adaptation to an existing Hera-based Web application [23], using a component-based implementation. To do so, they apply the Generic Adaptation Components (GAC) approach [16] provided by the AMACONT²⁷ project. Niederhausen et al. introduce further work over this foundation [32] that proposes an aspect-oriented view on adaptation engineering within the AMACONT framework. By separating the specification of adaptation from the underlying application in the form of so-called adaptation aspects, this work proposes to add new or modify existing adaptation concerns on demand. The authors also present an extension of their graphical authoring tool AMACONTBuilder [15]. This extension allows Web engineers to intuitively incorporate adaptation aspects into Web applications. Casteleyn et al. latest implementation [7] [8] proposes a Semantic-based Aspect-oriented adaptation approach materialized in the form of a domain specific language, which the authors called Semantic-based Aspect-oriented Adaptation Language (SEAL)²⁸. It is presented in the context of a Web Information System (WIS) design method, Hera-S, which combines the popular open source Resource Description Framework (RDF)²⁹

²⁷ System Architecture for Multimedia Adaptive WebCONTENT at http://www-mmt.inf.tu-dresden.de/Forschung/Projekte/AMACONT/index_en.xhtml

²⁸ SEAL BNF specification at <http://wise.vub.ac.be/downloads/research/seal/SEALBNF.pdf>

²⁹ W3C RDF/XML syntax specification at <http://www.w3.org/TR/REC-rdf-syntax/>

called Sesame [5] and the rich modeling capabilities of Hera [23], a model-driven approach for engineering Web applications based on semantically structure data. They choose Hera-S because: (i) it naturally builds on Semantic Web data and, (ii) it was conceived with adaptation in mind. An illustrative overview of Hera-S architecture is shown in Figure 2.12 from [8]. Basically, the architecture receives data from the actual source, which conforms to the Domain Model (DM). The Application Model (AM) is instantiated according to the context data provided by the Context Model (CM), resulting in so-called Application Model Pages (AMPs). The authors devised their own custom-made aspect language SEAL to provide adaptation support in the context of Hera-S. By using SEAL's syntax, which is based on BNF notation, they show their adaptation engineering perspective applying pointcuts and advices expressions.

```

:UniversityUnit a ams:NavigationalUnit ;
ams:hasInput [ a ams:Variable ;
ams:varName "U";
ams:varType uncdb:University] ;

ams:hasAttribute [
rdfs:label "UniversityName" ;
ams:hasQuery
"SELECT N1 FROM {$U} rdf:type {uncdb:University};
rdfs:label {N1}"];

ams:hasSetRelationship [
rdfs:label "Faculties" ;
ams:refersTo :FacultyUnit ;
ams:hasQuery
"SELECT F FROM {$U} rdf:type {uncdb:University};
uncdb:unversityFaculty {F}"
].

:FacultyUnit a ams:NavigationalUnit ;
ams:hasInput [ a ams:Variable ;
ams:varName "F";
ams:varType uncdb:Faculty] ;

ams:hasAttribute [
rdfs:label "FacultyName" ;
ams:hasQuery
"SELECT FN FROM {$F} rdf:type {uncdb:Faculty};
rdfs:label {FN}"
].

```

Figure 2.13: The Hera-S AM for the University home page example

To demonstrate the practicality of their proposal, they apply and integrate SEAL in the HydraGen engine³⁰ (an implementation generation tool for Hera-S developed externally by the University of Eindhoven).

Now, applying this proposal for developing our University home page example of Section 2.2.1, a Hera-S Application Model (AM) using Turtle RDF notation³¹ would include the statements shown in Figure 2.13.

An Hera-S Application Model (AM) is specified by means of navigational units (denoted by `ams:NavigationalUnit` and called shorthand: units). A unit can be used to represent a page and it is a primitive that (hierarchically) groups elements (called attributes) that will together be shown to the user. The type of a unit (denoted by `ams:varType`) refers to a domain data and the specification of this type is done by using the namespace-prefix from the Hera-S Domain Model (DM). Our Hera-S AM example bellow, consists of two units, *UniversityUnit* and *FacultyUnit*, which are of the type `uncdb:University` and `uncdb:Faculty` respectively (in this case this namespace-prefix from our Hera-S DM stands for “Universidad Nacional de Córdoba Data Base”). Both units are navigational units of Hera-S AM, each one representing a particularly grouping of information. For example, the *UniversityUnit* contains one attribute (denoted by `ams:hasAttribute`) representing the university’s name and a set of navigational relationships (denoted by `ams:hasSetRelationship`) from *UniversityUnit* to *FacultyUnit*. Note that the `ams:SetRelationship` “refersTo” the *FacultyUnit*, which specifies what exactly to show for every faculty. Since a unit will mostly correspond to (a) specific domain concept(s), one or several content elements are needed in order to instantiate the unit. For example, in the *UniversityUnit* the output of the SeRQL queries (denoted by `ams:hasQuery`) provides a university name and a number of members which will be used respectively to instantiate the *UniversityName* and the *Faculties* of the *UniversityUnit*.

Now, by using the domain specific language SEAL it is possible to apply the Casteleyn et al. proposal [8], to provide aspect-oriented adaptation support in the context of Hera-

³⁰ Hydragen: An implementation of Hera-S at <http://www.wis.win.tue.nl/~ksluijs/material/Singh-Master-Thesis-2007.pdf>

³¹ W3C-Turtle at <http://www.w3.org/TeamSubmission/turtle/>

S for the University home page example of Section 2.2.1. As Figure 2.14 shows, we have instantiated the adaptation requirement to stand for the Accessibility requirements of adjacent links. The *adaptation aspect* is composed of a pointcut and an advice; while pointcut expressions select exactly those elements from the Application Model (AM) where adaptation concerns need to be applied. Advices specify exactly what needs to be done to the element(s) selected in the pointcut [8]. Back to our example of Section 2.2.1, the pointcut in Figure 2.14 selects sets of relationships --i.e. consecutive links, which originate from a(ny) University unit and target a(ny) Faculty unit. The advice is conditioned to users using a “screen-reader” device. As we explained above, in Hera-S the user’s context is captured by the Context Model (CM) and with Hera-S notational conventions, referencing this user’s context is done using a “cm:” -prefix.

Adaptation REQUIREMENT: *for users using a screen-reader avoid consecutive links and clearly identify the target of each one of them.*

Adaptation ASPECT:

POINTCUT: type SetRelationship and from uncdb:University and to uncdb:Faculty

ADVICE: if (cm:userDevice.type = “screen-reader”) {

ADD attribute containing hasLabel “Faculty Name”, hasQuery “SELECT FN FROM {SF} rdf:type {uncdb:Faculty}; rdfs:label {FN}”;

ADD rdf:plainLiteral “[” and “]” surrounding;

};

Figure 2.14: Aspect-oriented adaptation using SEAL for Accessibility requirements of the University home page example

Firstly, the advice adds an AM attribute to the relationships selected in the pointcut showing the faculty name with the label “Faculty Name” and the corresponding query, if the user’s device is a “screen-reader”. Secondly, the advice also uses plain RDF(s)³² to add square brackets surrounding the relationships selected in the pointcut.

Although, this approach is primarily focused on adapting an existing Web application, we include it because the approach proposes to add relevant design concerns, like

³² W3C-RDF:PlainLiteral: A data type for RDF Plain Literals at http://www.w3.org/TR/rdf-plain-literal/#Syntax_for_rdf:PlainLiteral_Literals

Accessibility, in an aspect-oriented manner and, it is representative of other similar works in the adaptation field, like [1] [37]. For further details of this proposal, we refer the reader to [6] [7] [8].

2.2.5 User Needs through Personas

By using existing “best practices of software engineering” for Accessibility purposes, the approach by Zimmermann & Vanderheiden [53] presents a methodology for accessible design and testing to capture functional requirements. The approach defines a new way to use proven tools of software engineering, like use cases, scenarios, test cases, guidelines and checkpoints, for Accessibility purposes; and to relate them to each other, thus facilitating automation as much as possible. The resultant methodology or process model for accessible design and testing consist of: (i) capturing Accessibility requirements in a way that makes them tangible and comprehensible, through use cases and the technique of user profiling “personas” [53], (ii) making Accessibility requirements concrete through scenarios and guidelines for accessible design, (iii) manual and automatic testing based on test cases and Accessibility checkpoints that are derived from guidelines, and (iv) complementary user testing and expert reviews, thus evaluating intermediate and end results, and continuously improving the overall process model.

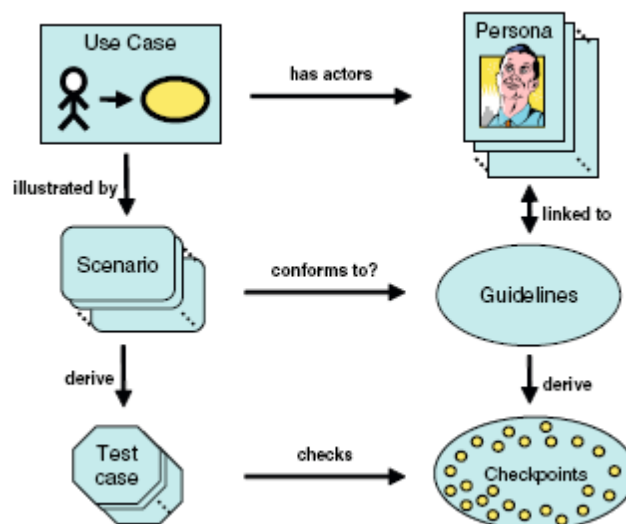


Figure 2.15: Components of the integrated approach and their relationships [53]

In this way for design projects that are employing a use case driven methodology, this approach allows to incorporate accessible design into the existing processes rather than having to add Accessibility as a new process [53]. Figure 2.15 from [53] shows how basic design tools as use cases, scenarios and test cases are linked to personas, guidelines and checkpoints respectively for Accessibility purpose.

Figure 2.16 shows the process model for accessible design and testing by Zimmermann & Vanderheiden [53] applied to our University home page example of Section 2.2.1 and using WCAG 1.0 Accessibility guidelines.

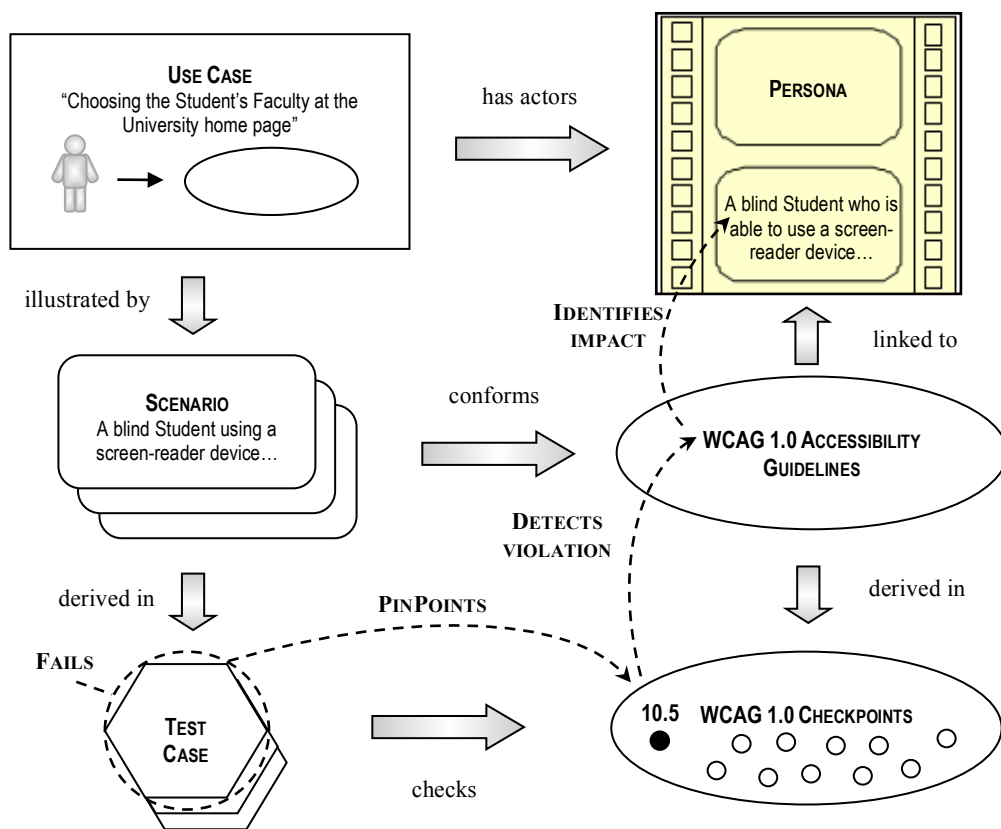


Figure 2.16: The Zimmermann & Vanderheiden [53] proposal for the University home page example

Figure 2.16 shows a situation where a test case is failing because an Accessibility requirement for adjacent links is not met. In this case, the proposed model makes it possible to pinpoint to a particular checkpoint that is causing the failure (10.5 checkpoint), and trace it back to a particular guideline that is violated (guideline 10 from WCAG 1.0). This allows identifying a particular persona (a blind Student) who

despite being able to use a screen-reader will not be able to access the application because of the Accessibility barrier identified by the test case failure. The model presented here is not only useful for fixing the Accessibility problems, but also provides a context to the developers for understanding the consequences of failure [53]. For further details of this proposal, we refer the reader to [53].

2.2.6 Model-Driven Development with AWA

Accessibility for Web Applications (AWA) [29] [30] offers a domain specific methodological framework for the development of accessible Web applications. The AWA framework provides: (i) a specific Accessibility process (which can be adopted by other processes), indicating activities, artifacts and their sequence in the different phases of integrating Accessibility criteria, and (ii) the support for modeling and using techniques provided by Web Engineering (WE) methods as well as Model-Driven Development (MDD), the focus of this work.

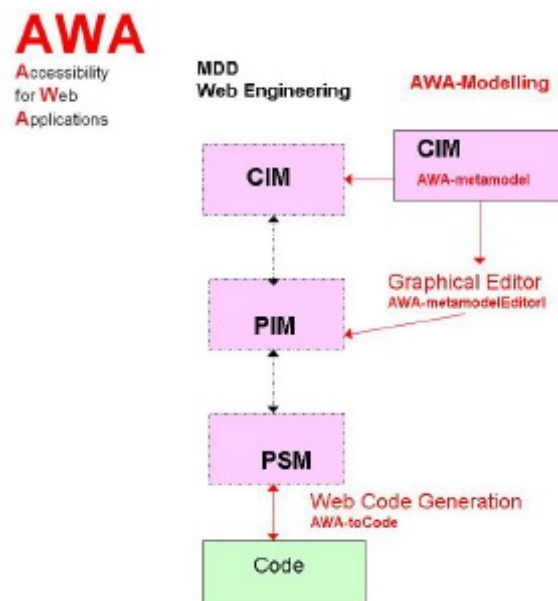


Figure 2.17: AWA for MDA development process [29]

As shown in Figure 2.17, the strategy in AWA consists of providing 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. The authors provide an AWA-

toCode resource and the strategy is based on a transformation Model-to-Text (M2T) to generate code from PSMs. In this work, they also announced that they have developed a CASE support for metamodeling, using the Ecore plugin from the Eclipse Modeling Framework (EMF)³³ [29].

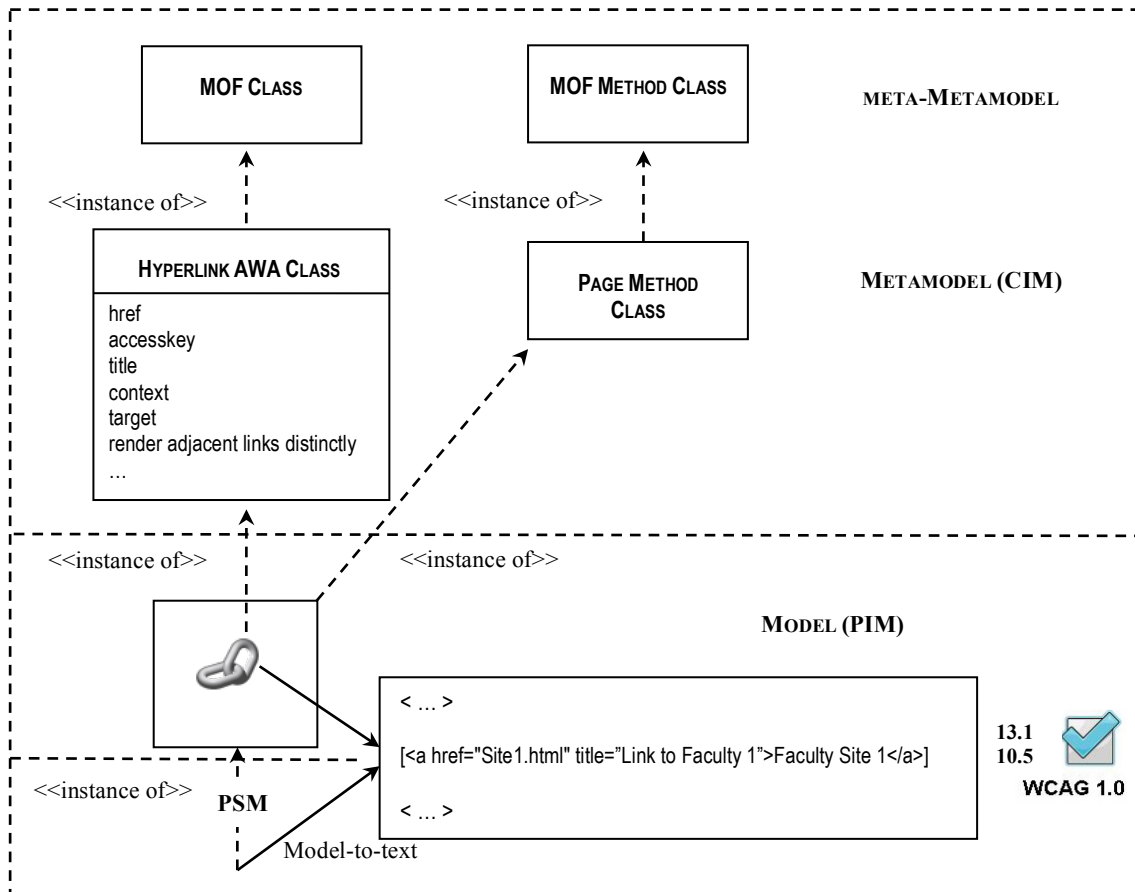


Figure 2.18: The Moreno et al. [29] proposal for the University home page

Figure 2.18 shows AWA for Model Driven Architecture (MDA)³⁴ applied to the Hyperlink concept required by our University home page example of Section 2.2.1. Here several constructors have been defined in the MetaObject Facility (MOF)³⁵ to support the abstraction of Web Accessibility concepts. The diagram develops the concept of hyperlink that includes required attributes to enable the hyperlink to meet the

³³ EMF overview at <http://help.eclipse.org/indigo/index.jsp?topic=/org.eclipse.emf.doc/references>

³⁴ OMG-MDA overview at <http://www.omg.org/mda/>

³⁵ OMG-MOF specification at <http://www.omg.org/mof/>

WCAG standard, such as the title attribute. This attribute contributes to satisfy the 13.1 checkpoint of WCAG 1.0 that establishes “Clearly identify the target of each link”. To continue with the example of Section 2.2.1, Moreno et al. [29] do not consider the 10.5 checkpoint of WCAG 1.0 as a property for the link/hyperlink concept. Although, notice that as we have done at the hyperlink AWA class in Figure 2.10, it is possible to include the “render adjacent links distinctly” attribute, to enable meeting this Accessibility requirement, if the presence of adjacent links makes it necessary.

A graphic element representing a hyperlink (MOF meta-object) has been defined in the AWA-Editor, and may be included in the PIM models, which contain knowledge provided by the AWA-Metamodel necessary for the Web code generation in the final phase [29]. For further details of this proposal, we refer the reader to [29] [30].

In this Chapter we presented Accessibility in the context of some WE approaches. We reviewed and applied in a case study five different proposals [35] [9] [6] [53] [30] that consider this quality factor in the development process of Web applications.

After introducing background (Chapter 3) and our proposal (Chapter 4), we will apply it (Chapter 5) and we will come back to the approaches summarized here to compare them to our proposal (Chapter 6).

3. BACKGROUND OF OUR PROPOSAL

3.1 Introducing the Basis

In the following Sections we introduce four key topics that we will use throughout the rest of the work, to make it self-contained. These are: (i) Aspect-Oriented Composition, (ii) Reference Frameworks and Ontologies, (iii) User Interaction Diagrams (UIDs), and (iv) Softgoal Interdependency Graphs (SIGs). Our aim is not to discuss these issues in detail; instead we intend to stress the most important concepts. We also devote a special section to the motivation for using the WCAG 1.0 [45] instead of WCAG 2.0 [46].

3.2 Aspect-Oriented Composition

A concern is an area of interest or focus in a system. Since Dijkstra [13], concerns are the primary criteria for decomposing software into smaller, more manageable and comprehensible parts that have meaning to a software engineer. Examples of concerns include requirements, use cases, features, data structures, quality-of-service issues, variants, intellectual property boundaries, collaborations, patterns and contracts. Thus, Separation Of Concerns (SOC), is a long standing idea that refers to the ability of identifying, encapsulating and manipulating parts of software that are crucial to a particular purpose [13]. Software engineering development methods have been created with this principle in mind. However, traditional paradigms to software development, such as Object-Oriented methods and languages, are not able to modularize crosscutting concerns effectively, because they suffer from a limitation called the “Tyranny of the Dominant Decomposition”. This limitation means that they allow modularization in only one way at a time, so they are unable to solve the many kinds of concerns that do not align with that main modularization. In other words, given one out of many possible decompositions of the problem (most of them are core functionality concerns), some sub-problems show, such as non-functional and functional requirements, added after facts, etc., which cannot be modularized. These problems are concerns that cut across many other concerns producing “crosscutting symptoms” resulting into representations - e.g. specifications, classes, code, etc., which are difficult to understand and maintain.

An important issue to underline about this kind of behavior is not only manifested for: (i) a given decomposition, but for all possible decompositions, (ii) a given paradigm, such as object-orientation, also in other paradigms and, (iii) at the implementation stage, also in other stages, such as analysis and design. Usually, these crosscutting symptoms manifest in “scattering” and “tangling” problems. We say that the representation of a concern is scattered over an artifact, when the code for the implementation of the concern’s body is spread out over multiple and different modules or classes rather than localized. While the representation of a concern is tangled within an artifact, when the code for the implementation of the concern’s body is intermixed with code that implements other concerns’ bodies. Scattering and tangling often go together, even though they are very different concepts [17].

Typical examples of such crosscutting concerns are non-functional requirements, such as security, availability, persistency, usability and Accessibility, the main topic of this paper. However, crosscutting concerns can also be functional requirements, such as order auditing, validation, and in the Web engineering domain, tracing the user navigation history [21].

SOC can be supported in many ways, such as by process, by notation, by organization, by language mechanism and, so on. Within the broad theme of SOC, Aspect-Oriented Software Development (AOSD) is distinguished by providing new insight on the separation of crosscutting concerns and in particular leads to the idea that single hierarchical structures are too limiting to effectively separate all concerns in complex systems³⁶. AOSD aims at handling such crosscutting concerns at the various levels of the process of software development, by providing means to their systematic identification, modularization and composition [17]. Crosscutting concerns are encapsulated in separate modules, known as “aspects”, and composition mechanisms are later used to weave them back with other core modules, at loading time, compilation time, or run-time. Since aspects are concerns that crosscut a primary or dominant decomposition (other core modules), aspect “weaving” is a composition mechanism that injects aspects into this primary or dominant decomposition.

However, aspects, as well as their compositions, also have an important role to play

³⁶ AOSD community at http://www.aosd.net/wiki/index.php?title=Main_Page

before the implementation. On one hand, the notion of “early aspects” means it is important to consider aspects early on in the software engineering lifecycle during analysis and design, as opposed to only at the implementation and testing stages. At these early stages of the development process, aspects will allow the modularization of crosscutting concerns that cannot be encapsulated by a single use case, for example, and are typically spread across several of them. Composition, on the other hand, allows the developers to picture the whole system and to identify conflicting situations whenever a concern contributes negatively to others [17].

Traditionally, AOSD has focused mainly on the implementation phase of the software lifecycle since aspects are identified and captured mainly at coding. But aspects have been also applied to former phases as design and even earlier as requirements to cover consistently the entire development process [2] [28].

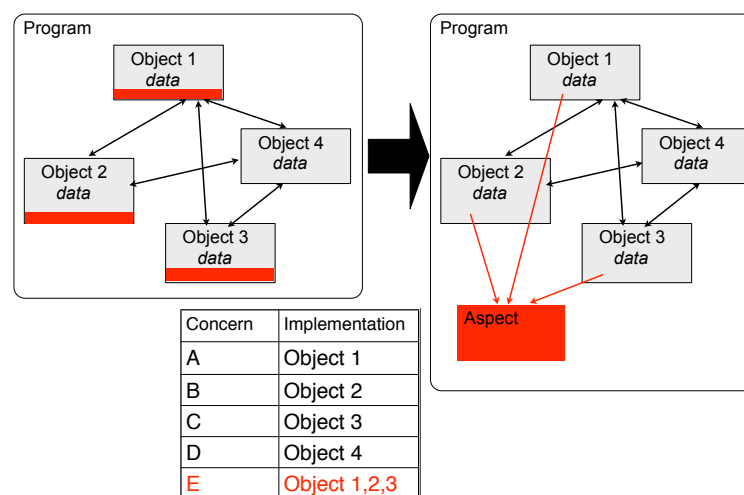


Figure 3.1: Aspects modularization [4]

3.2.1 Aspectual Implementation: Advices and Pointcuts

Aspect-orientation proposes a fundamentally new kind of modularization that goes beyond generalized procedures: an aspect. 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. Thus, the key to this modularization technique lies in its module composition mechanism. Figure 3.1 shows graphically the idea supporting aspects using an example at the implementation level. While subroutines explicitly

invoke the behaviors implemented by other subroutines, aspects have an implicit invocation mechanism [4]. This mechanism that injects aspects into the primary or dominant decomposition is called “aspect weaving”. The implicit invocation mechanism requires that the aspect itself specifies “where or when” it needs to be invoked and also “what” needs to be injected.

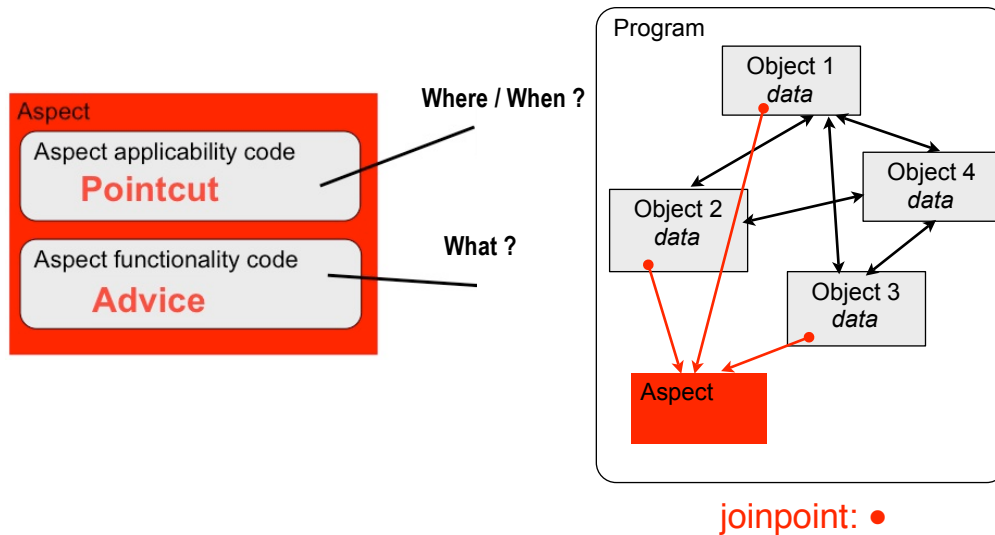


Figure 3.2: Aspects implementation [4]

Consequently, as Figure 3.2 shows, an aspect implementation consists of two conceptually different parts: the aspect functionality code --i.e. aspect functional implementation, and the aspect applicability code --i.e. aspect control over implicit invocation. The aspect functionality code is not essentially different from regular code and is executed when the aspect is invoked. This invocation of the aspect is determined by the aspect applicability code. This code contains statements that specify where or when the aspect needs to be invoked. In standard AOSD terminology, this aspect applicability code is referred to as a “pointcut” expression, which must match a join point, and the aspect functionality code is referred to as the aspect “advice” code. Since a single aspect can consist of multiple different functionalities that need to be invoked from various different places in the code, an aspect implementation can consist of several pointcuts and advice code segments.

3.3 Reference Frameworks and Ontologies

Our approach involves two main elements when designing the user interface towards achieving Accessibility of Web applications. Firstly, a reference framework can serve us as a conceptual structure for making design decisions when building useful user interface models for Accessibility purpose. Secondly, ontologies can provide us with a formal specification for the abstract interface vocabulary. In the following sections, we introduce these two main elements.

3.3.1 Design Decisions within a User Interface Framework

There are many decisions that developers must make during the design of a user interface. As with any complex decision-making process, it is useful to partition the set of decisions into classes and concentrate on the decision in each class, separately. A design decision framework consists of a collection of design decision classes. When decisions in each of the design decision classes are combined, an overall design is synthesized [27]. The criteria for identifying and constructing decision classes are separation, completeness, sufficiency, understandability, independence, reusability and soundness.

We applied in our work the Larson's user interface design decision framework [27] that defines the following five classes:

- *Structural* decision class, which specifies the structure of the end users' conceptual model. These specifications include a description of the conceptual objects that are consumed, produced, and/or accessed by the end users and application functions.
- *Functional* decision class, which specifies functions (operations), which the user can apply to the conceptual objects. Functional decisions determine what requests the users can express and what results the application functions can present to the user.
- *Dialog* decision class, which specifies the content and sequence of information exchange between the user and the application. In this class, the designer specifies the dialog style taking into account: (i) what the units of information exchanged between the user and the application are, (ii) how these units of information are structured into messages exchanged between the user and the application and, (iii)

what the appropriate sequences of message exchanged are. These units of information, which have a formally defined meaning, are called “semantic tokens”.

- *Presentation* decision class, where the designer chooses interaction objects that make up the end users’ interface. Informally, interaction objects are visible widgets on a screen that the user can manipulate to enter lexical tokens and which the user views to obtain lexical tokens. A “lexical token” is a keystroke, mouse movement, or mouse click entered by the user or a character, icon, or elementary sound presented to the user.
- *Pragmatic* decision class, which deals with issues of gesture, space, and hardware devices. Often these decisions are determine by designers in conjunction with ergonomic specialist.

Since the last three classes are related to the user interaction and activities with the application’s interface, and they are also directly involved with Web Accessibility, we ensure their inclusion in our approach. As an example, consider decisions involving Accessibility requirements in the case of playing a song’s track at a music Web site. The *Dialog* decision class must describe a sequence of commands for turn-on / turn-off the song’s track. While in the *Presentation* decision class, the designer chooses the appropriate vocabulary and widgets for individualizing these two commands clearly to the user. Finally, in the *Pragmatic* decision class, the designer chooses the hardware, such as a mouse or a touchscreen, for selecting these commands.

Larson's framework [27] gives us a comprehensive and general view that can be instantiated with different conceptual models, such as the approach proposed eleven years later by Baxley in [3]. This proposal describes a universal model of a user interface that can be applied to any interactive medium or product based on the established model of structure-behavior-presentation.

Table 3.1 shows how this early proposal, can be easily mapped to design decision classes introduced by the Larson’s framework to add additional levels of granularity or specificity. For example, Larson’s presentation class (corresponding to Baxley’s presentation tire) can be specified in depth at layout, style and Baxley’s text layers. This can be useful if the design for the user interface under development requires the explicit identification of these components at the presentation model.

Table 3.1: Mapping between Larson’s framework [27] and Baxley’s model [3]

Baxley’s Universal Model of User Interface		Larson’s User Interface Design Decision Framework
Tiers	Layers	Classes
Structure	Conceptual Model	Structural & Functional
	Task Flow	
	Organization Model	
Behaviour	Viewing & Navigational	Dialog
	Editing & Manipulation	
	User Assistance	
Presentation	Layout	Presentation
	Style	
	Text	

3.3.2 An Ontology to share Abstract Interface Vocabulary

Any hypermedia Web application exchange information through its user interface with its environment in order to fulfill a task. The most abstract level is called abstract user interface and focuses on the various types of functionality that can be played by interface widgets with respect to the information exchange between the user and the application.

We applied the Abstract Widget Ontology [36], which provides 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. This ontology can be thought of as a set of classes whose instances will comprise a given interface.

As shown in Figure 3.3, an abstract interface widget can be any of the following [36]:

- *SimpleActivator* widget, which represents elements capable of reacting to external events, such as mouse clicks on links or action buttons.
- *ElementExhibitor* widget, which represent elements able to exhibit some type of content, such as text or images.
- *VariableCapture* widget, which represent elements able to receive/capture, the value of one or more variables. As we can see in Figure 3.3, the *VariableCapture* widget generalizes two distinct (sub) concepts. The first one is the ontology (sub) concept *PredefinedVariable*, which represents elements that allow the selection of a subset from a set of predefined values, such as buttons and check boxes; often this selection must be a singleton. The second ontology (sub) concept is the

IndefiniteVariable, which represents elements that allow the user to enter data (previous unknown values) through the keyboard, such as text typed by the user in a text box on a form.

- *CompositeInterfaceElement* widget, which is a composition of any of the abstract interface widget represented by the ontology's previous concepts.

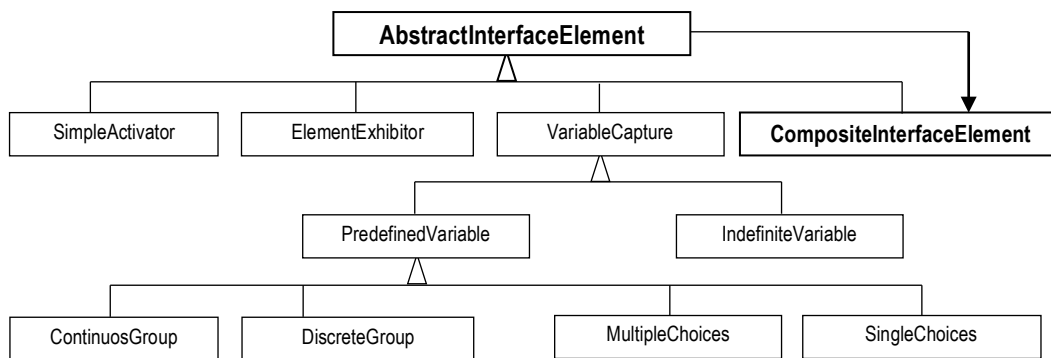


Figure 3.3: Abstract Widget Ontology [36]

It becomes evident from this ontology the essential roles that interface elements play with respect to the interaction --i.e. they exhibit information, or they react to external events, or they accept information. Composite elements allow us to build more complex interfaces out of simpler building blocks [36]. Once the abstract interface model has been defined, each widget is mapped onto a concrete widget to specify the concrete interface model. An abstract interface widget provides a type of functionality to the user by using an interface element, while a concrete interface widget is the actual implementation of that interface element in a given mark-up language or a runtime environment.

Since HTML is the “lingua franca” --i.e. a means of communication between people of different languages for publishing hypertext on the World Wide Web, in Sections 5.3.2 and 5.4 we map these ontology concepts onto HTML elements; this mapping is presented when we describe our model for user interface concerns.

3.4 User Interaction Diagrams

A User Interaction Diagram (UID) [44] is a diagrammatic modeling technique focusing exclusively on the information exchange between the application and the user. UIDs are

an outstanding tool to support the communication between different stakeholders during requirements specification and are particularly valuable considering the interactive nature of Web applications. UIDs can be used to enrich the use case 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 [43].

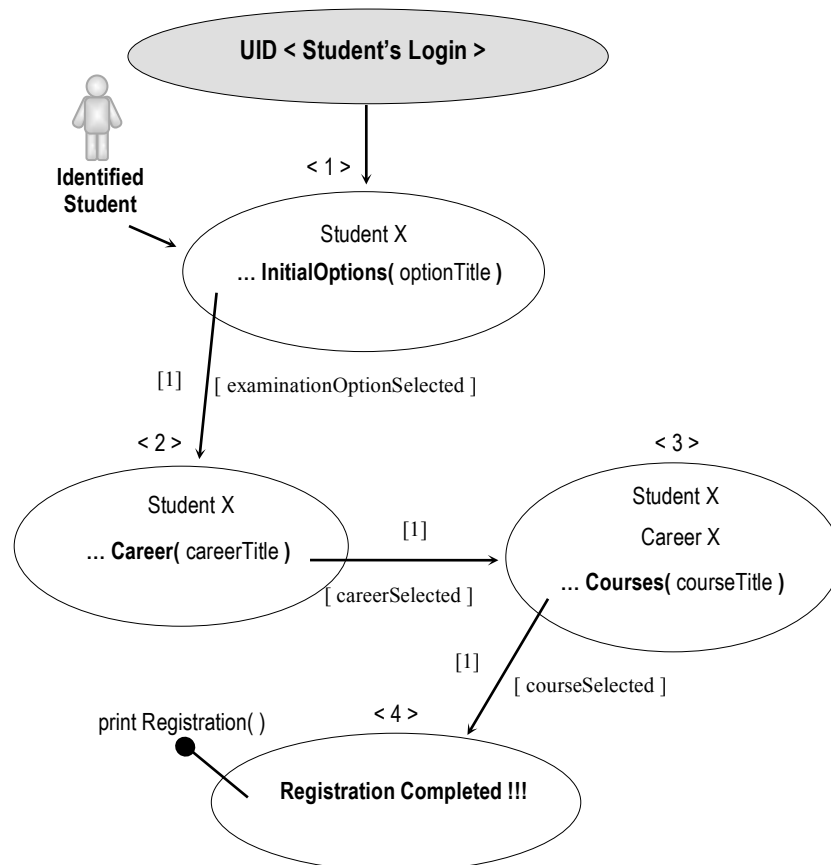


Figure 3.4: A simple UID: Enrolling a Student in an Examination Board given a Course

UIDs are simple state machines, and at the same time an effective instrument to convey the evolution of a Web application process and to support traceability from requirements to later design steps, smoothing the way to implementation. In Figure 3.4 we show a simple UID to express the use case “*Enrolling a Student in an Examination Board given a Course*” in the context of the SIU Guarani registration system.

To ease the comprehension of Figure 3.4, we include here some remarks about the UID’s notation. The ellipse represents an interaction between the user and the system and is assigned a number representing its order in the interaction sequence. An ellipse

with an arrow without a source particularly recognizes the initial interaction; the results of each subsequence interaction, which cause processing in the system, should be represented as a separate ellipse, connected to the preceding interaction by an arrow. Each ellipse offers content to the user that depends on the interaction sequence of the task represented by the UID. For example, an ellipse can provide the user with any of the following widgets: (i) a data entry i.e-- data entered by the user and graphically represented by a rectangle; (ii) text i.e--descriptive text represented by “XXXX”; (iii) a structure with their data items or a set of structures with their data items i.e--selectable elements represented by “element(data items)” or by “...element(data items)” respectively. A more formal description of the original UID’s notation can be found in [43] [44].

In the first interaction of Figure 3.4 (indicated by <1> and an incoming arrow), a student already identified at the SIU Guarani system by a previous UID corresponding to the use case “*Login a Student given the Student’s ID and Password*”, selects only the examination option (represented by “[1]”) from an initial set of options (represented by “...”). At interaction <2>, the response of the system is the set of careers in which a student is enrolled. Notice that this set always has at least two elements and this is because even if the student is enrolled in only one career, the SIU Guarani system offers examination enrolling for admission’s courses or career’s courses. The student chooses one of them and the system returns at interaction <3> a complete set of courses (related to the selected career) in which the student is able to enroll. The student selects a course and the system returns at interaction <4> the registration to an examination board for the course. Additionally, the user can perform the operation “*print Registration*” (indicated by a line with a black bullet) to get a receipt of the registration completed. The complete syntax for UIDs can be found in [44].

3.5 Softgoal Interdependency Graphs

Softgoal Interdependency Graphs (SIGs) have been intensively used in software engineering for modeling non-functional requirements [11] [12]. For example, a framework for integrating non-functional requirements (NFRs) with functional ones in the use case model is proposed in [12]. In this framework, NFRs are represented as

“softgoals” to be “satisfied”. To determine satisficeability, design alternatives or decisions (called operationalizing softgoals) are considered; design tradeoffs are analyzed, design rationale is recorded and design choices are made. The entire process is recorded in a “Softgoal Interdependency Graph” (SIG) and then the selected design decisions (operationalizing softgoals) can be used as a framework for architecture and design [12].

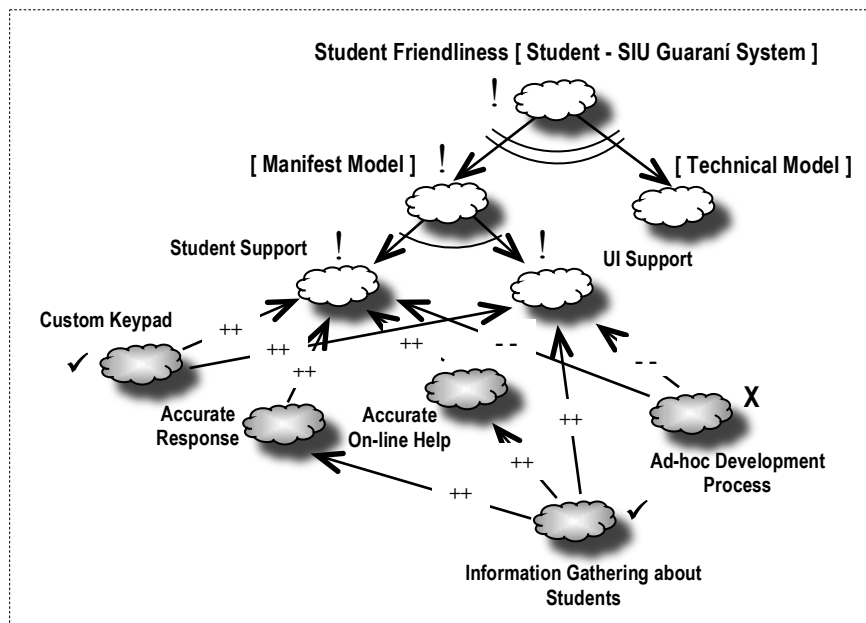


Figure 3.5: Softgoal Interdependency Graph (SIG) for Student Friendliness NFR

In Figure 3.5 we partially depict a SIG for the Student Friendliness softgoal in the context of the SIU Guarani registration system. The light cloud indicates an NFR softgoal, denoted with nomenclature Type[Topic] where Type is a non-functional aspect --e.g. Student Friendliness, and Topic is the context for the softgoal --e.g. a Student accessing the SIU Guarani registration system. Either Type or Topic of each NFR softgoals can be refined, one at a time, with either AND-decomposition (denoted with a single arc) or OR-decomposition (denoted with a double arc). For example, as shown in Figure 3.5, Student Friendliness[Student - SIU Guarani system] is OR-decomposed into Student Friendliness[Manifest Model] and Student Friendliness[Technical Model]. The manifest model is the UI model through which the software represents its functioning to the user and it is built around task, people and business objects; while the technical model is the model with which developers feel most comfortable and it is built around objects, method, algorithms and data structures [26].

Since student friendliness is the NFR under evaluation, the focus is on the Manifest Model token that is AND-decomposed into Student Support[Manifest Model] and UI Support [Manifest Model]. The dark cloud indicates an operationalizing softgoal. For example, in most development environments the developers agree on a basic framework and the UI is constructed in an ad-hoc manner when the screens are coded. This kind of practice has a highly negative contribution since a formal UI model is never constructed and this is the reason why in Figure 3.5, the operationalizing softgoal Ad-hoc Development Process is denied.

3.6 Web Content Accessibility Guidelines Documents

Since the WCAG has two documents (1.0 and 2.0), it is important to make clear at this point why we chose the 1.0 document. WCAG 1.0 has been used worldwide since 1999 as a reference material or cited as a normative from many other Accessibility documents in the world [34] [38] [40]. Many tools and approaches also have implemented it.

Although the WCAG 2.0 has been released in December 2008 and it is a fact that so far the rate of adoption has been relatively slow. For example, though it appears that within UK government departments there is a growing acceptance that websites under development should conform to WCAG 2.0, the official government policy still remains WCAG 1.0. As another example, in Germany, despite not using the WCAG, all public websites are beginning to use the usability regulation which incorporates WCAG 1.0 and migration of the Accessibility national guideline to WCAG 2.0 is just beginning; meanwhile in Spain, where any rule specified by legislation refers to a national standard based on WCAG 1.0, as far as we know, there is no regulation oriented toward WCAG 2.0 yet. Finally, since Section 508 [38] is undergoing a revision over the next couple of years [42], we have to wait approximately until 2011-2012 for the WCAG 2.0 to be harmonized into this Accessibility standard. At this point we emphasize that we are pre-supporting new issues addressed by W3C-WAI, but in light of how the migration of Accessibility regulations toward WCAG 2.0 is evolving, we think that the WCAG 2.0 is still in its infancy and therefore some time must pass before it is widespread adopted.

As we already mention in Section 2.1, the situation in Argentina is less developed, since Web Accessibility is an issue that has been recently included in the State's agenda. The

legislation 26.653 called “Guía de Accesibilidad para Sitios Web del Sector Público Nacional³⁷”, which adheres to WCAG 1.0 document, was approved by Resolution 69/2011 on June 27th 2011. In August 2011, Argentina became a member of the W3C³⁸. As argentine citizens committed with Accessibility, we have much expectation about this first steps towards an inclusive government Web for all.

In addition to the reasons stated above, we selected the WCAG 1.0 because it is a mature, committed to all possible Accessibility barriers and stable document version and part of a series of valuable and related Accessibility guidelines published by the W3C-WAI [50] with which WCAG 1.0 can be applied in conjunction. We revisit this discussion in Section 7.3.1 where we also provide some insights on how we upgraded our approach to WCAG 2.0 [46].

³⁷ Access to Public Information by Law 26.653 at

<http://www.infoleg.gov.ar/infolegInternet/anexos/175000-179999/175694/norma.htm>

³⁸ Argentina became a member of the W3C at <http://www.puntogov.com/nota.asp?nrc=2641>

4. AN APPROACH FOR ENGINEERING

ACCESSIBLE WEB APPLICATIONS

4.1 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.

The model we envisage to deal with Accessibility concerns within a Web engineering approach is illustrated in Figure 4.1. Columns in Figure 4.1 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 4.1, most arrows indicate an input or output, except for the UID and SIG diagrams as shown in Figure 4.1 (2.1) and (2.2), where the arrows are input/output. This is because there are cases 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. We revisit this issue in Chapter 5 and also in Chapter 6 where we also compare related work with ours indicating differences, advantages and drawbacks.

Firstly, we explain in general terms our approach to lead then to a detailed description of the proposed techniques for implementing our proposal step-by-step.

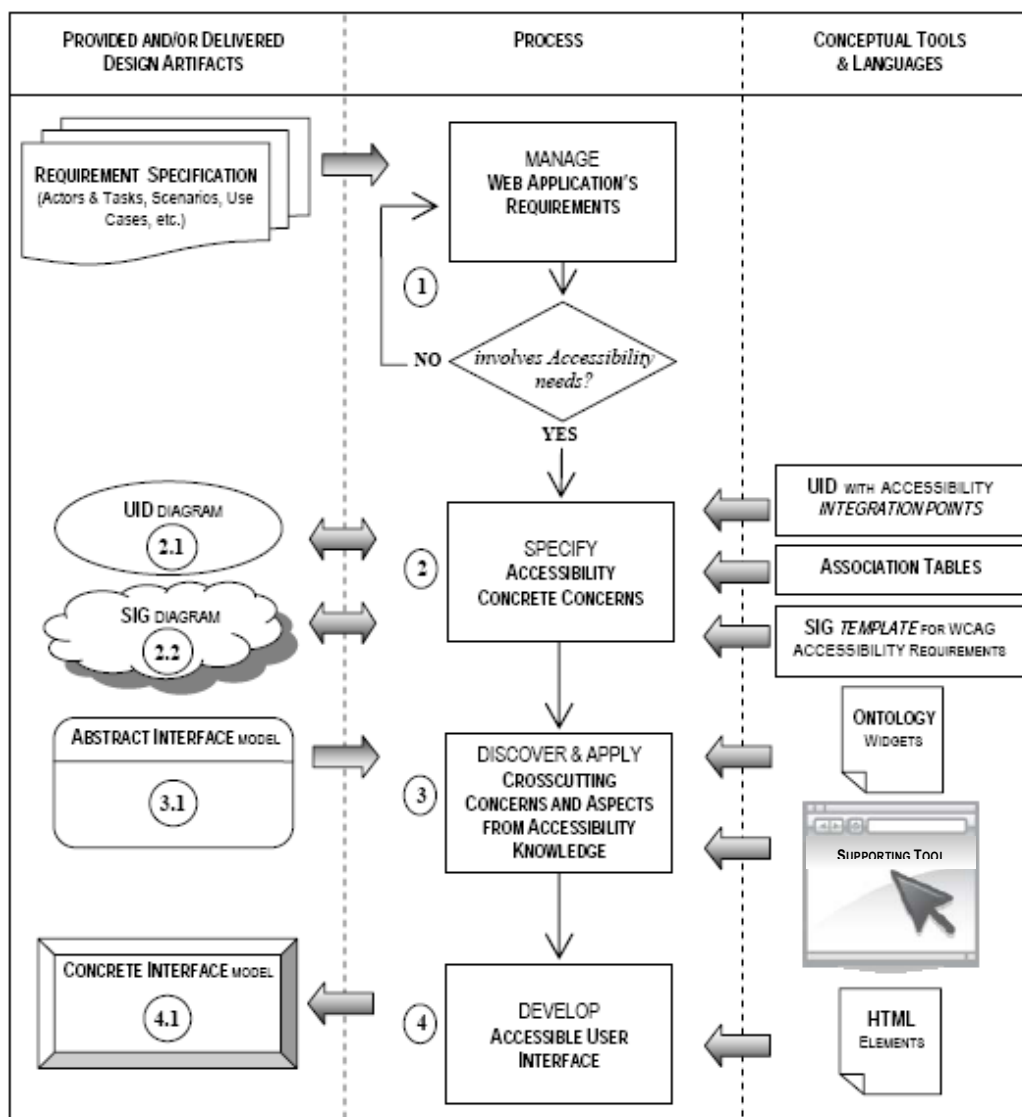


Figure 4.1: Overview of Our Approach

As highlighted in Figure 4.1 (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 barriers³⁹ finally show, so we are particularly interested in discovering Accessibility requirements at the user interface design. Then,

³⁹ Probably, the best-known definition of a barrier is the one given by Giorgio Brajnik at <http://users.dimi.uniud.it/~giorgio.brajnik/projects/bw/bw.html> <http://www.omg.org/mda/One>: “A barrier is any condition that hinders the user's progress towards achievement of a goal, when the user is a disabled person. A barrier is described in terms of: (i) the category of user and the type of disability, (ii) the type of assistive technology being used, (iii) the failure mode, that is the activity/task that is hindered and how it is hindered, and (iv) which features in the page raise the barrier.”

as shown in Figure 4.1 (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 4.1 (2.1) and (2.2) respectively. We propose these conceptual tools basically to allow the representation of Accessibility requirements while executing a user's task (the UID technique and the SIG model are described above in Sections 3.4 and 3.5 respectively). As indicated in Figure 4.1 (3), this Accessibility knowledge captured at early stages aids designers making decisions through the abstract interface model, as shown in Figure 4.1 (3.1), and then, as shown in Figure 4.1 (4) toward its implementation through the concrete interface model as shown in Figure 4.1 (4.1).

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 4.1 (3.1) and (4.1) respectively), to ensure an accessible user experience.

In the following Sections, we put all the pieces together to give a detailed step-by-step explanation of our Aspect-Oriented approach.

4.2 Identifying Application's Requirements that Involve Accessibility Needs

There is nothing new in saying that requirements are essential to create a model of the most relevant functional and non-functional application's concerns before writing one line of code. This is why any WE approach uses an explicit development activity for requirements gathering and specification. Most of these approaches apply some combination of UML⁴⁰ object-oriented techniques, like actors and tasks, scenarios, use

⁴⁰ OMG-UML: The Unified Modeling Language at <http://www.uml.org/>

cases, etc., to capture Web application’s requirements and deliver a model for handling complexity into parts. Since we are particularly interested in discovering Accessibility concerns at the user interface design, we propose as a first step, an iterative and incremental process over these Web application’s requirements looking specially those that involve user-system interaction but also those derived from all kind of user activity with the application’s interface. As an example, assume that we take into account the following use case “*Login a Student given the Student’s ID and Password*”:

Use Case 1: Login a Student given the Student’s ID and Password	
Brief Description: This use case describes how a Student logs into the SUI Guaraní registration system.	
Success End Condition: The Student is now logged into the system.	
Primary Actor: Student	
Description	
Main Success Scenario:	
Step	Action
1.	The system requests that the Student enter his/her ID and Password.
2.	The Student enters his/her ID and Password.
3.	The system validates the entered ID and Password and logs the Student into the system.
Extensions:	
Step	Branching Action
3.a	The Student enters an invalid ID and/or Password, the system displays an error message, the use case ends.

This use case describes the application’s requirements for the online student’s login Web page example (introduced in Section 1.1 by Figure 1.1). The functionality required for the online login involves user-system interaction, since at **Step 1** of the main success scenario, the student is requested by the system to enter his/her ID and password. At the registration system, **Step 2** is satisfied when the student enters its identity card number as an ID and a four-digit key as a password. Then at **Step 3** the system executes the validation process yielding the student logged into the system as a success end condition or displaying an error message to end the use case. This identification process is defined as **Step 1** and is graphically represented by (1) in Figure 4.1.

4.3 Specifying Accessibility Concrete Concerns

After requirements' identification in **Step 1** and because of the reasons related to Accessibility features and its relevance to the success of the Web, explained in Section 1.1 and Section 2.1, we propose the early capture of Accessibility concrete concerns that involve user interactions and activities with the application's interface. Mostly because of the non-functional, generic and crosscutting nature of Accessibility concerns of a user-system interaction, we developed two conceptual tools as extensions of the UID and SIG techniques (introduced earlier in Section 3.4 and 3.5 respectively): the UID technique with *integration points* and SIG *templates* for Accessibility.

As an example, let us return to the use case “*Login a Student given the Student's ID and Password*” in Section 4.2 and consider a scenario in which a blind student using an older “screen reader” device wishes to log into the registration system. The picture is easy to catch, just think about this student trying to deal with the online login Web page. It is a fact that Accessibility concerns related to the user layout and the user technology support must be considered to help blind student's interaction and browsing regardless of its assistive device. Specifically, in this case it means that the HTML elements required for the identification form must be accessible for students using “screen readers”. So, when developing the functional requirements captured by the use case, we need a way to record Accessibility concerns early and as a reminder for design. With this aim in mind we developed the UID technique with *integration points* and SIG *template* for Accessibility.

Following, in Sections 4.3.1 and 4.3.2, we describe these conceptual tools and we show how they work together to encourage the specification of Accessibility concrete concerns at **Step 2**.

4.3.1 Using UIDs with Integration Points Technique

For each application's requirement identified at **Step 1**, and at **Step 2** (graphically represented by (2) in Figure 4.1), we firstly develop an UID diagram focusing mainly on outlining *integration points* where Accessibility is crucial for ensuring a successful information exchange between the application and the user.

With the traditional perspective given by techniques like [11][12] in mind (depicted in Section 3.4), 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:

- **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. Then, choosing the appropriate granularity and selecting a U-UI or U-UIc *integration point* allow a better mapping of the elements composing the user interface design.

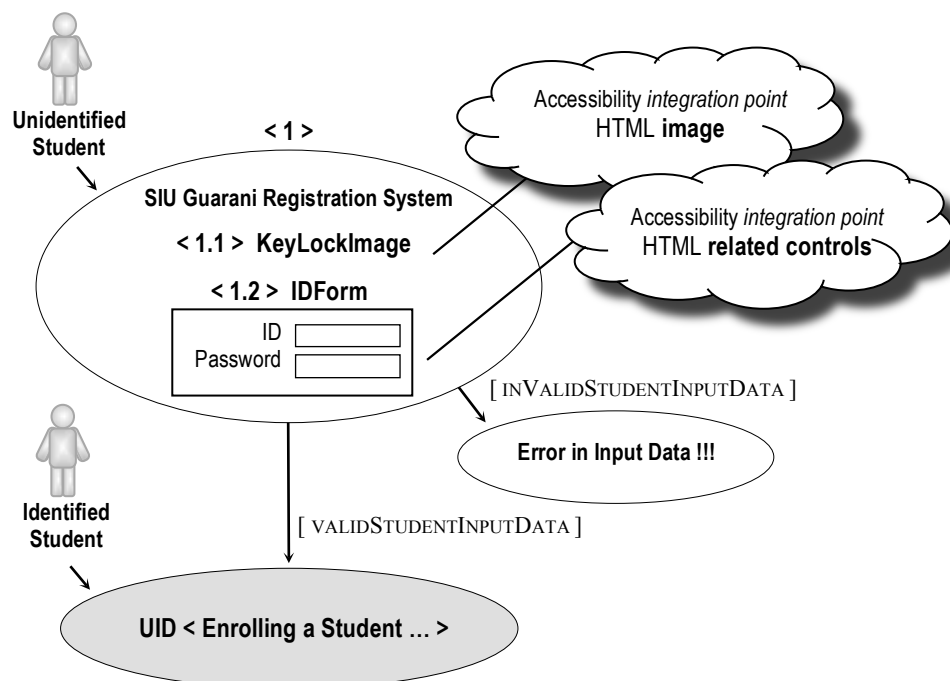


Figure 4.2: UID with Accessibility integration points: Login a Student given the Student's ID and Password

Figure 4.2 shows the resultant UID, corresponding to the use case “*Login a Student given the Student’s ID and Password*” (presented in Section 4.2), 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 (introduced in Section 1.1 by Figure 1.1). As we can see in the example shown in Figure 4.2, 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 enable the access for all the students.

The development of the UID diagram with *integration points* at **Step 2** is graphically represented by (2.1) in Figure 4.1.

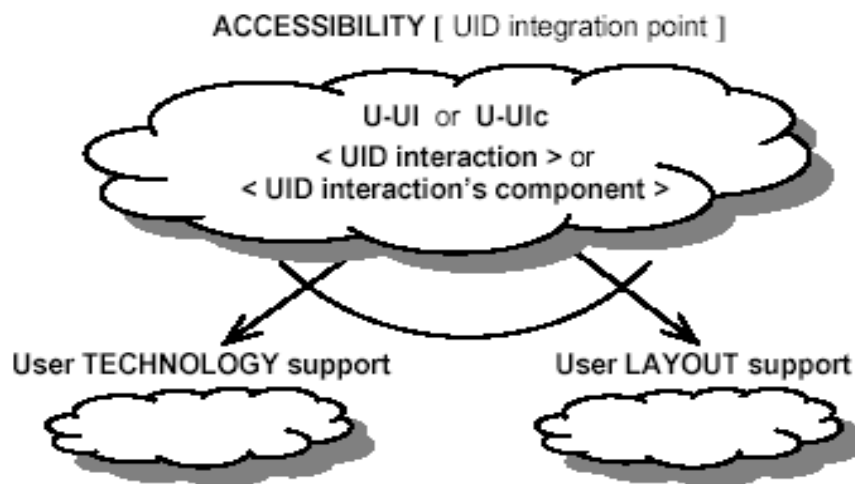


Figure 4.3: SIG Template for Accessibility

4.3.2 Applying the SIG Template

After specifying the Accessibility *integration points* of the UID diagrams at **Step 2**, we develop a SIG diagram for WCAG 1.0 Accessibility requirements. To do so, we take into consideration proposals from the user interface design literature [27][36] introduced in Section 3.3 as follows.

We have already seen that the dialogue class is directly represented by UIDs since they help in modeling the content and the sequence of the information exchange between the user and the system during navigation. However, presentation and pragmatic classes are

relevant too, so we propose considering the three classes --i.e. dialogue, presentation and pragmatic, when drawing a SIG for Accessibility.

Figure 4.3 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 enable 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 content⁴¹. 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 [48]; meanwhile, they

⁴¹ This last statement is compliant with the WCAG glossary that establishes three basic topics that compose an Internet document: (i) the presentation --i.e. how the document is rendered?, (ii) the structure -- i.e. how the document is organized logically?, and (iii) the content --i.e. what the document communicates to the user?

are associated with the three classes (dialogue, presentation and pragmatic) if they are hardware-dependent.

For example, returning to Figure 4.2, we establish the Accessibility softgoal for the interaction's components <1.1> KeyLockImage and <1.2> IDForm to support 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 gathering Accessibility concerns (shown in Figure 4.3) we work with the W3C-WAI WCAG 1.0 guidelines [45] as follows.

To facilitate this instantiation process of the SIG *template* we establish an *association table* for groups of related HTML elements. The instantiation process of the SIG *template* is conducted as a refinement process over the SIG tree using these *association tables* as a reference. For example, Table 4.1 introduces the *association table* that we have developed for the HTML *control* group. Basically, these *association tables* have the tasks of linking each ontology concept --i.e. abstract widget, with their respective HTML elements --i.e. concrete widgets, and with the Accessibility concerns prescribed for those widgets by the WCAG 1.0 checkpoints. It is important to clarify that we use "HTML elements" as a general term, including HTML elements and attributes, as well as embedded, internal and external objects like scripts, applets, style sheets, etc. This means, that the allusion to "HTML elements" is extensive to include all the possible widgets that may exist at a concrete user interface.

We will give a deeper explanation of the function of these *association tables* in Section 4.5.2 and since these *association tables* are developed for groups of related HTML elements, we also provide in Section 4.5.1 our own classification by mapping the ontology concepts (abstract widgets) onto five groups of HTML elements (concrete widgets).

Table 4.1: Association Table for the HTML Control Elements Group

ASPECT	ONTOLOGY WIDGETS (ABSTRACT WIDGETS)	HTML ELEMENTS (CONCRETE WIDGETS)		WCAG 1.0 CHECKPOINTS AND THEIR PRIORITIES: [1][2] OR [3]					DESIGN DECISION CLASS related to USER-APPLICATION INTERACTION	
				9.4	10.2	10.4	12.3	12.4		
				9.5 [3]	[2]	[3]	[2]	[2]		
									DIALOG (D) PRESENTATION (P) PRAGMATIC ✘	
I. TSCONTROL SIG's USER TECHNOLOGY SUPPORT BRANCH	INDEFINITEVARIABLE	TEXT FIELD	INPUT TEXT...	✓	✓	✓				
		TEXT AREA	TEXTAREA...	✓	✓	✓				
		RELATED CONTROLS	FIELDSET...		✓	✓				
	PREDEFINEDVARIABLE MULTIPLECHOICES	CHECK BOX	INPUT CHECKBOX...	✓	✓	M				
		MULTIPLE OPTION MENU	SELECT MULTIPLE...	✓	✓	✓				
		RELATED OPTIONS	OPTGROUP...		✓	✓				
	PREDEFINEDVARIABLE SINGLECHOICES	RADIO BUTTON	INPUT RADIO...	✓	✓	M				
		SIMPLE OPTION MENU	SELECT...	✓	✓	✓				
	II. LSCONTROL SIG's USER LAYOUT SUPPORT BRANCH	INDEFINITEVARIABLE	TEXT FIELD	INPUT TEXT...				✓	✓	
TEXT AREA			TEXTAREA...				✓	✓		
RELATED CONTROLS			FIELDSET...				✓	✓		
PREDEFINEDVARIABLE MULTIPLECHOICES		CHECK BOX	INPUT CHECKBOX...				✓	✓		
		MULTIPLE OPTION MENU	SELECT MULTIPLE ...				✓	✓		
		RELATED OPTIONS	OPTGROUP...				✓	✓		
PREDEFINEDVARIABLE SINGLECHOICES		RADIO BUTTON	INPUT RADIO...				✓	✓		
		SIMPLE OPTION MENU	SELECT...				✓	✓		

The development of the SIG diagram at **Step 2** is graphically represented by (2.2) in Figure 4.1.

4.4 Discovering Crosscutting and Applying Aspects

The activity of discovering Accessibility crosscutting concerns and applying Accessibility aspects properly at the user interface design is defined as **Step 3**.

We exploit the Accessibility knowledge captured by SIG diagrams built at the user

interface design activity (**Step 2**) to find out how WCAG 1.0 Accessibility concerns “crosscut” interface widgets. To achieve this, managing crosscutting in an aspect-oriented manner, we use again our *association tables* introduced in Section 4.3.2. As we said before, we will give a deeper explanation of the function of these *association tables* in Section 4.5.2.

Let us return again to the use case “*Login a Student given the Student’s ID and Password*” in Section 4.2, whose UID with Accessibility *integration points* is shown by Figure 4.2 in Section 4.3.1. The purpose at **Step 3** is to find out how WCAG 1.0 Accessibility concerns “crosscut” interface widgets required for the online login Web page, aided by the abstract interface model shown in Figure 4.1 (3.1). More specifically, the SIG diagrams and the *association tables* work together to discover the required WCAG 1.0 checkpoints for helping the student’s login but also to show how aspect-oriented “symptoms” (“scattering” and/or “tangling”) manifest their crosscutting nature on the HTML *image* and HTML *related control* elements. For example, and as we will see in-depth later, from guideline 10 responding to the statement “use interim⁴² solutions”, satisfying the 10.4 checkpoint is a “mandatory” goal (set with an “M”) or required for every HTML control element, and establishes that empty controls must be handled correctly until “user agents”. So, to ensure this Accessibility requirement, the checkpoint 10.4 will be “scatered” at the login Web page of the registration system every time that an HTML *text field* element (corresponding to an *IndefiniteVariable* widget) is present. It is important to highlight that ensuring compliance to Accessibility is, in several cases, similar for those HTML elements sharing the same HTML group. As we can see on Table 4.1, this is the case for the HTML *control* group. For those cases where these minor differences exist, the aspect-oriented paradigm provides key mechanisms to save distances smoothly --e.g. a variation in the application of the aspect by an aspect instantiation or by the way the “advice” (aspect functionality code) and “pointcut” (aspect applicability code) are specified.

⁴² Interim is used by the W3C as a temporary recommendation to ensure that while assistive technologies and older browsers exist they will operate correctly.

4.5 Designing with Accessible Interface Widgets

The development of an accessible user interface design is defined as **Step 4** and is graphically represented by (4) in Figure 4.1, while the corresponding abstract and concrete models are graphically represented by (3.1) and (4.1) respectively.

Having already completed the step-by-step description of our approach, we introduce now our classification of HTML elements and we also give an explanation of the *association tables* (used at **Step 2** and **Step 3**). We decided to introduce these conceptual tools in Section 4.5.1 and Section 4.5.2 respectively, since both are closely related to interface widgets issues.

4.5.1 A Mapping between Ontology Concepts and HTML Elements

Taking into account the Abstract Widget Ontology [36] described in Section 4.3, we map the ontology concepts onto HTML elements. We have materialized this mapping using UML class diagrams to explain the relationships between each abstract interface widget presented by the ontology concepts, and the concrete interface widget in HTML elements. Figure 4.4 shows the UML class diagram for the ontology concept *VariableCapture*, particularly for the ontology (sub) concepts *IndefiniteVariable*, *PredefinedVariable-SingleChoice* and *PredefinedVariable-MultipleChoice*. The ontology concept *CaptureVariable*, whose functionality is to capture the value of one or more variables, is implemented in HTML by *control* elements. HTML *control* elements can be grouped together in a form --i.e. an HTML *related controls* element, which is a possible implementation of the ontology concept *CompositeInterfaceElement*. Users interact with a form through HTML *related controls* by modifying their values before submitting the form to an agent, like a Web server or a mail server, for processing. Returning to the example of the login Web page for the student's login, the abstract interface model usually requests two *IndefiniteVariable* widgets of the *VariableCapture* type. A *CompositeInterfaceElement* groups together these two widgets required for receiving the user's identification and password login values respectively. On the other hand, the concrete interface model for the same login Web page maps these concepts on two HTML *text field* widgets of the *control* type. An HTML *related controls* element

groups together these two widgets, which allow entering the text strings typed by the user with previously unknown user's name and password values.

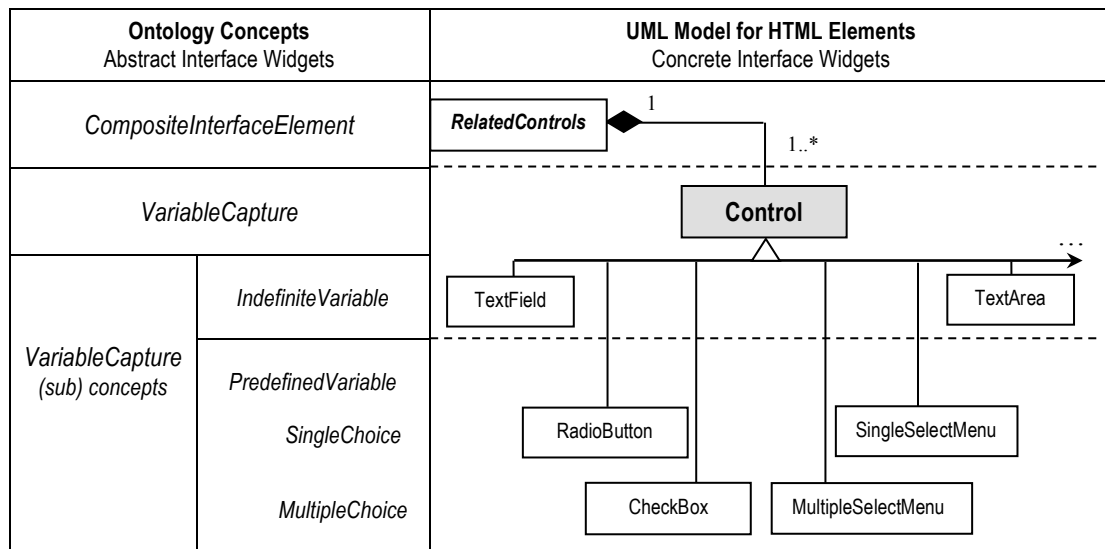


Figure 4.4: Mapping between some Ontology Concepts and HTML Elements

In this way, we map the ontology concepts onto five groups of HTML elements as follow:

- The *VariableCapture* maps onto the HTML *control* elements group, as we shown in Figure 4.4;
- The *SimpleActivator*, which is capable of reacting to external events such as mouse clicking, maps onto HTML *link* and *button* elements group;
- The *ElementExhibitor*, which is able to exhibit different types of content, such as text, images or applets, maps onto HTML *text* and *non-text* elements group;
- The *LogicalStructuring*, which is able to logically organize the HTML content of the document, maps onto the HTML *structural* elements group; and
- The *ElementStyling*, that is able to display the content with a certain appearance, maps onto *frame* and *style sheet* elements group.

As shown in Figure 4.1, only three of these five groups are characterized by their respective classes in the original abstract widget ontology [36]. Figure 4.5 shows how we have extended this ontology with the *LogicalStructuring* and *ElementStyling* widget classes in order to provide wider support to concrete widgets required by current user

interfaces, which are dynamic and with a high degree of complexity. The *LogicalStructuring* class, groups structural widgets to define how the content is organized logically, for example, with different levels of headers, by chapter, with an introduction and table of contents, etc. While the *ElementStyling* class, groups presentation widgets to define how the content is rendered, for example, as print, as a two-dimensional graphical presentation, as a text-only presentation, as synthesized speech, etc.

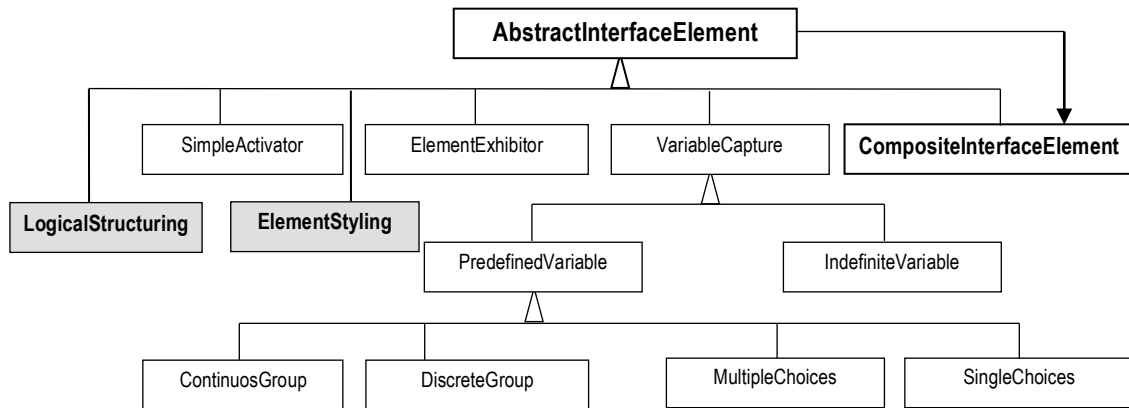


Figure 4.5: Extended Abstract Widget Ontology

Since most of the HTML elements are composed by other HTML elements, an accessible HTML element requires the Accessibility of all its components. So a deeper look about HTML elements composition is required to work properly with Accessibility issues. Figure 4.6 explains HTML elements composition providing a more detailed description of the HTML *control* elements: *text field* and *text area*; *radio button* and *single option menu*; and *check box* and *multiple option menu* (see Figures 4.6 (a), 4.6 (b) and 4.6 (c) respectively).

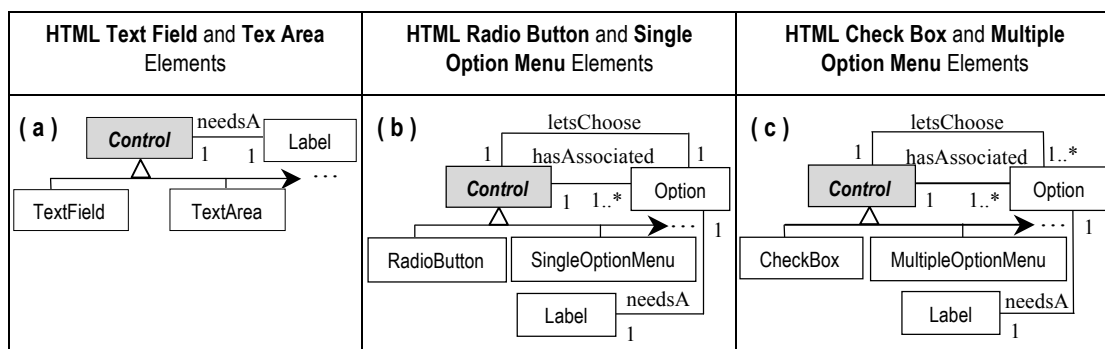


Figure 4.6: UML Model for HTML Control Elements

For example, the *label* is a very important element to achieve the goal of making a form --i.e. HTML *related controls* element, accessible, because, if used correctly, it can provide helpful support to people with disabilities. The WCAG 1.0 is very clear about the Accessibility role of the *label* element when developing an HTML *related controls* element. Specifically, the document provides two checkpoints, one related to the user layout support and the other to the user technology support --i.e. precisely the two initial branches of our SIG *template* for Accessibility, to be consider when “labeling” HTML *control* elements that are associated into a form --i.e. HTML *related controls* element.

4.5.2 Association between Ontology Concepts-HTML Elements-WCAG Checkpoints

To develop and exploit the SIG diagrams for managing crosscutting in an aspect-oriented manner, we establish five *association tables*, one for each group of HTML elements defined in Section 4.5.1: (i) the HTML *control* group as we shown in Figure 4.4 and Figure 4.6; (ii) the HTML *link* and *button* group; and (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. This is possible because we find out that ensuring compliance to Accessibility is in several cases very similar for those interface widgets that share the same HTML group. That is, ensuring Accessibility does not normally differ much between interface widgets that share the same group, and for those cases the aspect-oriented paradigm provides key mechanisms to save these distances smoothly --e.g. a variation in the application of the aspect by an aspect instantiation or by the way the “advice” and “pointcut” are specified. As we said before, Table 4.1 introduces the *association table* for the HTML *control* group. A checkpoint cell for a specific interface

widget is selected only when the HTML element requires considering the Accessibility by the checkpoint. As we can see in Table 4.1, this *association table* also indicates each checkpoint priority level assigned by the WCAG 1.0 [45]: (i) [Priority 1] checkpoints that “must” be satisfied, (ii) [Priority2] checkpoints that “should” be satisfied and, (iii) [Priority 3] checkpoints that “may” be satisfied. This information allows interface designers to keep in mind the impact of the Accessibility barrier when not satisfying each checkpoint. When a checkpoint cell is signed as “M” it means “mandatory” and the HTML element implementation for the interface widget helps by itself compliance to the checkpoint. To address Accessibility of the HTML *related controls*, guidelines 9, 10 and 12 deal with the question of what to do to make a form accessible [41][45][47].

On Table 4.1, Aspect I called “TSCControl” evaluates control’s widgets Accessibility to improve user’s current and earlier assistive devices and technologies; it is further supported by softgoals to be satisfied at the SIG’s user technology support branch.

The association between Accessibility softgoal concerns (represented by the WCAG 1.0 checkpoints and their priorities) and the design decision classes is showed in the table with a "P" for the presentation class, a "D" for the dialog class and by the "@" symbol for the pragmatic class. Here we must remember that to associate the three design decision classes --i.e. dialog, presentation and pragmatic, with the Accessibility softgoal concerns at the user technology support SIG’s branch, we take into account the considerations described in Section 4.3.2. Over this branch, satisfying checkpoints 9.4 and 9.5 responding to the statement “design for device-independence” of guideline 9 and, checkpoints 10.2 and 10.4 responding to the statement “use interim solutions” of guideline 10, are goals required for every HTML *control* element. The checkpoint 9.4 establishes that we should “create a logical tab order through links, form controls, and objects [Priority 3]” [45]. While the checkpoint 9.5 establishes that we should “provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls [Priority 3]” [45]. Checkpoints 9.4 and 9.5 are goals required for all the HTML *control* elements and are focused on providing alternative access by tabbing navigation or access keys to HTML *related controls* helping device- independency. This is important because it means that the user may interact with the “user agent” or document with a preferred input (or output) device -- e.g. mouse, keyboard, voice, head wand, or others [45]. If, for example, an HTML

control element can only be activated with a mouse or other pointing device, someone who is using the page without sight, with voice input, or with a keyboard or who is using some other non- pointing input device will not be able to use the form --i.e. people with motor, visual or cognitive disabilities who need these special devices to access the Web.

The checkpoint 10.2 establishes that “until user agents support explicit associations between labels and form control, for all form control with implicitly associated labels, ensure that the label is properly positioned [Priority 2]” [45]. While the checkpoint 10.4 establishes that “until user agents handle empty controls correctly, include default, place- holding characters in edit boxes and text areas [Priority 3]” [45]. Checkpoints 10.4 is a goal not required for HTML *checkBox* and *radioButton* elements since they have an obligatory attribute that specifies the initial value of the *control* element.

On Table 4.1, Aspect II called “LSControl” evaluates control’s widgets Accessibility to improve user’s interface issues, and it is supported by softgoals to be satisfied at the SIG’s user layout support branch. Here, we must highlight again that to associate the three design decision classes --i.e. dialog, presentation and pragmatic, with the Accessibility softgoal concerns at the user layout support SIG’s branch, we take into account the considerations described in Section 4.3.2. Over this branch, satisfying checkpoints 12.3 and 12.4 responding to the statement “provide context and orientation information” of guideline 12 are goals required for all the HTML *control* elements. The checkpoint 12.4 establishes that “associate labels explicitly with their controls [Priority 2]” [45]. While, checkpoint 12.3 establishes “divide large blocks of information into more manageable groups where natural and appropriated [Priority 2]” [45]. Checkpoints 10.3 and 10.4 are goals required for all the HTML *control* elements and are focused on providing context and orientation information to help users understand complex pages or HTML elements. For example, complex relationships between HTML *control* elements as parts of a form on a Web page may be difficult for people with cognitive disabilities and people with visual disabilities to interpret.

Similarly, to Table 4.1, we developed Tables to describe the rest of the four groups of HTML elements. Following, we include Tables 4.2, 4.3, 4.4 and 4.5 for the groups of HTML *link* and *button*, the HTML *text* and *non-text*, the HTML *structural* and, the

HTML *frame* and *style sheet* elements, respectively.

These five *association tables* cover thirteen out of the fourteen guidelines composing the WCAG 1.0 document [45]. Only guideline 11 (and its checkpoints 11.1, 11.2, 11.3 and 11.4) corresponding to the statement “use W3C technologies and guidelines” is not included in these *association tables* because this guideline is not required for specific HTML elements. They remind developers using W3C technologies (e.g., HTML, CSS, etc.) wherever possible because of the following reasons: (i) W3C technologies include "built-in" Accessibility features, (ii) W3C specifications undergo early review to ensure that Accessibility issues are considered during the design phase, and (iii) W3C specifications are developed in an open, industry consensus process. So, since checkpoints from guideline 11 provide generic recommendations for HTML documents, they cannot be associated to specific elements of any HTML group.

For a deeper understanding of our proposal, in Chapter 5, we illustrate with a complete case study, which we developed around the function for student’s login, shown in Section 1.1 by Figure 1.1, corresponding to a typical student’s registration system, such as the SIU Guarani registration system that we already mention.

Table 4.2: Association Table for the HTML Link and Button Elements Group

ASPECT	ONTOLOGY WIDGETS (ABSTRACT WIDGETS)	HTML ELEMENTS (CONCRETE WIDGETS)		WCAG 1.0 CHECKPOINTS AND THEIR PRIORITIES: [1][2] OR [3]										DESIGN DECISION CLASS related to USER-APPLICATION INTERACTION	
				1.2	1.5	9.1	9.4 9.5	10.5	13.4	13.5	13.6	1.1	13.1		
				[1]	[3]	[1]	[3]	[3]	[2]	[3]	[3]	[1]	[2]		
				D-P	D-P	D-P	D-P	D-P	D-P	D-P	P	D-P	DIALOG (D) PRESENTATION (P) PRAGMATIC ✘		
I. TSLINK&BUTTON SIG's USER TECHNOLOGY SUPPORT BRANCH	SIMPLEACTIVATOR	LINK	A HREF ...					✓							
		IMAGE LINK	A HREF, IMG, SRC...					✓							
		RELATED LINKS	A HREF, MAP...					✓	✓	✓	✓	✓			
		IMAGE MAP	SERVER-SIDE	A HREF, IMG, SRC, ISMAP...	✓		✓	✓	✓	✓	✓	✓			
			CLIENT-SIDE	IMG, SRC, USEMAP, MAP, AREA...		✓	✓	✓	✓	✓	✓	✓			
		PUSH BUTTON	INPUT SUBMIT, INPUT RESET...					✓							
		GENERALIZED & IMAGE BUTTON	BUTTON, INPUT IMAGE, IMG, SRC...					✓							
II. LSLINK&BUTTON SIG's USER LAYOUT SUPPORT BRANCH	SIMPLEACTIVATOR	LINK	A HREF...										✓		
		IMAGE LINK	A HREF, IMG, SRC...									✓	✓		
		RELATED LINKS	A HREF, MAP...										✓		
		IMAGE MAP	SERVER-SIDE	A HREF, IMG, SRC, ISMAP...									✓	✓	
			CLIENT-SIDE	IMG, SRC, USEMAP, MAP, AREA...									✓	✓	
		PUSH BUTTON	INPUT SUBMIT, INPUT RESET...												
		GENERALIZED & IMAGE BUTTON	BUTTON, INPUT IMAGE, IMG, SRC...										✓		

Table 4.3: Association Table for the HTML Text and Non-Text Elements Group

ASPECT	ONTOLOGY WIDGETS (ABSTRACT WIDGETS)	HTML ELEMENTS (CONCRETE WIDGETS)		WCAG 1.0 CHECKPOINTS AND THEIR PRIORITIES: [1][2] OR [3]																		DESIGN DECISION CLASS related to USER-APPLICATION INTERACTION		
				1.3	1.4	6.4	7.1	7.2	7.3	7.4	8.1	9.2	9.3	10.1	13.10	14.2	1.1	2.1	2.2	4.1	4.2		6.3	14.1
				[1]	[1]	[2]	[1]	[2]	[2]	[2]	[1]	[2]	[2]	[2]	[3]	[3]	[1]	[1]	[2]	[1]	[3]		[1]	[1]
				D-P	D-P	P	D-P	D-P	D-P	D	P	D-P	D-P	D-P	P	P	P	P	P	D-P	P			
				✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗			
I. TSText&NonText SIG's USER TECHNOLOGY SUPPORT BRANCH	ELEMENTEXHIBITOR	TEXT	ALT, TITLE, LONGDESC...												✓									
		NON-TEXT, MULTIMEDIA & PROGRAMMATIC OBJECTS	IMAGE	IMG, SRC...																				
			GRAPHIC	ASCII ART...										✓										
			AUDIO, VIDEO	OBJETC , PARAM ...	✓	✓						✓												
			APPLET, SCRIPT, NONSCRIPT...			✓	✓	✓	✓	✓	✓	✓	✓	✓										
II. LSText&NonText SIG's USER LAYOUT SUPPORT BRANCH	ELEMENTEXHIBITOR	TEXT	ALT, TITLE, LONGDESC...														✓	✓	✓	✓		✓		
		NON-TEXT, MULTIMEDIA & PROGRAMMATIC OBJECTS	IMAGE	IMG, SRC...													✓	✓	✓					
			GRAPHIC	ASCII ART...													✓							
			AUDIO, VIDEO	OBJETC , PARAM ...													✓							
			APPLET, SCRIPT, NONSCRIPT...														✓					✓		

Table 4.4: Association Table for the HTML Structural Elements Group

ASPECT	ONTOLOGY WIDGETS (ABSTRACT WIDGETS)	HTML ELEMENTS (CONCRETE WIDGETS)		WCAG 1.0 CHECKPOINTS AND THEIR PRIORITIES: [1][2] OR [3]																DESIGN DECISION CLASS related to USER-APPLICATION INTERACTION		
				10.3	13.2	13.3	3.1	3.2	3.5	3.6	3.7	4.3	5.1	5.2	5.3	5.4	5.5	5.6	13.7		13.8	13.9
				[3]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[3]	[1]	[1]	[2]	[2]	[3]	[3]	[3]		[3]	[3]
				P	D-P ✘	D-P ✘	P	P	P	P	P	P	P	P	P	P	D-P	D-P	D-P			
I. TSSTRUCTURAL SIG's USER TECHNOLOGY SUPPORT BRANCH	LOGICALSTRUCTURING	VALIDATOR & PROVIDER	HTML, HEAD, BODY, TITLE, DOCTYPE...		✓																	
		IN GRID	TABLE	✓		✓																
		GENERIC	DIV, SPAN...																			
		HEADING	H1-H6																			
		BLOCK	HEADER, MAIN, FOOTER, BLOCKQUOTE, ADDRESS...		✓																	
		IN SET	LIST			✓																
II. LSSTRUCTURAL SIG's USER LAYOUT SUPPORT BRANCH	LOGICALSTRUCTURING	VALIDATOR & PROVIDER	HTML, HEAD, BODY, TITLE, DOCTYPE...					✓			✓						✓		✓			
		IN GRID	TABLE				✓				✓	✓	✓	✓	✓	✓		✓				
		GENERIC	DIV, SPAN...				✓												✓			
		HEADING	H1-H6				✓		✓										✓			
		BLOCK	HEADER, MAIN, FOOTER, BLOCKQUOTE, ADDRESS...				✓			✓									✓			
		IN SET	LIST				✓			✓									✓			

Table 4.5: Association Table for the HTML Frame and Style Sheet Elements Group

ASPECT	ONTOLOGY WIDGETS (ABSTRACT WIDGETS)	HTML ELEMENTS (CONCRETE WIDGETS)		WCAG 1.0 CHECKPOINTS AND THEIR PRIORITIES: [1][2] OR [3]								DESIGN DECISION CLASS related to USER-APPLICATION INTERACTION		
				6.5	10.1	1.1	3.3	3.4	6.1	6.2	12.1		12.2	14.3
				[2]	[2]	[1]	[2]	[2]	[1]	[1]	[1]		[2]	[3]
											DIALOG (D) PRESENTATION (P) PRAGMATIC ✘			
I. TSFRAME&STYLESHEET SIG's USER TECHNOLOGY SUPPORT BRANCH	ELEMENTSTYLING	FORMATTING & POSITIONING	CSS, STYLE, STYLESHEET, LINK REL...											
		SECTIONING & SUBSPACES	FRAME, FRAMESET, NOFRAME, IFRAME...	✓	✓									
II. LSFRAME&STYLESHEET SIG's USER LAYOUT SUPPORT BRANCH	ELEMENTSTYLING	FORMATTING & POSITIONING	CSS, STYLE, STYLESHEET, LINK REL...				✓	✓	✓				✓	
		SECTIONING & SUBSPACES	FRAME, FRAMESET, NOFRAME, IFRAME...			✓		✓		✓	✓	✓	✓	

5. APPLYING OUR PROPOSAL

5.1 A Case Study

The SIU Guaraní student registration system is been used by a number of public universities in Argentina. It offers online information and/or diverse registration functionalities to their students. Since these kind of online systems give support to an educational organization, Accessibility is a main factor for all users but plays a key role for students with disabilities. In the spirit of such systems, we define the case study to apply our Aspect-Oriented approach, reusing the Student's login and the University home page examples, shown in Figures 1.1 and 2.1, respectively.

As Figure 5.1 shows, we propose a case study of 3 (three) level-deep navigation and 2 (two) optional anchors to get some help for data inputs ID and Password at the login Web page. The first level, shown in Figure 5.1 (a), is the student's University home page where the student selects the link to his/her respective Faculty site from a group of consecutive and related links. We highlight that we have already presented and explained this page example in Section 2.2.1 (as shown in Figure 2.1), since it is the one used to exemplify the related work. The second level, shown in Figure 5.1 (b), is the student's Faculty page that provides information about this institution among other functionalities and, offers a link to the SIU Guaraní student registration system. Finally, the third level, shown in Figure 5.1 (c), is the student's login page example, which we also have already presented and described in Section 1.1 (as shown in Figure 1.1) and then in Section 4.2 by the use case "*Login a Student given the Student's ID and Password*". From this third level, the student has the ability to browse for getting help to ID and/or Password if he/she fails to login to the system. These two pages, shown in Figure 5.1 (d), provide students with some helpful information and the chance to return to the login Web page.

To carry out the implementation of our approach clearly, in Section 5.2 we follow the step-by-step process as we described in Chapter 4 and depicted in Figure 4.1.

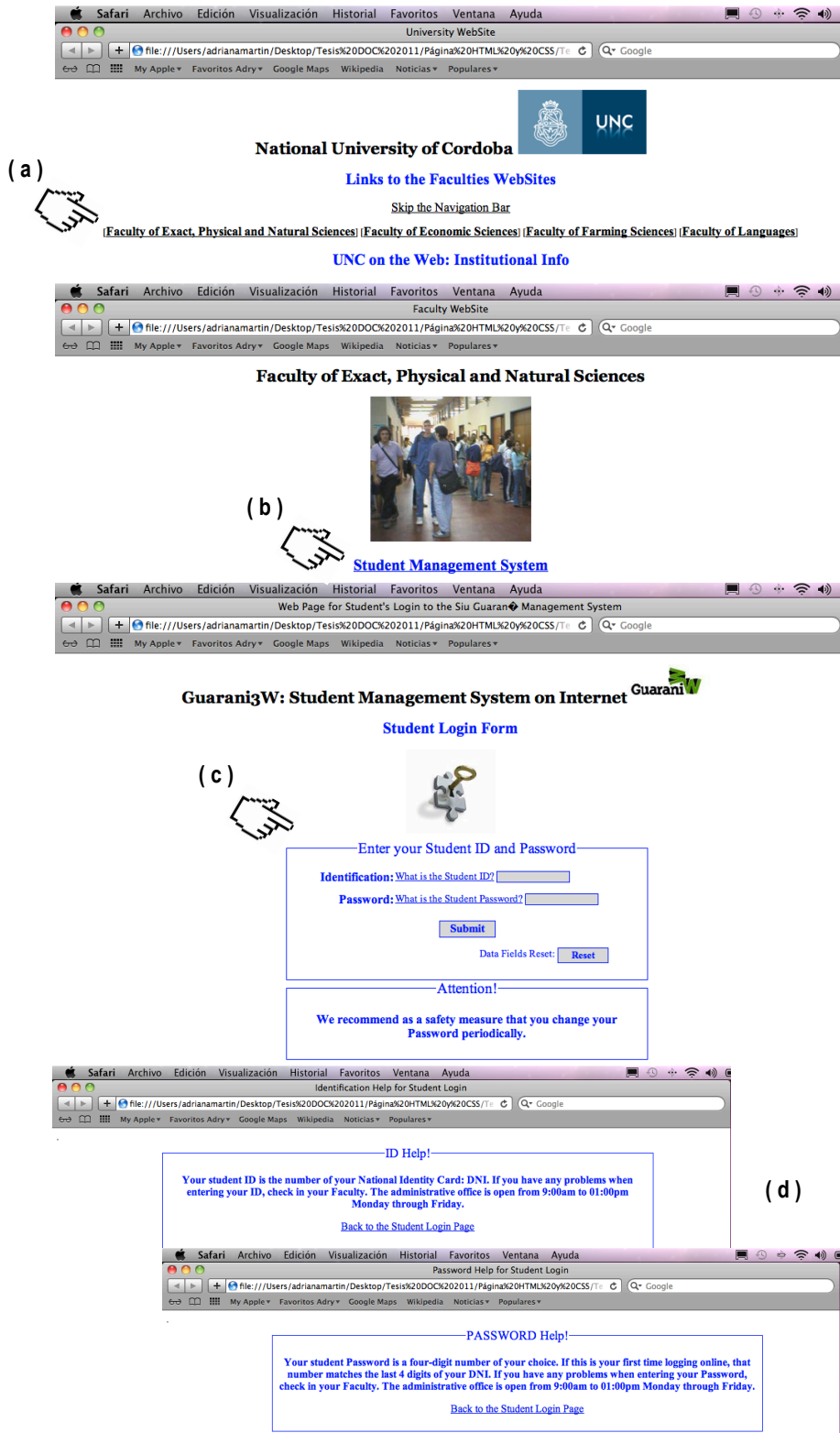


Figure 5.1: A Case Study

5.2 Our Proposal Step-by-Step on the Field

STEP 1. As highlighted in Figure 4.1 (1), we propose to manage the requirements of the case study to identify those that involve user-system interaction. Specifically, we focus on those requirements at the user interface (UI) that let the students reach the login Web page browsing through the three level-deep navigation, which we defined above for the case study, as follow:

- **Level 1 – The Student’s University home page.** The corresponding UI design provides the interface widgets⁴³ that allow the student to choose the anchor to his/her Faculty from a set of Faculty names, which make up the student’s University. In this case, as Figure 5.1 (a) shows, the UI design must include at least, for each link to Faculties, a widget of the type *SimpleActivator* at the abstract interface model mapped to the concrete interface model on a widget of the type *HTML link*. Also, as shown in Figure 5.1 (a), the UI design must include an extra link to skip the navigation bar. All these widgets are grouped together into a *CompositeInterfaceElement* at the abstract interface model and mapped to a concrete interface model on *HTML related links*. To complete de understanding of this mapping, refer to the *association table* for the *HTML link* and *button* group introduced in Section 4.5.2 by Table 4.2.
- **Level 2 – The Student’s Faculty page.** Basically, as Figure 5.1 (b) shows, the UI design must include, for the link to the SIU Guaraní registration system, a clear widget of the type *SimpleActivator* at the abstract interface model mapped to the concrete interface model on a widget of the type *HTML link*. To complete de understanding of this mapping, refer to the *association table* for the *HTML link* and *button* group introduced in Section 4.5.2 by Table 4.2.
- **Level 3 – The Student’s Login page.** The corresponding UI design provides the interface widgets that allow the student to login the SIU Guarani registration system. In this case, as Figure 5.1 (c) shows, the UI design must include at least, for the student’s identification purpose, two widgets of the type *IndefiniteVariable* at the

⁴³ To make this Step-by-Step explanation clearer, whenever we use “widgets” without specifying of which type, we are referring to both, abstract and concrete ones.

abstract interface model mapped to the concrete interface model on two widgets of the type *HTML text field*. The mission of these widgets is to receive the student's ID and Password values. Normally, these two widgets are grouped together into a *CompositeInterfaceElement* at the abstract interface model and mapped to the concrete interface model on *HTML related controls* to create a form. To complete the understanding of this mapping, refer to the *association table* for the *HTML control* group introduced in Section 4.3.2 by Table 4.1.

- **Levels 1, 2 and 3.** These three UI designs also provide text and images for student's information purpose. In this case, the UI designs must include three widgets of the type *ElementExhibitor* at the abstract interface models mapped to the concrete interface models on three widgets of the type *HTML image*. The mission of these widgets is to include the University logo (as shown in Figure 5.1 (a)), the Faculty picture (as shown in Figure 5.1 (b)), and the image of the key-lock (as shown in Figure 5.1 (c)). To complete the understanding of this mapping, refer to the *association table* for the *HTML text* and *non-text* group introduced in Section 4.5.2 by Table 4.3.
- **Level 4 – Help pages (Optional).** These two UI designs provide some instructive text about the data inputs ID and Password. In this case, as Figure 5.1 (d) shows, each UI design must include, for allowing the student to go back to the login page, a clear widget of the type *SimpleActivator* at the abstract interface model mapped to the concrete interface model on a widget of the type *HTML link*. To complete the understanding of this mapping, refer to the *association table* for the *HTML link* and *button* group introduced in Section 4.5.2 by Table 4.2.

It is important to highlight that browsing these pages is optional and therefore, if the student follows these help links, his/her decision will produce a different navigation path. At this point, we are focused on the UI models because, undoubtedly, is at the UI level where Accessibility barrier finally show; but in Section 6.3, we will revisit this argument to discuss the potential of our approach to deal with situations that could affect the Accessibility of the navigational models.

- **Levels 1, 2, 3 and 4.** Also, these four UI designs must consider widgets of the type *ElementStyling* at the abstract interface models mapped to the concrete interface

models on widgets of the type *HTML formatting & positioning*. The mission of these widgets is to define the appearance of the content --i.e. the look-&-feel of the UI. To complete the understanding of this mapping, refer to the *association table* for the *HTML frame* and *style sheet* group introduced in Section 4.5.2 by Table 4.5.

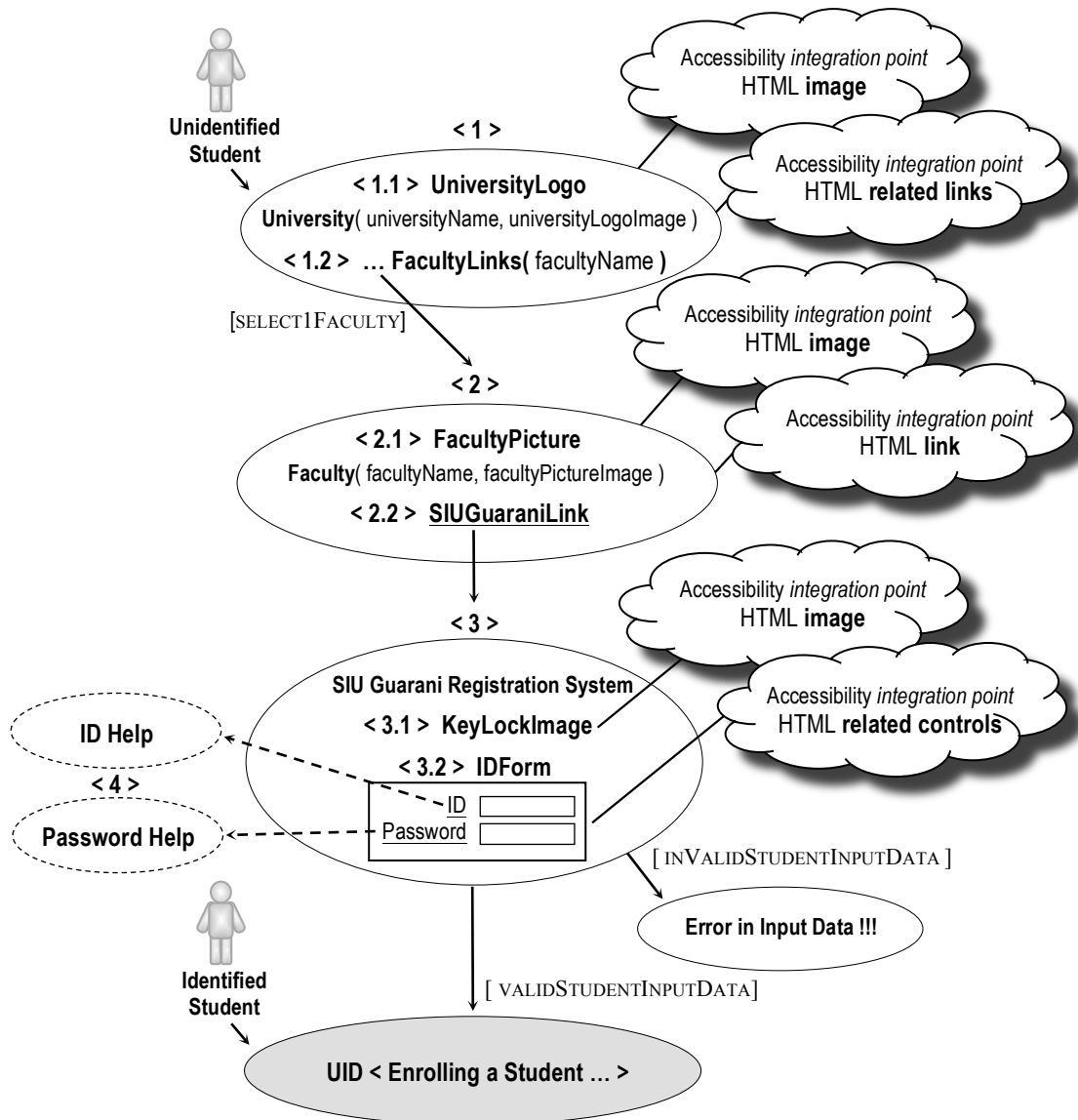


Figure 5.2: UID with integration points for the Case Study

STEP 2. As highlighted in Figure 4.1 (2.1) and (2.2), for specifying Accessibility concerns, we encourage the early capture of these Accessibility requirements by applying the UID and SIG conceptual tools.

STEP 2.1. We develop the UID diagram with *integration points* for the case study. As shown in Figure 5.2, at the UID interactions <1>, <2>, <3> and <4>, we outline the *integration points* that remain the Accessibility concerns that are crucial at each navigation level described above, as follow:

- **Level 1 – UID Interaction <1>.** We set <1.2> *integration point* for the HTML *HTML related links* corresponding to the links to Faculties.
- **Level 2 – UID interaction <2>.** We set <2.2> *integration point* for the HTML *link* corresponding to the link to the SIU Guarani registration.
- **Level 3 – UID interaction <3>.** We set <3.2> *integration point* for the HTML *related controls* corresponding to the form for the student’s identification. The Accessibility concerns, which are required by the related HTML *text fields* that make up the form, are relevant to a successful login information exchange between the student and the application, during the execution of the identification function.
- **Levels 1, 2 and 3 – UID interactions <1, 2, 3>.** We set <1.1>, <2.1> and <3.1> *integrations points* for the HTML *images* corresponding to the images of the University logo, the Faculty picture and the key-lock, respectively.
- **Level 4 – UID interactions <4> (Optional).** As we already said before, from **Level 3**, it is possible to browse to get some help for data inputs ID and Password. Although in Figure 5.2 we have not included details about the *integration points* required for these pages, we can set them for the HTML *text* and the HTML *link* corresponding to a helpful text and a link that clearly allows the student to return to the login Web page, respectively.
- **Levels 1, 2, 3 and 4 – UID interactions <1, 2, 3, 4>.** In Figure 5.2 we have not set *integrations points* for the HTML *formatting & positioning* to make simpler the understanding of the diagram and because, as we will see in **Step 2.2**, these are Accessibility concerns required in general for all Web pages.

STEP 2.2. We instantiate the SIG *template* for the Accessibility *integration points* outlined by the UID interactions <1>, <2>, <3> and <4> in **Step 2.1**, to identify WCAG 1.0 Accessibility requirements. In Section 3.5, we presented the basis of the SIG’s notation and vocabulary and then, in Section 4.3.2, we explained how we extended this

conceptual tool into a template to handle the Accessibility concerns. At this template, the focus of the Accessibility softgoal is highlighted into the root light cloud. 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. At this point, note that the *association tables* presented in Sections 4.3.1 and 4.5.2 help to the SIG instantiation process. Applying the SIG *template* for Accessibility, we develop the SIG diagrams at each navigation level, as follow:

- **Level 1 – SIG diagram at the UID interaction <1>**. As shown in Figure 5.3, we focus the main Accessibility softgoal on the UID interaction (U-UI) <1> called HTML University home. From this root, we define an Accessibility softgoal for the UID interaction component (U-UIC) <1.2> FacultyLinks, to help to accessible related links for all the students, including those with disabilities. In this case, to support the SIG instantiation process, we use Table 5.2 for the HTML *link* and *button* group, since the Accessibility softgoal is defined for the HTML *related links* element to Faculties. Next, we explain the refinement process for the SIG instantiation at the UID interaction <1>.

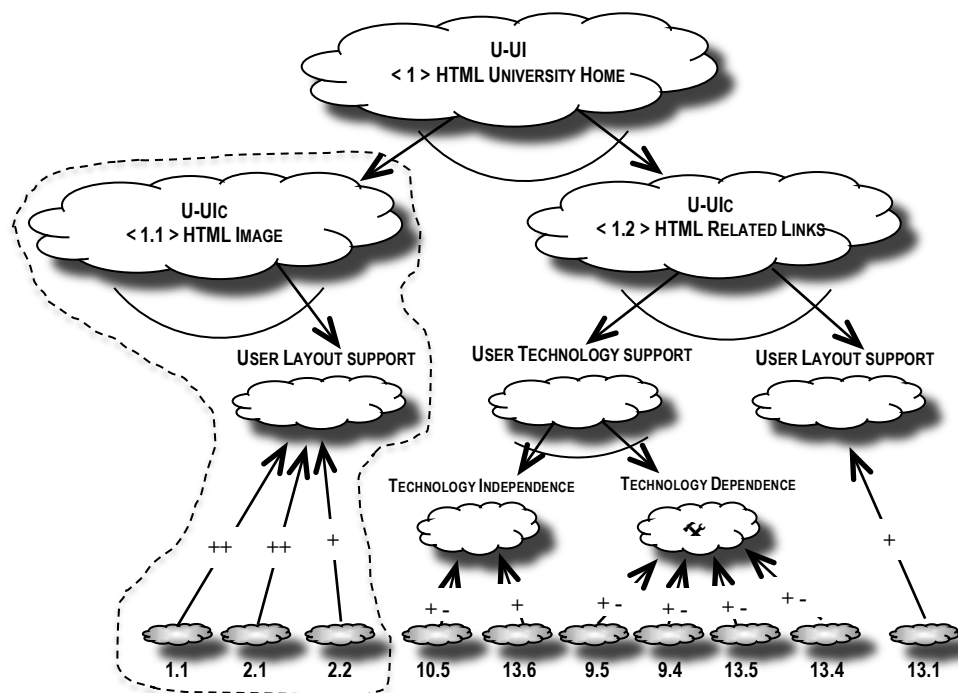


Figure 5.3: SIG instantiation for the UID interaction <1>

Firstly, looking at the user technology support branch in Figure 5.3, a distinction between “technology independence” and “technology dependence” is made in concordance with the distinction made in Section 4.3.2. To help to the universal access of devices to the HTML *related links* element, we chose an AND-decomposition; but the choice for an AND/OR decomposition will depend on the designer’s decisions and the application’s constraints. For “technology independence”, satisfying goals related to guidelines 10 and 13 for checkpoints 10.5 and 13.6 compliance are required. Otherwise for “technology dependence”, satisfying goals related to guidelines 9 and 13 for checkpoints 9.4 and 9.5; 13.5 and 13.4 compliance are required. Now looking at the user layout support, satisfying goals related to guideline 13 for checkpoint 13.1, compliance is required for the HTML *related links* element.

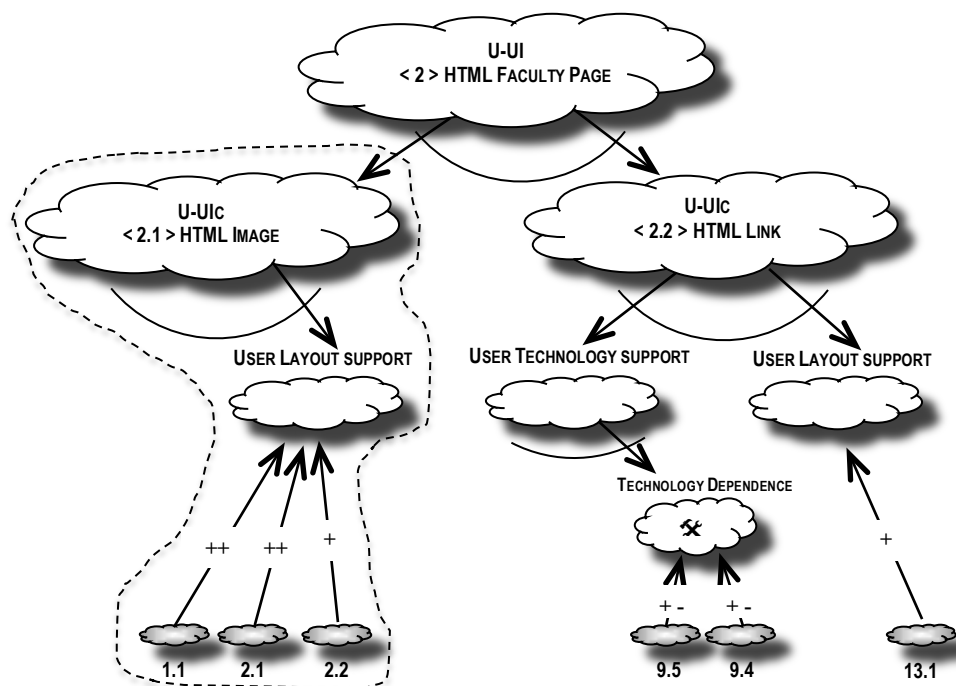


Figure 5.4: SIG instantiation for the UID interaction <2>

- **Level 2 – SIG diagram at the UID interaction <2>.** As shown in Figure 5.4, we focus the main Accessibility softgoal on the UID interaction (U-UI) <1> called HTML Faculty page. From this root, we define an Accessibility softgoal for the UID interaction component (U-Uic) <2.2> SIUGuaraniLink, to help to an accessible link. Here, to support the SIG instantiation process, we also use Table 5.3 for the HTML

link and *button* group, since the Accessibility softgoal is defined for the HTML *link* element to the SIU Guarani registration system. Next, we explain the refinement process for the SIG instantiation at the UID interaction <2>.

Firstly, looking at the user technology support branch in Figure 5.4, “technology dependence”, for satisfying goals related to guideline 9 for checkpoints 9.4 and 9.5, compliance are required for the HTML *link* element. Now looking at the user layout support, for satisfying goal related to guideline 13 for checkpoint 13.1, compliance is required for the HTML *related links* element.

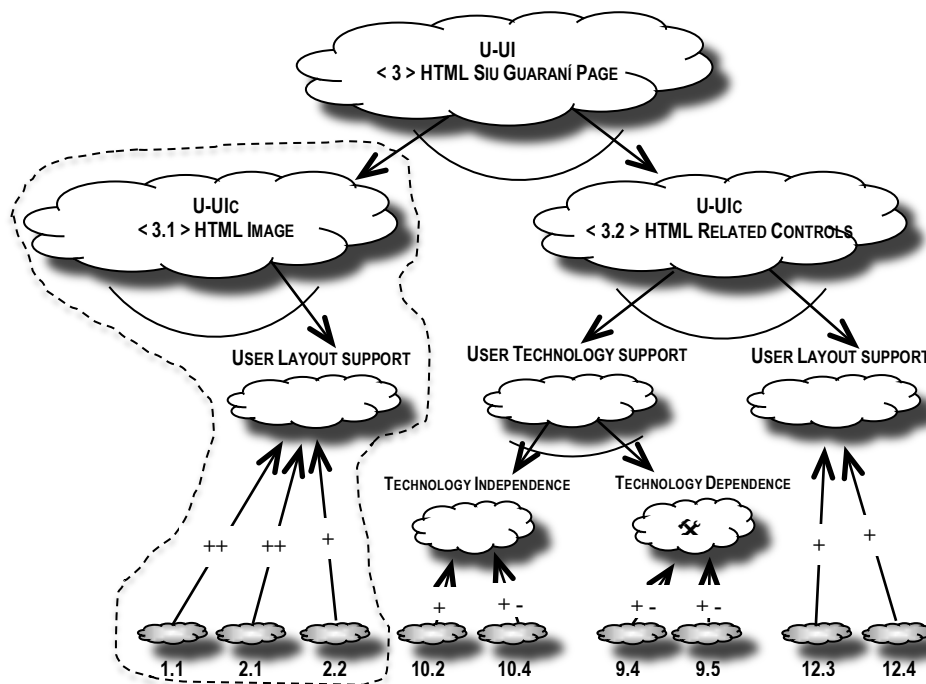


Figure 5.5: SIG instantiation for the UID interaction <3>

- **Level 3 – SIG diagram at the UID interaction <3>.** As shown in Figure 5.5, we focus the main Accessibility softgoal on the UID interaction (U-UI) <3> called HTML SIU Guarani page. From this root, we define an Accessibility softgoal for the UID interaction components (U-UIC) <3.2> IDForm, to help to accessible related controls. In this case, to support the SIG instantiation process, we use Table 5.1 for the HTML *control* group, since the Accessibility softgoal is defined for the HTML *related controls* element, which is a form composed of two HTML *text fields* for student identification purpose. Next, we explain the refinement process for the SIG instantiation at the UID interaction <3>.

Firstly, looking at the user technology support branch in Figure 5.5, we chose an AND-decomposition, as we already did at the SIG instantiation at UID interaction <1> and for the same reasons. For “technology independence”, for satisfying goals related to guideline 10 for checkpoints 10.2 and 10.4, compliance are required. Otherwise for “technology dependence”, for satisfying goals related to guideline 9 for checkpoints 9.4 and 9.5, compliance are required. Now looking at the user layout support, for satisfying goals related to guideline 12 for checkpoint 12.3 and 12.4, compliance are required for the HTML *related controls* element.

- **Levels 1, 2 and 3 – SIG diagrams at UID interactions <1, 2, 3>.** As shown in Figures 5.3, 5.4 and 5.5, we focus the main Accessibility softgoals on the UID interactions (U-UI) <1, 2, 3>. From these roots, we define Accessibility softgoals for the UID interaction components (U-UIc) <1.1> UniversityLogo, <2.1> FacultyPicture and <3.1> KeyLockImage to help to accessible HTML *image* elements at each page. In this case, to support the SIG instantiation process, we use Table 5.3 for the HTML *text* and *non-text* group, since these Accessibility softgoals are defined for the HTML *image* elements of the University logo, the Faculty picture and the key-lock respectively. Next, we explain the refinement process for the SIG instantiation at the UID interactions <1, 2, 3>.

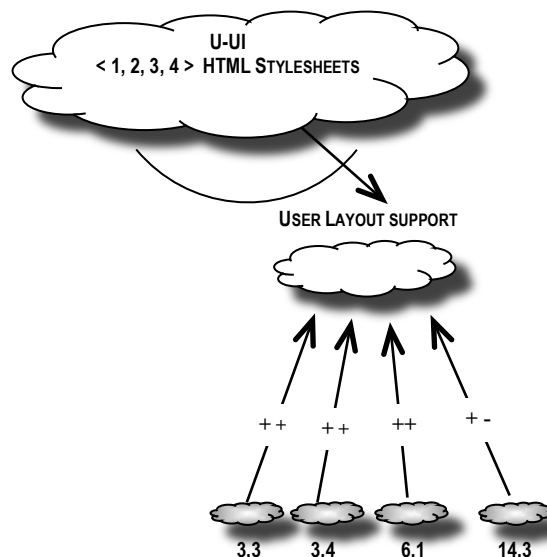


Figure 5.6: SIG instantiation for the UID interactions <1, 2, 3, 4>

Looking at the user layout support branches in Figures 5.3, 5.4 and 5.5, **for** satisfying goals related to guidelines 1 and 2 for checkpoints 1.1, 2.1 and 2.2,

compliance are required for the HTML *image* elements. In Section 4.1, we have already said, that there are situations in which we can develop artifacts once and then reused them, as they are required; at **Step 2** in Figure 4.1 (2.1) and (2.2), we have indicated the reuse capability of our approach with input/output arrows. Clearly, this is one of those situations, since the Accessibility softgoal for the HTML *image* element can be modeled once and then applied for the SIG instantiation, as they are required. As Figures 5.3, 5.4 and 5.5 show, we surrounded with dotted lines the UID interaction components (U-UIc) <1.1>, <2.1> and <3.1> for the HTML *image* elements to highlight the reusable artifact applied to the SIG diagrams of the case study.

- **Level 4 – SIG diagram at UID interactions <4> (Optional).** At this level, we proceed in the same way as for the previous levels. We do not give details about this optional level, because we consider it doesn't provide new knowledge about developing the SIG diagrams for Accessibility concerns.
- **Levels 1, 2, 3 and 4 – SIG diagram at UID interactions <1, 2, 3, 4>.** As shown in Figure 5.6, we focus the main Accessibility softgoal on the UID interactions (U-UI) <1, 2, 3, 4> called HTML Stylesheets. Here, to help the SIG instantiation process, we use Table 5.5 for the HTML *frame* and *style sheet* group, since the Accessibility softgoals are defined for the HTML *style sheet* elements to provide formatting and positioning support to the user layout. Next, we explain the refinement process for the SIG instantiation at the UID interactions <1>, <2>, <3> and <4>.

Looking at the user layout support branch in Figure 5.6, for satisfying goals related to guidelines 3, 6 and 14 for checkpoints 3.3 and 3.4, 6.1, 14.3, compliance are required for the HTML *style sheet* element.

STEP 3. As highlighted in Figure 4.1 (3), for the user interface design activity, we exploit the Accessibility knowledge captured and organized by SIG diagrams in **Step 2.2**. The purpose here is to find out how WCAG 1.0 Accessibility concerns “crosscut” the user interface widgets (abstract and concrete ones). In order to make our discussion clear, we focus on explaining how the SIG’s operationalizing goals --i.e. the required WCAG 1.0 checkpoints to be satisfied for an accessible student’s login -- “crosscut” the components of each HTML element corresponding to an abstract interface ontology

widget. 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 -- 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 elements at the user interface, but also allow Aspects to be modeled and instantiated appropriately to avoid “scattering” and “tangling” problems. Then Aspects can be seamless injected by the “weaving” mechanism into the core --i.e. user interface models, to achieve the Accessibility softgoal and as a consequence an HTML code with the desired conformance to the WCAG 1.0. As shown in Figure 4.1 (3.1), we work on the abstract user interface required at each navigation level, as follow:

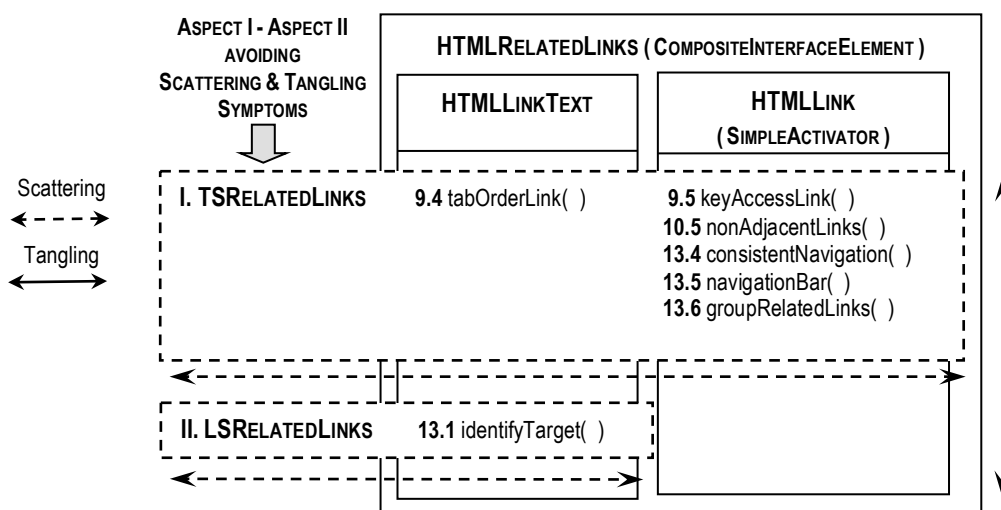


Figure 5.7: SIG’s operationalizing goals (WCAG 1.0 checkpoints) crosscutting an HTML *related links* element (Concrete Interface Widget) corresponding to a *CompositeInterfaceElement* (Abstract Interface Widget)

- **Level 1 – UI model at UID interaction <1>.** As shown in Figure 5.7 through a diagram similar to UML, whenever there is an HTML *related links* element at the user interface model, Aspect I “TSRelatedLink” and Aspect II “LSRelatedLinks”, focused on solving technology and layout Accessibility issues respectively, are injected to avoid the “scattered” and “tangling” nature of Accessibility checkpoints 9.4 and 9.5, 10.5, 13.4 and 13.5, 13.6 and 13.1 over HTML *related links* classes.

```

<?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>University WebSite</title>
<link rel="stylesheet" href="StudentID.css" type="text/css">
</head>
<body>
<h1> National University of Cordoba </h1>
<h2>Links to the Faculties WebSites</h2>
<map class="centerText" title="Navigation Bar to Faculties">
<p><a class="textLinks1" accesskey="S" href="#skip">Skip the Navigation Bar</a></p>
<p>
[<a class="textLinks2" accesskey="N" href="Student Faculty Home.html">Faculty of Exact, Physical
and Natural Sciences</a>]
[<a class="textLinks2" accesskey="E" href="other.html">Faculty of Economic Sciences</a>]
[<a class="textLinks2" accesskey="F" href="other.html">Faculty of Farming Sciences</a>]
[<a class="textLinks2" accesskey="L" href="other.html">Faculty of Languages</a>]
</p>
</map>
<h2><a name="skip">UNC on the Web: Institutional Info</a></h2>
</body>
</html>

```

CONFORMANCE TO WCAG 1.0
CHECKPOINTS 9.5, 10.5, 13.4, 13.5, 13.6
AND 13.1
ASPECT I - ASPECT II
AVOIDING

Figure 5.8: Accessible HTML code as a result of a “seamless” injection of Aspects I and II in the UI model at UID interaction <1>

The addition of Aspect I “TSRelatedLinks” and Aspect II “LSRelatedLinks” reminds later, at the implementation of the concrete interface model (as shown by Figure 4.1 (4.1), conformance to the following Accessibility concerns for each HTML *related links* element: (i) creating a logical tab order and/or providing keyboard shortcuts for links, (ii) including non-link, printable characters (surrounded by spaces) between adjacent links, (iii) using navigation mechanisms in a consistent manner and providing navigation bars to highlight and give access to the navigation mechanism, (iv) grouping related links, identifying the group and providing a way to bypass the group and, (v) clearly identifying the target of each link. Figure 5.8 shows the accessible HTML corresponding to the student’s University home example, whose screenshot is shown in Figures 2.1 and 5.1 (a).

- **Level 2 – UI model at UID interaction <2>**. As shown in Figure 5.9 through a diagram similar to UML, whenever there is an HTML *link* element at the user interface model, Aspect I “TSLink” and Aspect II “LSLink”, focused on solving technology and layout Accessibility issues respectively, are injected to avoid the

“scattered” and “tangling” nature of Accessibility checkpoints 9.4 and 9.5, and 13.1 over HTML *link* classes.

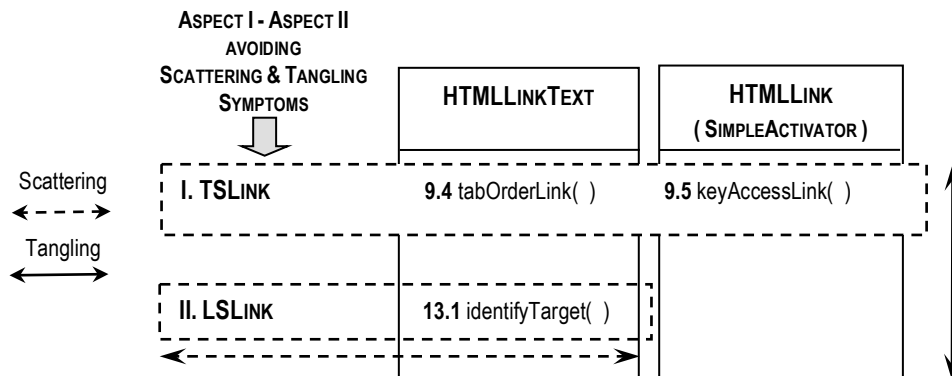


Figure 5.9: SIG’s operationalizing goals (WCAG 1.0 checkpoints) crosscutting an HTML *link* element (Concrete Interface Widget) corresponding to a *SimpleActivator* (Abstract Interface Widget)

The addition off Aspect I “TSLink” and Aspect II “LSLink” reminds later, at the implementation of the concrete interface model (as shown by Figure 4.1 (4.1)), conformance to the following Accessibility concerns for each HTML *link* element: (i) creating a logical tab order and/or providing keyboard shortcuts for links and, (ii) clearly identifying the target of each link.

```

Safari  Archivo  Edición  Visualización  Historial  Favoritos  Ventana  Ayuda
Código fuente de Student Faculty Home.html

<?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>Faculty WebSite</title>
<link rel="stylesheet" href="StudentID.css" type="text/css">
</head>
<body>
<h1>Faculty of Exact, Physical and Natural Sciences</h1>
<div class="centerText"></div>
<h2><a accesskey="G" href="Student Login Page.html">Student Management System</a></h2>
</body>
</html>

```

CONFORMANCE TO WCAG 1.0
CHECKPOINTS 9.5 AND 13.1
ASPECT I - ASPECT II
AVOIDING
SCATTERING & TANGLING SYMPTOMS

Figure 5.10: Accessible HTML code as a result of a “seamless” injection of Aspects I and II in the UI model at UID interaction <2>

Figure 5.10 shows the accessible HTML code corresponding to the student’s Faculty page example, whose screenshot is shown in 5.1 (b).

- **Level 3 – UI model at UID interaction <3>**. As shown in Figure 5.11 through a diagram similar to UML, whenever there is an HTML *related controls* element, which in this case comprises two HTML *text field* elements at the user interface model, Aspect I “TSRelatedControls” and Aspect II “LSRelatedControls”, focused on solving technology and layout Accessibility issues respectively, are injected to avoid the “scattered” and “tangling” nature of Accessibility checkpoints 9.4 and 9.5, 10.2 and 12.4, 10.4 and 12.3 and over HTML *related controls* classes.

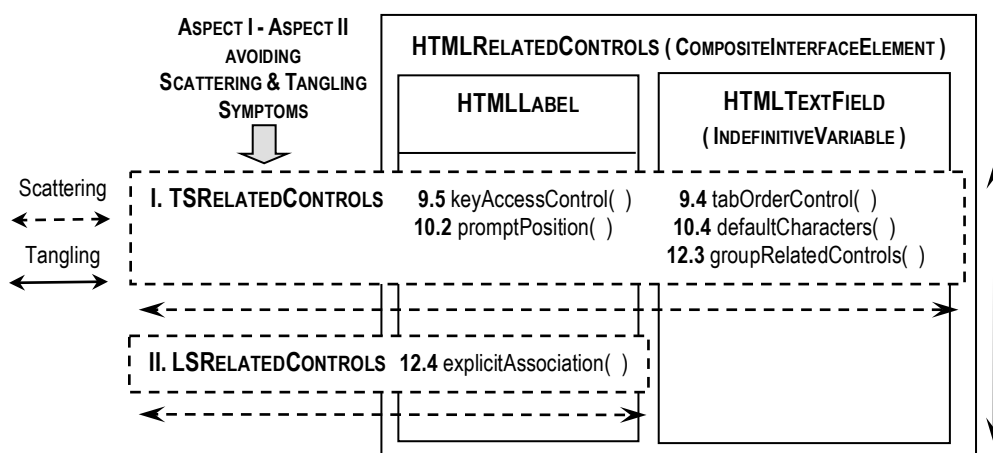


Figure 5.11: SIG’s operationalizing goals (WCAG 1.0 checkpoints) crosscutting an HTML *related controls* element (Concrete Interface Widget) corresponding to a *CompositeInterfaceElement* (Abstract Interface Widget)

The addition of Aspect I “TSRelatedControls” and Aspect II “LSRelatedControls” reminds later, at the implementation of the concrete interface model (as shown by Figure 4.1 (4.1)), conformance to the following Accessibility concerns for each HTML *related controls* element: (i) creating a logical tab order and/or providing keyboard shortcuts for controls, (ii) supporting explicit association between HTML *label* elements and controls, (iii) handling empty controls correctly by including default, place-holding characters and, (iv) grouping related controls with HTML *fieldset* and *legend* elements. Figure 5.12 shows the accessible HTML code corresponding to the student’s login page example, whose screenshot is shown in Figures 1.1 and 5.1 (c).

```

<?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 Guarani Management System</title>
<link rel="stylesheet" href="StudentID.css" type="text/css">
</head>
<body>
<h1>Guarani3W: Student Management System on Internet</h1>
<h2>Student Login Form</h2>
<div class="centerText"></div>
<form name="aspnetForm" method="post" action="">
<div class="topMargin">
<fieldset class="leftMargin">
<legend>Enter your Student ID and Password</legend>
<p>
<label for="idNumber">Identification:</label>
<a href="IDHelp.html" title="Identification Help for Student Login" accesskey="I">What is the Student ID?</a>
<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">What is the Student Password?</a>
<input type="text" id="idPassword" size="6" value=" " tabindex="2"/></p>
<p class="centerText"><input class="submit" type="submit" value=Submit tabindex="3"/></p>
<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 Password periodically.</p>
</fieldset>
</div>
</body>

```

CONFORMANCE TO WCAG 1.0
CHECKPOINTS 9.4, 9.5, 10.2, 10.4, 12.3
AND 12.4
ASPECT I - ASPECT II
AVOIDING
SCATTERING & TANGLING SYMPTOMS

Figure 5.12: Accessible HTML code as a result of a “seamless” injection of Aspects I and II in the UI model at UID interaction <3>

- **Level 1, 2 and 3 – UI models at UID interactions <1, 2, 3>.** As shown in Figure 5.13 through a diagram similar to UML, whenever there is an HTML *image* element, Aspect II “LSImage”, focused on solving layout Accessibility issues, is injected to avoid the “scattered” nature of Accessibility checkpoints 1.1, 1.2 and 2.2 over HTML *image* classes.

The addition of Aspect II “LSImage” reminds later, at the implementation of the concrete interface models (as shown by Figure 4.1 (4.1)), conformance to the following Accessibility concerns for each HTML *image* element: (i) adding a text equivalent for every image with a HTML *alt-text* element and, (ii) not relying on images’ color alone to convey information. Figures 5.8, 5.10 and 5.12 show the

accessible HTML corresponding to the student’s University home page, the Faculty page and the login page examples, whose screenshot are shown in Figures 5.1 (a), 5.1 (b) and 5.1 (c), respectively. As we can see in these HTML files, all the HTML *image* elements have their corresponding text equivalent.

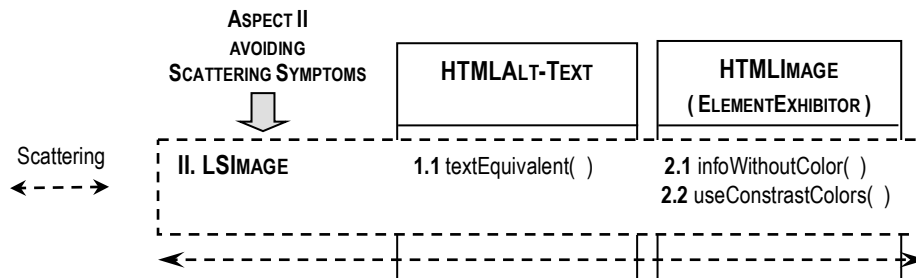


Figure 5.13: SIG’s operationalizing goals (WCAG 1.0 checkpoints) crosscutting an HTML *image* element (Concrete Interface Widget) corresponding to an *ElementExhibitor* (Abstract Interface Widget)

- **Level 4 – UI models at UID interaction <4> (Optional).** At this level, we proceed in the same way as for the previous levels. We do not give details about this optional level, because we consider it doesn’t provide new knowledge about developing the user interface models.
- **Level 1, 2, 3 and 4 – UI models at UID interactions <1, 2, 3, 4>.** As shown in Figure 5.14 through a diagram similar to UML, whenever there is an HTML *style sheet* element, Aspect II “LSStylesheet” focused on solving layout Accessibility issues, is injected to avoid the “scattered” nature of Accessibility checkpoints 3.3, 3.4, 6.1 and 14.3 over HTML *style sheet* classes.

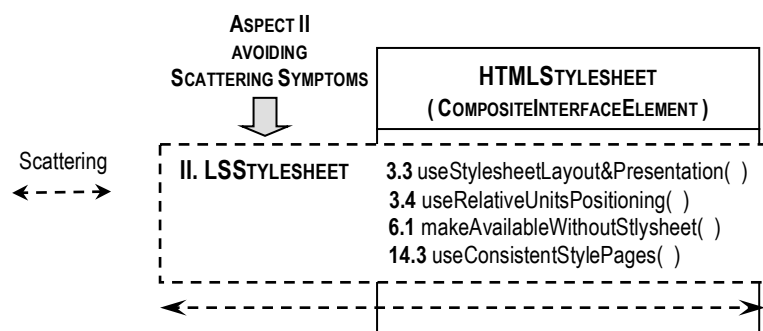


Figure 5.14: SIG’s operationalizing goals (WCAG 1.0 checkpoints) crosscutting an HTML *style sheet* element (Concrete Interface Widget) corresponding to a *CompositeInterfaceElement* (Abstract Interface Widget)

The addition of Aspect II “LSStylesheet” reminds later, at the implementation of the concrete interface models (as shown by Figure 4.1 (4.1)), conformance to the following Accessibility concerns for each HTML *style sheet* element: (i) using style sheets to control page layout and presentation, (ii) using relative rather than absolute units in markup language attribute values and style sheet property values, (iii) organizing documents so they may be read without style sheets and, (iv) creating a style of presentation that is consistent across pages. The HTML pages corresponding to the student’s University home page, the Faculty page, the login page and the help pages examples, whose screenshot are shown in Figures 5.1 (a), 5.1 (b), 5.1 (c) and 5.1 (d) respectively, keep a consistent styling across pages. As we can see in Figures 5.8, 5.10 and 5.12, for formatting and positioning purpose, these pages use an HTML *style sheet* element.

STEP 4. As highlighted in Figure 4.1 (4), for the user interface developing activity we exploit the aspects applied for solving Accessibility crosscutting concerns discovered in **Step 3.** As another way of illustrating how these aspects were seamless injected in an abstract user interface to obtain a concrete user interface (at the design level) and then an accessible and well formed HTML at the implementation level, we can express the Accessibility concerns conveyed by aspects using a pseudo-code language. We provide some examples for each level defined for the case study in Figure 5.1, as follow:

▪ **Level 1 – Aspect I and Aspect II in the UI model at UID interaction <1>.**

ASPECT I. TSRELATEDLINKS

POINTCUT ALL INTERFACE WIDGETS WITH CompositeInterfaceElement.SimpleActivator == HTML *related links*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITIONS

9.4 tabOrderLink == HTML *tabindex* element \wedge **9.5** keyAccessLink == HTML *accesskey* element \wedge

10.5 nonAdjacentLinks == HTML *printable characters* as “[” and “]” \wedge

13.4 consistentNavigation == W3C *Core Techniques* for navigation \wedge

13.5 navigationBar AND **13.6**groupRelatedLinks == HTML *map* element.

ASPECT II. LSRELATEDLINKS

POINTCUT ALL INTERFACE WIDGETS WITH CompositeInterfaceElement.SimpleActivator == HTML *related links*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITION **13.1** identifyTarget == HTML *clear link text* OR HTML *tittle* element.

- **Level 2 – Aspect I and Aspect II in the UI model at UID interaction <2>.**

ASPECT I. TSLINK

POINTCUT ALL INTERFACE WIDGETS WITH SimpleActivator == HTML *link*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITIONS

9.4 tabOrderLink == HTML *tabindex* element \wedge 9.5 keyAccessLink == HTML *accesskey* element.

ASPECT II. LSLINK

POINTCUT ALL INTERFACE WIDGETS WITH SimpleActivator == HTML *link* PROPERTY ADVICE ADD ACCESSIBILITY

CONDITION 13.1 identifyTarget == HTML *clear link text* OR HTML *title* element.

- **Level 3 – Aspect I and Aspect II in the UI model at UID interaction <3>.**

ASPECT I. TSRELATEDCONTROLS

POINTCUT ALL INTERFACE WIDGETS WITH CompositeInterfaceElement.IndefiniteVariable == HTML *related controls*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITIONS

9.4 tabOrderControl == HTML *tabindex* element \wedge 9.5 keyAccessControl == HTML *accesskey* element \wedge

10.2 promptPosition == HTML *for* element \wedge

10.4 defaultCharacters == HTML *value* element \wedge

12.3 groupRelatedControls == HTML *fieldset* element AND HTML *legend* element.

ASPECT II. LSRELATEDCONTROLS

POINTCUT ALL INTERFACE WIDGETS WITH CompositeInterfaceElement.IndefiniteVariable == HTML *related controls*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITION 12.4 explicitAssociation == HTML *for* element.

- **Level 1, 2 and 3 – Aspect II in UI models at UID interactions <1, 2, 3>.**

ASPECT II. LSIMAGE

POINTCUT ALL INTERFACE WIDGETS WITH ElementExhibitor == HTML *image*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITIONS

1.1 textEquivalent == HTML *alt* element OR HTML *longdesc* element \wedge

2.1 infoWithoutColor AND 2.2 useContrastColor == W3C *HTML, Core* AND *CSS Techniques* for color.

- **Level 4 – Aspects in UI models at UID interaction <4> (Optional).** At this level, we proceed in the same way as for the previous levels. We do not give details about this optional level, because we consider it doesn't provide new knowledge about injecting aspects in UI models.

- **Level 1, 2, 3 and 4 – Aspect II in UI models at UID interactions <1, 2, 3, 4>.**

ASPECT II. LSSYLESHEET

POINTCUT ALL INTERFACE WIDGETS WITH ElementStyling.Formatting&Positioning == HTML *stylesheet*

PROPERTY ADVICE ADD ACCESSIBILITY CONDITIONS

3.3 useStyleSheetLayout&Presentation AND 3.4 useRelativeUnitsPositioning AND

6.1 makeAvailableWithoutStylesheet AND 14.3 useConsistentStylePages == W3C HTML, Core AND CSS Techniques for controlling layout and presentation.

These are high-level specifications to avoid “scattering” and/or “tangling” symptoms caused by Accessibility concerns. The pointcut/advice pair specifies that, for all HTML widget of a specific kind (the pointcut specification), conditions satisfying Accessibility requirements are added (the advice specification).

As a result of modeling these aspects (using SIGs prescriptions for WCAG 1.0 checkpoints) and the addition of these aspects to deal with the targeted interface widgets, Figures 5.8, 5.10 and 5.12 show the accessible implementations for the concrete user interface models for the 3 (three) level-deep navigation case study in Figure 5.1, in terms of “well formed” HTML like W3C document [45].

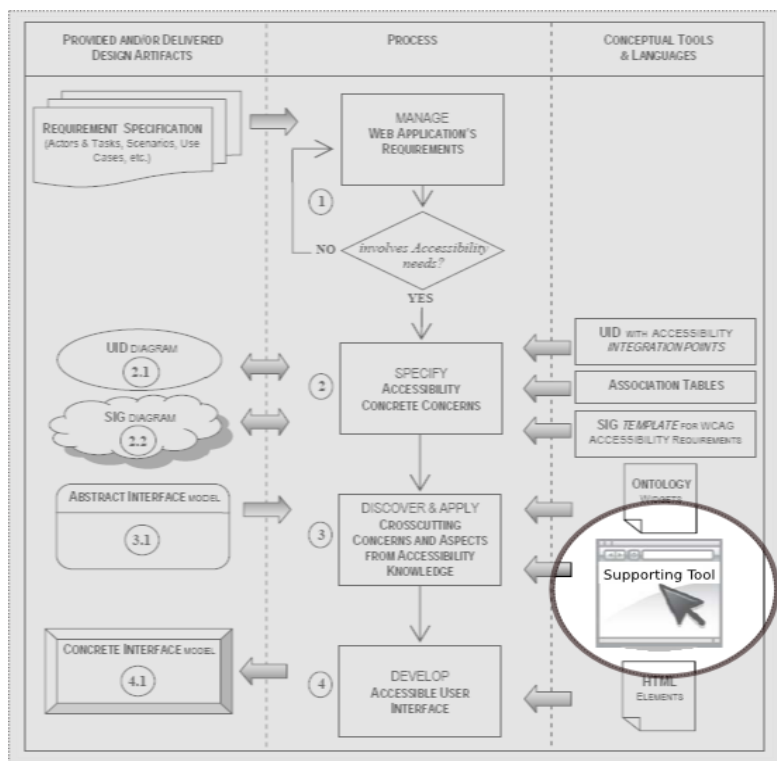


Figure 5.15: The supporting tool within our Aspect-Oriented design process

5.3 A Supporting Tool for Our Approach

Today, no one can deny the significance of having a supporting tool. The supporting tool and the kind of support given and features covered by the tool is relevant, especially to a design proposal. Related to this issue, our approach provides an initiative for a supporting tool to assist developers in the implementation of cases, and on the creation of their corresponding models by using reusable components. Currently, as Figure 5.15 shows, the tool provides assistance at **Step 3** of the design process for applying the Accessibility aspects (prescribed by the SIGs diagrams) to user interface models --i.e. abstract and concrete user interface models.

To achieve with its main purpose, the tool must deal with the concepts previously described, such as SIG diagrams, *association tables* and abstract user interface models. Also, the tool should be at the user's fingertips --i.e. the tool should be part of the users' development environment. To solve the second issue, the tool was developed as an Eclipse⁴⁴ plug-in, integrating an XML⁴⁵ editor in combination with the necessary views to inform the user about the missing information required for an accessible user interface --i.e. tags and attributes for a well-formed and accessible markup, as we describe in Section 5.3.2, and also to provide options to fix these missing information.

At this point, we introduce a brief explanation for the rational of choosing XML as the markup language to support resources and their future development as the tool evolves. Since XML allows writing our own markup language, we are not restricted to a limited set of tags defined by proprietary vendors. Custom tags are used to bring meaning to the data being displayed and when stored this way, data becomes extremely portable because it carry with their description rather than their display. In this way, XML allows the display to be extracted from the data and incorporated into a style sheet. Some of the benefits of this important XML characteristic are: (i) changes to display do not require futzing with the data, since a style sheet will specify the display, (ii) searching the data is easy and efficient, since tags provide the search engines with the intelligence they lack, (iii) complex relationships like trees and inheritance can be communicated and,

⁴⁴ The Eclipse Foundation at <http://www.eclipse.org/>

⁴⁵ W3C Extensible Markup Language (XML) at <http://www.w3.org/XML/>

(iv) the XML code is much more legible to a person coming into the environment with no prior knowledge. Other XML properties are: (i) it has stricter grammar rules than HTML that helps to develop well-formed documents --e.g. forgetting a label in an XML document makes the file unusable, (ii) it is a platform independent language and widely distributed and, (iii) it was developed by the W3C that also keeps its specification. The design goals of XML emphasize simplicity, generality, and usability over the Internet.

Following we introduce the proposed tool, describing the basis of its architecture, layers and classes, and also the resources and interfaces through which developers interact for designing accessible user interfaces.

5.3.1 Architecture's Overview: Layers and Classes

Figure 5.16 shows the tool's architecture and its three main layers, which are: *Presentation*, *Object Storage* and *Core*.

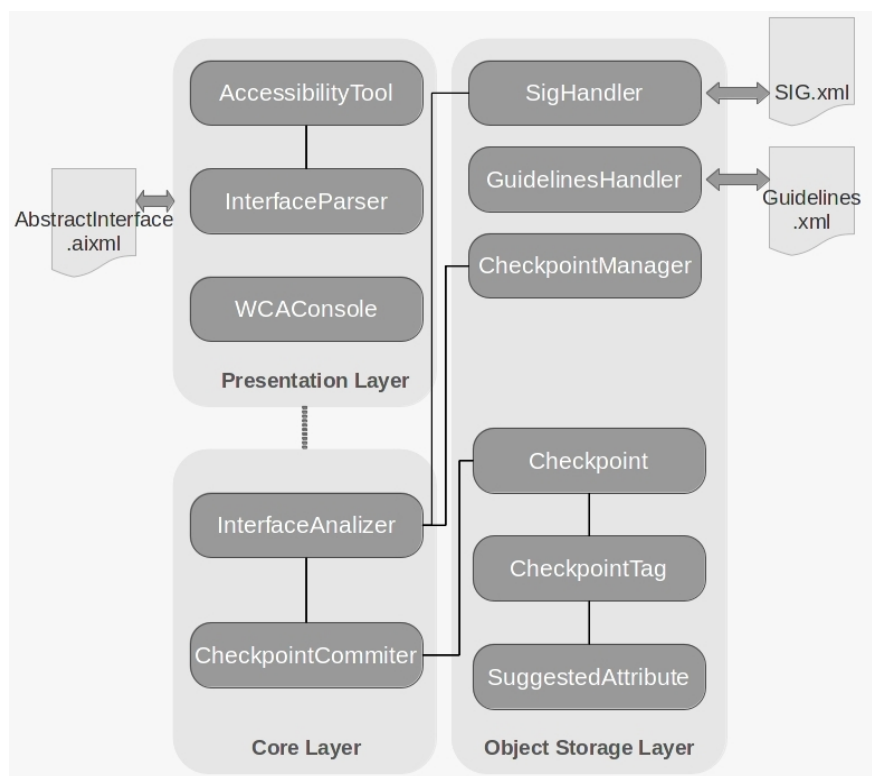


Figure 5.16: Main components of Our supporting tool

The *Presentation* layer represents the user interface for designers and developers. The main classes in the *Presentation* layer are:

-
- **AccessibilityTool** class, which represents the XML editor.
 - **InterfaceParser** class, which includes the functionality of identifying and highlighting syntax errors.
 - **WCAConsole** class, which provides functionality to show the non-commitment to the WCAG in a structured way. The name of this view stands for Web Content Accessibility Console, as a general view to include all the Accessibility issues.

The *Object Storage* layer represents an abstraction for the different underlying resource structures. Then, requests for information about WCAG 1.0 checkpoints [45], present in the SIG structure or in the tool database, are solved using the services of this layer. The main classes for the *Object Storage* layer are:

- **SIGHandler** class, which provides the necessary functionality to access the contained information in SIG structure file --i.e. the checkpoints to commit for a specified tag present in the abstract user interface.
- **GuidelinesHandler** class, which as the previous class, provides the needed functionality to access the contained information in the Guidelines file.
- **CheckpointManager** class, which provides the needed functionality to access information of different checkpoints. This class uses *CheckpointManager* to retrieve information about a checkpoint from the database file and maintain a pool of previously retrieved checkpoints.
- **Checkpoint**, **CheckpointTag** and **SuggestedAttribute** classes, which represent the models for accessing information about the element that each one represents. Specifically, *SuggestedAttribute* represents an attribute that needs to be added (or deleted) in a tag --i.e. *CheckpointTag*, to meet a specific *Checkpoint*.

Finally, the *Core* layer includes those classes that play a central role for the tool's functionality. Those classes are:

- **CheckpointCommitter** class, whose functionality includes the analysis and determination of commitment of an HTML tag to the WCAG recommendations. Also, it provides the functionality to generate the element code --i.e. HTML tag or attribute, to fix the non-commitment.

-
- **InterfaceAnalyzer** class, which provides the functionality of coordination for the analysis of the abstract user interface model. This class has an aspect-based implementation done in AspectJ⁴⁶, which is the central feature that will allow the completion of the analysis in a transparent manner --i.e. solving Accessibility crosscutting problems by injecting aspects smoothly.

Particularly, in Figure 5.16, 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 user interface model under treatment, takes place in a transparent manner. This abstract user interface model is an XML file, as we following see in Section 5.3.2. To reproduce this behavior, the tool uses the *Observer pattern*⁴⁷ and their classes *Subject* and *Observer*; each instance of the *Subject* class maintains a list of instances of the *Observer* class that are notified of the changes that occur in their respective instance of the *Subject* class. By applying these design concepts, the *AccessibilityTool* class plays the role of *Subject*, while the *InterfaceAnalyzer* class plays the role of *Observer*. Then, the aspects environment --i.e. the AspectJ capabilities, manages the update notifications. Thus, when the developer saves the XML document edited for the abstract user interface model, this automatically triggers this aspect-oriented functionality, which is not explicitly invoked by some element of the *Presentation* layer. As shown in Figure 5.15, the consequence at **Step 4.1** is the deliverable of a concrete HTML user interface model that improves conformance to WCAG 1.0 Accessibility requirements.

5.3.2 Tool's Resources: XML Schemas and Specifications

Figure 5.16 shows three XML files representing the input/output resources of the tool, which are *AbstractInterface*, *SIG*, and *Guidelines*. Following, we explain the relationship of these resources with our design proposal and we also provide their

⁴⁶ The AspectJ Development Tool at <http://www.eclipse.org/ajdt/>

⁴⁷ Object-Oriented Design and Programming: Observer Pattern at <http://www.oodesign.com/observer-pattern.html>

respective XML schema. Using examples, we show how to instantiate these XML schema for specifying the XML files.

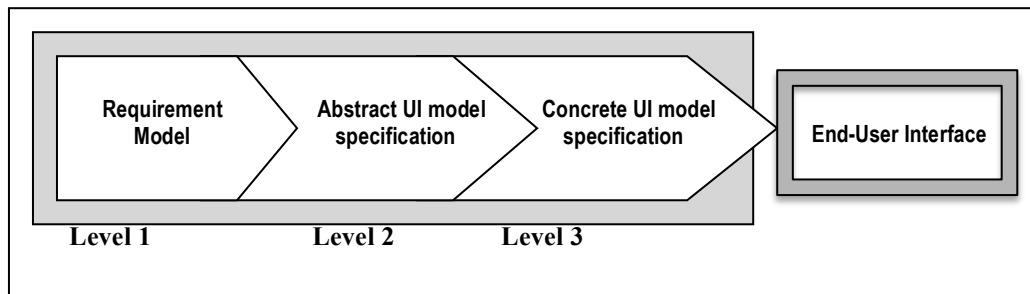


Figure 5.17: Model-driven principles applied to UI model development

The *AbstractInterface XML file* represents the abstract user interface model. As we have explained in previous chapters, our design approach uses the model-driven paradigm to develop high-level descriptions of the user interface structure and behavior and, from these declarative models to obtain the end-user interface. Figure 5.17 illustrates these design concepts, which are implemented by WE methods [31], such as OOHDM [36], which we have applied to develop our approach and supporting tool.

Figure 5.18 shows, the *AbstractInterface XML schema*⁴⁸ that we develop for specifying machine-understandable abstract user interface models. The most important tags of this XML schema are *Interface*, *Component*, *Composite* and *Attribute*.

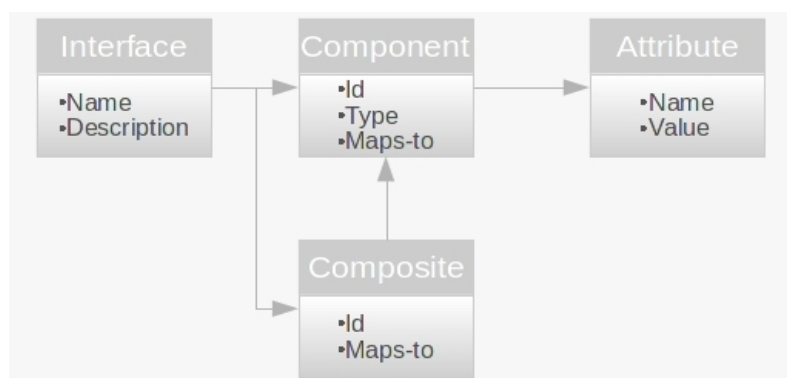


Figure 5.18: XML schema for the Abstract User Interface model

The specification of documents based on this schema begins with an *Interface* element, which can comprise *Composite* and *Component* elements. Also, a *Composite* element

⁴⁸ W3C XML Schema at <http://www.w3.org/XML/Schema>

can comprise *Component* elements resulting in a hierarchy of elements. Each tag has a modeling function within the *AbstractInterface* XML schema and its own descriptive attributes, as follow:

- The **Interface** tag is the container for the structure of an abstract user interface. The *Interface* tag has two descriptive attributes: (i) name, which identifies the *Interface* element under develop and, (ii) description, which states the purpose of the *Interface* element and the *Composite* and *Component* elements that are comprised within the *Interface* element.
- The **Component** tag represents the widgets that make up the abstract user interface. The *Component* tag has three descriptive attributes: (i) id, which identifies the *Component* element under development, (ii) type, which assign to the *Component* element a simple ontology widget and, (iii) maps-to, which links the *Component* element to a simple HTML element --e.g. an HTML *text field* element which is usually codified by using an HTML *input* element.
- The **Composite** tag is a container within an *Interface* element that comprises *Component* elements. The *Composite* tag has two descriptive attributes: (i) id, which identifies the *Composite* element under development and, (ii) maps-to, which links the *Composite* element to a composite HTML element --e.g. an HTML *related controls* element which is usually codified by using an HTML *fieldset* element.
- The **Attribute** tag represents the attributes that will be part of a concrete HTML element conveyed by “map-to” attributes. To complete the user interface design, the user adds some of these attributes, while the tool suggests others to solve Accesibility concerns.

Figure 5.19 shows the XML file specified applying the *AbstractInterface* XML schema to part of the case study shown in Figure 5.1 (c). As we can see in this specification, a *Composite* element is included at line 4 to represent the student identification FORM, which is a composite HTML element comprising two *Component* elements. These two INPUTs are *Component* elements included at lines 5 and 7 respectively, to represent the HTML *text field* elements required for the student’s name and password. The pair of attributes type and maps-to allow the association between ontology widget-HTML

element --e.g. the *Component* elements at lines 5 and 7 are of the ontology type **indefiniteVariable** and maps-to HTML *input* elements.

```

1. <interface name="student's login" description="An interface for
   the student's login at the SIU Guarani registration system">
2. <component id="guaraniLogo" type="elementExhibitor" maps-to="IMG">
3. </component>
4. <composite id="studentID" maps-to="FORM">
5. <component id="studentName" type="indefiniteVariable" maps-
   to="INPUT">
6. </component>
7. <component id="studentPassword" type="indefiniteVariable" maps-
   to="INPUT">
8. </component>
9. </composite>
10. </interface>

```

Figure 5.19: XML specification of an abstract user interface model

The **SIG XML file** represents the Softgoal Interdependency Graph (SIG) *template* for Accessibility and, as shown in Figure 5.20, we develop the **SIG XML schema** for specifying machine-understandable SIG diagrams. The most important tags of this *SIG XML schema* are *SIG*, *Node* and *Relation*.

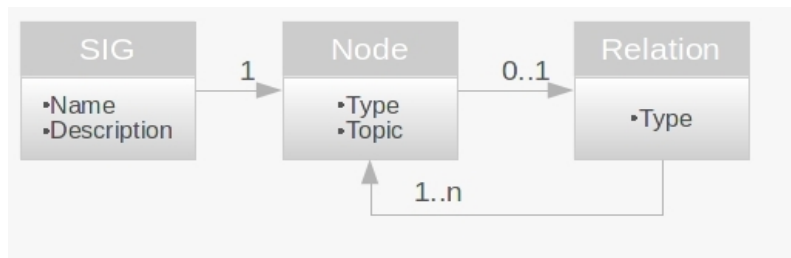


Figure 5.20: XML schema for the SIG template for Accessibility

The specification of documents based on this *SIG XML schema* begins with a *SIG* element linked to a main *Node* element, which in turn can comprises one or more *Node* elements through a *Relation* element. Thus, the *Relation* element allows a hierarchy specification for a *SIG* element. Each tag has a modeling function within the *SIG XML schema* and its own descriptive attributes, as follow:

-
- The **SIG** tag is the container for the structure of a SIG diagram for Accessibility. The *SIG* tag has two descriptive attributes: (i) name, which identifies the *SIG* element under develop and, (ii) description, which focus on the Accessibility softgoal of the *SIG* element through its main *Node* element --i.e. which, as we already explained in Section 5.2, is called the root light cloud of the SIG diagram applying the SIG terminology.
 - The **Node** tag represents a node, which, as we have already explained in Section 5.2, is called a cloud of the SIG diagram applying the SIG terminology. Thus, a *Node* element can represent a root or a refined Accessibility softgoal --i.e. a white cloud of the SIG diagram applying the SIG terminology, or an operationalizing goal for the required checkpoints to be satisfied --i.e. a dark cloud of the SIG diagram applying the SIG terminology. The *Node* tag has two descriptive attributes: (i) type, which specifies the type of a *Node* element depending on its Accessibility softgoal and, (ii) topic, which describes the Accessibility softgoal to be satisfied. While, the type of the *Node* attribute can be one of the following:
 - **U-UI** type, if the softgoal comprises Accessibility requirements to be satisfied at an interaction level in the UID diagram. We can use the U-UI type for a *Node* element representing a root Accessibility softgoal in the SIG diagram --e.g. in Figure 5.5, the U-UI root cloud for the SIU Guarani home page.
 - **U-UIc** type, if the softgoal represents Accessibility requirements to be satisfied at a component level in the UID interaction. We can use the U-UIc type for a *Node* element representing a refined or an operationalizing goal of the SIG diagram --i.e. in Figure 5.5, the U-UIc refined cloud for the HTML *related controls* element representing the student's identification form.
 - **Decomposition** type, if the *Node* element represents an Accessibility softgoal refinement by decomposition --i.e. in Figure 5.5, the Decomposition cloud at the User Technology Support branch for the HTML *related controls* element.
 - **Operationalizing** type, if the *Node* element represents an Accessibility operationalizing goal --i.e. in Figure 5.5, the Operationalizing dark clouds representing Accessibility requirements to be satisfied.
-

-
- The **Relation** tag applies for a parent *Node* element and its children, allowing a hierarchy specification for a *SIG* element. The *Relation* tag has only one descriptive attribute, *type*, which specifies the type of the relationship established between the parent *Node* element and its children. While, the type of the *Relation* attribute can be one of the following:
 - **AND** type, which represents the conjunction relationship, where all the children representing Accessibility softgoals must be satisfied to satisfy its parent *Node* element.
 - **OR** type, which represents the disjunction relationship, where satisfying some of the children representing Accessibility softgoals satisfied the parent *Node* element.
 - **OPERATIONALIZING** type, which represents the Accessibility operationalizing goal of the parent *Node* element. These operationalizing goals implement concrete Accessibility requirements on which a validation can be performed to establish conformance. For the instantiation of the Accessibility requirements, our tool applies the WCAG 1.0 checkpoint [45], but as we will explain in Chapter 6, our design proposal can work also with the WCAG 2.0 success criteria [46].
 - The **NodeList** tag is a container for a list of *Node* elements within a *Relation* element. Therefore, the *NodeList* tag can comprise one or more *Node* elements that are children of a parent *Node* element.

Figure 5.21 shows the XML file specified applying the *SIG* XML schema to part of the XML specification of the abstract user interface model in Figure 5.20. As shown at line 1, the softgoal to be satisfied --i.e. the Accessibility concern of the *SIG* diagram, is set in order to improve the Accessibility for all the students accessing the SIU Guarani registration system. The root *Node* element at line 2 is of the type **U-UI** because its Accessibility softgoal targets the UID interaction representing the home page of the system. This root *Node* element is decomposed into two refined *Node* elements at lines 5 and 19 by a *Relation* element of the type **AND** at line 3. These two *Node* elements are of the type **U-UIc** because their Accessibility softgoals target the IMG and FORM components at the UID interaction representing the home page of the system. The softgoal refinement process continues over the tree to develop the *SIG* diagram for

Accessibility, until specific operationalizing goals are met. For example, at line 11 the *Node* element is of the type **operationalizing** and in consequence instantiates the topic attribute with the **checkpoint 1.1** to establish a concrete Accessibility requirement to be satisfied.

```
1. <sig name="student's login" description="SIG instantiation for
   an accessible user interface for the student's login at the SIU
   Guarani registration system">
2. <node type="U-UI" topic="HTML SIU Guarani Page">
3. <relation type="AND">
4. <nodeList>
5.   <node type="U-UIC" topic="IMG">
6.   <relation type="AND">
7.   <nodeList>
8.     <node type="decomposition" topic="USER LAYOUT SUPPORT">
9.     <relation type="OPERATIONALIZING">
10.    <nodeList>
11.      <node type="operationalizing" topic="1.1" />
12.      ...
13.    </nodeList>
14.   </relation>
15.   </node>
16. </nodeList>
17. </relation>
18. </node>
19. <node type="U-UIC" topic="FORM">
20. <relation type="AND">
21. <nodeList>
22.   <node type="decomposition" topic="USER TECHNOLOGY LAYOUT">
23.   ...
```

Figure 5.21: XML specification of a SIG diagram for Accessibility

The *Guidelines XML file* represents the Accessibility guidelines from the WCAG 1.0 recommendations [45], which are stored accordingly to a structured language we especially develop. As we have already seen in previous chapters, there is a gap between the abstract knowledge transmitted by guidelines, which are expressed in natural language, and their implementation using a markup language such as HTML,

which is based on a technical specification⁴⁹. Trying to reduce this gap, we propose a structured language for guidelines, which we called in Spanish LEP (Lenguaje de Estructura de Pautas). As Figure 5.22 shows, LEP is positioned between natural language and HTML, simplifying not only the human comprehension of guidelines but also their storage as structures specified by a XML schema. Therefore, LEP is a specification language to adapt the structure of the Accessibility guidelines from WCAG 1.0 recommendations and make them possible to be managed by our tool.

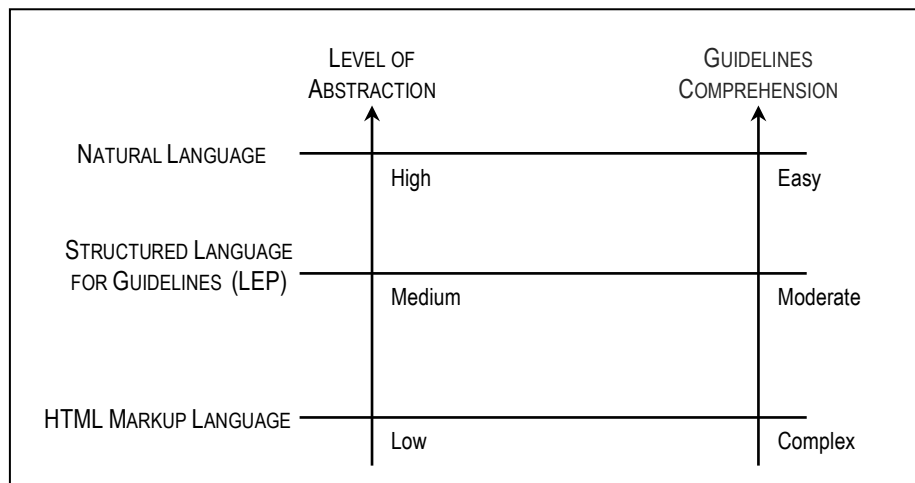


Figure 5.22: Levels of expressiveness to Accessibility Guidelines comprehension

The W3C-WAI [50] has specified systematically the 14 (fourteen) guidelines of the WCAG 1.0 recommendations (see the complete document at Appendix I). Each guideline within the WCAG 1.0 recommendations [45] includes: (i) the guideline number, (ii) the statement of the guideline (iii) the rationale behind the guideline and some groups of users who benefit from it and, (iv) a list of checkpoint definitions. The checkpoint definitions in each guideline explain how the guideline applies in typical content development scenarios. Each checkpoint definition includes: (i) the checkpoint number, (ii) the statement of the checkpoint, (iii) the priority of the checkpoint (the priority levels are 1, 2, 3), (iv) optional informative notes, clarifying examples, and cross references to related guidelines or checkpoints and, (v) a list of techniques where implementations and examples of the checkpoint are discussed to facilitate the checkpoint evaluation and conformance.

⁴⁹ W3C HTML 4 Specification at <http://dev.w3.org/html5/spec/Overview.html>

Now, to adapt this Accessibility information provided by WCAG 1.0 recommendations, we consider the formalization of those elements that are relevant to the expressiveness of the stored structures for providing the proper support required by the tool. Figure 5.23 shows the *Guidelines XML schema* we develop based on LEP --i.e. our supporting language, to allow the adaptation of the Accessibility guidelines and to store their structures as machine-understandable representations. The most important tags of the *Guidelines XML schema* are *Guidelines*, *Guideline*, *Checkpoint*, *Tag* and *Attribute*.

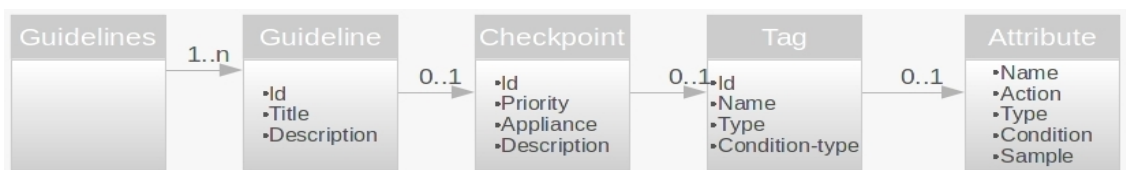


Figure 5.23: XML schema for the Accessibility guidelines from WCAG 1.0

As we can see in Figure 5.23, each *Guideline* element has a list of *Checkpoint* elements and each *Checkpoint* element has a list of *Tag* elements --i.e. HTML tags, which are the target of the *Checkpoint* element. For example, if a *Checkpoint* element establishes that an HTML *table* element must summary its content --i.e. checkpoint 5.5 from WCAG 1.0, the *Checkpoint* element will include a *Tag* element for the HTML *table* element and, the *Tag* element will include an *Attribute* element for the HTML *summary* element.

[GUIDELINE NUMBER] – [STATEMENT OF THE GUIDELINE]	
[CHECKPOINT NUMBER] – [STATEMENT OF THE CHECKPOINT] – [PRIORITY OF THE CHECKPOINT]	
PRESCRIPTION OF THE CHECKPOINT	APPLIANCE
Provides an explanation of the checkpoint and its foundations to compliance.	[SEMI-AUTOMATIC] <i>Requires the developer's manual intervention with the tool's support.</i> OR [MANUAL] <i>Requires the developer's manual intervention without the tool's support.</i>
SAMPLE: Provides topics on how to implement the checkpoint using well-formed and accessible HTML.	
SAMPLE IN LEP SPECIFICATION: Provides examples of how the checkpoints are specified in LEP.	

Figure 5.24: Adapting the WCAG 1.0 checkpoints to the schema based on LEP

The *Guidelines* XML schema based on LEP, convey information through the following tags:

- The **Guidelines**, which allow beginning a new file and containing its structure.
- The **Guideline**, which provides id, title and description of a specific WCAG 1.0 guideline; also includes a list of its checkpoints.

GUIDELINE 1. PROVIDE EQUIVALENT ALTERNATIVES TO AUDITORY AND VISUAL CONTENT	
<p>CHECKPOINT 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). <i>This includes:</i> images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [PRIORITY 1]</p>	
PRESCRIPTION OF THE CHECKPOINT	APPLIANCE
<ul style="list-style-type: none"> • Use "alt" for the IMG, INPUT, and APPLETT elements, or provide a text equivalent in the content of the OBJECT and APPLETT elements. • For complex content (e.g., a chart) where the "alt" text does not provide a complete text equivalent, provide an additional description using, for example, "longdesc" with IMG or FRAME, a link inside an OBJECT element, or a description link. • For image maps, either use the "alt" attribute with AREA, or use the MAP element with A elements (and other text) as content. 	[SEMI-AUTOMATIC]
<p>SAMPLE:</p> <pre></pre>	
<p>SAMPLE IN LEP SPECIFICATION:</p> <pre><tagList> <tag id="1" name="IMG" type="" condition-type=""> <attributes> <attribute name="ALT" sample="img src="guarani3w.jpg" alt="" action="add" type="HTMLAttribute" condition="mandatory"/> </attributes> </tag> </tagList></pre>	

Figure 5.25: Adapting checkpoints 1.1 to the schema based on LEP

- The **Checkpoint**, which provides id, priority (1, 2, 3) and description of a specific WCAG 1.0 checkpoint; also includes the appliance, which is “semi-automatic” when the checkpoint requires the developer’s manual intervention with the tool’s support or is “manual” when requires the developer’s manual intervention without the tool’s support, and a list of the HTML tags concerning to the checkpoint.
- The **Tag**, which provides id, which is a number assigned for identification purpose and is not related with WCAG 1.0 guidelines and checkpoints numbers, name (the

HTML tag name), and type/condition-type, which allow to specify the tag use case/s where the guideline/checkpoint applies to the tag; also includes a list of its attributes.

- The **Attribute**, which provides name (the HTML attribute or tag name), action (add, modify, update or delete), type (HTML tag, HTML attribute, text attributes, etc.), condition, which allows specifying if the attribute is mandatory or optional, and sample, which provides an application example.

The preservation of the WCAG philosophy was our goal when we worked on the Accessibility guidelines seeking for a specification manageable by the tool. Figure 5.24 summarizes the basis for analyzing and adapting the WCAG 1.0 checkpoints to the *Guidelines* XML schema based on LEP, while Figure 5.25 shows part of the analysis and adaptation for **checkpoint 1.1**. For example, this specification applies to satisfy the **operationalizing** softgoal in the SIG diagram shown in Figure 5.21, line 11.

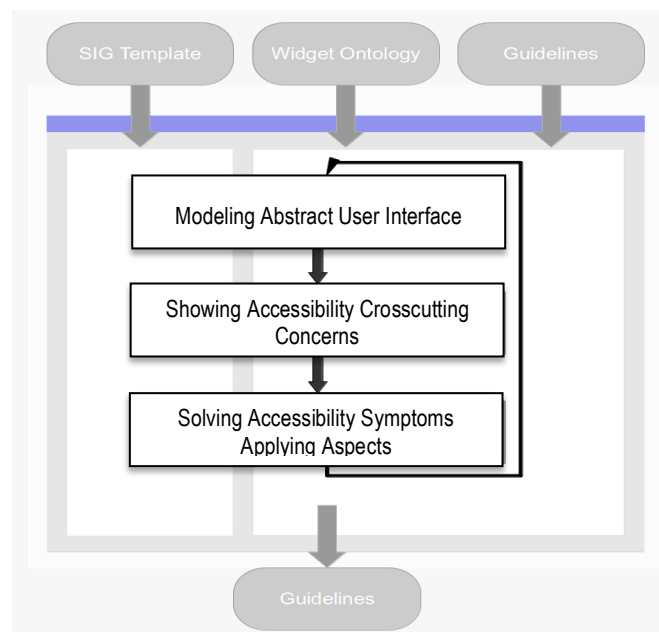


Figure 5.26: Basis of the Aspect-Oriented design cycle

5.3.3 Tool’s User Interfaces

From the user’s point of view the interaction with the tool applies an “open-save-close” cycle to the document under develop. The developer designs an abstract user interface

for a given Web page by editing and saving changes in an XML-based document. This mode for developing documents is usually known as document-centered work schema.

Figure 5.26 shows the basis of the aspect-oriented design cycle in the interaction between the developer and our tool, where we can identify the following steps:

- **Modeling Abstract User Interface**, the developer designs the abstract user interface model choosing widgets from the abstract widget ontology.
- **Showing Accessibility Crosscutting Concerns**, the tool shows how the Accessibility concerns crosscut the interface widgets selected to compose the user interface by the developer.
- **Solving Accessibility Symptoms Applying Aspects**, the developer decides, based on the information provided by the tool and the tool wraps, these Accessibility crosscutting concerns into Accessibility aspects for their modularization and transparent injection in the user interface under design.

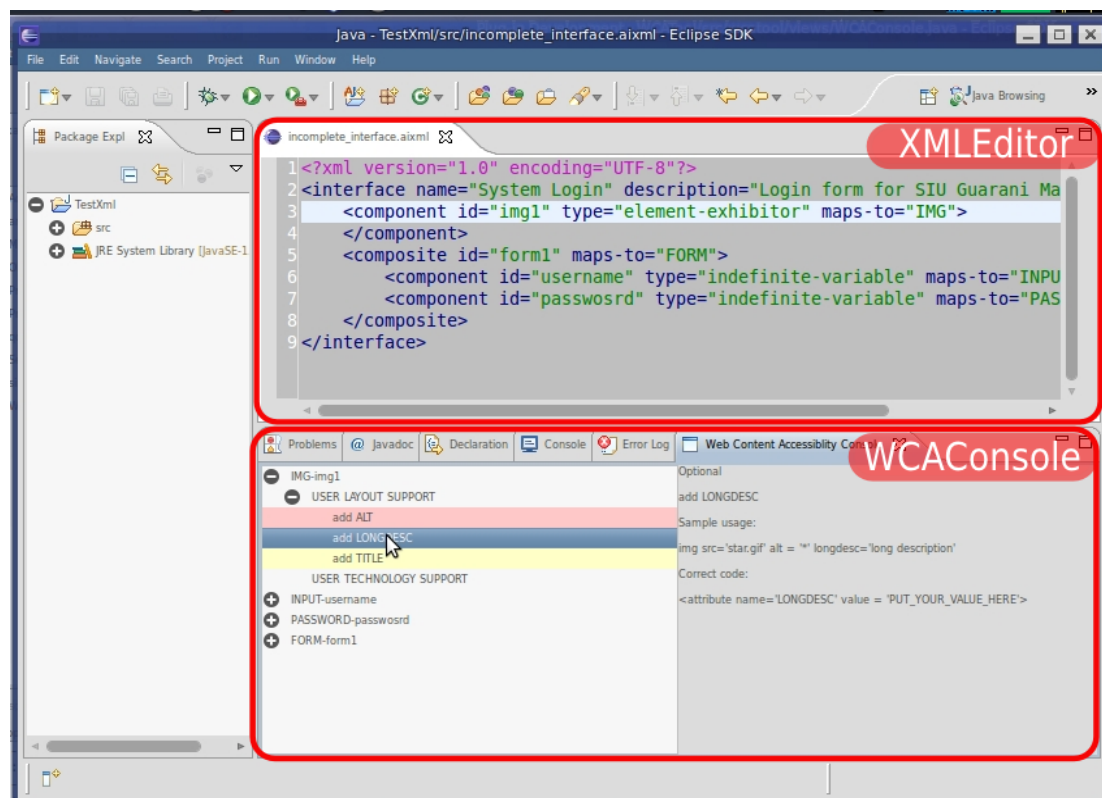


Figure 5.27: The components integrated in the Eclipse platform

For this reason, one of the main components of the tool's UI is the *XML Editor*, which is complemented with the view *WCAConsole* for showing, and allow solving the non-

commitment to the Accessibility guidelines. Figure 5.27 shows a screenshot of these tool components integrated in the Eclipse platform. The *XMLEditor* is shown in the upper box of screen in Figure 5.27 and is used by the developer to edit the abstract user interface model. When the developer saves the XML file and its changes, the analysis of the structure and commitment to the Accessibility guidelines is launched. The analysis result is shown in a structured manner using the view *WCAConsole*, which is shown in the lower box of the screen in Figure 5.27 and also and also in Figure 5.28. The *WCAConsole* comprises two other components. The one on the left side of the *WCAConsole* is a tree view, which shows to the developer the missing elements and/or errors in the implementation of elements for every tag present in the abstract user interface. This tree view is based on the SIG diagram for Accessibility and also shows related tags that should be in an accessible a well-formed user interface.

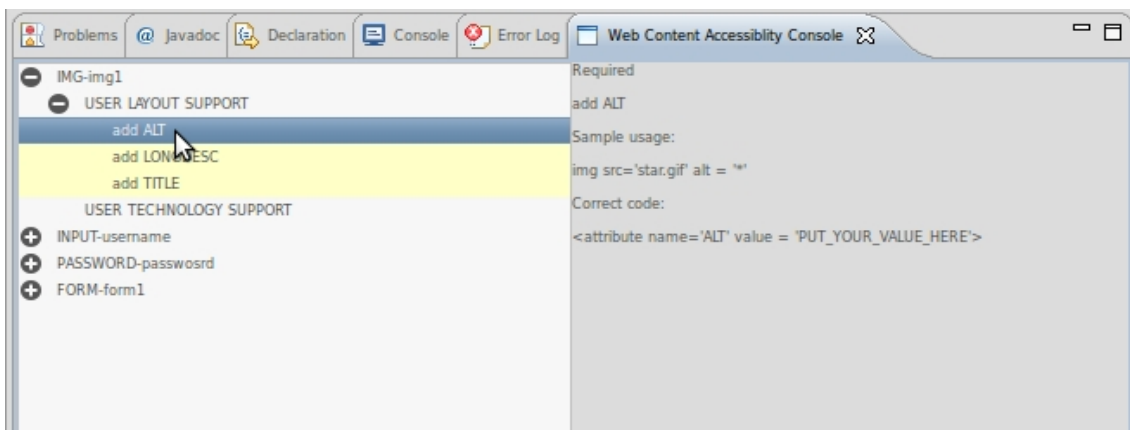


Figure 5.28: The WCAConsole component

The other component on the right side of the *WCAConsole* is a read-only description view, which shows to the developer the following information, for each selected element of the component on the left side:

- **Attribute/Tag condition (Mandatory/Optional):** Indicates to the developer whether the selected element (tag or attribute), is mandatory, as shown in Figure 5.28, or optional, as shown in Figure 5.27, to satisfy the guideline/checkpoint.
- **Action (Add/Remove):** Indicates to the developer the action to perform with the selected element (tag or attribute), if the element should be added (or must be added if the condition is mandatory) to the abstract user interface or removed.

-
- **Sample usage:** Provides to the developer an example on how to properly use in HTML the element (tag or attribute).
 - **Correct code:** Shows to the developer the necessary XML code to insert the element (tag or attribute) in the abstract interface model to commit to the Accessibility guidelines.

5.3.4 Some Insights about the Tool

Our supporting tool, which was conceived prioritizing early Accessibility design, helps developers on the application of our Aspect-Oriented proposal to create user interfaces. The tool provides support at **Step 3** of the design process to discover crosscutting concerns and apply aspects from the knowledge captured about Accessibility requirements in previous stages. Following the approach's basis, the type of support and features covered by the tool can be described as those that usually provide a Computer-Aided Software Engineering (CASE) tool with model-driven properties. As a CASE tool, our supporting tool results helpful in creating models of cases. These models can be developed using reusable components and this is possible because of two reasons. On one hand, the Accessibility guidelines are quite independent from the Web application under development, so there are many cases to which the same Accessibility solution can be applied. Then, recording such recurrent situations (e.g., using patterns) enables to reuse them, which contribute to reduce the development effort when implementing our proposal. On the other hand, the Accessibility aspects as we proposed, could be developed once and be reused in different Web projects. For example, returning to the student's login Web page example in Figure 5.1 (c), establishing a logical tab order for accessing the HTML text field elements for the student ID and password, is an Accessibility concern that forces crosscutting in the implementation. The early identification of this situation allows modeling a reusable Accessibility aspect that is going to be in charge of providing an HTML *tabindex* element for each text field element at the user's layout. Currently, since the function for reusing components is not fully implemented, our tool provides assistance for applying the Accessibility aspects (prescribed by some predefined and stored SIG diagrams) to an abstract user interface model loaded by the designer.

As visible disadvantages of our supporting tool, we believe it is important to highlight the following issues: (i) although the part of the approach that is supported by the tool is completely documented and self-contained within a well-known Web engineering approach, its comprehension requires a prior knowledge of the WCAG 1.0 (or 2.0) guidelines and their specific terminology and also of the AOSD basis; (ii) although the tool helps to transfer Accessibility concerns, 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.

As a final note, we provide our supporting tool aiming to help and, as a consequence, encourage, Web development in designing user interfaces with the Accessibility quality factor in mind.

6. COMPARING OUR PROPOSAL

6.1 Comparison Criteria

In order to compare and discuss the main characteristics of the different approaches, we developed an evaluation framework, as Figure 6.1 shows, which is divided into three main criteria: *Accessibility*, *Design* and *Other criteria*. Each of these topics deals with different issues of the approaches in order to describe them and analyze their strengths and weaknesses when developing an accessible Web site and from a Web engineering perspective. Following, we explain the meaning of the three main criteria through their set of topics.

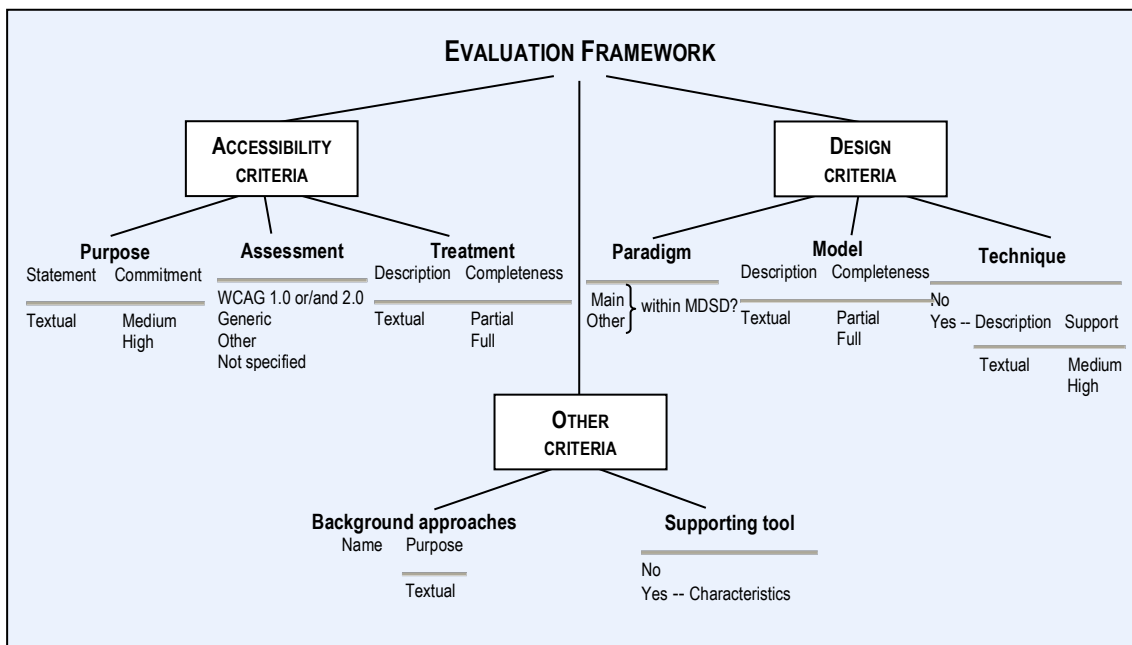


Figure 6.1: Evaluation Framework

Accessibility criteria. We propose these criteria to assess the degree of commitment with Accessibility by evaluating three topics: *purpose*, *assessment* and *treatment*.

We analyze the *purpose* earliest and in the context of the *Accessibility criterion*, because the main focus of our evaluation is on the support given to Accessibility during a Web site development process. Here we evaluate the degree of commitment to Accessibility by considering only two possible scores --i.e. “medium” and “high”, because we have already selected approaches with a certain relation with Web

Accessibility. So a “low” score is out of range for the purpose of this comparison. The differences between the “medium” and “high” scores are set depending on whether Accessibility is the main concern of the approach under consideration.

In addition, because the results can be broadly different depending on the applied reference guidelines, the *assessment* topic aims to establishing the Accessibility conformance criteria applied by the approach. In this case the options are “WCAG” (1.0 or/and 2.0)⁵⁰ [48][49], “generic”, “other” or “not specified”. We are particularly interested on those approaches applying WCAG guidelines because as we said before it is a World-Wide reference normative. We choose “generic” when the approach proposes to consider standards and guidelines develop for several domains⁵¹, such as Accessibility for e-Learning, software, PDF format, Java language, media and Web content, but it does not apply directly to any particularly. An “other” choice states that the approach can apply any “other” practice --e.g. using an ontology, an heuristic, a markup framework, etc., to analyze and treat Web page Accessibility at some stages of the development process --e.g. analysis and design, implementation, etc., and to generate an accessible Web page version. Finally, we decided to include a "not specified" choice for those approaches whose focus is not exclusively on Accessibility, so they do not need to model using a particular Accessibility principle, standard or guideline.

Finally, the *treatment* topic refers to the way Accessibility is handled by the approach. In addition it is important to highlight that many other issues can be taken into account related to Web Accessibility requirements, for example, the type of user disability --i.e. visual, motor, cognitive, deaf, etc. For the *treatment* topic, we are particularly interested in establishing how the approach deals with Accessibility requirements during a Web site development. We believe that Accessibility should be considered as part of the Web design process instead of being evaluated by a post-design repair process. This is the reason why at the analysis of this topic we are mainly interested on establishing the degree of completeness with which the approach handles Accessibility through the stages of the development process. For the purpose of evaluating the *treatment* topic we

⁵⁰ An Overview to WCAG Standards at <http://www.w3.org/WAI/intro/wcag.php>

⁵¹ A list of Accessibility resources at <http://www.accesstechnologiesgroup.com/Resources>

provide a brief description to highlight the stage (or stages) of the design process where the approach concentrates the Accessibility efforts. Then we evaluate the degree of completeness using only two possible scores --i.e. “partial” and “full”, because we selected approaches with a certain relation with modeling Accessibility. So a “low” score is out of range for the purpose of this comparison. We set a “full” score when the approach allows the integration of Accessibility from an early stage, and gives support through the whole Web design process; otherwise, a “partial” score is set.

Design criteria. We propose these criteria to evaluate design issues of the approaches under consideration by using three topics: *paradigm*, *model* and *techniques*.

At the *paradigm* topic, firstly we are interested in identifying if a main paradigm or some other combination of paradigms is used by the approach to deal with Accessibility at design. Since our comparison is framed within Web Engineering (WE) principles, we are also interested in identifying if the approach follows a Model-Driven Software Development (MDSD)⁵² as the core operational *paradigm* to drive the development process. This kind of approaches are usually classified as Model-Driven Web Engineering (MDWE) [31], since they address the different concerns involved in the design and development of a Web application using separate models (such as content, navigation and presentation), and these models can then be supported by model compilers that produce most of the application’s Web pages and logic right from the original models [31]. In consequence, we propose “main”, “other” or “main/other within MDSD” options for the *paradigm* topic. At this point it is important to highlight that we are specially focusing on approaches using the AOSD paradigm to deal with Accessibility at design, because we believe that aspect orientation allows managing Accessibility’s nature properly and as a first-class citizen.

The *model* topic refers to models provided by the approach to deal with Accessibility, and in particular the user interface *model*, since it is at the user’s interface level where

⁵² As we already said, one of the best-known MDSD initiatives is called Model-Driven Architecture (MDA) from OMG at <http://www.omg.org/mda/One>. The MDA framework, together with its related acronym Model-Driven Development (MDD), are registered trademark of the OMG, trademarks within the Unified Modeling Language (UML) is central. Web Engineering is a specific domain in which MDSD can be successfully applied.

Accessibility barriers mostly shown. We introduce in first place a brief description of the basis of the *model* proposed by the approach. It is highly desirable that this *model* fully maps the criteria assumed for treating Accessibility --i.e. the *treatment* and *model* topics must be in concordance and reinforce each other. For the purpose of the *model* topic evaluation, we focus on what elements of an interface model are addressed by the approach and how they are addressed taking into account the fact that these elements are the media for holding an Accessible user-system interaction. We suggest two possible scores, “partial” and “full”, to define the degree of completeness with which the *model* specifies the interface elements. We propose to analyze this degree of *model* completeness from three perspectives: (i) the quantity and granularity of the interface elements considered by the *model*; (ii) the level of detail with which the *model* represents these elements; and further, (iii) the consistency and continuity of a main paradigm with which the approach defines and applies the *model* to deal with the Accessibility of the interface elements. We attach a “full” score, when the model provides the necessary mechanisms for dealing with the Accessibility required by the interface elements. Otherwise, we set a “partial” score. Again, a “low” score is out of range because of the selected approaches for the purpose of the comparison.

Finally, we introduce the *technique* topic to consider the case in which the approach proposes some proprietary technique to complement itself. In the case of an affirmative answer, we provide a brief description of the *technique* and its name --if any, and we also evaluate this *technique* from the perspective of providing support to enrich the design level and to reinforce the Accessibility treatment. When the technique is specifically proposed to provide this kind of support we score it as “high”; otherwise we use a “medium” score.

Other criteria. We propose these criteria to consider two additional topics: *background* and *supporting tool*. We include the *background* topic to consider the case in which the approach takes into account and/or is based-on previous work. Since we believe that the approach’s basis is relevant to the approach’s strength, for each previous work we provide the name and the purpose within its respective approach.

Finally, we introduce the *supporting tool* topic to indicate whether the approach has an associated supporting tool or not. Also it is important the kind of support given and

features covered by the tool in order to contribute to the development of an accessible Web application. Therefore, if the approach provides a tool, some extra considerations about the characteristics of the tool are also given here.

Table 6.1: Accessibility Criteria applied to the six approaches

ACCESSIBILITY CRITERIA					
Approach	Purpose		Assessment	Treatment	
	Statement	Commitment		Description	Completeness
A1 Plessers et al. [35]	Generate the semantic annotations (authoring and mobility Accessibility concepts) for visually impaired users as a by-product of the Web design process.	High	Other	Applies its own developed semantic annotations through a transformation process at the WSDM Implementation Design phase.	Full
A2 Centeno et al. [9]	Provide Accessibility support in a Web composition process managed by a design tool.	High	WCAG (1.0)	Uses a set of compliance rules, which are based on the WCAG 1.0 checkpoints, to ensure accessible Web pages from the composition of accessible HTML snippets.	Partial
A3 Casteleyn et al. [6][7][8]	Engineering Adaptation concerns to extend an existing HERA-based [23] Web application.	Medium	Not specified	Applies aspect-oriented techniques to add Adaptation concerns in a high-level specification and separate from the regular Web process.	Partial
A4 Zimmermann & Vanderheiden [53]	Introduce a process model for Accessibility design that includes well-known software engineering tools.	High	Generic	Develops Personas to support Accessibility requirements and links them to Accessibility guidelines and checkpoints for conformance testing.	Full
A5 Moreno et al. [29][30]	Introduce AWA module that is a domain-specific metamodel of the Web Accessibility domain.	High	WCAG (1.0) (2.0)	Identifies meta-objects following the standard WCAG.	Full
Ours Martin et al.	Early engineering of Accessibility concerns within a Web development process.	High	WCAG (1.0) (2.0)	Models Accessibility as an aspect-oriented concern moving from abstract to concrete architectural views.	Full

6.2 Discussion

At this point we are ready to evaluate the six approaches in accordance with the characteristics defined by our evaluation framework. To make more understandable our explanation, we refer to the approaches as A1 [35], A2 [9], A3 [6][7][8], A4 [53], A5 [29][30] and Ours.

Accessibility criteria. Table 6.1 shows the resultant evaluation of the Accessibility criteria applied to the six approaches. As we can see, A3 is the only one that has a “medium” score at the *purpose* commitment column. We evaluate its grade of commitment to Accessibility with a “medium” score because when analyzing its *purpose* statement, the approach is not focused on the Accessibility concern, but on a wide range of adaptation concerns --i.e. omnipresence, device independence, personalization, localization, privacy, etc.

Accordingly to the fact stated above at the *purpose* commitment column, we set A3 *assessment* column as “not specified”, because the intent of this approach does not make any reference to a particularly Accessibility conformance criteria. On the other hand and since Accessibility is the main intent of A1, A2, A4, A5 and Ours, we set all the approaches’ *purpose* commitment with a “high” score. A2 applies the W3C WCAG 1.0 for Accessibility conformance, and for that reason we set the approach’s *assessment* column with the “WCAG 1.0” option. We set A1 *assessment* column with “other” because this approach applies its own practice to assess Accessibility instead of using a World-Wide reference guideline. A1 uses the DANTE tool [52] to extract visual objects from the page that support navigation. DANTE annotates the objects based on the Web Authoring for Accessibility (WafA)⁵³ travel ontology. We set A4 *assessment* column with “generic” because this approach focuses on accessible design by using scenarios and guidelines, where “guidelines” means Accessibility standards or guidelines that contain interoperability techniques and heuristics for accessible design [52]. Finally, we set A5 and Ours *assessment* column with “WCAG 1.0 and 2.0”. Both approaches originally were conceived to work with WCAG 1.0 checkpoints, but in [29], A5 shows how the proposal can work with WCAG 2.0. Also, we have already finished the migration of Ours to work with the W3C WCAG 2.0 success criteria.

At the *treatment* completeness column, A2 and A3 are the only ones that have “partial” scores but for different reasons. A2 aims to ensure an accessible Web page (or site) during a Web composition process that is managed by an authoring tool. We set a “partial” score at the *treatment* completeness column because the main focus of A2 is not placed on design issues but on evaluation to guarantee that no kind of new

⁵³ Web Authoring for Accessibility (WafA) at <http://augmented.man.ac.uk/ontologies/wafa.owl>

Accessibility barriers can be introduced during a Web composition process. On the other hand, A3 completely illustrates how adaptation concerns can be added to an existing Hera-based Web application at the design level using aspect-oriented techniques. Despite to this fact, we also set a “partial” score for A3 at the *treatment* completeness column because the approach is not focused on adding Accessibility concerns. For A1, A4, A5 and Ours, the *treatment* completeness column is set with “full” scores and this is because these methods allow in different ways, early integration of the Accessibility in the design process. For example, A1 takes the WSDM design models as inputs --i.e. conceptual, navigation and implementation, and generates a set of annotations to improve Accessibility for visually impaired users. A4 defines a new way to take advantage of use cases, scenarios, test cases, personas, guidelines and checkpoints for Accessibility purposes during a design project employing a use case driven methodology. A5 follows the standard WCAG to model concepts and their relationships for AWA-Metamodel at the Compute Independent Model (CIM) of the MDA framework. Finally, Ours focuses on Accessibility requirements early taking advantages of applying AOSD principles to handle them properly as concerns during a Web development process.

Design criteria. Table 6.2 shows the resultant evaluation of the Design criteria. As we can see, we set the *paradigm* column for A1, A3, A5 and Ours as “main within MDSD” because these approaches show commitment and are fully identified with a particular *paradigm* to deal with Accessibility at design within different MDWE approaches. For example, at A1 the DANTE [52] annotation process uses a rule-based mapping model as a foundation *paradigm* to drive the authoring and mobility Accessibility annotations within WSDM [13]. A5 applies the MDA *paradigm* to define a domain-specific metamodel for Accessibility within the OOWS Navigational Model [18]. A3 and Ours apply consistently the AOSD *paradigm* when focusing on solving adaptation and Accessibility concerns, respectively. A3 adds aspect-oriented adaptation engineering to elements of the HERA Application Model [23], while Ours exploits the modeling capabilities of OOHDM Interface Models [36] to inject aspect-oriented Accessibility concerns identified at requirements elicitation. In the cases of A2 and A4, we set their *paradigm* column as “other” because they implement more than one

paradigm to deal with Accessibility. A2 applies a rule-based model as a foundation *paradigm* to drive the conditions under an accessible composition process takes place. But also, A2 proposes the Service-Oriented *paradigm* when using the Web Composition Service Linking System (WSLS) [20] as the authoring tool which enables the process of generating new and accessible Web content. Finally, A4 defines itself like tailored for design project employing a use-case driven methodology, so we say that A4 follows the Object-Oriented *paradigm* but combined with a user profile-based technique called “Personas” [53].

Table 6.2: Design Criteria applied to the six approaches

DESIGN CRITERIA					
Approach	Paradigm	Model		Technique	
		Description	Completeness	Description / Name	Support
A1 Plessers et al. [35]	Main Within MDSD	Identifies the interface elements, which may represent Accessibility barriers for visually impaired users, and annotates these interface elements with the semantic annotations.	Full	Yes Mapping rules established from the relationship between the concepts in the WSDM ontology and DANTE's WafA ontology.	High
A2 Centeno et al. [9]	Other	Works on compositions, which are made of accessible chunks of HTML code, and evaluates these compositions with the compliance rules.	Partial	Yes Compliance rules established for Web compositions and formalized with W3C standards (XPath and XQuery expressions).	Medium
A3 Casteleyn et al. [6][7][8]	Main within MDSD	Selects the elements (units, attributes, relationships, etc.) from an HERA Application Model and injects these elements with the required Adaptation concerns.	Partial	Yes A domain specific language, baptized SEAL, which is custom-made to provide Adaptation support (through a set of constructs for aspects specification) in the context of Hera-S.	Medium
A4 Zimmermann & Vanderheiden [53]	Other	Models primary and secondary Personas to drive the user interface design for each use case.	Partial	No	
A5 Moreno et al. [29][30]	Main within MDSD	Defines several constructs in UML metamodel (MOF) to support the abstraction of Web Accessibility concepts based on WCAG standards.	Full	No	
Ours Martin et al.	Main within MDSD	Identifies Accessibility concerns in Web application requirements and maps them to widgets from abstract and concrete interface models using aspect orientation to meet the WCAG standards.	Full	Yes Three conceptual tools: ▪ UID with Integration Points, ▪ Association Tables, and ▪ SIG template for Accessibility that working together manage Accessibility concerns in an aspect-oriented manner.	High

Albeit for different reasons, A2, A3 and A4 have “partial” scores at the *model* completeness column. A2 is focused on formalizing the Accessibility conditions to be met by a Web composition of prewritten accessible chunks of Web pages, usually called “snippets”. The approach proposes a set of Accessibility extra conditions for a range of possible Web compositions given a pair of accessible HTML snippets. We set a “partial” score for A2 at the *model* completeness column because the approach works over coarse-grained interface elements (existing accessible chunks composed of fine-grained elements as the raw material of the Web composition process) and, as a consequence, A2 focus its design effort on the evaluation over these coarse-grained elements. Also, it is a fact that the Service-Oriented paradigm is not inherent of the basic *model* (which is rule-based) but of the WSLS [20] proposed by the approach as the Accessibility enabled authoring tool for the *model*'s implementation. A3 proposes a general *model* to extend an application with new functionality, considered as adaptation concerns, without having to redesign the entire application. We set a “partial” score for A3 at the *model* completeness column because the approach is focused on showing how the transformations required by an adaption concern can be specified independently from the original presentation level of a Web application using a generic transcoding tool. Therefore the *model* is not concerned on a detailed representation of the interface elements for an accessible design, but on showing how high-level support for adaptation specifications can be realized applying aspect-oriented techniques. A4 proposes a method that draws from the work on Accessibility guidelines and combines them with existing Object-Oriented techniques in software development. The approach encourages the early capture of Accessibility requirements using use cases, personas, scenarios and guidelines, and promotes manual/automatic testing based on test cases and Accessibility checkpoints (derived from guidelines) and expert reviews. In this case we set a “partial” score for A4 at the *model* completeness column because the proposed *model* does not represent these requirements into accessible interface elements at later stages of design. On the other hand, we set “full” scores for A1, A4 and Ours at the *model* completeness column. We set a “full” score for A1 at the *model* completeness column because the approach uses the DANTE's WafA ontology to manage Accessibility of elementary interface elements for visually impaired users. The proposed *model* for the transformation process consists of two steps based on “authoring” and “mobility”

concepts and takes also into account the context of the journey --i.e. the purpose of the user's navigation. The conceptual knowledge captured at the WSDM design process is exploited by the *model* during the transformation because it provides mapping rules between modeling concepts in the WSDM ontology and the authoring concepts from WafA ontology. A4 defines several meta-objects in MOF⁵⁴ to support the abstraction of Web Accessibility concepts and their relationships based on WCAG standards. Although A4 focuses its efforts on the meta-model, we set a "full" score for A4 *model* completeness column because the concepts provided by the approach can become concrete interface elements at the Platform Specific Model (PSM) for the MDA development process. Finally, we set a "full" score for Ours at the *model* completeness column because from the very beginning of the development process the approach focuses on identifying Accessibility requirements and managing them as AOSD concerns, consistently through abstract and concrete widgets of the OOHDM interface models. As a result of this proposal, the approach adds aspect-oriented Accessibility concerns early since requirement elicitation are weaved together using specialized techniques (for a thorough discussion on AOSD principles see [2][28]).

At the *techniques* support column, A4 and A5 do not propose any proprietary *technique* to complement themselves, since they apply existing design tools of software engineering and concepts from the MDA framework, respectively. As we can see at Table 6.2, A2 and A3 have "partial" scores at the *technique* support column. A2 offers a rule-based technique for a safe compound process delivering an accessible Web page from WCAG point of view. A2 has a "medium" score at the *technique* support column because the proposed technique is close to implementation and not focused on giving support to Accessibility design issues. Although the fact that A3 provides a domain specific language called SEAL⁵⁵, we set a "medium" score for A3 at the *technique* support column because the purpose of this proprietary custom-made language is to enrich the design level for adaptation support and not to reinforce the Accessibility

⁵⁴ OMG-MOF The Model-Object Facility at <http://www.omg.org/mof/>

⁵⁵ SEmanatics-based Aspect-oriented Adaptation Language (SEAL) at <http://wise.vub.ac.be/downloads/research/seal/SEALBNF.pdf>

treatment. A1 and Ours have “high” scores at the *technique* support column. A1 provides mapping rules between the concepts in the WSDM ontology and DANTE’s WafA ontology which enable enriching the design level to reinforce the Accessibility propose by taking the WSDM conceptual models as input and annotating them with authoring and mobility concepts. Finally, Ours provides the User-Interaction Diagram (UID) with Integration Points and the Softgoal Interdependency Graph (SIG) template for Accessibility linked by the Association Tables. We set a “high” score for Ours at the *technique* support column because these conceptual tools where specially developed to provide aspect-oriented support at the design level for Accessibility purpose.

Table 6.3: Other Criteria applied to the six approaches

OTHER CRITERIA			
Approach	Background approaches		Supporting tool
	Name	Purpose	
A1 Plessers et al. [35]	DANTE [52]	Used to perform the semantic annotation process of Web pages.	Yes Implements WSDM-DANTE mapping rules to automatically generate semantic annotations.
A2 Centeno et al. [9]	WSLS: A Service-based System for Reuse-Oriented Web engineering [20]	Used as the Accessibility enabled authoring tool.	Yes Shows for some selected rules (based on automatable WCAG checkpoints) how WSLS can afford compliance to these rules.
A3 Casteleyn et al. [6][7][8]	Component-based AMACONT framework [15][16] [32]	Used as the first implementation of a presentation engine for HERA-S.	Yes Integrates SEAL in HydraGen system, which is the latest implementation generation tool for Hera-S.
A4 Zimmermann & Vanderheiden [53]	Use Cases and Personas	Applied to model user profiles linked to their Accessibility requirements.	No
A5 Moreno et al. [29][30]	MDA framework	Applied to support AWA for MDA development process.	Yes Provides AWA-MetamodelEditor for graphical support to AWA-Metamodel.
Ours Martin et al.	User Interaction Diagram (UID) for modeling user-system interaction [43] Softgoal Interdependency Graph (SIG) for modeling non-functional requirements (NFRs) [11][12]	Extended for supporting Accessibility requirements.	Yes Provides a supporting tool to discover crosscutting concerns and apply aspects at the Abstract User Interface model.

Other criteria. Table 6.3 shows the resultant evaluation of the Other criteria. At the *background approach* column, we can see that all the proposals have previous works

and these works are fundamental pieces to the operation of the approaches. A1 founds its work on DANTE's WafA ontology [52] that is applied to enhance the mobility of visually impaired Web users by providing screen readers with extra knowledge to better facilitate the audio presentation of the Web page. A2 uses the WSLS system [20], which is a component-based system applying the service-oriented paradigm to compound, discover and reuse services. The GAC transcoder [16] provided by the ANACONT framework [15] is foundational to A3, since this approach exploits a transcoding tool for making Web application adaptive. A4 applies uses cases and scenarios extended with the "personas" profiling technique for describing Accessibility interfaces' needs and other usage requirements of users with disabilities.

As we can see in Table 6.3, A4 is the only one that has "No" at the *supporting tool* column, while A1, A2, A3, A5 and Ours offer at least some kind of executable implementation of their proposals. A1 presented a combined approach where the mapping rules between the WSDM [13] concepts and the DANTE [52] concepts are implemented. This implementation allows about +/- 70% of the DANTE concepts annotations to be generated automatically without any extra effort from designers. A2 extends the WSLS system [20] to afford compliance to a set of selected rules that guarantee accessible Web composition. The tool seems to give already some promising results since the fact that the WSLS framework is implemented on the top of the .NET framework and gives support to XML technologies. A3 offers a latest implementation of the approach generation tool for HERA-S that integrates SEAL in HydraGen engine⁵⁶ (an implementation generation tool for Hera-S developed externally by the University of Eindhoven), to show their adaptation engineering perspective applying pointcuts and advices expressions. A5 provides the AWA-MetamodelEditor for graphical metamodel support that is based on the Graphical Modeling Framework (GMF)⁵⁷. Finally, Ours provides a tool at Stage 3 of the proposed development process that helps designer and developers to produce accessible interfaces by moving from abstract to concrete architectural views using aspect-orientation --i.e. discovering

⁵⁶ Hydragen: An implementation of Hera-S at <http://www.wis.win.tue.nl/~ksluijs/material/Singh-Master-Thesis-2007.pdf>

⁵⁷ The Eclipse Graphical Modeling Project (GMP) at <http://www.eclipse.org/modeling/gmp/>

crosscutting concerns and applying aspects at the abstract user interface model from knowledge about Accessibility obtained in previous stages. Related to Ours, it is also important to highlight that as we have already indicated in Chapter 4 and later, we have showed with the case study in Chapter 5, there are cases in which we can develop artifacts once and then reused them, as we required. The reuse capabilities of Ours is a main advantage, because propitiates the supporting tool to have a design artifacts repository. For example, and as we have showed in Figures 5.3, 5.4 and 5.5, the Accessibility softgoal for the HTML *image* element can be modeled once and then applied for the SIG instantiation any time is required.

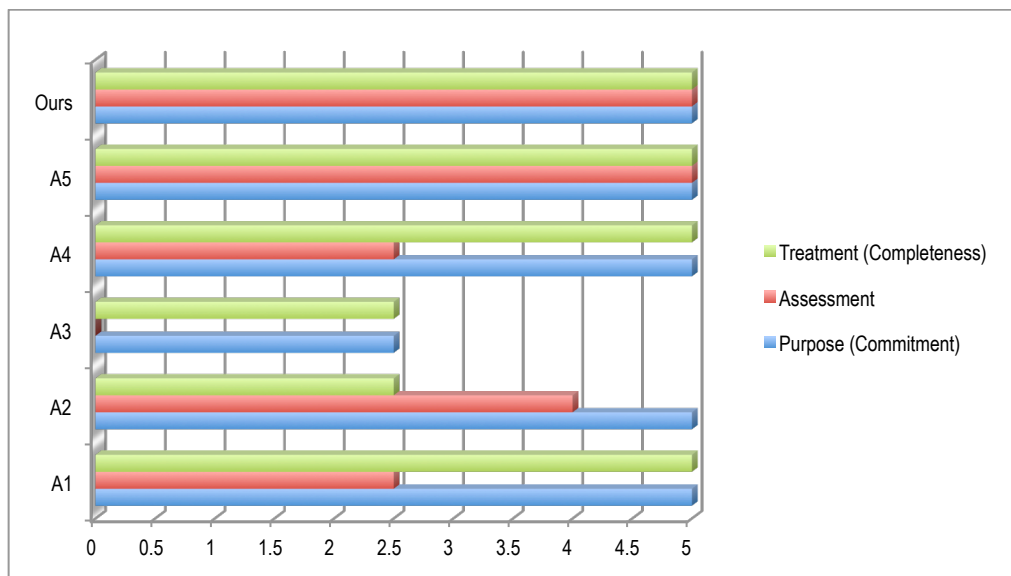


Figure 6.2: Scoring the six approaches for the Accessibility Criteria

To summarize the results of the six approaches' comparison, 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.

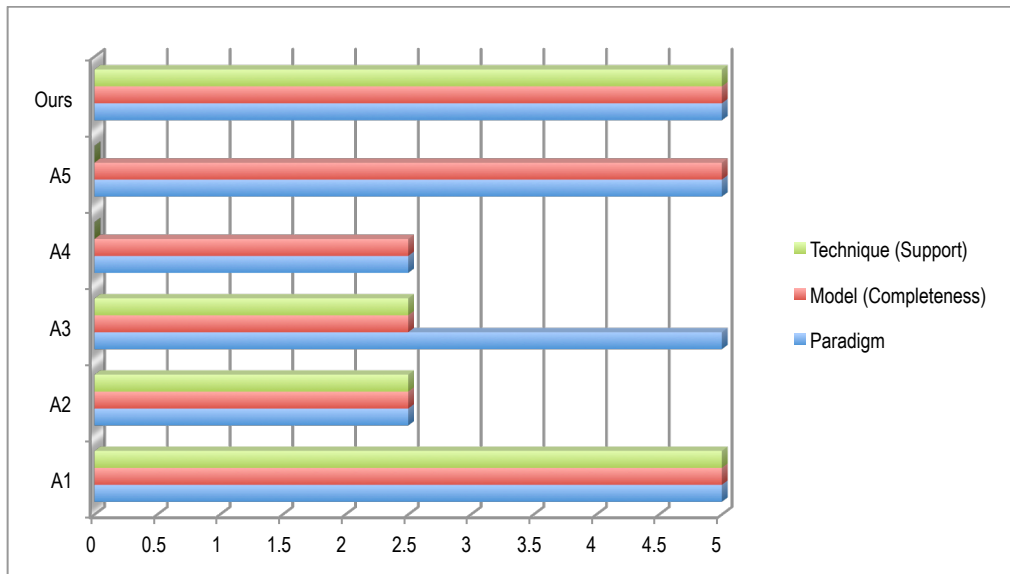


Figure 6.3: Scoring the six approaches for the Design Criteria

To complete this summary, Figure 6.4 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.

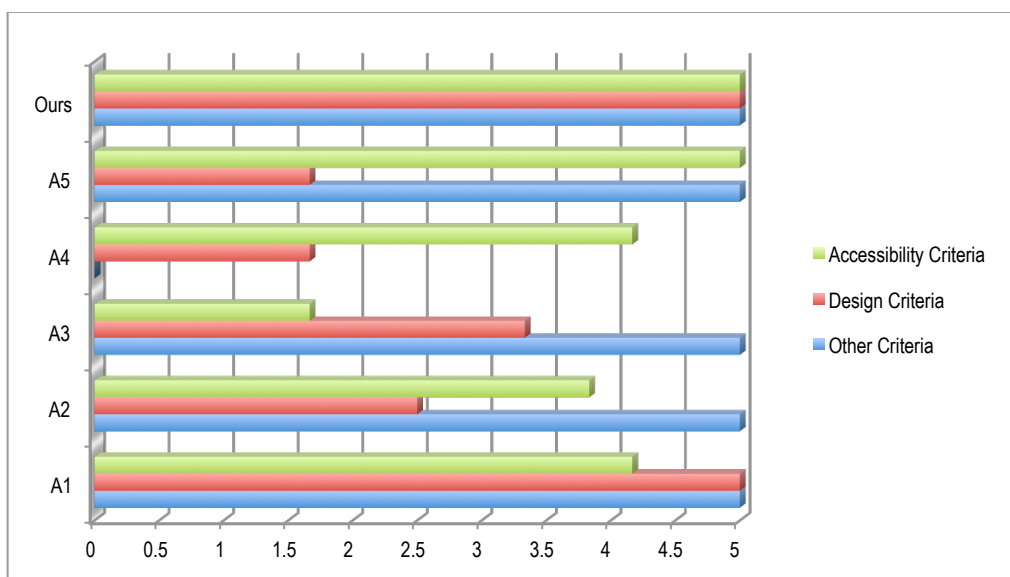


Figure 6.4: The average of scores for the six approaches by Criteria

6.3 Focusing on Ours

We dedicate this Section to provide some extra discussion about our proposal. As we already said, Ours 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 revisit the argument, which we stated when applying Ours in Section 5.2, to the case study in Section 5.1, about alternatives in the navigation path. As Figure 5.1 (d) shows, the case study offers the student two pages to help to the login process in Figure 5.1 (c). 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.

As Figure 6.5 shows, 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 Figure 6.5 shows, 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.

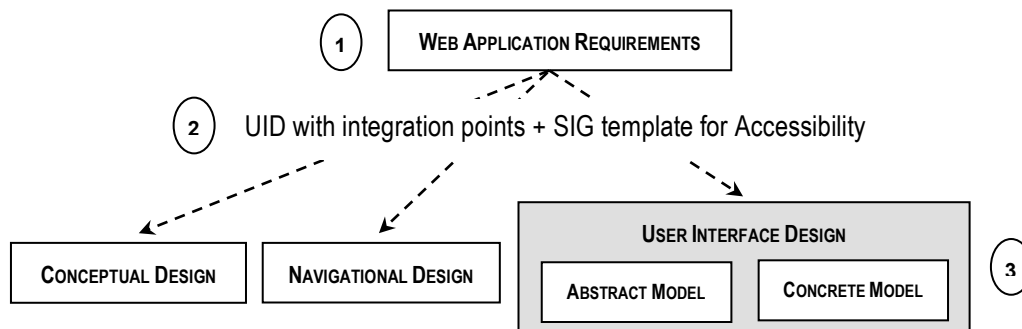


Figure 6.5: Ours within MDSO paradigm

In second place, since Ours is conceived within the MDSO 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.

6.3.1 Migrating to WCAG 2.0

We have already given part of our motivation for applying WCAG 1.0 [45] instead of WCAG 2.0 [46] in Section 3.6.

In first place, and to avoid linking the selection of the WCAG 1.0 only to issues related to the adoption rate in the world, it seems appropriate to highlight that as we are concerned with Accessibility, we have a few quibbles about the decision made on the usefulness of certain checkpoints in the WCAG 2.0 document.

Table 6.1: Association Table for the HTML Control Elements Group using WCAG 2.0

ASPECT	ONTOLOGY WIDGETS (ABSTRACT WIDGETS)	HTML ELEMENTS (CONCRETE WIDGETS)		WCAG 2.0 SUCCESS CRITERIA AND THEIR LEVELS OF CONFORMANCE: [A][AA]OR [AAA]			DESIGN DECISION CLASS related to USER-APPLICATION INTERACTION
				2.4.3 [A]	1.3.1 [A]	4.1.2 [A]	
				D-P ✘	P	D-P	
I. TSCONTROL SIG's USER TECHNOLOGY SUPPORT BRANCH	INDEFINITEVARIABLE	TEXT FIELD	INPUT TEXT...	✓			
		TEXT AREA	TEXTAREA...	✓			
		RELATED CONTROLS	FIELDSET...				
	PREDEFINEDVARIABLE MULTIPLECHOICES	CHECK BOX	INPUT CHECKBOX...	✓			
		MULTIPLE OPTION MENU	SELECT MULTIPLE...	✓			
		RELATED OPTIONS	OPTGROUP...				
	PREDEFINEDVARIABLE SINGLECHOICES	RADIO BUTTON	INPUT RADIO...	✓			
		SIMPLE OPTION MENU	SELECT...	✓			
II. LSCONTROL SIG's USER LAYOUT SUPPORT BRANCH	INDEFINITEVARIABLE	TEXTFIELD	INPUT TEXT...		✓	✓	
		TEXTAREA	TEXTAREA...		✓	✓	
		RELATED CONTROLS	FIELDSET...		✓	✓	
	PREDEFINEDVARIABLE MULTIPLECHOICES	CHECKBOX	INPUT CHECKBOX...		✓	✓	
		MULTIPLE OPTION MENU	SELECT MULTIPLE ...		✓	✓	
		RELATED OPTIONS	OPTGROUP...		✓	✓	
	PREDEFINEDVARIABLE SINGLECHOICES	RADIO BUTTON	INPUT RADIO...		✓	✓	
		SIMPLE OPTION MENU	SELECT...		✓	✓	

For example, WCAG 1.0 provides the checkpoint 12.3 which basically states that the information should be grouped to divide large blocks of information into more manageable groups and this is especially true for the HTML *related controls* element (a set of HTML *text field* elements). The WCAG 2.0 version from January 2006 was also clear on this point, providing the criterion 4.1.3, which basically says that the label of each user interface control in the Web content that accepts input from the user can be

programmatically determined and explicitly associated with the control. Unfortunately, success criterion 4.1.3 has been removed and WCAG 2.0 relies on success criterion 1.3.1 to cover the labeling of related controls, which is not explicit enough to ensure the absence of this important accessibility barrier. In this sense, we fully agree with the statement about the WCAG 2.0 on [41]: “not having any success criteria specifically dealing with forms is certainly a mistake”.

However, aware that the new guidelines and the move to technological neutrality are undoubtedly good, we don't see major inconveniences to upgrade our approach to WCAG 2.0 when necessary. 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 conceptual tools are able to support the success criteria from WCAG 2.0 instead of checkpoints from WCAG 1.0 applying some straightforward redefinitions and adjustments. As an example, Table 7.1 shows the association table for HTML control elements group using WCAG 2.0 success criteria. We highlight that to realize this upgrade we use the comparison provided by W3C-WAI in [49], since there are still some discrepancies at the Accessibility community⁵⁸ when providing mappings between the WCAG 1.0 checkpoints onto the WCAG 2.0 success criteria.

⁵⁸ Examples of these comparisons at <http://www.w3.org/WAI/WCAG20/from10/comparison/>; <http://wipa.org.au/papers/wcag-migration.htm>; <http://www.usability.com.au/resources/wcag2./>

7. CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Web Engineering (WE) is essential to the development of systems that are accessible, usable and acceptable to everybody. Accessibility relies on formulating and promulgating principles, methods and tools of universal design in order to develop applications that are accessible and usable by everybody. Web Engineering starts with a perceived problem and represents a problem solving process, which aims to come up with a model of the implementation of the proposed solution. The discipline of design therefore provides the interface between understanding and creation, and a multitude of acceptable solutions for designing Accessibility may exist, as we summarized in this work. The multiplicity of feasible directions is significant, as it implies a need to choose from among a set of potential alternatives that address different aspects of the problem and provide different levels of solutions with regard to the users' needs. However, as we have already seen in Chapter 2, when we presented and applied related works, there are not so many similar efforts for early design with the principles of Accessibility in mind. In general, the WE proposals do not consider Accessibility as a main driver of the process; which might hinder the identification and evaluation of relevant design elements from early stages.

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. In Chapter 5, we used a real application example of 3 (three) level-deep navigation and 2 (two) optional anchor, to illustrate our ideas and point out the advantages of a clear separation of concerns throughout the development life-cycle. First of all, aspect-orientation capabilities constitute an important driver to efficiently capturing the orthogonal properties that are typical of the Accessibility's nature. Secondly, 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. This is especially important when reasoning about the different properties, because their complexity may be adequately addressed.

In addition, we provided explicit analysis and design techniques aiming at facilitating the capture of early Accessibility concerns. These techniques might be combined with traditional WE methods, which would help introduce and deploy our approach in the industry. 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 design tools to assist developers to design model-driven accessible Web applications. In Section 5.3, we have introduced a supporting tool that is already developed to provide assistance for applying the Accessibility aspects, which avoid crosscutting symptoms when applying the Accessibility concerns prescribed by the SIGs diagrams, to user interface models (abstract and concrete ones).

Since our proposal is strongly linked to the model-driven paradigm, we would like to close this section, reflecting 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-known 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 a 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 [33] 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 could be developed once and be reused in different Web projects. This reinforces what we have already said in Sections 4.1, 5.2 and 6.2 for SIGs diagrams, about how our proposal propitiates the reuse of design artifacts.

7.2 Future Work

Considering the extensibility of our approach, it is important to highlight, that although in this work we 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. The reason why we use visual impairment is based on the fact that ensuring Accessibility requirements for blind people, to a certain extent, covers Accessibility requirements for other disabilities. For example, the checkpoint 1.1 of the WCAG 1.0 establishes that text equivalents must be written to convey all essential content; therefore ensuring compliance to checkpoint 1.1 is vital for visually impaired users. The fact is that the absence of non-text equivalents represents a critical Accessibility barrier for people with visual disabilities, but ensuring text-equivalent also improves Accessibility for users with deafness, cognitive and learning disabilities. So, we considered the treatment of visual impairments as a good starting point.

Finally, we should further validate our proposal working with WCAG 2.0 [46] beyond the case study, which we used in Section 5.1 to apply our Aspect-Oriented approach, and make some comparisons between case studies that we have been applying during the validating process. To do so, we are currently following two different but related paths: (i) migrating the supporting tool to work with the WCAG 2.0 version of our approach and extending the tool's functionality to cover all the approach's development

process to propitiate industry adoption and, (ii) 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.

7.3 Publications related to this Thesis

The partial results obtained during this investigation have been published and presented in different forums. Following, in sections 7.3.1, 7.3.2, 7.3.3 and 7.3.4, we present some of these work ordered according to whether they correspond to Journals, Book Chapters, International Conferences and National Conferences, respectively.

7.3.1 Journals

- **(WWWJ 2010) World Wide Web: Internet and Web Information Systems Journal⁵⁹**

Title: *Engineering Accessible Web Applications. An Aspect-Oriented Approach*

Authors: Adriana Martín, Gustavo Rossi, Alejandra Cechich, Silvia Gordillo

In: World Wide Web: Internet and Web Information Systems Journal (WWWJ)

ISBN: 978-1-59904-847-5

Volume-Number: 13 (4)

Pages: 419-440

DOI: 10.1007/s11280-010-0091-3

Abstracted/Indexed in: Academic OneFile, ACM Computing Reviews, ACM Digital Library, Cabell's, Computer and Communication Security Abstracts, Computer Science Index, Current Abstracts, Current Contents/Engineering, Computing and Technology, DBLP, EBSCO, EI-Compendex, Gale, Google Scholar, INSPEC, io-port.net, Journal Citation Reports/Science Edition, OCLC, Science Citation Index Expanded (SciSearch), SCOPUS, Summon by Serial Solutions.

Impact Factor: 1.0

⁵⁹ (WWWJ 2010) at

<http://www.informatik.uni-trier.de/~ley/db/journals/www/www13.html#MartinRCG10>

7.3.2 Book Chapters

- **(Book Chapter 2008) Handbook of Research on Web Information Systems Quality⁶⁰**

Title: *Comparing Approaches to Web Accessibility Assessment*

Authors: Adriana Martín, Alejandra Cechich, Gustavo Rossi

In: Coral Calero, M^a Ángeles Moraga and Mario Piattini (Editors) Handbook of Research on Web Information Systems Quality, 2008

ISBN₁₃: 9781599048475 - ISBN₁₀: 1599048477 - ISBN₁₃: 9781599048482

Publisher: IGI Global

Chapter: XI

Pages: 181-205

DOI: 10.4018/978-1-59904-847-5.ch011

7.3.3 International Conferences

- **(W4A 2011) World Wide Web 8th International Cross-Disciplinary Conference on Web Accessibility⁶¹**

Title: *Accessibility at Early Stages: Insights from the Designer Perspective*

Authors: Adriana Martín, Alejandra Cechich, Gustavo Rossi

In: Proceedings of 8th International Cross-Disciplinary Conference on Web Accessibility (W4A), Hyderabad, Andhra Pradesh, India, 2011

ISBN: 978-1-4503-0476-4

Publisher: ACM

Pages: 9

DOI: 10.1145/1969289.1969302

⁶⁰ (Chapter XI) at <http://www.igi-global.com/bookstore/chapter.aspx?titleid=21973>

⁶¹ (W4A 2011) at <http://www.informatik.uni-trier.de/~ley/db/conf/w4a/w4a2011.html#MartinCR11>

-
- **(ICSEA 2010) 5th International Conference on Software Engineering Advances⁶²**

Title: *Supporting an Aspect-Oriented Approach to Web Accessibility Design*

Authors: Adriana Martín, Rafaela Mazalú, Alejandra Cechich

In: Proceedings of 5th International Conference on Software Engineering Advances (ICSEA), Nice, France, 2010

ISBN: 978-0-7695-4144-0

Publisher: IEEE

Pages: 20-25

DOI: 10.1109/ICSEA.2010.10

- **(LA-WEB 2007) Fifth Latin American Web Congress⁶³**

Title: *A Three-Layered Approach to Model Web Accessibility for Blind Users*

Authors: Adriana Martín, Alejandra Cechich, Silvia Gordillo, Gustavo Rossi

In: Proceedings of 5th Latin American Web Congress (LA-WEB), Santiago de Chile, Chile, 2007

ISBN: 0-7695-3008-7

Publisher: IEEE

Pages: 76-83

DOI: 10.1109/LA-WEB.2007.56

7.3.4 National Conferences

- **(ASSE 2011) 12th Argentine Symposium on Software Engineering⁶⁴**

Title: *AO -WAD: A Supporting Tool to Aspect-Oriented Web Accessibility Design*

Authors: Rafaela Mazalú, Fabián Huenuman, Adriana Martín, Alejandra Cechich

In: Proceedings of 12th Argentine Symposium on Software Engineering (ASSE), Córdoba, Argentina, 2011

⁶² (ICSEA 2010) at <http://www.informatik.uni-trier.de/~ley/db/conf/icsea/icsea2010.html#MartinMC10>

⁶³ (LA-WEB 2007) at <http://www.informatik.uni-trier.de/~ley/db/conf/la-web/la-web2007.html#MartinCGR07>

⁶⁴ (ASSE 2011) at <http://www.40jaiio.org.ar/node/85>

ISBN: 1850-2792

Pages: 108-119

- **(CACIC 2009) XV Congreso Argentino en Ciencias de la Computación**⁶⁵

Title: *Hacia una Herramienta de Soporte para el Modelado Web con Accesibilidad*

Authors: Rafaela Mazalu, Adriana Martín, Alejandra Cechich

In: Proceedings of XV Congreso Argentino en Ciencias de la Computación (CACIC), San Salvador de Jujuy, Jujuy, Argentina, 2009

ISBN: 978-897-24068-4-1

Pages: 663-672

7.4 Other related Publications

Following, in sections 7.4.1 and 7.4.2, we present other related work ordered according to whether they correspond to International Conferences and National Conferences, respectively.

7.4.1 International Conferences

- **(CibSE 2010) XIII Congreso Americano en “Software Engineering**⁶⁶

Title: *Diseño de Interfaces Guiado por Restricciones de Accesibilidad Web*

Authors: Brenda Bustos, Adriana Martín, Alejandra Cechich

In: Proceedings of XIII Congreso Americano en “Software Engineering” (CibSE), Universidad del Azuay, Cuenca, Ecuador, 2010

Pages: 229-242

⁶⁵ (CACIC 2009) http://redunci.info.unlp.edu.ar/files/indice_Cacic_2009.pdf

⁶⁶ (CibSE 2010) at http://www.uazuay.edu.ec/cibse/2_sessions.php

- **(LA-WEB 2005) Third Latin American Web Congress⁶⁷**

Title: *A Model-Driven Reengineering Approach to Web Site Personalization*

Authors: Adriana Martín, Alejandra Cechich

In: Proceedings of 3rd Latin American Web Congress (LA-WEB), Buenos Aires, Argentina, 2005

ISBN: 0-7695-2471-0

Publisher: IEEE

Pages: 14-22

DOI: 10.1109/LAWEB.2005.5

7.4.2 National Conferences

- **(CACIC 2008) XIV Congreso Argentino en Ciencias de la Computación**

Title: *Extendiendo MVC para Diseñar Interfaces de Usuario Accesibles*

Authors: Brenda Bustos Torres, Adriana Martín, Alejandra Cechich

In: Proceedings of XIV Congreso Argentino en Ciencias de la Computación (CACIC), Chilecito, La Rioja, Argentina, 2008

ISBN: 978-987-24611-0-2

Pages: 1163-1174

⁶⁷ (LA-WEB 2005) at

<http://www.informatik.uni-trier.de/~ley/db/conf/la-web/la-web2005.html#MartinC05>

APPENDIX I



Web Content Accessibility Guidelines (WCAG) 1.0

W3C Recommendation 5-May-1999

This version:

<http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505>
(plain text, PostScript, PDF, gzip tar file of HTML, zip archive of HTML)

Latest version:

<http://www.w3.org/TR/WAI-WEBCONTENT>

Previous version:

<http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990324>

Editors:

Wendy Chisholm, Trace R & D Center, University of Wisconsin -- Madison
Gregg Vanderheiden, Trace R & D Center, University of Wisconsin -- Madison
Ian Jacobs, W3C

Copyright W3C (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

These guidelines explain how to make Web content [p. 26] accessible to people with disabilities. The guidelines are intended for all Web content developers [p. 26] (page authors and site designers) and for developers of authoring tools [p. 25]. The primary goal of these guidelines is to promote accessibility. However, following them will also make Web content more available to *all* users, whatever user agent [p. 30] they are using (e.g., desktop browser, voice browser, mobile phone, automobile-based personal computer, etc.) or constraints they may be operating under (e.g., noisy surroundings, under- or over-illuminated rooms, in a hands-free environment, etc.). Following these guidelines will also help people find information on the Web more quickly. These guidelines do not discourage content developers from using images, video, etc., but rather explain how to make multimedia content more accessible to a wide audience.

This is a reference document for accessibility principles and design ideas. Some of the strategies discussed in this document address certain Web internationalization and mobile access concerns. However, this document focuses on accessibility and does not fully address the related concerns of other W3C Activities. Please consult the W3C Mobile Access Activity home page and the W3C Internationalization Activity home page for more information.

This document is meant to be stable and therefore does not provide specific information about browser support for different technologies as that information

changes rapidly. Instead, the Web Accessibility Initiative (WAI) Web site provides such information (refer to [WAI-UA-SUPPORT] [p. 33]).

This document includes an appendix that organizes all of the checkpoints [p. 7] by topic and priority. The checkpoints in the appendix link to their definitions in the current document. The topics identified in the appendix include images, multimedia, tables, frames, forms, and scripts. The appendix is available as either a tabular summary of checkpoints or as a simple list of checkpoints.

A separate document, entitled "Techniques for Web Content Accessibility Guidelines 1.0" ([TECHNIQUES] [p. 33]), explains how to implement the checkpoints defined in the current document. The Techniques Document discusses each checkpoint in more detail and provides examples using the Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), Synchronized Multimedia Integration Language (SMIL), and the Mathematical Markup Language (MathML). The Techniques Document also includes techniques for document validation and testing, and an index of HTML elements and attributes (and which techniques use them). The Techniques Document has been designed to track changes in technology and is expected to be updated more frequently than the current document. **Note.** Not all browsers or multimedia tools may support the features described in the guidelines. In particular, new features of HTML 4.0 or CSS 1 or CSS 2 may not be supported.

"Web Content Accessibility Guidelines 1.0" is part of a series of accessibility guidelines published by the Web Accessibility Initiative. The series also includes User Agent Accessibility Guidelines ([WAI-USERAGENT] [p. 33]) and Authoring Tool Accessibility Guidelines ([WAI-AUTOOLS] [p. 33]).

Status of this document

This document has been reviewed by W3C Members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another documents. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and universality of the Web.

The English version of this specification is the only normative version. However, for translations in other languages see <http://www.w3.org/WAI/GL/WAI-WEBCONTENT-TRANSLATIONS>.

The list of known errors in this document is available at <http://www.w3.org/WAI/GL/WAI-WEBCONTENT-ERRATA>. Please report errors in this document to wai-wcag-editor@w3.org.

A list of current W3C Recommendations and other technical documents can be found at <http://www.w3.org/TR>.

This document has been produced as part of the W3C Web Accessibility Initiative. The goal of the Web Content Guidelines Working Group is discussed in the Working Group charter.

1. Introduction

For those unfamiliar with accessibility issues pertaining to Web page design, consider that many users may be operating in contexts very different from your own:

- They may not be able to see, hear, move, or may not be able to process some types of information easily or at all.
- They may have difficulty reading or comprehending text.
- They may not have or be able to use a keyboard or mouse.
- They may have a text-only screen, a small screen, or a slow Internet connection.
- They may not speak or understand fluently the language in which the document is written.
- They may be in a situation where their eyes, ears, or hands are busy or interfered with (e.g., driving to work, working in a loud environment, etc.).
- They may have an early version of a browser, a different browser entirely, a voice browser, or a different operating system.

Content developers must consider these different situations during page design. While there are several situations to consider, each accessible design choice generally benefits several disability groups at once and the Web community as a whole. For example, by using style sheets [p. 29] to control font styles and eliminating the FONT element, HTML authors will have more control over their pages, make those pages more accessible to people with low vision, and by sharing the style sheets, will often shorten page download times for all users.

The guidelines discuss accessibility issues and provide accessible design solutions. They address typical scenarios (similar to the font style example) that may pose problems for users with certain disabilities. For example, the first guideline [p. 10] explains how content developers can make images accessible. Some users may not be able to see images, others may use text-based browsers that do not support images, while others may have turned off support for images (e.g., due to a slow Internet connection). The guidelines do not suggest avoiding images as a way to improve accessibility. Instead, they explain that providing a text equivalent [p. 27] of the image will make it accessible.

How does a text equivalent make the image accessible? Both words in "text equivalent" are important:

- Text content can be presented to the user as synthesized speech, braille, and visually-displayed text. Each of these three mechanisms uses a different sense -- ears for synthesized speech, tactile for braille, and eyes for visually-displayed text -- making the information accessible to groups representing a variety of sensory and other disabilities.
- In order to be useful, the text must convey the same function or purpose as the image. For example, consider a text equivalent for a photographic image of the Earth as seen from outer space. If the purpose of the image is mostly that of decoration, then the text "Photograph of the Earth as seen from outer space" might fulfill the necessary function. If the purpose of the photograph is to illustrate specific information about world geography, then the text equivalent should convey that information. If the photograph has been designed to tell the user to select the image (e.g., by clicking on it) for information about the earth, equivalent text would be "Information about the Earth". Thus, if the text conveys the same function or purpose for the user with a disability as the image does for other users, then it can be considered a text equivalent.

Note that, in addition to benefitting users with disabilities, text equivalents can help all users find pages more quickly, since search robots can use the text when indexing the pages.

While Web content developers must provide text equivalents for images and other multimedia content, it is the responsibility of user agents [p. 30] (e.g., browsers and assistive technologies such as screen readers [p. 29], braille displays [p. 26] , etc.) to present the information to the user.

Non-text equivalents of text (e.g., icons, pre-recorded speech, or a video of a person translating the text into sign language) can make documents accessible to people who may have difficulty accessing written text, including many individuals with cognitive disabilities, learning disabilities, and deafness. Non-text equivalents of text can also be helpful to non-readers. An auditory description [p. 28] is an example of a non-text equivalent of visual information. An auditory description of a multimedia presentation's visual track benefits people who cannot see the visual information.

2. Themes of Accessible Design

The guidelines address two general themes: ensuring graceful transformation, and making content understandable and navigable.

2.1 Ensuring Graceful Transformation

By following these guidelines, content developers can create pages that transform gracefully. Pages that transform gracefully remain accessible despite any of the constraints described in the introduction [p. 5] , including physical, sensory, and cognitive disabilities, work constraints, and technological barriers. Here are some keys to designing pages that transform gracefully:

- Separate structure from presentation (refer to the difference between content, structure, and presentation [p. 26]).
- Provide text (including text equivalents [p. 27]). Text can be rendered in ways that are available to almost all browsing devices and accessible to almost all users.
- Create documents that work even if the user cannot see and/or hear. Provide information that serves the same purpose or function as audio or video in ways suited to alternate sensory channels as well. This does not mean creating a prerecorded audio version of an entire site to make it accessible to users who are blind. Users who are blind can use screen reader [p. 29] technology to render all text information in a page.
- Create documents that do not rely on one type of hardware. Pages should be usable by people without mice, with small screens, low resolution screens, black and white screens, no screens, with only voice or text output, etc.

The theme of graceful transformation is addressed primarily by guidelines 1 to 11.

2.2 Making Content Understandable and Navigable

Content developers should make content understandable and navigable. This includes not only making the language clear and simple, but also providing understandable mechanisms for navigating within and between pages. Providing navigation tools and orientation information in pages will maximize accessibility and usability. Not all users can make use of visual clues such as image maps, proportional scroll bars, side-by-

side frames, or graphics that guide sighted users of graphical desktop browsers. Users also lose contextual information when they can only view a portion of a page, either because they are accessing the page one word at a time (speech synthesis or braille display [p. 26]), or one section at a time (small display, or a magnified display). Without orientation information, users may not be able to understand very large tables, lists, menus, etc.

The theme of making content understandable and navigable is addressed primarily in guidelines 12 to 14.

3. How the Guidelines are Organized

This document includes fourteen guidelines, or general principles of accessible design. Each guideline includes:

- The guideline number.
- The statement of the guideline.
- Guideline navigation links. Three links allow navigation to the next guideline (right arrow icon), the previous guideline (left arrow icon), or the current guideline's position in the table of contents (up arrow icon).
- The rationale behind the guideline and some groups of users who benefit from it.
- A list of checkpoint definitions.

The checkpoint definitions in each guideline explain how the guideline applies in typical content development scenarios. Each checkpoint definition includes:

- The checkpoint number.
- The statement of the checkpoint.
- The priority of the checkpoint. Priority 1 checkpoints are highlighted through the use of style sheets.
- Optional informative notes, clarifying examples, and cross-references to related guidelines or checkpoints.
- A link to a section of the Techniques Document ([TECHNIQUES] [p. 33]) where implementations and examples of the checkpoint are discussed.

Each checkpoint is intended to be specific enough so that someone reviewing a page or site may verify that the checkpoint has been satisfied.

3.1 Document conventions

The following editorial conventions are used throughout this document:

- Element names are in uppercase letters.
- Attribute names are quoted in lowercase letters.
- Links to definitions are highlighted through the use of style sheets.

4. Priorities

Each checkpoint has a priority level assigned by the Working Group based on the checkpoint's impact on accessibility.

[Priority 1] A Web content developer **must** satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web documents.

[Priority 2] A Web content developer **should** satisfy this checkpoint. Otherwise, one or more groups will find it difficult to access information in the document. Satisfying this checkpoint will remove significant barriers to accessing Web documents.

[Priority 3] A Web content developer **may** address this checkpoint. Otherwise, one or more groups will find it somewhat difficult to access information in the document. Satisfying this checkpoint will improve access to Web documents.

Some checkpoints specify a priority level that may change under certain (indicated) conditions.

5. Conformance

This section defines three levels of conformance to this document:

- **Conformance Level "A"**: all Priority 1 checkpoints are satisfied;
- **Conformance Level "Double-A"**: all Priority 1 and 2 checkpoints are satisfied;
- **Conformance Level "Triple-A"**: all Priority 1, 2, and 3 checkpoints are satisfied;

Note. Conformance levels are spelled out in text so they may be understood when rendered to speech.

Claims of conformance to this document must use one of the following two forms. Form 1: Specify:

- The guidelines title: "Web Content Accessibility Guidelines 1.0"
- The guidelines URI: <http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505>
- The conformance level satisfied: "A", "Double-A", or "Triple-A".
- The scope covered by the claim (e.g., page, site, or defined portion of a site.).

Example of Form 1:

This page conforms to W3C's "Web Content Accessibility Guidelines 1.0", available at <http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505>, level Double-A.

Form 2: Include, on each page-claiming conformance, one of three icons provided by W3C and link the icon to the appropriate W3C explanation of the claim. Information about the icons and how to insert them in pages is available at [WCAG-ICONS] [p. 33].

6. Web Content Accessibility Guidelines

Guideline 1. Provide equivalent alternatives to auditory and visual content.

Provide content that, when presented to the user, conveys essentially the same function or purpose as auditory or visual content.

Although some people cannot use images, movies, sounds, applets, etc. directly, they may still use pages that include equivalent [p. 27] information to the visual or auditory content. The equivalent information must serve the same purpose as the visual or auditory content. Thus, a text equivalent for an image of an upward arrow that links to a table of contents could be "Go to table of contents". In some cases, an equivalent should also describe the appearance of visual content (e.g., for complex charts, billboards, or diagrams) or the sound of auditory content (e.g., for audio samples used in education).

This guideline emphasizes the importance of providing text equivalents [p. 27] of

non-text content (images, pre-recorded audio, video). The power of text equivalents lies in their capacity to be rendered in ways that are accessible to people from various disability groups using a variety of technologies. Text can be readily output to speech synthesizers and braille displays [p. 26] , and can be presented visually (in a variety of sizes) on computer displays and paper. Synthesized speech is critical for individuals who are blind and for many people with the reading difficulties that often accompany cognitive disabilities, learning disabilities, and deafness. Braille is essential for individuals who are both deaf and blind, as well as many individuals whose only sensory disability is blindness. Text displayed visually benefits users who are deaf as well as the majority of Web users.

Providing non-text equivalents (e.g., pictures, videos, and pre-recorded audio) of text is also beneficial to some users, especially nonreaders or people who have difficulty reading. In movies or visual presentations, visual action such as body language or other visual cues may not be accompanied by enough audio information to convey the same information. Unless verbal descriptions of this visual information are provided, people who cannot see (or look at) the visual content will not be able to perceive it.

Checkpoints:

1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes:* images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [Priority 1]

For example, in HTML:

- Use "alt" for the IMG, INPUT, and APPLET elements, or provide a text equivalent in the content of the OBJECT and APPLET elements.
- For complex content (e.g., a chart) where the "alt" text does not provide a complete text equivalent, provide an additional description using, for example, "longdesc" with IMG or FRAME, a link inside an OBJECT element, or a description link [p. 28].
- For image maps, either use the "alt" attribute with AREA, or use the MAP element with A elements (and other text) as content.

Refer also to checkpoint 9.1 and checkpoint 13.10.

Techniques for checkpoint 1.1

1.2 Provide redundant text links for each active region of a server-side image map. [Priority 1]

Refer also to checkpoint 1.5 and checkpoint 9.1.

Techniques for checkpoint 1.2

1.3 Until user agents [p. 30] can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation. [Priority 1]

Synchronize the auditory description [p. 28] with the audio track as per checkpoint 1.4. Refer to checkpoint 1.1 for information about textual equivalents for visual information.

Techniques for checkpoint 1.3

1.4 For any time-based multimedia presentation (e.g., a movie or animation),

synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation. [Priority 1]

Techniques for checkpoint 1.4

1.5 Until user agents [p. 30] render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map. [Priority 3]

Refer also to checkpoint 1.2 and checkpoint 9.1.

Techniques for checkpoint 1.5

Guideline 2. Don't rely on color alone.

Ensure that text and graphics are understandable when viewed without color.

If color alone is used to convey information, people who cannot differentiate between certain colors and users with devices that have non-color or non-visual displays will not receive the information. When foreground and background colors are too close to the same hue, they may not provide sufficient contrast when viewed using monochrome displays or by people with different types of color deficits.

Checkpoints:

2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1]

Techniques for checkpoint 2.1

2.2 Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen. [Priority 2 for images, Priority 3 for text].

Techniques for checkpoint 2.2

Guideline 3. Use markup and style sheets and do so properly.

Mark up documents with the proper structural elements. Control presentation with style sheets rather than with presentation elements and attributes.

Using markup improperly -- not according to specification -- hinders accessibility. Misusing markup for a presentation effect (e.g., using a table for layout or a header to change the font size) makes it difficult for users with specialized software to understand the organization of the page or to navigate through it. Furthermore, using presentation markup rather than structural markup to convey structure (e.g., constructing what looks like a table of data with an HTML PRE element) makes it difficult to render a page intelligibly to other devices (refer to the description of difference between content, structure, and presentation [p. 26]).

Content developers may be tempted to use (or misuse) constructs that achieve a desired formatting effect on older browsers. They must be aware that these practices cause accessibility problems and must consider whether the formatting effect is so critical as to warrant making the document inaccessible to some users.

At the other extreme, content developers must not sacrifice appropriate markup because a certain browser or assistive technology does not process it correctly. For example, it is appropriate to use the TABLE element in HTML to mark up tabular

information [p. 29] even though some older screen readers may not handle side-by-side text correctly (refer to checkpoint 10.3). Using TABLE correctly and creating tables that transform gracefully (refer to guideline 5) makes it possible for software to render tables other than as two-dimensional grids.

Checkpoints:

- 3.1 When an appropriate markup language exists, use markup rather than images to convey information. [Priority 2]
For example, use MathML to mark up mathematical equations, and style sheets [p. 29] to format text and control layout. Also, avoid using images to represent text -- use text and style sheets instead. Refer also to guideline 6 and guideline 11.
Techniques for checkpoint 3.1
- 3.2 Create documents that validate to publish formal grammars. [Priority 2] For example, include a document type declaration at the beginning of a document that refers to a published DTD (e.g., the strict HTML 4.0 DTD).
Techniques for checkpoint 3.2
- 3.3 Use style sheets to control layout and presentation. [Priority 2] For example, use the CSS 'font' property instead of the HTML FONT element to control font styles.
Techniques for checkpoint 3.3
- 3.4 Use relative rather than absolute units in markup language attribute values and style sheet property values. [Priority 2]
For example, in CSS, use 'em' or percentage lengths rather than 'pt' or 'cm', which are absolute units. If absolute units are used, validate that the rendered content is usable (refer to the section on validation [p. 24]).
Techniques for checkpoint 3.4
- 3.5 Use header elements to convey document structure and use them according to specification. [Priority 2]
For example, in HTML, use H2 to indicate a subsection of H1. Do not use headers for font effects.
Techniques for checkpoint 3.5
- 3.6 Mark up lists and list items properly. [Priority 2] For example, in HTML, nest OL, UL, and DL lists properly. Techniques for checkpoint 3.6
- 3.7 Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2]
For example, in HTML, use the Q and BLOCKQUOTE elements to markup short and longer quotations, respectively.
Techniques for checkpoint 3.7

Guideline 4. Clarify natural language usage

Use markup that facilitates pronunciation or interpretation of abbreviated or foreign text.

When content developers mark up natural language changes in a document, speech synthesizers and braille devices can automatically switch to the new language, making the document more accessible to multilingual users. Content developers should identify the predominant natural language [p. 29] of a document's content (through markup or HTTP headers). Content developers should also provide expansions of abbreviations

and acronyms.

In addition to helping assistive technologies, natural language markup allows search engines to find key words and identify documents in a desired language. Natural language markup also improves readability of the Web for all people, including those with learning disabilities, cognitive disabilities, or people who are deaf.

When abbreviations and natural language changes are not identified, they may be indecipherable when machine-spoken or brailled.

Checkpoints:

4.1 Clearly identify changes in the natural language of a document's text and any text equivalents [p. 27] (e.g., captions). [Priority 1]

For example, in HTML use the "lang" attribute. In XML, use "xml:lang".

Techniques for checkpoint 4.1

4.2 Specify the expansion of each abbreviation or acronym in a document where it first occurs. [Priority 3]

For example, in HTML, use the "title" attribute of the ABBR and ACRONYM elements. Providing the expansion in the main body of the document also helps document usability.

Techniques for checkpoint 4.2

4.3 Identify the primary natural language of a document. [Priority 3] For example, in HTML set the "lang" attribute on the HTML element. In XML, use "xml:lang".

Server operators should configure servers to take advantage of HTTP content negotiation mechanisms ([RFC2068] [p. 33] , section 14.13) so that clients can automatically retrieve documents of the preferred language.

Techniques for checkpoint 4.3

Guideline 5. Create tables that transform gracefully.

Ensure that tables have necessary markup to be transformed by accessible browsers and other user agents.

Tables should be used to mark up truly tabular information [p. 29] ("data tables"). Content developers should avoid using them to lay out pages ("layout tables"). Tables for any use also present special problems to users of screen readers [p. 29] (refer to checkpoint 10.3). Some user agents [p. 30] allow users to navigate among table cells and access header and other table cell information. Unless marked-up properly, these tables will not provide user agents with the appropriate information. (Refer also to guideline 3.)

The following checkpoints will directly benefit people who access a table through auditory means (e.g., a screen reader or an automobile-based personal computer) or who view only a portion of the page at a time (e.g., users with blindness or low vision using speech output or a braille display, [p. 26] or other users of devices with small displays, etc.).

Checkpoints:

5.1 For data tables, identify row and column headers. [Priority 1] For example, in HTML, use TD to identify data cells and TH to identify headers.

Techniques for checkpoint 5.1

-
- 5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells. [Priority 1]
For example, in HTML, use THEAD, TFOOT, and TBODY to group rows, COL and COLGROUP to group columns, and the "axis", "scope", and "headers" attributes, to describe more complex relationships among data.
Techniques for checkpoint 5.2
- 5.3 Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version [p. 28]). [Priority 2]
Note. Once user agents [p. 30] support style sheet positioning, tables should not be used for layout. Refer also to checkpoint 3.3.
Techniques for checkpoint 5.3
- 5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]
For example, in HTML do not use the TH element to cause the content of a (non-table header) cell to be displayed centered and in bold.
Techniques for checkpoint 5.4
- 5.5 Provide summaries for tables. [Priority 3] For example, in HTML, use the "summary" attribute of the TABLE element.
Techniques for checkpoint 5.5
- 5.6 Provide abbreviations for header labels. [Priority 3] For example, in HTML, use the "abbr" attribute on the TH element.
Techniques for checkpoint 5.6
Refer also to checkpoint 10.3.

Guideline 6. Ensure that pages featuring new technologies transform gracefully.

Ensure that pages are accessible even when newer technologies are not supported or are turned off.

Although content developers are encouraged to use new technologies that solve problems raised by existing technologies, they should know how to make their pages still work with older browsers and people who choose to turn off features.

Checkpoints:

- 6.1 Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document. [Priority 1]
When content is organized logically, it will be rendered in a meaningful order when style sheets are turned off or not supported.
Techniques for checkpoint 6.1
- 6.2 Ensure that equivalents for dynamic content are updated when the dynamic content changes. [Priority 1]
Techniques for checkpoint 6.2
- 6.3 Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. [Priority 1]

For example, ensure that links that trigger scripts work when scripts are turned off or not supported (e.g., do not use "javascript:" as the link target). If it is not possible to make the page usable without scripts, provide a text equivalent with the NOSCRIPT element, or use a server-side script instead of a client-side script, or provide an alternative accessible page as per checkpoint 11.4. Refer also to guideline 1.

Techniques for checkpoint 6.3

- 6.4 For scripts and applets, ensure that event handlers are input device-independent. [Priority 2]

Refer to the definition of device independence [p. 26] .

Techniques for checkpoint 6.4

- 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page. [Priority 2]

For example, in HTML, use NOFRAMES at the end of each frameset. For some applications, server-side scripts may be more accessible than client-side scripts.

Techniques for checkpoint 6.5

Refer also to checkpoint 11.4.

Guideline 7. Ensure user control of time-sensitive content changes.

Ensure that moving, blinking, scrolling, or auto-updating objects or pages may be paused or stopped.

Some people with cognitive or visual disabilities are unable to read moving text quickly enough or at all. Movement can also cause such a distraction that the rest of the page becomes unreadable for people with cognitive disabilities. Screen readers [p. 29] are unable to read moving text. People with physical disabilities might not be able to move quickly or accurately enough to interact with moving objects.

Note. All of the following checkpoints involve some content developer responsibility until user agents [p. 30] provide adequate feature control mechanisms.

Checkpoints:

- 7.1 Until user agents [p. 30] allow users to control flickering, avoid causing the screen to flicker. [Priority 1]

Note. People with photosensitive epilepsy can have seizures triggered by flickering or flashing in the 4 to 59 flashes per second (Hertz) range with a peak sensitivity at 20 flashes per second as well as quick changes from dark to light (like strobe lights). Techniques for checkpoint 7.1

- 7.2 Until user agents [p. 30] allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off). [Priority 2]

Techniques for checkpoint 7.2

- 7.3 Until user agents [p. 30] allow users to freeze moving content, avoid movement in pages. [Priority 2]

When a page includes moving content, provide a mechanism within a script or applet to allow users to freeze motion or updates. Using style sheets with scripting to create movement allows users to turn off or override the effect more

easily. Refer also to guideline 8.
Techniques for checkpoint 7.3

7.4 Until user agents [p. 30] provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2]

For example, in HTML, don't cause pages to auto-refresh with "HTTP-EQUIV=refresh" until user agents allow users to turn off the feature.

Techniques for checkpoint 7.4

7.5 Until user agents [p. 30] provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2]

Techniques for checkpoint 7.5

Note. The BLINK and MARQUEE elements are not defined in any W3C HTML specification and should not be used. Refer also to guideline 11.

Guideline 8. Ensure direct accessibility of embedded user interfaces.

Ensure that the user interface follows principles of accessible design: device-independent access to functionality, keyboard operability, self-voicing, etc.

When an embedded object has its "own interface", the interface -- like the interface to the browser itself -- must be accessible. If the interface of the embedded object cannot be made accessible, an alternative accessible solution must be provided.

Note. For information about accessible interfaces, please consult the User Agent Accessibility Guidelines ([WAI-USERAGENT] [p. 33]) and the Authoring Tool Accessibility Guidelines ([WAI-AUTOOL] [p. 33]).

Checkpoint:

8.1 Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important [p. 28] and not presented elsewhere, otherwise Priority 2.]

Refer also to guideline 6.

Techniques for checkpoint 8.1

Guideline 9. Design for device-independence.

Use features that enable activation of page elements via a variety of input devices.

Device-independent [p. 26] access means that the user may interact with the user agent or document with a preferred input (or output) device -- mouse, keyboard, voice, head wand, or other. If, for example, a form control can only be activated with a mouse or other pointing device, someone who is using the page without sight, with voice input, or with a keyboard or who is using some other non-pointing input device will not be able to use the form.

Note. Providing text equivalents for image maps or images used as links makes it possible for users to interact with them without a pointing device. Refer also to guideline 1.

Generally, pages that allow keyboard interaction are also accessible through speech input or a command line interface.

Checkpoints:

- 9.1 Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape. [Priority 1]
Refer also to checkpoint 1.1, checkpoint 1.2, and checkpoint 1.5.
Techniques for checkpoint 9.1
- 9.2 Ensure that any element that has its own interface can be operated in a device-independent manner. [Priority 2]
Refer to the definition of device independence [p. 26] . Refer also to guideline 8.
Techniques for checkpoint 9.2
- 9.3 For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2]
Techniques for checkpoint 9.3
- 9.4 Create a logical tab order through links, form controls, and objects. [Priority 3]
For example, in HTML, specify tab order via the "tabindex" attribute or ensure a logical page design.
Techniques for checkpoint 9.4
- 9.5 Provide keyboard shortcuts to important links (including those in client-side image maps [p. 28]), form controls, and groups of form controls. [Priority 3]
For example, in HTML, specify shortcuts via the "accesskey" attribute.
Techniques for checkpoint 9.5

Guideline 10. Use interim solutions.

Use interim accessibility solutions so that assistive technologies and older browsers will operate correctly.

For example, older browsers do not allow users to navigate to empty edit boxes. Older screen readers read lists of consecutive links as one link. These active elements are therefore difficult or impossible to access. Also, changing the current window or popping up new windows can be very disorienting to users who cannot see that this has happened.

Note. The following checkpoints apply until user agents [p. 30] (including assistive technologies [p. 25]) address these issues. These checkpoints are classified as "interim", meaning that the Web Content Guidelines Working Group considers them to be valid and necessary to Web accessibility *as of the publication of this document*. However, the Working Group does not expect these checkpoints to be necessary in the future, once Web technologies have incorporated anticipated features or capabilities.

Checkpoints:

- 10.1 Until user agents [p. 30] allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user. [Priority 2]
For example, in HTML, avoid using a frame whose target is a new window.
Techniques for checkpoint 10.1
- 10.2 Until user agents [p. 30] support explicit associations between labels and form

controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2]

The label must immediately precede its control on the same line (allowing more than one control/label per line) or be in the line preceding the control (with only one label and one control per line). Refer also to checkpoint 12.4.

Techniques for checkpoint 10.2

10.3 Until user agents [p. 30] (including assistive technologies) render side-by-side text correctly, provide a linear text alternative (on the current page or some other) for *all* tables that lay out text in parallel, word-wrapped columns. [Priority 3]

Note. Please consult the definition of linearized table [p. 28] . This checkpoint benefits people with user agents [p. 30] (such as some screen readers [p. 29]) that are unable to handle blocks of text presented side-by-side; the checkpoint should not discourage content developers from using tables to represent tabular information [p. 29] .

Techniques for checkpoint 10.3

10.4 Until user agents [p. 30] handle empty controls correctly, include default, placeholder characters in edit boxes and text areas. [Priority 3]

For example, in HTML, do this for TEXTAREA and INPUT.

Techniques for checkpoint 10.4

10.5 Until user agents [p. 30] (including assistive technologies) render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links. [Priority 3]

Techniques for checkpoint 10.5

Guideline 11. Use W3C technologies and guidelines.

Use W3C technologies (according to specification) and follow accessibility guidelines. Where it is not possible to use a W3C technology, or doing so results in material that does not transform gracefully, provide an alternative version of the content that is accessible.

The current guidelines recommend W3C technologies (e.g., HTML, CSS, etc.) for several reasons:

- W3C technologies include "built-in" accessibility features.
- W3C specifications undergo early review to ensure that accessibility issues are considered during the design phase.
- W3C specifications are developed in an open, industry consensus process.

Many non-W3C formats (e.g., PDF, Shockwave, etc.) require viewing with either plug-ins or stand-alone applications. Often, these formats cannot be viewed or navigated with standard user agents [p. 30] (including assistive technologies [p. 25]). Avoiding non-W3C and non-standard features (proprietary elements, attributes, properties, and extensions) will tend to make pages more accessible to more people using a wider variety of hardware and software. When inaccessible technologies (proprietary or not) must be used, equivalent accessible pages must be provided.

Even when W3C technologies are used, they must be used in accordance with accessibility guidelines. When using new technologies, ensure that they transform gracefully (Refer also to guideline 6.).

Note. Converting documents (from PDF, PostScript, RTF, etc.) to W3C markup

languages (HTML, XML) does not always create an accessible document. Therefore, validate each page for accessibility and usability after the conversion process (refer to the section on validation [p. 24]). If a page does not readily convert, either revise the page until its original representation converts appropriately or provide an HTML or plain text version.

Checkpoints:

11.1 Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported. [Priority 2]

Refer to the list of references [p. 32] for information about where to find the latest W3C specifications and [WAI-UA-SUPPORT] [p. 33] for information about user agent support for W3C technologies.

Techniques for checkpoint 11.1

11.2 Avoid deprecated features of W3C technologies. [Priority 2] For example, in HTML, don't use the deprecated [p. 26] FONT element; use style sheets instead (e.g., the 'font' property in CSS).

Techniques for checkpoint 11.2

11.3 Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.) [Priority 3]

Note. Use content negotiation where possible.

Techniques for checkpoint 11.3

11.4 If, after best efforts [p. 21] , you cannot create an accessible [p. 25] page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent [p. 27] information (or functionality), and is updated as often as the inaccessible (original) page. [Priority 1]

Techniques for checkpoint 11.4

Note. Content developers should only resort to alternative pages when other solutions fail because alternative pages are generally updated less often than "primary" pages. An out-of-date page may be as frustrating as one that is inaccessible since, in both cases, the information presented on the original page is unavailable. Automatically generating alternative pages may lead to more frequent updates, but content developers must still be careful to ensure that generated pages always make sense, and that users are able to navigate a site by following links on primary pages, alternative pages, or both. Before resorting to an alternative page, reconsider the design of the original page; making it accessible is likely to improve it for all users.

Guideline 12. Provide context and orientation information.

Provide context and orientation information to help users understand complex pages or elements.

Grouping elements and providing contextual information about the relationships between elements can be useful for all users. Complex relationships between parts of a page may be difficult for people with cognitive disabilities and people with visual disabilities to interpret.

Checkpoints:

12.1 Title each frame to facilitate frame identification and navigation. [Priority 1] For example, in HTML use the "title" attribute on FRAME elements.

Techniques for checkpoint 12.1

- 12.2 Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone. [Priority 2]

For example, in HTML, use "longdesc," or a description link. [p. 28]

Techniques for checkpoint 12.2

- 12.3 Divide large blocks of information into more manageable groups where natural and appropriate. [Priority 2]

For example, in HTML, use OPTGROUP to group OPTION elements inside a SELECT; group form controls with FIELDSET and LEGEND; use nested lists where appropriate; use headings to structure documents, etc. Refer also to guideline 3.

Techniques for checkpoint 12.3

- 12.4 Associate labels explicitly with their controls. [Priority 2]

For example, in HTML use LABEL and its "for" attribute.

Techniques for checkpoint 12.4

Guideline 13. Provide clear navigation mechanisms.

Provide clear and consistent navigation mechanisms -- orientation information, navigation bars, a site map, etc. -- to increase the likelihood that a person will find what they are looking for at a site.

Clear and consistent navigation mechanisms [p. 29] are important to people with cognitive disabilities or blindness, and benefit all users.

Checkpoints:

- 13.1 Clearly identify the target of each link. [Priority 2] Link text [p. 29] should be meaningful enough to make sense when read out of context -- either on its own or as part of a sequence of links. Link text should also be terse. For example, in HTML, write "Information about version 4.3" instead of "click here". In addition to clear link text, content developers may further clarify the target of a link with an informative link title (e.g., in HTML, the "title" attribute).

Techniques for checkpoint 13.1

- 13.2 Provide metadata to add semantic information to pages and sites. [Priority 2] For example, use RDF ([RDF] [p. 32]) to indicate the document's author, the type of content, etc. **Note.** Some HTML user agents [p. 30] can build navigation tools from document relations described by the HTML LINK element and "rel" or "rev" attributes (e.g., rel="next", rel="previous", rel="index", etc.). Refer also to checkpoint 13.5.

Techniques for checkpoint 13.2

- 13.3 Provide information about the general layout of a site (e.g., a site map or table of contents). [Priority 2]

In describing site layout, highlight and explain available accessibility features.

Techniques for checkpoint 13.3

- 13.4 Use navigation mechanisms in a consistent manner. [Priority 2]

Techniques for checkpoint 13.4

- 13.5 Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]

Techniques for checkpoint 13.5

- 13.6 Group related links, identify the group (for user agents), and, until user agents [p. 30] do so, provide a way to bypass the group. [Priority 3]

Techniques for checkpoint 13.6

- 13.7 If search functions are provided, enable different types of searches for different skill levels and preferences. [Priority 3]

Techniques for checkpoint 13.7

- 13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc. [Priority 3]

Note. This is commonly referred to as "front-loading" and is especially helpful for people accessing information with serial devices such as speech synthesizers. Techniques for checkpoint 13.8

- 13.9 Provide information about document collections (i.e., documents comprising multiple pages.). [Priority 3]

For example, in HTML specify document collections with the LINK element and the "rel" and "rev" attributes. Another way to create a collection is by building an archive (e.g., with zip, tar and gzip, stuffit, etc.) of the multiple pages. **Note.** The performance improvement gained by offline processing can make browsing much less expensive for people with disabilities who may be browsing slowly.

Techniques for checkpoint 13.9

- 13.10 Provide a means to skip over multi-line ASCII art. [Priority 3]

Refer to checkpoint 1.1 and the example of ascii art in the glossary [p. 25] .

Techniques for checkpoint 13.10

Guideline 14. Ensure that documents are clear and simple.

Ensure that documents are clear and simple so they may be more easily understood.

Consistent page layout, recognizable graphics, and easy to understand language benefit all users. In particular, they help people with cognitive disabilities or who have difficulty reading. (However, ensure that images have text equivalents for people who are blind, have low vision, or for any user who cannot or has chosen not to view graphics. Refer also to guideline 1.)

Using clear and simple language promotes effective communication. Access to written information can be difficult for people who have cognitive or learning disabilities. Using clear and simple language also benefits people whose first language differs from your own, including those people who communicate primarily in sign language.

Checkpoints:

- 14.1 Use the clearest and simplest language appropriate for a site's content. [Priority 1]

Techniques for checkpoint 14.1

- 14.2 Supplement text with graphic or auditory presentations where they will facilitate comprehension of the page. [Priority 3]

Refer also to guideline 1.

Techniques for checkpoint 14.2

- 14.3 Create a style of presentation that is consistent across pages. [Priority 3]
-

Appendix A. -- Validation

Validate accessibility with automatic tools and human review. Automated methods are generally rapid and convenient but cannot identify all accessibility issues. Human review can help ensure clarity of language and ease of navigation.

Begin using validation methods at the earliest stages of development. Accessibility issues identified early are easier to correct and avoid.

Following are some important validation methods, discussed in more detail in the section on validation in the Techniques Document.

1. Use an automated accessibility tool and browser validation tool. Please note that software tools do not address all accessibility issues, such as the meaningfulness of link text, the applicability of a text equivalent [p. 27], etc.
2. Validate syntax (e.g., HTML, XML, etc.).
3. Validate style sheets (e.g., CSS).
4. Use a text-only browser or emulator.
5. Use multiple graphic browsers, with:
 - sounds and graphics loaded,
 - graphics not loaded,
 - sounds not loaded,
 - no mouse,
 - frames, scripts, style sheets, and applets not loaded
6. Use several browsers, old and new.
7. Use a self-voicing browser, a screen reader, magnification software, a small display, etc.
8. Use spell and grammar checkers. A person reading a page with a speech synthesizer may not be able to decipher the synthesizer's best guess for a word with a spelling error. Eliminating grammar problems increases comprehension.
9. Review the document for clarity and simplicity. Readability statistics, such as those generated by some word processors may be useful indicators of clarity and simplicity. Better still, ask an experienced (human) editor to review written content for clarity. Editors can also improve the usability of documents by identifying potentially sensitive cultural issues that might arise due to language or icon usage.
10. Invite people with disabilities to review documents. Expert and novice users with disabilities will provide valuable feedback about accessibility or usability problems and their severity.

Appendix B. -- Glossary

Accessible

Content is accessible when it may be used by someone with a disability.

Applet

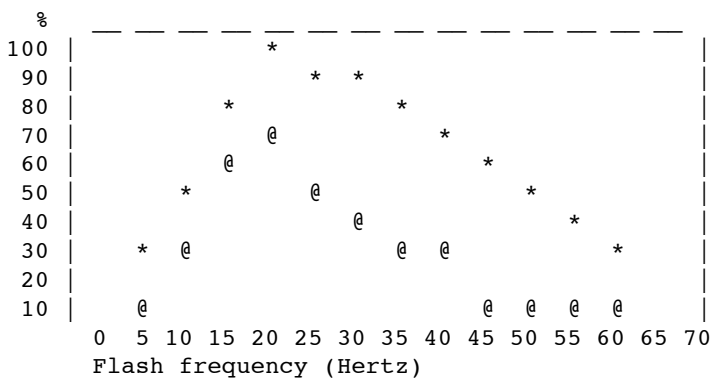
A program inserted into a Web page.

Assistive technology

Software or hardware that has been specifically designed to assist people with disabilities in carrying out daily activities. Assistive technology includes wheelchairs, reading machines, devices for grasping, etc. In the area of Web Accessibility, common software-based assistive technologies include screen readers, screen magnifiers, speech synthesizers, and voice input software that operate in conjunction with graphical desktop browsers (among other user agents [p. 30]). Hardware assistive technologies include alternative keyboards and pointing devices.

ASCII art

ASCII art refers to text characters and symbols that are combined to create an image. For example ";-)" is the smiley emoticon. The following is an ascii figure showing the relationship between flash frequency and photoconvulsive response in patients with eyes open and closed [skip over ascii figure [p. 25] or consult a description of chart]:



Authoring tool

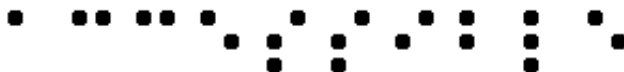
HTML editors, document conversion tools, tools that generate Web content from databases are all authoring tools. Refer to the "Authoring Tool Accessibility Guidelines" ([WAI-AUTOOLS] [p. 33]) for information about developing accessible tools.

Backward compatible

Design that continues to work with earlier versions of a language, program, etc.

Braille

Braille uses six raised dots in different patterns to represent letters and numbers to be read by people who are blind with their fingertips. The word "Accessible" in braille follows:



A **braille display**, commonly referred to as a "dynamic braille display," raises or lowers dot patterns on command from an electronic device, usually a computer. The result is a line of braille that can change from moment to moment. Current dynamic braille displays range in size from one cell (six or eight dots) to an eighty-cell line, most having between twelve and twenty cells per line.

Content developer

Someone who authors Web pages or designs Web sites.

Deprecated

A deprecated element or attribute is one that has been outdated by newer constructs. Deprecated elements may become obsolete in future versions of HTML. The index of HTML elements and attributes in the Techniques Document indicates which elements and attributes are deprecated in HTML 4.0. Authors should avoid using deprecated elements and attributes. User agents should continue to support for reasons of backward compatibility.

Device independent

Users must be able to interact with a user agent (and the document it renders) using the supported input and output devices of their choice and according to their needs. Input devices may include pointing devices, keyboards, braille devices, head wands, microphones, and others. Output devices may include monitors, speech synthesizers, and braille devices. Please note that "device-independent support" does not mean that user agents must support every input or output device. User agents should offer redundant input and output mechanisms for those devices that are supported. For example, if a user agent supports keyboard and mouse input, users should be able to interact with all features using either the keyboard or the mouse.

Document Content, Structure, and Presentation

The content of a document refers to what it says to the user through natural language, images, sounds, movies, animations, etc. The structure of a document is how it is organized logically (e.g., by chapter, with an introduction and table of contents, etc.). An element [p. 27] (e.g., P, STRONG, BLOCKQUOTE in HTML) that specifies document structure is called a structural element. The presentation of a document is how the document is rendered (e.g., as print, as a two-dimensional graphical presentation, as an text-only presentation, as synthesized speech, as braille, etc.) An element [p. 27] that specifies document presentation (e.g., B, FONT, CENTER) is called a presentation element. Consider a document header, for example. The content of the header is what the header says (e.g., "Sailboats"). In HTML, the header is a structural element marked up with, for example, an H2 element. Finally, the presentation of the header might be a bold block text in the margin, a centered line of text, a title spoken with a certain voice style (like an aural font), etc.

Dynamic HTML (DHTML)

DHTML is the marketing term applied to a mixture of standards including HTML, style sheets [p. 29] , the Document Object Model [DOM1] [p. 32] and scripting. However, there is no W3C specification that formally defines DHTML. Most guidelines may be applicable to applications using DHTML, however the following guidelines focus on issues related to scripting and style sheets: guideline 1, guideline 3, guideline 6, guideline 7, and guideline 9.

Element

This document uses the term "element" both in the strict SGML sense (an element is a syntactic construct) and more generally to mean a type of content (such as video or sound) or a logical construct (such as a header or list). The second sense emphasizes that a guideline inspired by HTML could easily apply to another markup language. Note that some (SGML) elements have content that is rendered (e.g., the P, LI, or TABLE elements in HTML), some are replaced by external content (e.g., IMG), and some affect processing (e.g., STYLE and SCRIPT cause

information to be processed by a style sheet or script engine). An element that causes text characters to be part of the document is called a text element.

Equivalent

Content is "equivalent" to other content when both fulfill essentially the same function or purpose upon presentation to the user. In the context of this document, the equivalent must fulfill essentially the same function for the person with a disability (at least insofar as is feasible, given the nature of the disability and the state of technology), as the primary content does for the person without any disability. For example, the text "The Full Moon" might convey the same information as an image of a full moon when presented to users. Note that equivalent information focuses on **fulfilling the same function**. If the image is part of a link and understanding the image is crucial to guessing the link target, an equivalent must also give users an idea of the link target. Providing equivalent information for inaccessible content is one of the primary ways authors can make their documents accessible to people with disabilities.

As part of fulfilling the same function of content an equivalent may involve a description of that content (i.e., what the content looks like or sounds like). For example, in order for users to understand the information conveyed by a complex chart, authors should describe the visual information in the chart. Since text content can be presented to the user as synthesized speech, braille, and visually-displayed text, these guidelines require **text equivalents** for graphic and audio information. Text equivalents must be written so that they convey all essential content. **Non-text equivalents** (e.g., an auditory description of a visual presentation, a video of a person telling a story using sign language as an equivalent for a written story, etc.) also improve accessibility for people who cannot access visual information or written text, including many individuals with blindness, cognitive disabilities, learning disabilities, and deafness. Equivalent information may be provided in a number of ways, including through attributes (e.g., a text value for the "alt" attribute in HTML and SMIL), as part of element content (e.g., the OBJECT in HTML), as part of the document's prose, or via a linked document (e.g., designated by the "longdesc" attribute in HTML or a *description link*). Depending on the complexity of the equivalent, it may be necessary to combine techniques (e.g., use "alt" for an abbreviated equivalent, useful to familiar readers, in addition to "longdesc" for a link to more complete information, useful to first-time readers). The details of how and when to provide equivalent information are part of the Techniques Document ([TECHNIQUES] [p. 33]).

A **text transcript** is a text equivalent of audio information that includes spoken words and non-spoken sounds such as sound effects. A **caption** is a text transcript for the audio track of a video presentation that is synchronized with the video and audio tracks. Captions are generally rendered visually by being superimposed over the video, which benefits people who are deaf and hard-of-hearing, and anyone who cannot hear the audio (e.g., when in a crowded room). A **collated text transcript** combines (collates) captions with text descriptions of video information (descriptions of the actions, body language, graphics, and scene changes of the video track). These text equivalents make presentations accessible to people who are deaf-blind and to people who cannot play movies, animations, etc. It also makes the information available to search engines. One example of a non-text equivalent is an **auditory description** of the key visual elements of a presentation. The description is either a prerecorded human voice or a synthesized voice (recorded or generated on the fly). The auditory description is synchronized with the

audio track of the presentation, usually during natural pauses in the audio track. Auditory descriptions include information about actions, body language, graphics, and scene changes.

Image

A graphical presentation.

Image map

An image that has been divided into regions with associated actions. Clicking on an active region causes an action to occur. When a user clicks on an active region of a client-side image map, the user agent calculates in which region the click occurred and follows the link associated with that region. Clicking on an active region of a server-side image map causes the coordinates of the click to be sent to a server, which then performs some action. Content developers can make client-side image maps accessible by providing device-independent access to the same links associated with the image map's regions. Client-side image maps allow the user agent to provide immediate feedback as to whether or not the user's pointer is over an active region.

Important

Information in a document is important if understanding that information is crucial to understanding the document.

Linearized table

A table rendering process where the contents of the cells become a series of paragraphs (e.g., down the page) one after another. The paragraphs will occur in the same order as the cells are defined in the document source. Cells should make sense when read in order and should include structural elements [p. 26] (that create paragraphs, headers, lists, etc.) so the page makes sense after linearization.

Link text

The rendered text content of a link.

Natural Language

Spoken, written, or signed human languages such as French, Japanese, American Sign Language, and braille. The natural language of content may be indicated with the "lang" attribute in HTML ([HTML40] [p. 32], section 8.1) and the "xml:lang" attribute in XML ([XML] [p. 33], section 2.12).

Navigation Mechanism

A navigation mechanism is any means by which a user can navigate a page or site. Some typical mechanisms include:

navigation bars

A navigation bar is a collection of links to the most important parts of a document or site.

site maps

A site map provides a global view of the organization of a page or site.

tables of contents

A table of contents generally lists (and links to) the most important sections of a document.

Personal Digital Assistant (PDA)

A PDA is a small, portable computing device. Most PDAs are used to track personal data such as calendars, contacts, and electronic mail. A PDA is generally a handheld device with a small screen that allows input from various sources.

Screen magnifier

A software program that magnifies a portion of the screen, so that it can be more easily viewed. Screen magnifiers are used primarily by individuals with low vision.

Screen reader

A software program that reads the contents of the screen aloud to a user. Screen readers are used primarily by individuals who are blind. Screen readers can usually only read text that is printed, not painted, to the screen.

Style sheets

A style sheet is a set of statements that specify presentation of a document. Style sheets may have three different origins: they may be written by content providers, created by users, or built into user agents. In CSS ([CSS2] [p. 32]), the interaction of content provider, user, and user agent style sheets is called the *cascade*.

Presentation markup

Is markup that achieves a stylistic (rather than structuring) effect such as the B or I elements in HTML. Note that the STRONG and EM elements are not considered presentation markup since they convey information that is independent of a particular font style.

Tabular information

When tables are used to represent logical relationships among data -- text, numbers, images, etc., that information is called "tabular information" and the tables are called "data tables". The relationships expressed by a table may be rendered visually (usually on a two-dimensional grid), aurally (often preceding cells with header information), or in other formats.

Until user agents...

In most of the checkpoints, content developers are asked to ensure the accessibility of their pages and sites. However, there are accessibility needs that would be more appropriately met by user agents [p. 30] (including assistive technologies [p. 25]). As of the publication of this document, not all user agents or assistive technologies provide the accessibility control users require (e.g., some user agents may not allow users to turn off blinking content, or some screen readers may not handle tables well). Checkpoints that contain the phrase "until user agents ..." require content developers to provide additional support for accessibility until most user agents readily available to their audience include the necessary accessibility features. **Note.** The W3C WAI Web site (refer to [WAI-UA-SUPPORT] [p. 33]) provides information about user agent support for accessibility features. Content developers are encouraged to consult this page regularly for updated information.

User agent

Software to access Web content, including desktop graphical browsers, text browsers, voice browsers, mobile phones, multimedia players, plug-ins, and some software assistive technologies used in conjunction with browsers such as screen readers, screen magnifiers, and voice recognition software.

Acknowledgments

Web Content Guidelines Working Group Co-Chairs:

Chuck Letourneau, Starling Access Services

Gregg Vanderheiden, Trace Research and Development

W3C Team contacts:

Judy Brewer and Daniel Dardailler

We wish to thank the following people who have contributed their time and valuable comments to shaping these guidelines:

Harvey Bingham, Kevin Carey, Chetz Colwell, Neal Ewers, Geoff Freed, Al Gilman, Larry Goldberg, Jon Gunderson, Eric Hansen, Phill Jenkins, Leonard Kasday, George Kerscher, Marja-Riitta Koivunen, Josh Krieger, Scott Luebking, William Loughborough, Murray Maloney, Charles McCathieNevile, MegaZone (Livingston Enterprises), Masafumi Nakane, Mark Novak, Charles Oppermann, Mike Paciello, David Pawson, Michael Pieper, Greg Rosmaita, Liam Quinn, Dave Raggett, T.V. Raman, Robert Savellis, Jutta Treviranus, Steve Tyler, Jaap van Lelieveld, and Jason White

The original draft of this document is based on "The Unified Web Site Accessibility Guidelines" ([UWSAG] [p. 33]) compiled by the Trace R & D Center at the University of Wisconsin. That document includes a list of additional contributors.

References

For the latest version of any W3C specification please consult the list of W3C Technical Reports.

[CSS1]

"CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. The CSS1 Recommendation is: <http://www.w3.org/TR/1999/REC-CSS1-19990111>. The latest version of CSS1 is available at: <http://www.w3.org/TR/REC-CSS1>.

[CSS2]

"CSS, level 2 Recommendation", B. Bos, H. Wium Lie, C. Lilley, and I. Jacobs, eds., 12 May 1998. The CSS2 Recommendation is: <http://www.w3.org/TR/1998/REC-CSS2-19980512>. The latest version of CSS2 is available at: <http://www.w3.org/TR/REC-CSS2>.

[DOM1]

"Document Object Model (DOM) Level 1 Specification", V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood, eds., 1 October 1998. The DOM Level 1 Recommendation is: <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001>. The latest version of DOM Level 1 is available at: <http://www.w3.org/TR/REC-DOM-Level-1>

[HTML40]

"HTML 4.0 Recommendation", D. Raggett, A. Le Hors, and I. Jacobs, eds., 17 December 1997, revised 24 April 1998. The HTML 4.0 Recommendation is: <http://www.w3.org/TR/1998/REC-html40-19980424>. The latest version of HTML 4.0 is available at: <http://www.w3.org/TR/REC-html40>.

[HTML32]

"HTML 3.2 Recommendation", D. Raggett, ed., 14 January 1997. The latest version of HTML 3.2 is available at: <http://www.w3.org/TR/REC-html32>.

[MATHML]

"Mathematical Markup Language", P. Ion and R. Miner, eds., 7 April 1998. The MathML 1.0 Recommendation is: <http://www.w3.org/TR/1998/REC-MathML-19980407>. The latest version of MathML 1.0 is available at: <http://www.w3.org/TRREC-MathML>.

[PNG]

"PNG (Portable Network Graphics) Specification", T. Boutell, ed., T. Lane, contributing

ed., 1 October 1996. The latest version of PNG 1.0 is: <http://www.w3.org/TR/REC-png>.

[RDF]

"Resource Description Framework (RDF) Model and Syntax Specification", O. Lassila, R. Swick, eds., 22 February 1999. The RDF Recommendation is: <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>. The latest version of RDF 1.0 is available at:

<http://www.w3.org/TR/REC-rdf-syntax>

[RFC2068]

"HTTP Version 1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, and T. Berners-Lee, January 1997.

[SMIL]

"Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", P. Hoschka, ed., 15 June 1998. The SMIL 1.0 Recommendation is: <http://www.w3.org/TR/1998/REC-smil-19980615> The latest version of SMIL 1.0 is available at: <http://www.w3.org/TR/REC-smil>

[TECHNIQUES]

"Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, I. Jacobs, eds. This document explains how to implement the checkpoints defined in "Web Content Accessibility Guidelines 1.0". The latest draft of the techniques is available at: <http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/>

[WAI-AUTOOLS]

"Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, eds. The latest Working Draft of these guidelines for designing accessible authoring tools is available at: <http://www.w3.org/TR/WAI-AUTOOLS/>

[WAI-UA-SUPPORT]

This page documents known support by user agents (including assistive technologies) of some accessibility features listed in this document. The page is available at: <http://www.w3.org/WAI/Resources/WAI-UA-Support>

[WAI-USERAGENT]

"User Agent Accessibility Guidelines", J. Gunderson and I. Jacobs, eds. The latest Working Draft of these guidelines for designing accessible user agents is available at: <http://www.w3.org/TR/WAI-USERAGENT/>

[WCAG-ICONS]

Information about conformance icons for this document and how to use them is available at <http://www.w3.org/WAI/WCAG1-Conformance.html>

[UWSAG]

"The Unified Web Site Accessibility Guidelines", G. Vanderheiden, W. Chisholm, eds. The Unified Web Site Guidelines were compiled by the Trace R & D Center at the University of Wisconsin under funding from the National Institute on Disability and Rehabilitation Research (NIDRR), U.S. Dept. of Education. This document is available at: http://www.tracecenter.org/docs/html_guidelines/version8.htm

[XML]

"Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds., 10 February 1998. The XML 1.0 Recommendation is: <http://www.w3.org/TR/1998/REC-xml-19980210>. The latest version of XML 1.0 is available at: <http://www.w3.org/TR/REC-xml>

APPENDIX II



Web Content Accessibility Guidelines (WCAG) 2.0

W3C Recommendation 11 December 2008

This version:

<http://www.w3.org/TR/2008/REC-WCAG20-20081211/>

Latest version:

<http://www.w3.org/TR/WCAG20/>

Previous version:

<http://www.w3.org/TR/2008/PR-WCAG20-20081103/>

Editors:

Ben Caldwell, Trace R&D Center, University of Wisconsin-Madison Michael Cooper, W3C Loretta Guarino Reid, Google, Inc. Gregg Vanderheiden, Trace R&D Center, University of Wisconsin-Madison

Previous Editors:

Wendy Chisholm (until July 2006 while at W3C) John Slatin (until June 2006 while at Accessibility Institute, University of Texas at Austin) Jason White (until June 2005 while at University of Melbourne)

Please refer to the [errata](#) for this document, which may include normative corrections.

See also [translations](#).

This document is also available in non-normative formats, available from [Alternate Versions of Web Content Accessibility Guidelines 2.0](#).

Copyright © 2008 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

Web Content Accessibility Guidelines (WCAG) 2.0 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these. Following these guidelines will also often make your Web content more usable to users in general.

WCAG 2.0 success criteria are written as testable statements that are not technology-specific. Guidance about satisfying the success criteria in specific technologies, as well as general information about interpreting the success criteria, is provided in separate

documents. See [Web Content Accessibility Guidelines \(WCAG\) Overview](#) for an introduction and links to WCAG technical and educational material.

WCAG 2.0 succeeds [Web Content Accessibility Guidelines 1.0 \[WCAG10\]](#), which was published as a W3C Recommendation May 1999. Although it is possible to conform either to WCAG 1.0 or to WCAG 2.0 (or both), the W3C recommends that new and updated content use WCAG 2.0. The W3C also recommends that Web accessibility policies reference WCAG 2.0.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the [W3C technical reports index](#) at <http://www.w3.org/TR/>.

This is the Web Content Accessibility Guidelines (WCAG) 2.0 [W3C Recommendation](#) from the [Web Content Accessibility Guidelines Working Group](#).

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

WCAG 2.0 is supported by the associated non-normative documents, [Understanding WCAG 2.0](#) and [Techniques for WCAG 2.0](#). Although those documents do not have the formal status that WCAG 2.0 itself has, they provide information important to understanding and implementing WCAG.

The Working Group requests that any comments be made using the provided [online comment form](#). If this is not possible, comments can also be sent to public-comments-wcag20@w3.org. The [archives for the public comments list](#) are publicly available. Comments received on the WCAG 2.0 Recommendation cannot result in changes to this version of the guidelines, but may be addressed in errata or future versions of WCAG. The Working Group does not plan to make formal responses to comments. Archives of the [WCAG WG mailing list discussions](#) are publicly available, and future work undertaken by the Working Group may address comments received on this document.

This document has been produced as part of the W3C [Web Accessibility Initiative \(WAI\)](#). The goals of the WCAG Working Group are discussed in the [WCAG Working Group charter](#). The WCAG Working Group is part of the [WAI Technical Activity](#).

This document was produced by a group operating under the [5 February 2004 W3C Patent Policy](#). W3C maintains a [public list of any patent disclosures](#) made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains [Essential Claim\(s\)](#) must disclose the information in accordance with [section 6 of the W3C Patent Policy](#).

Table of Contents

[Introduction](#)

- [WCAG 2.0 Layers of Guidance](#)
- [WCAG 2.0 Supporting Documents](#)
- [Important Terms in WCAG 2.0](#)

[WCAG 2.0 Guidelines](#)

[1 Perceivable](#)

- [1.1 Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.](#)
- [1.2 Provide alternatives for time-based media.](#)
- [1.3 Create content that can be presented in different ways \(for examplesimpler layout\) without losing information or structure.](#)
- [1.4 Make it easier for users to see and hear content including separating foreground from background.](#)

[2 Operable](#)

- [2.1 Make all functionality available from a keyboard.](#)
- [2.2 Provide users enough time to read and use content.](#)
- [2.3 Do not design content in a way that is known to cause seizures.](#)
- [2.4 Provide ways to help users navigate, find content, and determine where they are.](#)

[3 Understandable](#)

- [3.1 Make text content readable and understandable.](#)
- [3.2 Make Web pages appear and operate in predictable ways.](#)
- [3.3 Help users avoid and correct mistakes.](#)

[4 Robust](#)

- [4.1 Maximize compatibility with current and future user agents, including assistive technologies.](#)

[Conformance](#)

[Conformance Requirements](#)

[Conformance Claims \(Optional\)](#)

[Statement of Partial Conformance - Third Party Content](#)

[Statement of Partial Conformance - Language](#)

Appendix A:

[Glossary \(Normative\)](#)

Appendix B: [Acknowledgments](#)

Appendix C: [References](#)

Introduction

This section is informative.

Web Content Accessibility Guidelines (WCAG) 2.0 defines how to make Web content more accessible to people with disabilities. Accessibility involves a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities. Although these guidelines cover a wide range of issues,

they are not able to address the needs of people with all types, degrees, and combinations of disability. These guidelines also make Web content more usable by older individuals with changing abilities due to aging and often improve usability for users in general.

WCAG 2.0 is developed through the [W3C process](#) in cooperation with individuals and organizations around the world, with a goal of providing a shared standard for Web content accessibility that meets the needs of individuals, organizations, and governments internationally. WCAG 2.0 builds on WCAG 1.0 [[WCAG10](#)] and is designed to apply broadly to different Web technologies now and in the future, and to be testable with a combination of automated testing and human evaluation. For an introduction to WCAG, see the [Web Content Accessibility Guidelines \(WCAG\) Overview](#).

Web accessibility depends not only on accessible content but also on accessible Web browsers and other user agents. Authoring tools also have an important role in Web accessibility. For an overview of how these components of Web development and interaction work together, see:

- [Essential Components of Web Accessibility](#)
- [User Agent Accessibility Guidelines \(UAAG\) Overview](#)
- [Authoring Tool Accessibility Guidelines \(ATAG\) Overview](#)

WCAG 2.0 Layers of Guidance

The individuals and organizations that use WCAG vary widely and include Web designers and developers, policy makers, purchasing agents, teachers, and students. In order to meet the varying needs of this audience, several layers of guidance are provided including overall *principles*, general *guidelines*, testable *success criteria* and a rich collection of *sufficient techniques*, *advisory techniques*, and *documented common failures* with examples, resource links and code.

Principles - At the top are four principles that provide the foundation for Web accessibility: *perceivable*, *operable*, *understandable*, and *robust*. See also [Understanding the Four Principles of Accessibility](#).

Guidelines - Under the principles are guidelines. The 12 guidelines provide the basic goals that authors should work toward in order to make content more accessible to users with different disabilities. The guidelines are not testable, but provide the framework and overall objectives to help authors understand the success criteria and better implement the techniques.

Success Criteria - For each guideline, testable success criteria are provided to allow WCAG 2.0 to be used where requirements and conformance testing are necessary such as in design specification, purchasing, regulation, and contractual agreements. In order to meet the needs of different groups and different situations, three levels of conformance are defined: A (lowest), AA, and AAA (highest). Additional information on WCAG levels can be found in [Understanding Levels of Conformance](#).

Sufficient and Advisory Techniques - For each of the *guidelines* and *success criteria* in the WCAG 2.0 document itself, the working group has also documented a wide variety of *techniques*. The techniques are informative and fall into two categories: those that are *sufficient* for meeting the success criteria and those that are *advisory*. The advisory techniques go beyond what is required by the individual success criteria and

allow authors to better address the guidelines. Some advisory techniques address accessibility barriers that are not covered by the testable success criteria. Where common failures are known, these are also documented. See also [Sufficient and Advisory Techniques in Understanding WCAG 2.0](#).

All of these layers of guidance (principles, guidelines, success criteria, and sufficient and advisory techniques) work together to provide guidance on how to make content more accessible. Authors are encouraged to view and apply all layers that they are able to, including the advisory techniques, in order to best address the needs of the widest possible range of users.

Note that even content that conforms at the highest level (AAA) will not be accessible to individuals with all types, degrees, or combinations of disability, particularly in the cognitive language and learning areas. Authors are encouraged to consider the full range of techniques, including the advisory techniques, as well as to seek relevant advice about current best practice to ensure that Web content is accessible, as far as possible, to this community. [Metadata](#) may assist users in finding content most suitable for their needs.

WCAG 2.0 Supporting Documents

The WCAG 2.0 document is designed to meet the needs of those who need a stable, referenceable technical standard. Other documents, called supporting documents, are based on the WCAG 2.0 document and address other important purposes, including the ability to be updated to describe how WCAG would be applied with new technologies.

Supporting documents include:

1. [How to Meet WCAG 2.0](#) - A customizable quick reference to WCAG 2.0 that includes all of the guidelines, success criteria, and techniques for authors to use as they are developing and evaluating Web content.
2. [Understanding WCAG 2.0](#) - A guide to understanding and implementing WCAG 2.0. There is a short "Understanding" document for each guideline and success criterion in WCAG 2.0 as well as key topics.
3. [Techniques for WCAG 2.0](#) - A collection of techniques and common failures, each in a separate document that includes a description, examples, code and tests.
4. [The WCAG 2.0 Documents](#) - A diagram and description of how the technical documents are related and linked.

See [Web Content Accessibility Guidelines \(WCAG\) Overview](#) for a description of the WCAG 2.0 supporting material, including education resources related to WCAG 2.0. Additional resources covering topics such as the business case for Web accessibility, planning implementation to improve the accessibility of Web sites, and accessibility policies are listed in [WAI Resources](#).

Important Terms in WCAG 2.0

WCAG 2.0 includes three important terms that are different from WCAG 1.0. Each of these is introduced briefly below and defined more fully in the glossary.

Web Page

It is important to note that, in this standard, the term "Web page" includes much more than static HTML pages. It also includes the increasingly dynamic Web pages that are emerging on the Web, including "pages" that can present entire virtual interactive communities. For example, the term "Web page" includes an immersive, interactive movie-like experience found at a single URI. For more information, see [Understanding "Web Page"](#).

Programmatically Determined

Several success criteria require that content (or certain aspects of content) can be "programmatically determined." This means that the content is delivered in such a way that user agents, including assistive technologies, can extract and present this information to users in different modalities. For more information, see [Understanding Programmatically Determined](#).

Accessibility Supported

Using a technology in a way that is accessibility supported means that it works with assistive technologies (AT) and the accessibility features of operating systems, browsers, and other user agents. Technology features can only be relied upon to conform to WCAG 2.0 success criteria if they are used in a way that is "accessibility supported". Technology features can be used in ways that are not accessibility supported (do not work with assistive technologies, etc.) as long as they are not relied upon to conform to any success criterion (i.e., the same information or functionality is also available another way that is supported).

The definition of "accessibility supported" is provided in the [Appendix A: Glossary](#) section of these guidelines. For more information, see [Understanding Accessibility Support](#).

WCAG 2.0 Guidelines

This section is normative.

Principle 1: Perceivable - Information and user interface components must be presentable to users in ways they can perceive.

Guideline 1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language. [Understanding Guideline 1.1](#) ⁶⁸

1.1.1 Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A)

How to Meet 1.1.1 Understanding 1.1.1
--

⁶⁹

⁶⁸ Any Guideline provides a link to help understanding the Guideline; but since this appendix does not include active links, we are not including this link in the following Guidelines.

⁶⁹ Any Success Criteria provide a box with two links to help on how meeting and understanding the Success Criteria; since this appendix does not include active links, we do not include this box in the following Success Criteria.

-
- **Controls, Input:** If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to [Guideline 4.1](#) for additional requirements for controls and content that accepts user input.)
 - **Time-Based Media:** If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to [Guideline 1.2](#) for additional requirements for media.)
 - **Test:** If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content.
 - **Sensory:** If non-text content is primarily intended to create a specific sensory experience, then text alternatives at least provide descriptive identification of the non-text content.
 - **CAPTCHA:** If the purpose of non-text content is to confirm that content is being accessed by a person rather than a computer, then text alternatives that identify and describe the purpose of the non-text content are provided, and alternative forms of CAPTCHA using output modes for different types of sensory perception are provided to accommodate different disabilities.
 - **Decoration, Formatting, Invisible:** If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.

Guideline 1.2 Time-based Media: Provide alternatives for time-based media.

1.2.1 Audio-only and Video-only (Prerecorded): For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such: (Level A)

- **Prerecorded Audio-only:** An alternative for time-based media is provided that presents equivalent information for prerecorded audio-only content.
- **Prerecorded Video-only:** Either an alternative for time-based media or an audio track is provided that presents equivalent information for prerecorded video-only content.

1.2.2 Captions (Prerecorded): Captions are provided for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A)

1.2.3 Audio Description or Media Alternative (Prerecorded): An alternative for time-based media or audio description of the prerecorded video content is provided for synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A)

1.2.4 Captions (Live): Captions are provided for all live audio content in synchronized media. (Level AA)

1.2.5 Audio Description (Prerecorded): Audio description is provided for all prerecorded video content in synchronized media. (Level AA)

1.2.6 Sign Language (Prerecorded): Sign language interpretation is provided for all prerecorded audio content in synchronized media. (Level AAA)

1.2.7 Extended Audio Description (Prerecorded): Where pauses in foreground audio are insufficient to allow audio descriptions to convey the sense of the video, extended audio description is provided for all prerecorded video content in synchronized media. (Level AAA)

1.2.8 Media Alternative (Prerecorded): An alternative for time-based media is provided for all prerecorded synchronized media and for all prerecorded video-only media. (Level AAA)

1.2.9 Audio-only (Live): An alternative for time-based media that presents equivalent information for live audio-only content is provided. (Level AAA)

Guideline 1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)

1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)

1.3.3 Sensory Characteristics: Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, size, visual location, orientation, or sound. (Level A)

Note: For requirements related to color, refer to [Guideline 1.4](#).

Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)

Note: This success criterion addresses color perception specifically. Other forms of perception are covered in [Guideline 1.3](#) including programmatic access to color and other visual presentation coding.

1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A)

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether or not it is used to meet other success criteria) must meet this success criterion. See [Conformance Requirement 5: Non- Interference](#).

1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA)

- **Large Text:** Large-scale text and images of large-scale text have a contrast ratio of at least 3:1; **Incidental:** Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
- **Logotypes:** Text that is part of a logo or brand name has no minimum contrast requirement.

1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA)

1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA)

- **Customizable:** The image of text can be visually customized to the user's requirements;
- **Essential:** A particular presentation of text is essential to the information being conveyed.

Note: Logotypes (text that is part of a logo or brand name)

1.4.6 Contrast (Enhanced): The visual presentation of text and images of text has a contrast ratio of at least 7:1, except for the following: (Level AAA)

- **Large Text:** Large-scale text and images of large-scale text have a contrast ratio of at least 4.5:1; **Incidental:** Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
- **Logotypes:** Text that is part of a logo or brand name has no minimum contrast requirement.

1.4.7 Low or No Background Audio: For prerecorded audio-only content that (1) contains primarily speech in the foreground, (2) is not an audio CAPTCHA or audio logo, and (3) is not vocalization intended to be primarily musical expression such as singing or rapping, at least one of the following is true: (Level AAA)

- **No Background:** The audio does not contain background sounds.
- **Turn Off:** The background sounds can be turned off.
- **20 dB:** The background sounds are at least 20 decibels lower than the foreground speech content, with the exception of occasional sounds that last for only one or two seconds.

Note: Per the definition of "decibel," background sound that meets this requirement will be approximately four times quieter than the foreground speech content.

1.4.8 Visual Presentation: For the visual presentation of blocks of text, a mechanism is available to achieve the following: (Level AAA)

1. Foreground and background colors can be selected by the user.
2. Width is no more than 80 characters or glyphs (40 if CJK).
3. Text is not justified (aligned to both the left and the right margins).
4. Line spacing (leading) is at least space-and-a-half within paragraphs, and paragraph spacing is at least 1.5 times larger than the line spacing.
5. Text can be resized without assistive technology up to 200 percent in a way that does not require the user to scroll horizontally to read a line of text on a full-screen window.

1.4.9 Images of Text (No Exception): Images of text are only used for pure decoration or where a particular presentation of text is essential to the information being conveyed. (Level AAA)

Note: Logotypes (text that is part of a logo or brand name) are considered essential.

Principle 2: Operable - User interface components and navigation must be operable.

Guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard

2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)

Note 1: This exception relates to the underlying function, not the input technique. For example, if using handwriting to enter text, the input technique (handwriting) requires path-dependent input but the underlying function (text input) does not.

Note 2: This does not forbid and should not discourage providing mouse input or other input methods in addition to keyboard operation.

2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See [Conformance Requirement 5: Non- Interference](#).

2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)

Guideline 2.2 Enough Time: Provide users enough time to read and use content.

2.2.1 Timing Adjustable: For each time limit that is set by the content, at least one of the following is true: (Level A)

- **Turn off:** The user is allowed to turn off the time limit before encountering it; or
- **Adjust:** The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or
- **Extend:** The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or
- **Real-time Exception:** The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or
- **Essential Exception:** The time limit is essential and extending it would invalidate the activity; or
- **20 Hour Exception:** The time limit is longer than 20 hours.

Note: This success criterion helps ensure that users can complete tasks without unexpected changes in content or context that are a result of a time limit. This success criterion should be considered in conjunction with [Success Criterion 3.2.1](#), which puts limits on changes of content or context as a result of user action.

2.2.2 Pause, Stop, Hide: For moving, blinking, scrolling, or auto-updating information, all of the following are true: (Level A)

- **Moving, blinking, scrolling:** For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is essential; and

-
- **Auto-updating:** For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

Note 1: For requirements related to flickering or flashing content, refer to [Guideline 2.3](#).

Note 2: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See [Conformance Requirement 5: Non- Interference](#).

Note 3: Content that is updated periodically by software or that is streamed to the user agent is not required to preserve or present information that is generated or received between the initiation of the pause and resuming presentation, as this may not be technically possible, and in many situations could be misleading to do so.

Note 4: An animation that occurs as part of a preload phase or similar situation can be considered essential if interaction cannot occur during that phase for all users and if not indicating progress could confuse users or cause them to think that content was frozen or broken.

2.2.3 No Timing: Timing is not an essential part of the event or activity presented by the content, except for non- interactive synchronized media and real-time events. (Level AAA)

2.2.4 Interruptions: Interruptions can be postponed or suppressed by the user, except interruptions involving an emergency. (Level AAA)

2.2.5 Re-authenticating: When an authenticated session expires, the user can continue the activity without loss of data after re-authenticating. (Level AAA)

[Guideline 2.3](#) Seizures: Do not design content in a way that is known to cause seizures.

2.3.1 Three Flashes or Below Threshold: Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds. (Level A)

Note: Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion. See [Conformance Requirement 5: Non- Interference](#)

2.3.2 Three Flashes: Web pages do not contain anything that flashes more than three times in any one second period. (Level AAA)

[Guideline 2.4](#) Navigable: Provide ways to help users navigate, find content, and determine where they are.

2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A)

2.4.2 Page Titled: Web pages have titles that describe topic or purpose. (Level A)

2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)

2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A)

2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA)

2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)

2.4.7 Focus Visible: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA)

2.4.8 Location: Information about the user's location within a set of Web pages is available. (Level AAA)

2.4.9 Link Purpose (Link Only): A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general. (Level AAA)

2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA)

Note 1: "Heading" is used in its general sense and includes titles and other ways to add a heading to different types of content.

Note 2: This success criterion covers sections within writing, not user interface components. User Interface components are covered under [Success Criterion 4.1.2](#).

Principle 3: Understandable - Information and the operation of user interface must be understandable.

Guideline 3.1 Readable: Make text content readable and understandable.

3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A)

3.1.2 Language of Parts: The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text. (Level AA)

3.1.3 Unusual Words: A mechanism is available for identifying specific definitions of words or phrases used in an unusual or restricted way, including idioms and jargon. (Level AAA)

3.1.4 Abbreviations: A mechanism for identifying the expanded form or meaning of abbreviations is available. (Level AAA)

3.1.5 Reading Level: When text requires reading ability more advanced than the lower secondary education level after removal of proper names and titles, supplemental content, or a version that does not require reading ability more advanced than the lower secondary education level, is available. (Level AAA)

3.1.6 Pronunciation: A mechanism is available for identifying specific pronunciation of words where meaning of the words, in context, is ambiguous without knowing the

pronunciation. (Level AAA)

Guideline 3.2 Predictable: Make Web pages appear and operate in predictable ways.

3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A)

3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)

3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)

3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)

3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA) component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)

3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)

3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)

3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA)

Guideline 3.3 Input Assistance: Help users avoid and correct mistakes.

3.3.1 Error Identification: If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text. (Level A)

3.3.2 Labels or Instructions: Labels or instructions are provided when content requires user input. (Level A)

3.3.3 Error Suggestion: If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content. (Level AA)

3.3.4 Error Prevention (Legal, Financial, Data): For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true: (Level AA)

1. **Reversible**: Submissions are reversible.
2. **Checked**: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.
3. **Confirmed**: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

3.3.5 Help: Context-sensitive help is available. (Level AAA)

3.3.6 Error Prevention (All): For Web pages that require the user to submit information, at least one of the following is true: (Level AAA)

-
1. **Reversible:** Submissions are reversible.
 2. **Checked:** Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.
 3. **Confirmed:** A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

Principle 4: Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

Guideline 4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies

4.1.1 Parsing: In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A)

Note: Start and end tags that are missing a critical character in their formation, such as a closing angle bracket or a mismatched attribute value quotation mark are not complete.

4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)

Note: This success criterion is primarily for Web authors who develop or script their own user interface components. For example, standard HTML controls already meet this success criterion when used according to specification.

Conformance

This section is normative.

This section lists requirements for conformance to WCAG 2.0. It also gives information about how to make conformance claims, which are optional. Finally, it describes what it means to be accessibility supported, since only accessibility-supported ways of using technologies can be relied upon for conformance. [Understanding Conformance](#) includes further explanation of the accessibility-supported concept.

Conformance Requirements

In order for a Web page to conform to WCAG 2.0, all of the following conformance requirements must be satisfied:

1. **Conformance Level:** One of the following levels of conformance is met in full.
 - **Level A:** For Level A conformance (the minimum level of conformance), the Web page satisfies all the Level A Success Criteria, or a conforming alternate version is provided.
 - **Level AA:** For Level AA conformance, the Web page satisfies all the Level A and

-
- Level AA Success Criteria, or a Level AA conforming alternate version is provided.
 - **Level AAA:** For Level AAA conformance, the Web page satisfies all the Level A, Level AA and Level AAA Success Criteria, or a Level AAA conforming alternate version is provided.

Note 1: Although conformance can only be achieved at the stated levels, authors are encouraged to report (in their claim) any progress toward meeting success criteria from all levels beyond the achieved level of conformance.

Note 2: It is not recommended that Level AAA conformance be required as a general policy for entire sites because it is not possible to satisfy all Level AAA Success Criteria for some content.

2. Full pages: Conformance (and conformance level) is for full Web page(s) only, and cannot be achieved if part of a Web page is excluded.

Note 1: For the purpose of determining conformance, alternatives to part of a page's content are considered part of the page when the alternatives can be obtained directly from the page, e.g., a long description or an alternative presentation of a video. *Note 2:* Authors of Web pages that cannot conform due to content outside of the author's control may consider a [Statement of Partial Conformance](#).

3. Complete processes: When a Web page is one of a series of Web pages presenting a process (i.e., a sequence of steps that need to be completed in order to accomplish an activity), all Web pages in the process conform at the specified level or better. (Conformance is not possible at a particular level if any page in the process does not conform at that level or better.)

Example: An online store has a series of pages that are used to select and purchase products. All pages in the series from start to finish (checkout) conform in order for any page that is part of the process to conform.

4. Only Accessibility-Supported Ways of Using Technologies: Only accessibility-supported ways of using technologies are relied upon to satisfy the success criteria. Any information or functionality that is provided in a way that is not accessibility supported is also available in a way that is accessibility supported. (See [Understanding accessibility support](#).)

5. Non-Interference: If technologies are used in a way that is not accessibility supported, or if they are used in a non-conforming way, then they do not block the ability of users to access the rest of the page. In addition, the Web page as a whole continues to meet the conformance requirements under each of the following conditions:

1. when any technology that is not relied upon is turned on in a user agent,
2. when any technology that is not relied upon is turned off in a user agent, and
3. when any technology that is not relied upon is not supported by a user agent

In addition, the following success criteria apply to all content on the page, including content that is not otherwise relied upon to meet conformance, because failure to meet them could interfere with any use of the page:

- **1.4.2 - Audio Control,**
- **2.1.2 - No Keyboard Trap,**
- **2.3.1 - Three Flashes or Below Threshold,** and
- **2.2.2 - Pause, Stop, Hide.**

Note: If a page cannot conform (for example, a conformance test page or an example page), it cannot be included in the scope of conformance or in a conformance claim.

For more information, including examples, see [Understanding Conformance Requirements](#).

Conformance Claims (Optional)

Conformance is defined only for Web pages. However, a conformance claim may be made to cover one page, a series of pages, or multiple related Web pages.

Required Components of a Conformance Claim

Conformance claims are **not required**. Authors can conform to WCAG 2.0 without making a claim. However, if a conformance claim is made, then the conformance claim **must** include the following information:

1. **Date** of the claim
2. **Guidelines title, version and URI** "Web Content Accessibility Guidelines 2.0 at <http://www.w3.org/TR/2008/REC-WCAG20-20081211/>"
3. **Conformance level** satisfied: (Level A, AA or AAA)
4. **A concise description of the Web pages**, such as a list of URIs for which the claim is made, including whether subdomains are included in the claim.

Note 1: The Web pages may be described by list or by an expression that describes all of the URIs included in the claim.

Note 2: Web-based products that do not have a URI prior to installation on the customer's Web site may have a statement that the product would conform when installed.

5. A list of the **Web content technologies relied upon**.

Note: If a conformance logo is used, it would constitute a claim and must be accompanied by the required components of a conformance claim listed above.

Optional Components of a Conformance Claim

In addition to the required components of a conformance claim above, consider providing additional information to assist users. Recommended additional information includes:

- A list of success criteria beyond the level of conformance claimed that have been met. This information should be provided in a form that users can use, preferably machine-readable metadata.
- A list of the specific technologies that are "*used but not relied upon*."
- A list of user agents, including assistive technologies that were used to test the content.
- Information about any additional steps taken that go beyond the success criteria to enhance accessibility.
- A machine-readable metadata version of the list of specific technologies that are relied upon.
- A machine-readable metadata version of the conformance claim.

Note 1: Refer to [Understanding Conformance Claims](#) for more information and example conformance claims.

Note 2: Refer to [Understanding Metadata](#) for more information about the use of

metadata in conformance claims.

Statement of Partial Conformance - Third Party Content

Sometimes, Web pages are created that will later have additional content added to them. For example, an email program, a blog, an article that allows users to add comments, or applications supporting user-contributed content. Another example would be a page, such as a portal or news site, composed of content aggregated from multiple contributors, or sites that automatically insert content from other sources over time, such as when advertisements are inserted dynamically.

In these cases, it is not possible to know at the time of original posting what the uncontrolled content of the pages will be. It is important to note that the uncontrolled content can affect the accessibility of the controlled content as well. Two options are available:

1. A determination of conformance can be made based on best knowledge. If a page of this type is monitored and repaired (non-conforming content is removed or brought into conformance) within two business days, then a determination or claim of conformance can be made since, except for errors in externally contributed content which are corrected or removed when encountered, the page conforms. No conformance claim can be made if it is not possible to monitor or correct non-conforming content;

OR

2. A "statement of partial conformance" may be made that the page does not conform, but could conform if certain parts were removed. The form of that statement would be, "This page does not conform, but would conform to WCAG 2.0 at level X if the following parts from uncontrolled sources were removed." In addition, the following would also be true of uncontrolled content that is described in the statement of partial conformance:

- a. It is not content that is under the author's control.
- b. It is described in a way that users can identify (e.g., they cannot be described as "all parts that we do not control" unless they are clearly marked as such.)

Statement of Partial Conformance – Language

A "statement of partial conformance due to language" may be made when the page does not conform, but would conform if accessibility support existed for (all of) the language(s) used on the page. The form of that statement would be, "This page does not conform, but would conform to WCAG 2.0 at level X if accessibility support existed for the following language(s):"

Appendix A: Glossary

This section is normative.

abbreviation

shortened form of a word, phrase, or name where the abbreviation has not become part of the language

Note 1: This includes initialisms and acronyms where:

1. **initialisms** are shortened forms of a name or phrase made from the initial letters of words or syllables contained in that name or phrase *Note 1:* Not defined in all languages.

Example 1: SNCF is a French initialism that contains the initial letters of the Société Nationale des Chemins de Fer, the French national railroad.

Example 2: ESP is an initialism for extrasensory perception.

2. **acronyms** are abbreviated forms made from the initial letters or parts of other words (in a name or phrase) which may be pronounced as a word

Example: NOAA is an acronym made from the initial letters of the National Oceanic and Atmospheric Administration in the United States.

Note 2: Some companies have adopted what used to be an initialism as their company name. In these cases, the new name of the company is the letters (for example, Ecma) and the word is no longer considered an abbreviation.

accessibility supported

supported by users' assistive technologies as well as the accessibility features in browsers and other user agents To qualify as an accessibility-supported use of a Web content technology (or feature of a technology), both 1 and 2 must be satisfied for a Web content technology (or feature):

1. **The way that the Web content technology is used must be supported by users' assistive technology (AT).** This means that the way that the technology is used has been tested for interoperability with users' assistive technology in the human language(s) of the content, **AND**

2. **The Web content technology must have accessibility-supported user agents that are available to users.** This means that at least one of the following four statements is true:

a. The technology is supported natively in widely-distributed user agents that are also accessibility supported (such as HTML and CSS);

OR

b. The technology is supported in a widely-distributed plug-in that is also accessibility supported;

OR

c. The content is available in a closed environment, such as a university or corporate network, where the user agent required by the technology and used by the organization is also accessibility supported;

OR

d. The user agent(s) that support the technology are accessibility supported and are available for download or purchase in a way that:

- does not cost a person with a disability any more than a person without a disability **and**
- is as easy to find and obtain for a person with a disability as it is for a person without disabilities.

Note 1: The WCAG Working group and the W3C do not specify which or how much support by assistive technologies there must be for a particular use of a Web technology in order for it to be classified as accessibility supported. (See [Level of Assistive Technology Support Needed for "Accessibility Support"](#).)

Note 2: Web technologies can be used in ways that are not accessibility supported as long as they are not relied upon and the page as a whole meets the conformance requirements, including [Conformance Requirement 4: Only Accessibility-Supported Ways of Using Technologies](#) and [Conformance Requirement 5: Non-Interference](#), are met.

Note 3: When a Web Technology is used in a way that is "accessibility supported," it does not imply that the entire technology or all uses of the technology are supported. Most technologies, including HTML, lack support for at least one feature or use. Pages conform to WCAG only if the uses of the technology that are accessibility supported can be relied upon to meet WCAG

requirements.

Note 4: When citing Web content technologies that have multiple versions, the version(s) supported should be specified.

Note 5: One way for authors to locate uses of a technology that are accessibility supported would be to consult compilations of uses that are documented to be accessibility supported. (See [Understanding Accessibility-Supported Web Technology Uses](#).) Authors, companies, technology vendors, or others may document accessibility-supported ways of using Web content technologies. However, all ways of using technologies in the documentation would need to meet the definition of accessibility-supported Web content technologies above.

alternative for time-based media

document including correctly sequenced text descriptions of time-based visual and auditory information and providing a means for achieving the outcomes of any time-based interaction

Note: A screenplay used to create the synchronized media content would meet this definition only if it was corrected to accurately represent the final synchronized media after editing.

ambiguous to users in general

the purpose cannot be determined from the link and all information of the Web page presented to the user simultaneously with the link (i.e., readers without disabilities would not know what a link would do until they activated it)

Example: The word guava in the following sentence "One of the notable exports is guava" is a link. The link could lead to a definition of guava, a chart listing the quantity of guava exported or a photograph of people harvesting guava. Until the link is activated, all readers are unsure and the person with a disability is not at any disadvantage.

ASCII art

picture created by a spatial arrangement of characters or glyphs (typically from the 95 printable characters defined by ASCII).

assistive technology (as used in this document)

hardware and/or software that acts as a user agent, or along with a mainstream user agent, to provide functionality to meet the requirements of users with disabilities that go beyond those offered by mainstream user agents

Note 1: *functionality provided by assistive technology includes alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).*

Note 2: *Assistive technologies often communicate data and messages with mainstream user agents by using and monitoring APIs.*

Note 3: *The distinction between mainstream user agents and assistive technologies is not absolute. Many mainstream user agents provide some features to assist individuals with disabilities. The basic difference is that mainstream user agents target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The mainstream user agent may provide important functionality to assistive technologies like retrieving Web content from program objects or parsing markup*

into identifiable bundles.

Example: Assistive technologies that are important in the context of this document include the following:

- screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;
- screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;
- text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;
- speech recognition software, which may be used by people who have some physical disabilities;
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use head pointers, single switches, sip/puff and other special input devices.);
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

audio

the technology of sound reproduction

Note: Audio can be created synthetically (including speech synthesis), recorded from real world sounds, or both.

audio description

narration added to the soundtrack to describe important visual details that cannot be understood from the main soundtrack alone

Note 1: Audio description of video provides information about actions, characters, scene changes, on-screen text, and other visual content.

Note 2: In standard audio description, narration is added during existing pauses in dialogue. (See also extended audio description.)

Note 3: Where all of the video information is already provided in existing audio, no additional audio description is necessary.

Note 4: Also called "video description" and "descriptive narration."

audio-only

a time-based presentation that contains only audio (no video and no interaction)

blinking

switch back and forth between two visual states in a way that is meant to draw attention

Note: See also flash. It is possible for something to be large enough and blink brightly enough at the right frequency to be also classified as a flash.

blocks of text

more than one sentence of text

CAPTCHA

initialism for "Completely Automated Public Turing test to tell Computers and Humans Apart"

Note 1: CAPTCHA tests often involve asking the user to type in text that is displayed in an obscured image or audio file.

Note 2: A Turing test is any system of tests designed to differentiate a human from a computer. It is named after famed computer scientist Alan Turing. The term was coined by researchers at Carnegie Mellon University. [\[CAPTCHA\]](#)

captions

synchronized visual and/or text alternative for both speech and non-speech audio information needed to understand the media content

Note 1: Captions are similar to dialogue-only subtitles except captions convey not only the content of spoken dialogue, but also equivalents for non-dialogue audio information needed to understand the program content, including sound effects, music, laughter, speaker identification and location.

Note 2: Closed Captions are equivalents that can be turned on and off with some players.

Note 3: Open Captions are any captions that cannot be turned off. For example, if the captions are visual equivalent images of text embedded in video.

Note 4: Captions should not obscure or obstruct relevant information in the video.

Note 5: In some countries, captions are called subtitles.

Note 6: Audio descriptions can be, but do not need to be, captioned since they are descriptions of information that is already presented visually.

changes of context

major changes in the content of the Web page that, if made without user awareness, can disorient users who are not able to view the entire page simultaneously Changes in context include changes of:

1. user agent;
2. viewport;
3. focus;
4. content that changes the meaning of the Web page.

Note: A change of content is not always a change of context. Changes in content, such as an expanding outline, dynamic menu, or a tab control do not necessarily change the context, unless they also change one of the above (e.g., focus).

Example: Opening a new window, moving focus to a different component, going to a new page (including anything that would look to a user as if they had moved to a new page) or significantly re-arranging the content of a page are examples of changes of context.

conformance

satisfying all the requirements of a given standard, guideline or specification

conforming alternate version

version that

1. conforms at the designated level, and
2. provides all of the same information and functionality in the same human language, and
3. is as up to date as the non-conforming content, and
4. for which at least one of the following is true:
 - a. the conforming version can be reached from the non-conforming page via an accessibility-supported mechanism, or
 - b. the non-conforming version can only be reached from the conforming version, or
 - c. the non-conforming version can only be reached from a conforming page that also provides a mechanism to reach the conforming version

Note 1: In this definition, "can only be reached" means that there is some mechanism, such as a conditional redirect, that prevents a user from "reaching" (loading) the non-conforming page unless the user had just come from the conforming version.

Note 2: The alternate version does not need to be matched page for page with the original (e.g., the conforming alternate version may consist of multiple pages).

Note 3: If multiple language versions are available, then conforming alternate versions are required for each language offered.

Note 4: Alternate versions may be provided to accommodate different technology environments or user groups. Each version should be as conformant as possible. One version would need to be fully conformant in order to meet [conformance requirement 1](#).

Note 5: The conforming alternative version does not need to reside within the scope of conformance, or even on the same Web site, as long as it is as freely available as the non-conforming version.

Note 6: Alternate versions should not be confused with supplementary content, which support the original page and enhance comprehension.

Note 7: Setting user preferences within the content to produce a conforming version is an acceptable mechanism for reaching another version as long as the method used to set the preferences is accessibility supported.

See [Understanding Conforming Alternate Versions content \(Web content\)](#)

content (Web content)

information and sensory experience to be communicated to the user by means of a user agent, including code or markup that defines the content's structure, presentation, and interactions.

context-sensitive help

help text that provides information related to the function currently being performed

Note: Clear labels can act as context-sensitive help.

contrast ratio

$(L1 + 0.05) / (L2 + 0.05)$, where L1 is the relative luminance of the lighter of the colors, and L2 is the relative luminance of the darker of the colors. Note 1: Contrast ratios can range from 1 to 21 (commonly written 1:1 to 21:1).

Note 2: Because authors do not have control over user settings as to how text is rendered (for example font smoothing or anti-aliasing), the contrast ratio for text can be evaluated with anti-aliasing turned off.

Note 3: For the purpose of Success Criteria 1.4.3 and 1.4.6, contrast is measured with respect to the specified background over which the text is rendered in normal usage. If no background color is specified, then white is assumed.

Note 4: Background color is the specified color of content over which the text is to be rendered in normal usage. It is a failure if no background color is specified when the text color is specified, because the user's default background color is unknown and cannot be evaluated for sufficient contrast. For the same reason, it is a failure if no text color is specified when a background color is specified.

Note 5: When there is a border around the letter, the border can add contrast and would be used in calculating the contrast between the letter and its background. A narrow border around the letter would be used as the letter. A wide border around the letter that fills in the inner details of the letters acts as a halo and would be considered background.

Note 6: WCAG conformance should be evaluated for color pairs specified in the

content that an author would expect to appear adjacent in typical presentation. Authors need not consider unusual presentations, such as color changes made by the user agent, except where caused by authors' code.

correct reading sequence

any sequence where words and paragraphs are presented in an order that does not change the meaning of the content.

emergency

a sudden, unexpected situation or occurrence that requires immediate action to preserve health, safety, or property.

essential

if removed, would fundamentally change the information or functionality of the content, **and** information and functionality cannot be achieved in another way that would conform.

extended audio description

audio description that is added to an audiovisual presentation by pausing the video so that there is time to add additional description

Note: This technique is only used when the sense of the video would be lost without the additional audio description and the pauses between dialogue/narration are too short.

flash

a pair of opposing changes in relative luminance that can cause seizures in some people if it is large enough and in the right frequency range

Note 1: See general flash and red flash thresholds for information about types of flash that are not allowed.

Note 2: See also blinking. **Functionality** processes and outcomes achievable through user action

general flash and red flash thresholds

a flash or rapidly changing image sequence is below the threshold (i.e., content **passes**) if any of the following are true:

1. there are no more than three **general flashes** and / or no more than three **red flashes** within any one-second period; or
2. the combined area of flashes occurring concurrently occupies no more than a total of .006 steradians within any 10 degree visual field on the screen (25% of any 10 degree visual field on the screen) at typical viewing distance where:
 - A **general flash** is defined as a pair of opposing changes in relative luminance of 10% or more of the maximum relative luminance where the relative luminance of the darker image is below 0.80; and where "a pair of opposing changes" is an increase followed by a decrease, or a decrease followed by an increase, and
 - A **red flash** is defined as any pair of opposing transitions involving a saturated red.

Exception: Flashing that is a fine, balanced, pattern such as white noise or an alternating checkerboard pattern with "squares" smaller than 0.1 degree (of visual field at typical viewing distance) on a side does not violate the thresholds.

Note 1: For general software or Web content, using a 341 x 256 pixel rectangle anywhere on the displayed screen area when the content is viewed at 1024 x 768 pixels will provide a good estimate of a 10 degree visual field for standard screen sizes and viewing distances (e.g., 15-17 inch screen at 22-26 inches). (Higher

resolutions displays showing the same rendering of the content yield smaller and safer images so it is lower resolutions that are used to define the thresholds.)

Note 2: A transition is the change in relative luminance (or relative luminance/color for red flashing) between adjacent peaks and valleys in a plot of relative luminance (or relative luminance/color for red flashing) measurement against time. A flash consists of two opposing transitions.

Note 3: The current working definition in the field for "**pair of opposing transitions involving a saturated red**" is where, for either or both states involved in each transition, $R/(R+G+B) \geq 0.8$, and the change in the value of $(R-G-B) \times 320$ is > 20 (negative values of $(R-G-B) \times 320$ are set to zero) for both transitions. R, G, B values range from 0-1 as specified in "relative luminance" definition. [HARDING-BINNIE]

Note 4: Tools are available that will carry out analysis from video screen capture. However, no tool is necessary to evaluate for this condition if flashing is less than or equal to 3 flashes in any one second. Content automatically passes (see #1 and #2 above).

human language

language that is spoken, written or signed (through visual or tactile means) to communicate with humans

Note: See also sign language.

idiom

phrase whose meaning cannot be deduced from the meaning of the individual words and the specific words cannot be changed without losing the meaning

Note: idioms cannot be translated directly, word for word, without losing their (cultural or language-dependent) meaning.

Example 1: In English, "spilling the beans" means "revealing a secret." However, "knocking over the beans" or "spilling the vegetables" does not mean the same thing.

Example 2: In Japanese, the phrase “さじを投げる” literally translates into "he throws a spoon," but it means that there is nothing he can do and finally he gives up.

Example 3: In Dutch, "Hij ging met de kippen op stok" literally translates into "He went to roost with the chickens," but it means that he went to bed early.

image of text

text that has been rendered in a non-text form (e.g., an image) in order to achieve a particular visual effect

Note: This does not include text that is part of a picture that contains significant other visual content.

Example: A person's name on a nametag in a photograph. **Informative** for information purposes and not required for conformance *Note:* Content required for conformance is referred to as "normative."

input error

information provided by the user that is not accepted *Note:* This includes:

1. Information that is required by the Web page but omitted by the user
2. Information that is provided by the user but that falls outside the required data format or values

jargon

words used in a particular way by people in a particular field

Example: The word StickyKeys is jargon from the field of assistive technology/accessibility.

keyboard interface

interface used by software to obtain keystroke input

Note 1: A keyboard interface allows users to provide keystroke input to programs even if the native technology does not contain a keyboard.

Example: A touchscreen PDA has a keyboard interface built into its operating system as well as a connector for external keyboards. Applications on the PDA can use the interface to obtain keyboard input either from an external keyboard or from other applications that provide simulated keyboard output, such as handwriting interpreters or speech-to-text applications with "keyboard emulation" functionality.

Note 2: Operation of the application (or parts of the application) through a keyboard-operated mouse emulator, such as MouseKeys, does not qualify as operation through a keyboard interface because operation of the program is through its pointing device interface, not through its keyboard interface.

label

text or other component with a text alternative that is presented to a user to identify a component within Web content

Note 1: A label is presented to all users whereas the name may be hidden and only exposed by assistive technology. In many (but not all) cases the name and the label are the same.

Note 2: The term label is not limited to the label element in HTML.

large scale (text)

with at least 18 point or 14 point bold or font size that would yield equivalent size for Chinese, Japanese and Korean (CJK) fonts

Note 1: Fonts with extraordinarily thin strokes or unusual features and characteristics that reduce the familiarity of their letter forms are harder to read, especially at lower contrast levels.

Note 2: Font size is the size when the content is delivered. It does not include resizing that may be done by a user.

Note 3: The actual size of the character that a user sees is dependent both on the author-defined size and the user's display or user-agent settings. For many mainstream body text fonts, 14 and 18 point is roughly equivalent to 1.2 and 1.5 em or to 120% or 150% of the default size for body text (assuming that the body font is 100%), but authors would need to check this for the particular fonts in use. When fonts are defined in relative units, the actual point size is calculated by the user agent for display. The point size should be obtained from the user agent, or calculated based on font metrics as the user agent does, when evaluating this success criterion. Users who have low vision would be responsible for choosing appropriate settings.

Note 4: When using text without specifying the font size, the smallest font size used on major browsers for unspecified text would be a reasonable size to assume for the font. If a level 1 heading is rendered in 14pt bold or higher on major browsers, then it would be reasonable to assume it is large text. Relative scaling can be calculated from the default sizes in a similar fashion.

Note 5: The 18 and 14 point sizes for roman texts are taken from the minimum size for large print (14pt) and the larger standard font size (18pt). For other fonts such as CJK languages, the "equivalent" sizes would be the minimum large print

size used for those languages and the next larger standard large print size.

legal commitments

transactions where the person incurs a legally binding obligation or benefit

Example: A marriage license, a stock trade (financial and legal), a will, a loan, adoption, signing up for the army, a contract of any type, etc.

link purpose

nature of the result obtained by activating a hyperlink

live

information captured from a real-world event and transmitted to the receiver with no more than a broadcast delay

Note 1: A broadcast delay is a short (usually automated) delay, for example used in order to give the broadcaster time to queue or censor the audio (or video) feed, but not sufficient to allow significant editing.

Note 2: If information is completely computer generated, it is not live.

lower secondary education level

the two or three year period of education that begins after completion of six years of school and ends nine years after the beginning of primary education

Note: This definition is based on the International Standard Classification of Education [UNESCO].

mechanism

process or technique for achieving a result

Note 1: The mechanism may be explicitly provided in the content, or may be relied upon to be provided by either the platform or by user agents, including assistive technologies.

Note 2: The mechanism needs to meet all success criteria for the conformance level claimed.

media alternative for text

media that presents no more information than is already presented in text (directly or via text alternatives)

Note: A media alternative for text is provided for those who benefit from alternate representations of text. Media alternatives for text may be audio-only, video-only (including sign-language video), or audio-video.

name

text by which software can identify a component within Web content to the user

Note 1: The name may be hidden and only exposed by assistive technology, whereas a label is presented to all users. In many (but not all) cases, the label and the name are the same.

Note 2: This is unrelated to the name attribute in HTML.

navigated sequentially

navigated in the order defined for advancing focus (from one element to the next) using a keyboard interface

non-text content

any content that is not a sequence of characters that can be programmatically determined or where the sequence is not expressing something in human language

Note: This includes ASCII Art (which is a pattern of characters), emoticons, leetspeak (which uses character substitution), and images representing text

normative

required for conformance *Note 1*: One may conform in a variety of well-defined ways to this document.

Note 2: Content identified as "informative" or "non-normative" is never required for conformance.

on a full-screen window

on the most common sized desktop/laptop display with the viewport maximized

Note: Since people generally keep their computers for several years, it is best not to rely on the latest desktop/laptop display resolutions but to consider the common desktop/laptop display resolutions over the course of several years when making this evaluation.

paused

stopped by user request and not resumed until requested by user

prerecorded

information that is not live

presentation

rendering of the content in a form to be perceived by users

primary education level

six year time period that begins between the ages of five and seven, possibly without any previous education

Note: This definition is based on the International Standard Classification of Education [UNESCO].

process

series of user actions where each action is required in order to complete an activity

Example 1: Successful use of a series of Web pages on a shopping site requires users to view alternative products, prices and offers, select products, submit an order, provide shipping information and provide payment information.

Example 2: An account registration page requires successful completion of a Turing test before the registration form can be accessed.

programmatically determined (programmatically determinable)

determined by software from author-supplied data provided in a way that different user agents, including assistive technologies, can extract and present this information to users in different modalities

Example 1: Determined in a markup language from elements and attributes that are accessed directly by commonly available assistive technology.

Example 2: Determined from technology-specific data structures in a non-markup language and exposed to assistive technology via an accessibility API that is supported by commonly available assistive technology.

programmatically determined link context

additional information that can be programmatically determined from relationships with a link, combined with the link text, and presented to users in different modalities

Example: In HTML, information that is programmatically determinable from a link in English includes text that is in the same paragraph, list, or table cell as the link or in a table header cell that is associated with the table cell that contains the link.

Note: Since screen readers interpret punctuation, they can also provide the context from the current sentence, when the focus is on a link in that sentence.

programmatically set

set by software using methods that are supported by user agents, including assistive technologies

pure decoration

serving only an aesthetic purpose, providing no information, and having no functionality

Note: Text is only purely decorative if the words can be rearranged or substituted without changing their purpose.

Example: The cover page of a dictionary has random words in very light text in the background.

real-time event

event that a) occurs at the same time as the viewing and b) is not completely generated by the content

Example 1: A Webcast of a live performance (occurs at the same time as the viewing and is not prerecorded).

Example 2: An on-line auction with people bidding (occurs at the same time as the viewing).

Example 3: Live humans interacting in a virtual world using avatars (is not completely generated by the content and occurs at the same time as the viewing).

relationships

meaningful associations between distinct pieces of content

relative luminance

the relative brightness of any point in a colorspace, normalized to 0 for darkest black and

1 for lightest white

Note 1: For the sRGB colorspace, the relative luminance of a color is defined as $L = 0.2126 * R + 0.7152 * G + 0.0722 * B$ where **R**, **G** and **B** are defined as:

- if $R_{sRGB} \leq 0.03928$ then $R = R_{sRGB}/12.92$ else $R = ((R_{sRGB}+0.055)/1.055)^{2.4}$
- if $G_{sRGB} \leq 0.03928$ then $G = G_{sRGB}/12.92$ else $G = ((G_{sRGB}+0.055)/1.055)^{2.4}$
- if $B_{sRGB} \leq 0.03928$ then $B = B_{sRGB}/12.92$ else $B = ((B_{sRGB}+0.055)/1.055)^{2.4}$

and R_{sRGB} , G_{sRGB} , and B_{sRGB} are defined as:

- $R_{sRGB} = R_{8bit}/255$
- $G_{sRGB} = G_{8bit}/255$
- $B_{sRGB} = B_{8bit}/255$

The "^" character is the exponentiation operator. (Formula taken from [\[sRGB\]](#) and [\[IEC-4WD\]](#)).

Note 2: Almost all systems used today to view Web content assume sRGB encoding. Unless it is known that another color space will be used to process and display the content, authors should evaluate using sRGB colorspace. If using other color spaces, see [Understanding Success Criterion 1.4.3](#).

Note 3: If dithering occurs after delivery, then the source color value is used. For colors that are dithered at the source, the average values of the colors that are dithered should be used (average R, average G, and average B).

Note 4: Tools are available that automatically do the calculations when testing contrast and flash.

Note 5: A [MathML version of the relative luminance definition](#) is available.

relied upon (technologies that are)

the content would not conform if that technology is turned off or is not supported

role

text or number by which software can identify the function of a component within Web content

Example: A number that indicates whether an image functions as a hyperlink, command button, or check box.

same functionality

same result when used

Example: A submit "search" button on one Web page and a "find" button on another Web page may both have a field to enter a term and list topics in the Web site related to the term submitted. In this case, they would have the same functionality but would not be labeled consistently.

same relative order

same position relative to other items

Note: Items are considered to be in the same relative order even if other items are inserted or removed from the original order. For example, expanding navigation menus may insert an additional level of detail or a secondary navigation section may be inserted into the reading order.

satisfies a success criterion

the success criterion does not evaluate to 'false' when applied to the page

section

A self-contained portion of written content that deals with one or more related topics or thoughts

Note: A section may consist of one or more paragraphs and include graphics, tables, lists and sub-sections.

set of Web pages

collection of Web pages that share a common purpose and that are created by the same author, group or organization

Note: Different language versions would be considered different sets of Web pages.

sign language

a language using combinations of movements of the hands and arms, facial expressions, or body positions to convey meaning

sign language interpretation

translation of one language, generally a spoken language, into a sign language

Note: True sign languages are independent languages that are unrelated to the spoken language(s) of the same country or region.

specific sensory experience

a sensory experience that is not purely decorative and does not primarily convey important information or perform a function

Example: Examples include a performance of a flute solo, works of visual art etc.

structure

1. The way the parts of a Web page are organized in relation to each other; and

2. The way a collection of Web pages is organized

supplemental content

additional content that illustrates or clarifies the primary content *Example 1:* An audio version of a Web page.

Example 2: An illustration of a complex process. *Example 3:* A paragraph summarizing the major outcomes and recommendations made in a research study.

synchronized media

audio or video synchronized with another format for presenting information and/or with time-based interactive components, unless the media is a media alternative for text that is clearly labeled as such **technology (Web content)**

mechanism for encoding instructions to be rendered, played or executed by user agents

Note 1: As used in these guidelines "Web Technology" and the word "technology" (when used alone) both refer to Web Content Technologies.

Note 2: Web content technologies may include markup languages, data formats, or programming languages that authors may use alone or in combination to create end- user experiences that range from static Web pages to synchronized media presentations to dynamic Web applications.

Example: Some common examples of Web content technologies include HTML, CSS, SVG, PNG, PDF, Flash, and JavaScript.

text

sequence of characters that can be programmatically determined, where the sequence is expressing something in human language

text alternative

text that is programmatically associated with non-text content or referred to from text that is programmatically associated with non-text content. Programmatically associated text is text whose location can be programmatically determined from the non-text content.

Example: An image of a chart is described in text in the paragraph after the chart. The short text alternative for the chart indicates that a description follows.

Note: Refer to [Understanding Text Alternatives](#) for more information.

used in an unusual or restricted way

words used in such a way that requires users to know exactly which definition to apply in order to understand the content correctly

Example: The term "gig" means something different if it occurs in a discussion of music concerts than it does in article about computer hard drive space, but the appropriate definition can be determined from context. By contrast, the word "text" is used in a very specific way in WCAG 2.0, so a definition is supplied in the glossary.

user agent

any software that retrieves and presents Web content for users

Example: Web browsers, media players, plug-ins, and other programs — including assistive technologies — that help in retrieving, rendering, and interacting with Web content.

user-controllable

data that is intended to be accessed by users

Note: This does not refer to such things as Internet logs and search engine monitoring data.

Example: Name and address fields for a user's account.

user interface component

a part of the content that is perceived by users as a single control for a distinct function

Note 1: Multiple user interface components may be implemented as a single programmatic element. Components here is not tied to programming techniques, but rather to what the user perceives as separate controls.

Note 2: User interface components include form elements and links as well as components generated by scripts.

Example: An applet has a "control" that can be used to move through content by line or page or random access. Since each of these would need to have a name and be settable independently, they would each be a "user interface component."

video

the technology of moving or sequenced pictures or images *Note:* Video can be made up of animated or photographic images, or both.

video-only

a time-based presentation that contains only video (no audio and no interaction)

viewport

object in which the user agent presents content

Note 1: The user agent presents content through one or more viewports. Viewports include windows, frames, loudspeakers, and virtual magnifying glasses. A viewport may contain another viewport (e.g., nested frames). Interface components created by the user agent such as prompts, menus, and alerts are not viewports.

Note 2: This definition is based on [User Agent Accessibility Guidelines 1.0 Glossary](#). **visually customized** the font, size, color, and background can be set

Web page

a non-embedded resource obtained from a single URI using HTTP plus any other resources that are used in the rendering or intended to be rendered together with it by a user agent

Note 1: Although any "other resources" would be rendered together with the primary resource, they would not necessarily be rendered simultaneously with each other.

Note 2: For the purposes of conformance with these guidelines, a resource must be "non-embedded" within the scope of conformance to be considered a Web page.

Example 1: A Web resource including all embedded images and media.

Example 2: A Web mail program built using Asynchronous JavaScript and XML (AJAX). The program lives entirely at <http://example.com/mail>, but includes an inbox, a contacts area and a calendar. Links or buttons are provided that cause the inbox, contacts, or calendar to display, but do not change the URI of the page as a whole.

Example 3: A customizable portal site, where users can choose content to display from a set of different content modules.

Example 4: When you enter "<http://shopping.example.com/>" in your browser, you enter a movie-like interactive shopping environment where you visually move

around in a store dragging products off of the shelves around you and into a visual shopping cart in front of you. Clicking on a product causes it to be demonstrated with a specification sheet floating alongside. This might be a single-page Web site or just one page within a Web site.

Appendix B: Acknowledgments

This section is informative.

This publication has been funded in part with Federal funds from the U.S. Department of Education, National Institute on Disability and Rehabilitation Research (NIDRR) under contract number ED05CO0039. The content of this publication does not necessarily reflect the views or policies of the U.S. Department of Education, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

Additional information about participation in the Web Content Accessibility Guidelines Working Group (WCAG WG) can be found on the [Working Group home page](#).

Participants active in the WCAG WG at the time of publication

- Bruce Bailey (U.S. Access Board)
- Frederick Boland (NIST)
- Ben Caldwell (Trace R&D Center, University of Wisconsin)
- Sofia Celic (W3C Invited Expert)
- Michael Cooper (W3C)
- Roberto Ellero (International Webmasters Association / HTML Writers Guild)
- Bengt Farre (Rigab)
- Loretta Guarino Reid (Google)
- Katie Haritos-Shea Andrew Kirkpatrick (Adobe)
- Drew LaHart (IBM) Alex Li (SAP AG)
- David MacDonald (E-Ramp Inc.)
- Roberto Scano (International Webmasters Association / HTML Writers Guild)
- Cynthia Shelly (Microsoft)
- Andi Snow-Weaver (IBM)
- Christophe Strobbe (DocArch, K.U.Leuven)
- Gregg Vanderheiden (Trace R&D Center, University of Wisconsin)

Other previously active WCAG WG participants and other contributors to WCAG 2.0

Shadi Abou-Zahra, Jim Allan, Jenae Andershonis, Avi Arditti, Aries Arditi, Mike Barta, Sandy Bartell, Kynn Bartlett, Marco Bertoni, Harvey Bingham, Chris Blouch, Paul Bohman, Patrice Bourlon, Judy Brewer, Andy Brown, Dick Brown, Doyle Burnett, Raven Calais, Tomas Caspers, Roberto Castaldo, Sambhavi Chandrashekar, Mike Cherim, Jonathan Chetwynd, Wendy Chisholm, Alan Chuter, David M Clark, Joe Clark, James Coltham, James Craig, Tom Croucher, Nir Dagan, Daniel Dardailier, Geoff Deering, Pete DeVasto, Don Evans, Neal Ewers, Steve Faulkner, Lainey Feingold, Alan J. Flavell, Nikolaos Floratos, Kentarou Fukuda, Miguel Garcia, P.J. Gardner, Greg Gay, Becky Gibson, Al Gilman, Kerstin Goldsmith, Michael Grade, Jon Gunderson, Emmanuelle Gutiérrez y Restrepo, Brian Hardy, Eric Hansen, Sean Hayes, Shawn Henry, Hans Hillen, Donovan Hipke, Bjoern Hoehrmann, Chris Hofstader, Yvette

Hoitink, Carlos Iglesias, Ian Jacobs, Phill Jenkins, Jyotsna Kaki, Leonard R. Kasday, Kazuhito Kidachi, Ken Kipness, Marja- Riitta Koivunen, Preety Kumar, Gez Lemon, Chuck Letourneau, Scott Luebking, Tim Lacy, Jim Ley, William Loughborough, Greg Lowney, Luca Mascaro, Liam McGee, Jens Meiert, Niqui Merret, Alessandro Miele, Mathew J Mirabella, Charles McCathieNevile , Matt May, Marti McCuller, Sorcha Moore, Charles F. Munat, Robert Neff, Bruno von Niman, Tim Noonan, Sebastiano Nutarelli, Graham Oliver, Sean B. Palmer, Sailesh Panchang, Nigel Peck, Anne Pemberton, David Poehlman, Adam Victor Reed, Chris Ridpath, Lee Roberts, Gregory J. Rosmaita, Matthew Ross, Sharron Rush, Gian Sampson-Wild, Joel Sanda, Gordon Schantz, Lisa Seeman, John Slatin, Becky Smith, Jared Smith, Neil Soiffer, Jeanne Spellman, Mike Squillace, Michael Stenitzer, Jim Thatcher, Terry Thompson, Justin Thorp, Makoto Ueki, Eric Velleman, Dena Wainwright, Paul Walsch, Takayuki Watanabe, Jason White.

Appendix C: References

This section is informative.

CAPTCHA

The CAPTCHA Project, Carnegie Mellon University. The project is online at <http://www.captcha.net>.

HARDING-BINNIE

Harding G. F. A. and Binnie, C.D., Independent Analysis of the ITC Photosensitive Epilepsy Calibration Test Tape. 2002.

IEC-4WD

IEC/4WD 61966-2-1: Colour Measurement and Management in Multimedia Systems and Equipment - Part 2.1: Default Colour Space - sRGB. May 5, 1998.

sRGB

"A Standard Default Color Space for the Internet - sRGB," M. Stokes, M. Anderson, S. Chandrasekar, R. Motta, eds., Version 1.10, November 5, 1996. A copy of this paper is available at <http://www.w3.org/Graphics/Color/sRGB.html>.

UNESCO

International Standard Classification of Education, 1997. A copy of the standard is available at http://www.unesco.org/education/information/nfsunesco/doc/isced_1997.htm.

WCAG10

Web Content Accessibility Guidelines 1.0, G. Vanderheiden, W. Chisholm, I. Jacobs, Editors, W3C Recommendation, 5 May 1999, <http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/>. The latest version of WCAG 1.0 is available at <http://www.w3.org/TR/WAI-WEBCONTENT/>.

REFERENCES

- [1] Baumeister, H., Knapp, A., Koch, N. & Zhang, G.. Modelling Adaptivity with Aspects. In ICWE (2005) doi.org/10.1007/11531371_53
- [2] Baniassad, E. L. A., Clements, P. C., Araújo, J., Moreira, A., Rashid, A., Tekinerdogra, B.: Discovering Early Aspects. IEEE Software 23(1), 61-70 (2006)
- [3] Baxley, B. Universal Model of a User Interface. DUX (2003) doi:10.1145/997078.997090
- [4] Brichau, J., D'Hondt, T. Aspect-Oriented Software Development: An Introduction. AOSD Europe Project. <http://www.aosd-europe.net/>. Accessed 15th April 2010.
- [5] Broekstra, J., Kampman, A., van Harmelen, F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. ISWC 2342, 54-68 (2002)
- [6] 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
- [7] 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
- [8] 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)
- [9] Centeno, V., Kloos, C., Gaedke, M., Nussbaumer, M. Web Composition with WCAG in Mind. W4A (2005) doi:10.1145/1061811.1061819
- [10] Ceri, S., Brambilla, M., Fraternali, P. The History of WebML Lessons Learned from 10 Years of Model-Driven Development of Web Applications. Conceptual Modeling (2009) doi: 10.1007/978-3-642-02463-4_15
- [11] Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston (2000)
- [12] Chung, L., Supakkul, S. Representing FRs and NFRs: A Goal-oriented and Use Case Driven Approach. SERA (2004) doi:10.1007/11668855_3

-
- [13] De Troyer, Casteleyn, S., 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)
- [14] Dijkstra, E. W.: A Discipline of Programming. Prentice-Hall, NJ (1976)
- [15] Fiala, Z., Hinz, M., Meissner, K. Developing Component-based Adaptive Web Applications with the AMACONTbuilder. WSE (2005) doi:10.1109/WSE.2005.5
- [16] Fiala, Z., Houben G.J. A Generic Transcoding Tool for Making Web Applications Adaptive. CAiSE (2005) doi:10.1.1.124.7045
- [17] Filman, R., Elrad, T., Clarke, S., Aksit, M.: Aspect-oriented Software Development. Addison-Wesley, Vancouver BC (2004)
- [18] 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: Modeling and Implementing Web Applications. pp. 65-108. Springer-Verlag, London (2008)
- [19] Frasincar, F., Houben, G.J., Vdovjak, R. Specification Framework for Engineering Adaptive Web Applications. WWW (2002) doi:10.1.1.9.9912
- [20] Gaedke M., Nussbaumer, M., Meinecke, J.: WSLS: A Service-Based System for Reuse-Oriented Web Engineering. In: Matera, M., Comai, S. (eds.) Engineering Advanced Web Applications, pp. 26-37. Rinton Press, NJ (2004)
- [21] Gordillo, S., Rossi, G., Araújo, Moreira, J.A., Vairetti, C., Urbieto, M. Modeling and Composing Navigational Concerns in Web Applications. Requirements and Design Issues. LA-WEB (2006) doi:10.1109/LA-WEB.2006.21
- [22] Hoffman, D., Grivel, E., Battle, L.: Designing Software Architectures to Facilitate Accessible Web Applications. IBM Systems Journal 44(3), 467-484 (2005)
- [23] Houben, G-J., van der Sluijs, K., Barna, P., Broekstra, J., Casteleyn, S., Fiala, Z., Fransincar, F. Hera. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modeling and Implementing Web Applications. pp. 163-302. Springer-Verlag, London (2008)
- [24] ISO International Organization for Standardization/Technical Specification. Ergonomics of human-system interaction - Guidance on Accessibility for human-
-

-
- computer interfaces.
http://www.jtc1access.org/documents/swga_204/ISO_DIS_9241-171__E_.pdf
(2002); http://www.iso.org/iso/catalogue_detail?csnumber=30858 (2003).
Accessed 15 April 2010.
- [25] 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: Modeling and Implementing Web Applications. pp. 157-191. Springer-Verlag, London (2008)
- [26] Kreitzberg C. B., Little, A.: Useful, Usable and Desirable: Usability as a Core Development Competence. <http://msdn.microsoft.com/en-us/magazine/dd727512.aspx> (2009). Accessed April 15th 2009.
- [27] Larson, J.: Interactive Software: Tools for Building Interactive User Interfaces. Prentice Hall, NJ (1992)
- [28] Moreira, A., Araújo, J., Rashid, A. A Concern-oriented Requirements Engineering Model. CAiSE (2005) doi:10.1007/11431855_21
- [29] Moreno, L., Martinez, P., Ruiz, B. A MDD Approach for Modeling Web Accessibility. IWWOST (2008) doi:10.1.1.163.9478
- [30] Moreno, L. AWA, 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.
- [31] 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: Modeling and Implementing Web Applications. pp. 109-155. Springer-Verlag, London (2008)
- [32] Niederhausen, M., Fiala, Z., Kopcsek, N., Meissner, K. Web Software Evolution by Aspect-Oriented Adaptation Engineering. WSE (2007) doi:10.1109/WSE.2007.4380237
- [33] Offutt, J. Quality Attributes of Web Software Applications. *IEEE Software*, 19(2), 2002, 25-32 doi:10.1002/stvr.425
- [34] PAS 78. Publicly Available Specification: A Guide to Good Practice in Commissioning Accessible Websites, ICS 35.240.30. Disability Rights
-

-
- Commission (DRC) <http://www.hobo-web.co.uk/seo-blog/pas-78/> (2006-2011). Accessed 15 April 2010.
- [35] 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
- [36] Rossi. G., Schwabe, D. Modeling and Implementing Web Applications with OOADM. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modeling and Implementing Web Applications. pp. 109-155. Springer-Verlag, London (2008)
- [37] Schauerhuber, A., Wimmer M., Schwinger, W., Kapsammer, E. & Retschitzegger, W. Aspect-oriented Modeling of Ubiquitous Web Applications: The AspectWebML Approach. ECBS MBD (2007) doi:10.1109/ECBS.2007.20
- [38] Section 508. Electronic and Information Technology Accessibility Standards <http://www.section508.gov/index.cfm?fuseAction=stdsdoc> (2000-2010). Accessed 15 April 2010
- [39] Sommerville, I.: Software Engineering 8th Edition. Pearson Education Limited, Harlow (2007)
- [40] Stanca Law. Italian Legislation on Accessibility. http://www.pubbliaccesso.it/biblioteca/documentazione/guidelines_study/index.htm (2004). Accessed 25 January 2010.
- [41] Thatcher, J., Burks, M., Heilmann, Ch., Henry, S., Kirpatrick, A., Lauke, P., Lawson, B., Regan, B., Rutter, R., Urban, M., Waddell, C.: Web Accessibility - Web Standards and Regulatory Compliance. Friendsof ED, USA (2006)
- [42] Update of the 508 Standards - Draft Information and Communication Technology (ICT) Standards and Guidelines and the Telecommunications Act Guidelines. <http://www.access-board.gov/sec508/refresh/draft-rule.htm#e106> (2010). Accessed July 14th 2011.
- [43] Vilain, P., Schwabe, D.: Improving the Web Application Design process with UIDs, IWWOST Workshop Program. <http://users.dsic.upv.es/~west/iwwost02/papers/vilain.pdf> (2002). Accessed June 1st 2009.
-

-
- [44] 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
- [45] W3C: Web Content Accessibility Guidelines 1.0. (WCAG 1.0). <http://www.w3.org/TR/WCAG10/> (1999). Accessed April 15th 2010.
- [46] W3C: Web Content Accessibility Guidelines 2.0 (WCAG 2.0). <http://www.w3.org/TR/WCAG20/> (2008). Accessed April 15th 2010.
- [47] W3C: HTML Techniques for Web Content Accessibility Guidelines 1.0. <http://www.w3.org/TR/WCAG10-HTML-TECHS/> (2000). Accessed April 15th 2010.
- [48] W3C: User Agent Accessibility Guidelines 1.0 (UAAG 1.0). <http://www.w3.org/TR/WAI-USERAGENT/> (2002). Accessed April 22th 2010.
- [49] W3C-WAI: Comparison of WCAG 1.0 Checkpoints to WCAG 2.0. <http://www.w3.org/WAI/WCAG20/from10/comparison/> (2008). Accessed April 22th 2010.
- [50] W3C-WAI: Web Content Accessibility Guidelines (WCAG) 1.0 Documents. <http://www.w3.org/WAI/intro/wcag10docs.php> (2006). Accessed April 22th 2010.
- [51] Woods, S. Websites for Visually Impaired Users. Thesis <http://wise.vub.ac.be/Downloads/Theses/Woods-thesis.pdf> (2006-2007). Accessed April 15th 2009.
- [52] 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
- [53] 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).