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.

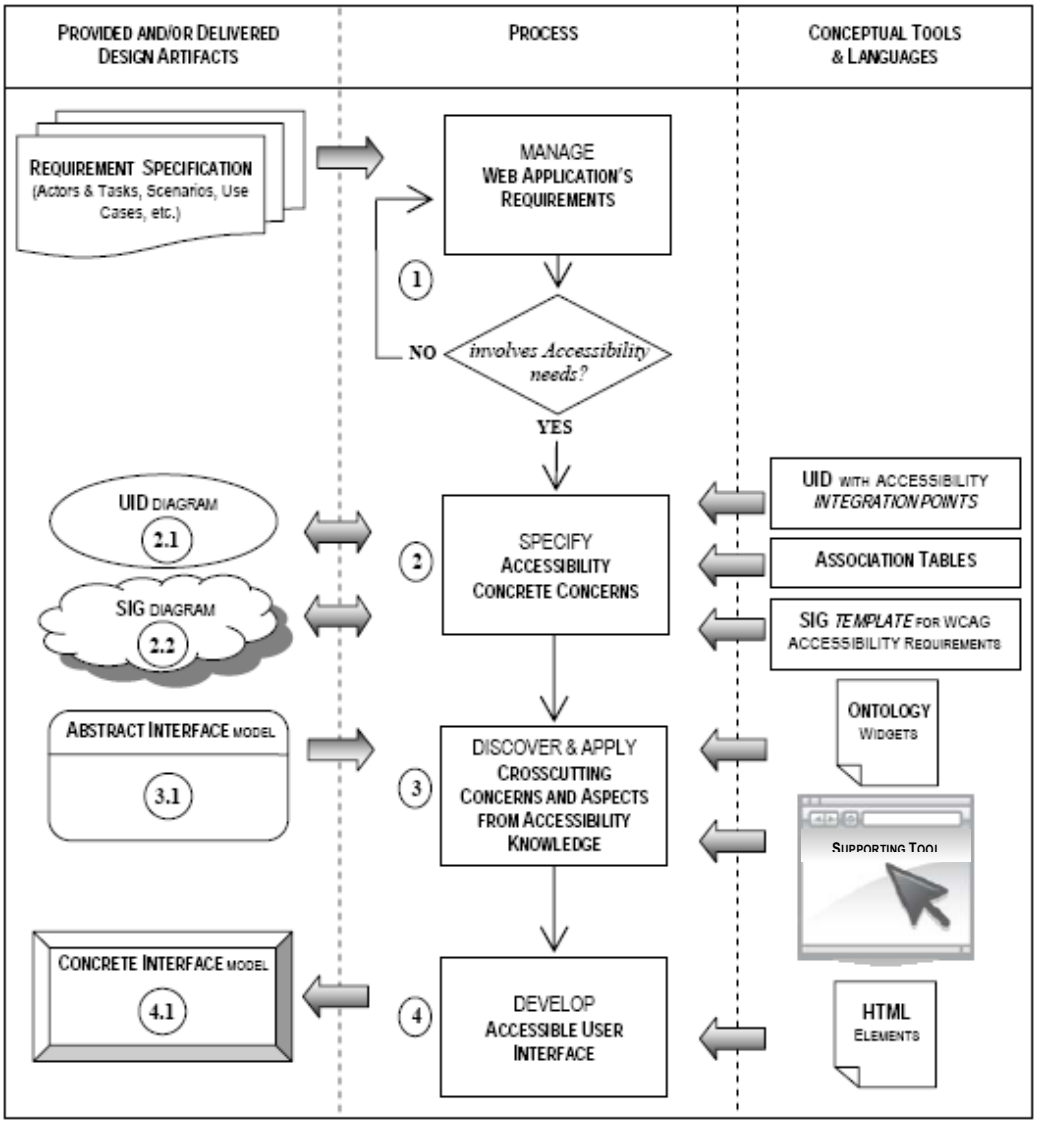

Figure 4.1: Overview of Our Approach

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 barriers[39] 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.htmlhttp://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 UML[40] 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.

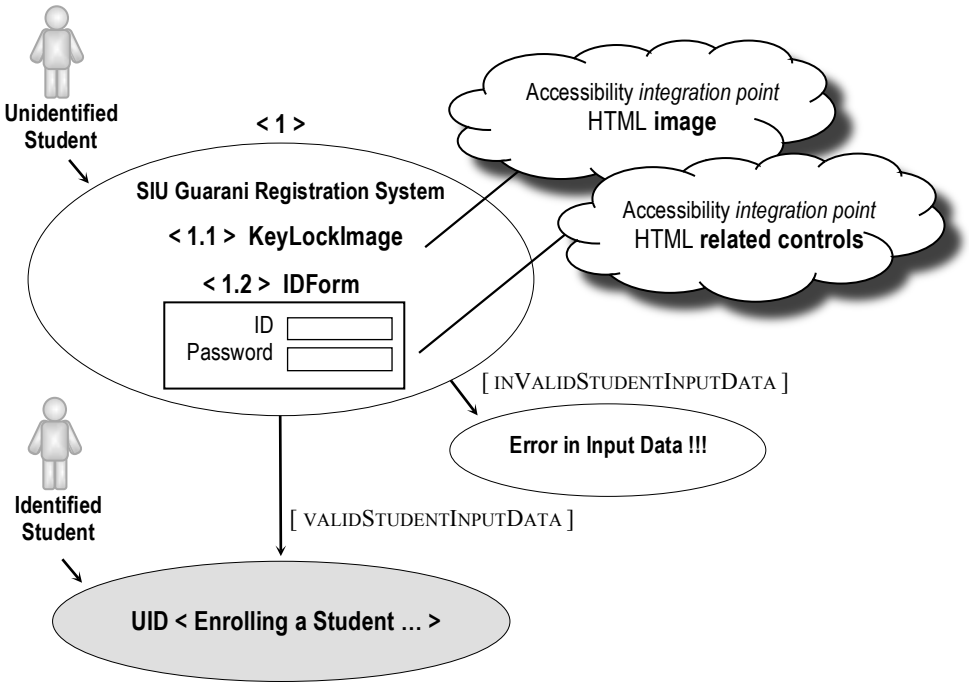

Figure 4.2: UID with Accessibility integration points: Login a Student given the Student's ID and Password

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.

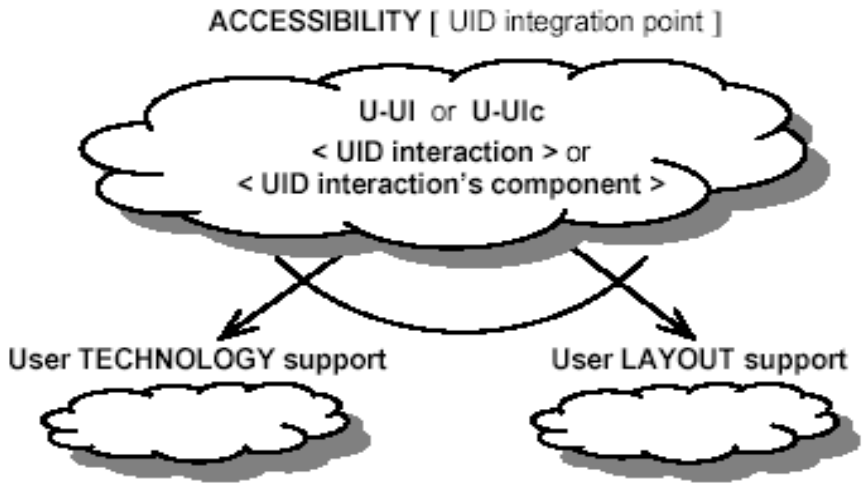

Figure 4.3: SIG Template for Accessibility

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 content[41]. 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 9.5 [3]	10.2 [2]	10.4 [3]	12.3 [2]	12.4 [2]	
				D-P 🛠	P	P	D-P	D-P	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 interim[42] solutions", satisfacing 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.

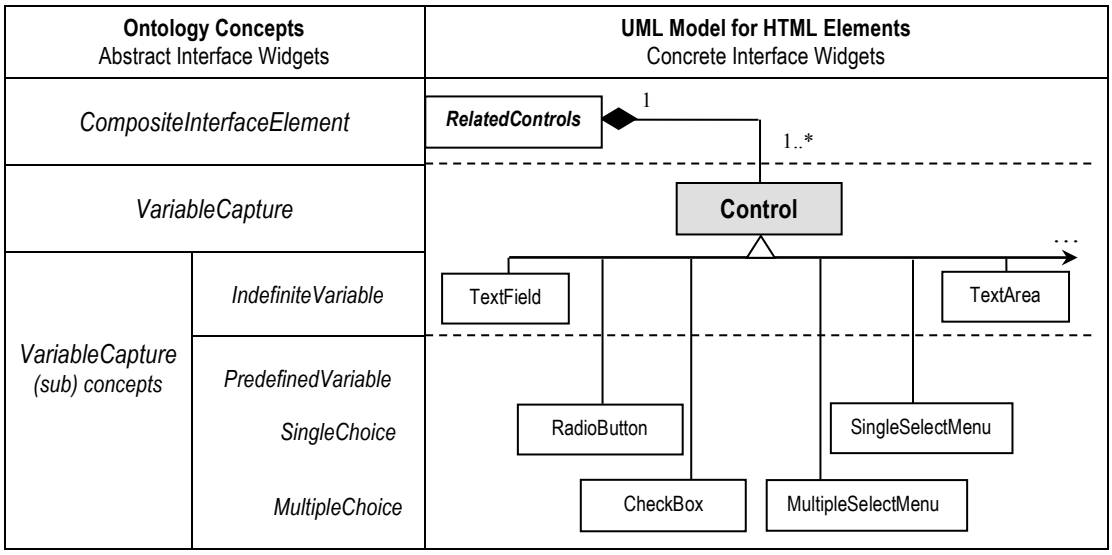

Figure 4.4: Mapping between some Ontology Concepts and HTML Elements

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.

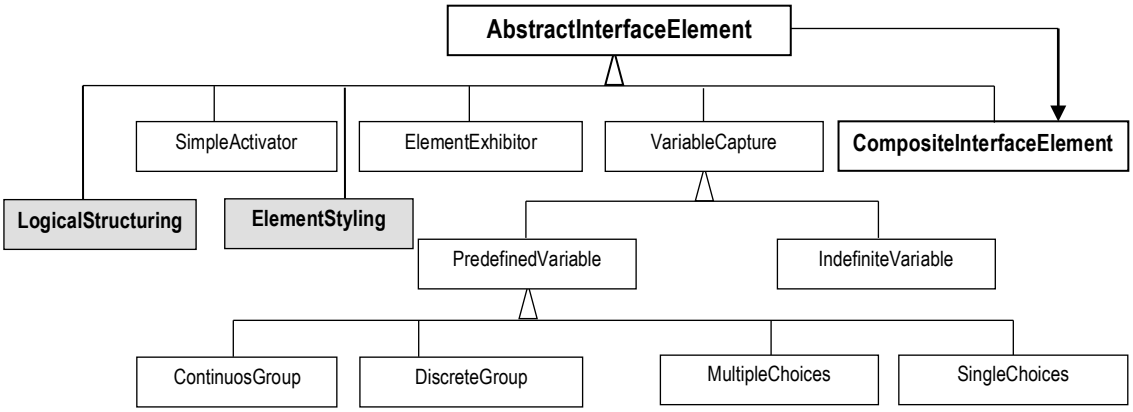

Figure 4.5: Extended Abstract Widget Ontology

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).

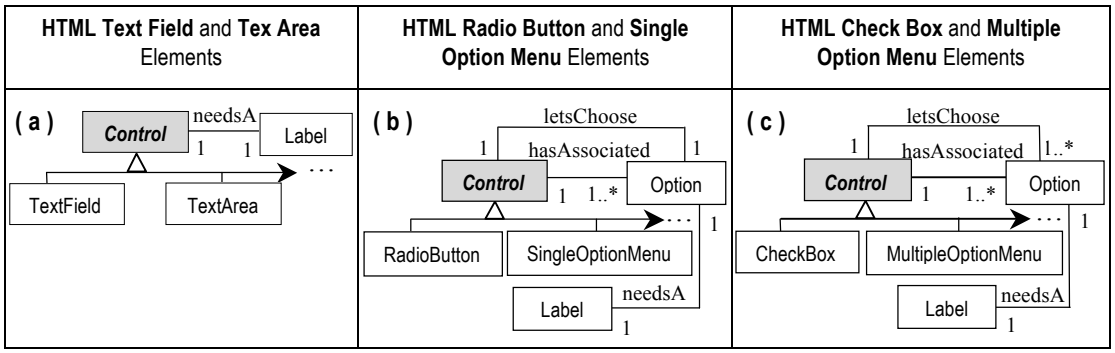

Figure 4.6: UML Model for HTML Control Elements

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 "TSControl" 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]	1.5 [3]	9.1 [1]	9.4 9.5 [3]	10.5 [3]	13.4 [2]	13.5 [3]	13.6 [3]	1.1 [1]	13.1 [2]	
				D-P	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]	1.4 [1]	6.4 [2]	7.1 [1]	7.2 [2]	7.3 [2]	7.4 7.5 [2]	8.1 [1] [2]	9.2 [2]	9.3 [2]	10.1 [2]	13.10 [3]	14.2 [3]	1.1 [1]	2.1 [1]	2.2 [2] [3]	4.1 [1]	4.2 [3]	6.3 [1]	14.1 [1]	
				D-P 🛠	D-P 🛠	P 🛠	D-P 🛠	D-P 🛠	D-P 🛠	D-P 🛠	D-P 🛠	D 🛠	P 🛠	D-P	D-P 🛠	D-P 🛠	P	P	P	P	P	D-P	P	DIALOG (D) PRESENTATION (P) PRAGMATIC 🛠
I. TSTEXT&NONTEXT SIG'S USER TECHNOLOGY SUPPORT BRANCH	ELEMENTEXHIBITOR	TEXT	ALT, TITTLE, 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, TITTLE, LONGDESC…															✓	✓ [3]	✓	✓		✓	
		NON-TEXT, MULTIMEDIA & PROGRAMMATIC OBJECTS	IMAGE, IMG, SRC…														✓	✓	✓ [2]					
			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 [3]	13.2 [2]	13.3 [2]	3.1 [2]	3.2 [2]	3.5 [2]	3.6 [2]	3.7 [2]	4.3 [3]	5.1 [1]	5.2 [1]	5.3 [2]	5.4 [2]	5.5 [3]	5.6 [3]	13.7 [3]	13.8 [3]	13.9 [3]	
				P	D-P 🛠	D-P 🛠	P	P	P	P	P	P	P	P	P	P	P	P	D-P	D-P	D-P	DIALOG (D) PRESENTATION (P) PRAGMATIC 🛠
I. TSSTRUCTURAL SIG'S USER TECHNOLOGY SUPPORT BRANCH	LOGICALSTRUCTURING	VALIDATOR & PROVIDER	HTML, HEAD, BODY, TITTLE, 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, TITTLE, 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 [2]	10.1 [2]	1.1 [1]	3.3 [2]	3.4 [2]	6.1 [1]	6.2 [1]	12.1 [1]	12.2 [2]	14.3 [3]	
				D-P 🛠	D-P	P	P	P	P	D-P	D-P	D-P	P	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…			✓		✓		✓	✓	✓	✓	