Accessibility: A Web Engineering Approach

Peter Plessers[1], Sven Casteleyn[1], Yeliz Yesilada[2], Olga De Troyer[1], Robert Stevens[2], Simon Harper[2], and Carole Goble[2]

[1]Vrije Universiteit Brussel
Pleinlaan 2
1050 Elsene – Brussel
Belgium
+32 2 629 37 54
{Peter.Plessers | Sven.Casteleyn | Olga.DeTroyer}@vub.ac.be

[2]School of Computer Science
The University of Manchester
Oxford Road, Manchester
UK M13 9PL
+44 161 275 6160.
{yesilady | stevensr | harpers | cgoble} @cs.man.ac.uk

ABSTRACT

Currently, the vast majority of web sites do not support accessibility for visually impaired users. Usually, these users have to rely on screen readers: applications that sequentially read the content of a web page in audio. Unfortunately, screen readers are not able to detect the meaning of the different page objects, and thus the implicit semantic knowledge conveyed in the presentation of the page is lost. One approach described in literature to tackle this problem, is the Dante approach, which allows semantic annotation of web pages to provide screen readers with extra (semantic) knowledge to better facilitate the audio presentation of a web page. Until now, such annotations were done manually, and failed for dynamic pages. In this paper, we combine the Dante approach with a web design method, WSDM, to fully automate the generation of the semantic annotation for visually impaired users. To do so, the semantic knowledge gathered during the design process is exploited, and the annotations are generated as a by-product of the design process, requiring no extra effort from the designer.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *architecture, user issues*; K.4.2 [**Computers and Society**]: Social Issues – *assistive technologies for persons with disabilities*.

General Terms

Design, Human Factors

Keywords

Accessibility, Visual Impairment, Web Engineering, Semantic Web.

1. INTRODUCTION

The majority of resources on the Web primarily focus on good visual presentation to implicitly help users navigate, understand, and interact with the content. However, problems arise when visually impaired users try to interact with such resources because the implicit visual cues presented cannot be accessed and used.

Visually impaired users usually access the Web by using screen readers (e.g., Jaws[1]). These applications render pages in audio and allow either simple line based interaction or full reading of a page, top to bottom; one word at a time. However, these screen readers cannot 'see' the structure of a web page or the intended meaning of the page objects. Consequently, no distinction between relevant and irrelevant information can be made (e.g., tables used for formatting versus tables containing relevant data; or images, such as bullets, used purely for presentation versus images conveying relevant schematics). Moreover, the visual cues used by (sighted) users to identify the semantics of the page objects are lost. For example, due to the visual organization a sighted user can (easily) detect a menu structure; a screen reader can only detect a collection of links, thereby losing the meaning of the concept menu. Even though some screen readers do access the HTML source code, they still cannot discover the (implicitly present) semantics of the different page objects. Most of the time HTML source code represents the visual presentation of the page rather than its logical structure.

Screen readers work satisfactorily as long as the pages are laid out linearly. Unfortunately this is not often the case, although there are guidelines to promote accessibility[2], these guidelines do not address the importance of navigation support [1].

The Dante approach is proposed to overcome these limitations. Dante analyses Web pages to extract visual objects that support navigation [23]. The identified objects are then annotated with terms from an ontology, the Web Authoring for Accessibility (WAfA)[3] ontology, in order to make their roles explicit. This markup is then used to transcode pages into a form that is easy to travel. The WAfA ontology aims to encapsulate extensive knowledge to make structural and navigational information about Web pages explicit. Fundamentally, the WAfA ontology defines concepts concerning how objects on a page are presented (their structural properties) and used (the role they fulfil) to complete a successful journey. This ontology is used to annotate Web pages manually. These annotations make the implicit knowledge about the role the page objects fulfil explicit, and allow screen readers to improve their reading capabilities significantly [22].

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

[1] See Jaws, http://www.freedomscientific.com/.

[2] See WCAG 1.0, http://www.w3.org/TR/WAI-WEBCONTENT/.

[3] Formerly known as the Travel Ontology [22].

One of the major drawbacks of this approach is the fact that the annotations have to be specified manually. While this may be feasible for small websites, the annotation task becomes too laborious and the associated cost too high for larger websites. The manual annotations also pose problems with dynamic pages, where the amount of data shown on a web page is variable and the layout may depend on the availability of certain data. Furthermore, updates to the website, both in structure, content or layout, possibly invalidate the annotations. Maintainability of the manual annotation approach is thus problematic. Taking this into account, together with the relatively short life span of a website, companies in general will consider the cost/benefit ratio of manual annotation too meager.

Despite these problems, we are convinced that Dante is a very useful approach for providing accessibility for visually impaired users. Accessibility for visually impaired users is a major issue, particularly for official (governmental) websites, who cannot deny any citizen access to official information and services. In this paper, we present a web engineering-based approach to support web accessibility, where the WAfA annotations needed to assist visually impaired users are generated automatically (along with the actual implementation of the website). The WAfA annotations are generated from the design specifications collected during the design process. It is important to highlight that this does not require any additional effort from the designer. The approach we present here is based on the web design method WSDM [8, 9, 18]. The current generation phase of WSDM has been adapted to support the generation of the WAfA annotations. The result is that the modeling concepts already used in WSDM are sufficient to generate the annotations for 70% of the concepts defined in the WAfA ontology. This can be easily increased to 84% by slightly extending the current generation process. In this paper we describe how the modeling concepts of WSDM can be used to generate WAfA annotations in the actual implementation.

The proposed approach solves all disadvantages of the manual annotations done in the Dante approach. Since the generation of the annotations is completely automatic, there is no additional cost. Furthermore, the maintenance problem is solved since changes to a website are realised through a new iteration of the web engineering process which results in the generation of a new website implementation including the annotations. Finally, typical problems with manual annotations for dynamic pages are eliminated as our approach does not make a distinction between static or dynamic websites.

The paper is structured as follows. **Section 2** summarises the Dante approach, and **Section 3** presents an overview of WSDM. In **Section 4**, the mapping between the WAfA concepts and WSDM concepts is described. **Section 5** presents an overview of the implementation architecture along with an example. **Section 6** highlights the advantages of our approach and discusses the main outcomes. **Section 7** describes and discusses some related works. Finally, **Section 8** provides some conclusions.

2. DANTE APPROACH

The Dante approach proposes a semi-automated tool that aims to improve the navigation and movement support of Web pages for visually impaired users. The main goal of Dante is to (i) analyse Web pages to identify objects that support navigation and movement; (ii) discover their roles; (iii) annotate them with concepts from an ontology, WAfA ontology, to make their roles explicit; and (iv) transcode[4] pages by using these annotations so that they can be easily accessed by using screen readers [10]. Dante uses a pipeline approach that is driven by the WAfA ontology. The extensive knowledge encoded in the ontology enables Dante to automatically transcode a Web page in a way that the visual objects on that page can play their intended roles in an audio presentation. The following sections first introduce the WAfA ontology and then explain the two main components of Dante: the annotation and transcoding components.

2.1 The WAfA Ontology

This ontology represents concepts and relationships that are necessary for the modeling of the structural and navigational organization of Web pages in a computable form. In Dante, this ontology is used as a controlled vocabulary to drive the transcodings. The WAfA ontology has three components[5]:

- *Taxonomy:* taxonomy of visual components that constitute a Web page, e.g., Sidebar *is-a* Chunk, Header *is-a* Chunk.
- *Structural abstraction:* the relationships between components and their connectivity, e.g., Logo *part-of* Header.
- *Meta-knowledge:* principles and sets of rules, e.g., Disjoint (Header, Footer) which means an object cannot be a header and a footer at the same time.

The WAfA ontology consists of a number of sub-ontologies. The two main sub-ontologies are:

1. **Authoring concepts:** encapsulate information about how the objects are *structured* in a Web page to form the overall structure of that page. These concepts originate from traditional hypermedia [16], previous work on transcoding [2], mark-up languages[6], content management systems, etc. The four higher-level concepts in this part of the ontology are: Atom (e.g., Headline, Caption), Chunk (e.g, Header, Footer, Sidebar), Node (represents a page) and Collection (represents a site).
2. **Mobility concepts:** focus on how the objects are *used* to navigate within and between Web pages. Depending on the context of the navigation task, objects can have a specific role (e.g, Cue, Obstacle) and depending on the environment, objects can also play different roles (e.g., WayPoint, ReferencePoint, DecisionPoint).

The combination of these two sub-ontologies provides extensive knowledge about pages that can be used to transcode them into a more accessible form.

2.2 Dante: Annotating Web pages

In the current prototype of Dante, pages are manually annotated with the COHSE[7] annotator[8] by using the authoring concepts described above. After annotating pages with authoring concepts, a set of mapping rules along with the underlying HTML source

[4] See http://www.w3.org/1999/07/NOTE-annot-19990710/.

[5] See http://augmented.man.ac.uk/ontologies/wafa.owl.

[6] See http://www.docbook.org/.

[7] See http://cohse.man.ac.uk/.

[8] For available tools, see http://annotation.semanticweb.org/tools.

code and the ontology are used to map annotated authoring concepts to mobility concepts. Therefore, the extended DOM in Dante includes both authoring and mobility concepts that provide enough knowledge to transcode pages into a more accessible form.

The COHSE annotator is implemented as a Mozilla plug-in so that an ontology can easily be loaded and used for annotating pages. The COHSE annotator uses XPointer[9] expressions to identify a region of a document and annotations are held in an external store. Although this provides us the flexibility to annotate third party pages, it is a long and expensive process. Moreover, it is not an easy task to capture the intention of a designer. Therefore, the annotations cannot be as good as the ones that could be done by the designer of the pages or that could be generated using a design methodology. These third party annotations can only be best-practices.

2.3 Dante: Transcoding Web Pages

Since the currently available screen readers cannot directly access these kinds of annotations and are bound to the HTML source code, Dante transcodes pages into a form so that screen readers can easily and properly render them in audio. The transcoding techniques focus on enhancing navigation support and presenting identified objects in a way that they can fulfil their intended roles. Some of the transcoding techniques are summarized as follows:

- *Physical fragmentation of pages:* Dante uses objects annotated as "Chunk" to break a complex page into a number of simpler and smaller pages and creates a table of contents that provides links to those smaller pages. In this way, users can easily get an overview of a page and locate the information that they really want or need to read. Therefore, they do not need to read an entire, long and complex page to find what they are looking for. This technique turns a complex page into a number of simpler and more manageable pages.
- *Enhancing intra-page movement:* a skip link is a popular accessibility feature. It is a link provided at the top of the page to move to the main content or to skip repeated objects in a page. Therefore Dante adds a skip link either to move to the object annotated as a "Headline" or to skip "Header" or "Sidebar" objects.
- *Eliminating repetitions:* Dante removes objects that are annotated as "Header" and "Footer" from the page so that users do not need to read them every time they access the page. This technique provides a short and concise version of the page. Dante also removes objects annotated as "Advertisement", so that the users are not distracted by these objects.
- *Alternative Views:* Dante also provides different views of a page. For example, Dante uses "LinkMenu" annotations to present only link menus in the page. This enables visually impaired users to scan pages as sighted people do[10]. Similarly, only "Sidebar" objects can be presented.

These are some of the transcodings that Dante performs by using the WAfA annotations[11]. These techniques can also be supported by screen readers but unfortunately existing screen readers do not have access to such kinds of annotations. In the meantime, tools like Dante can play an intermediary role to enhance the accessibility of Web pages for visually impaired users.

The annotation approach in Dante is different from existing semantic annotation techniques as it annotates the role and structure of the resources rather than their meaning by using an ontology [22]. Therefore, it provides better support for visually impaired users. Until now, the Dante annotations were done manually, yet in this paper we will explain a technique to generate the annotations automatically by WSDM.

3. WSDM

WSDM is a web site design method. It allows web sites and web applications to be developed in a systematic way. Other web site design methods exist in literature. The most representative ones are WebML [7], OOHDM [19], Hera [10] and OOH [11].

Figure 1 shows an overview of the different phases of WSDM. In this section, we briefly explain each phase by focusing on the phases relevant to this paper: the implementation design and the actual implementation. A detailed overview of all the phases of WSDM can be found in [9].

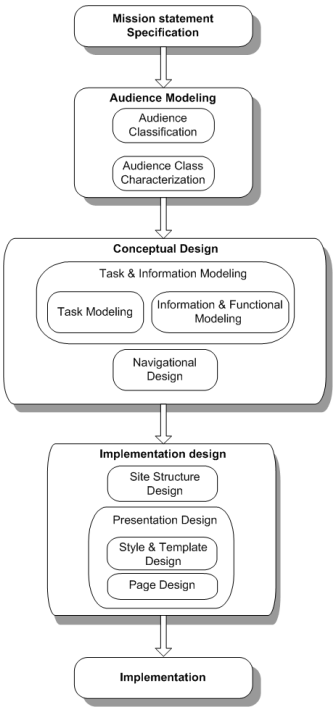

Figure 1 - WSDM overview

In the first phase of WSDM, the **mission statement** of the website, the purpose, subject and intended users are specified. By doing so, WSDM ensures that the designer clearly establishes the borders of the design.

During the **audience modeling phase**, the users identified in the mission statement are taken as a starting point and classified into different audience classes based on their information and

[9] See http://www.w3.org/TR/xptr/.

[10] See http://www.useit.com/alertbox/9710a.html.

[11] For more examples, see [20].

functional requirements. The audience classes form a hierarchy. The root of the hierarchy is the *visitor* audience class. The requirements *all* users share are associated with this class. Each audience sub-class represents a group of users that has additional requirements. During Audience Class Characterisation, the characteristics of each audience class are specified.

The **conceptual design phase** consists of two subphases: task & information modeling and navigational design. During task & information modeling, the user requirements, which were informally specified in the previous phase, are elaborated. For each requirement, a task model is specified using hierarchical task decomposition. WSDM uses the Concurrent Task Tree technique [17] for its task modeling because it allows not only to hierarchically decompose a task in subtasks but also to specify temporal relationships between different subtasks. Next, for each elementary task, an object chunk is created. Such a object chunk formally describes the data or functionality needed by a particular elementary task.

During navigational design, the conceptual structure of the website is built. A navigational model consists of nodes, grouping information and/or functionality (elementary tasks) that logically belong together. Links between those nodes specify navigation paths. WSDM provides a systematic method for obtaining the navigation structure of a website: first, the audience class hierarchy is one-to-one mapped onto a navigational model, creating a navigation track for each audience class. Next, each navigation track is further refined using the information in the task models. Nodes are created for elementary tasks; navigation links are used to reflect the temporal relationships between tasks and the relevant object chunks are connected to the nodes.

The goal of the **implementation design** is to provide, for the conceptual design, the necessary implementation aspects. The implementation design consists of three subphases, which are discussed in detail below.

The subject of the first subphase, the *Site Structure Design*, is to decide how the nodes, defined in the navigation model, will be grouped into pages. Note that in this phase we define *abstract* pages; each abstract page possibly gives rise to multiple concrete page instances in the actual implementation.

The goal of the next sub phase, the *Presentation Design*, is to describe the layout (i.e., positioning & style) for all pages. WSDM provides three different sets of modeling concepts to describe the layout of a page. Each set provides a different level of abstraction.

- *Primitive presentation concepts:* a basic set of building blocks needed to layout a page. There are two kind of such primitives: *positioning elements* and *multimedia elements* (primitive data types: Audio, Email, Image, Integer, String and Video).
- *High-level presentation concepts:* express more high-level presentation concepts like lists, menus and sections, logos, banners, breadcrumbs, etc. These concepts are defined on top of the primitive presentation concepts.
- *Template concepts:* are defined on top of the primitive presentation concepts and meant to define templates. WSDM supports the most commonly found types of templates. A template is defined as the combination of the following elements: footer, header, right- and left-sidebar, content-pane and editable region.

During the *Template & Style design*, the designer specifies templates that will be used for different types of pages. Templates define the general layout of a page. To specify the website style (fonts, colours, graphics, etc), WSDM currently uses Style Sheets.

In the *Page Design* sub phase the designer describes where the information on a page (defined in the object chunks) should be positioned and how it should be laid out. It is also decided how and where the links (defined in the navigational design) should be presented. The template defined for the page is taken as a starting point.

In the final phase, the actual **implementation**, all the information collected so far is taken as input and a website in the chosen implementation environment is automatically generated.

4. INTEGRATION OF DANTE ANNOTATION IN WSDM

Having explained the Dante approach and WSDM, we are now ready to clarify how the Dante annotation process can be integrated in WSDM.

Our main goal is to generate the annotations for visually impaired users automatically from the explicit conceptual knowledge existing during the design process. Such conceptual knowledge is captured during the design process by means of WSDM's modeling concepts. All WSDM modelling concepts used in the different phases are described in an (OWL[12]) ontology. This ontology is called the WSDM ontology[13]. To be able to generate code that is annotated with concepts from the WAfA ontology, we need to establish a relationship between the modeling concepts of WSDM and the concepts in the WAfA. We need to establish a link between the concepts in the WSDMOntology and those in the WAfA ontology. Therefore, we have defined a set of rules that map the WSDMOntology concepts onto the WAfAOntology concepts. Using these mapping rules a transformation process can be established that takes the conceptual design models as inputs and in consequence generates a set of annotations. Here we describe the mapping rules as well as the transformation process that uses them.

The transformation process consists of two steps (see T6 and T7 in Figure 2). Their roles can be described as follows:

- **T6: Authoring Annotation Transformation.** In this transformation, only the annotations for the authoring concepts of the WAfA ontology are generated. This transformation takes as inputs: the models made during the conceptual design, navigational design and presentation design. Information specified in these models is used to generate the authoring annotation.
- **T7: Mobility Annotation Transformation.** This transformation takes as inputs: the output of the previous transformation (the authoring annotations) as well as the design models. A derived set of mapping rules as defined in the original Dante approach [22] is used to extend the authoring annotations generated in the previous transformation with mobility annotations.

Note that this two-step transformation resembles the original annotation process of the Dante approach. The difference is that

[12] See http://www.w3.org/2004/OWL/.

[13] See http://wise.vub.ac.be/ontologies/WSDMOntology.owl.

the authoring annotation in the Dante framework is completed manually and based on the HTML source code of the website.

Let us now consider the transformations T6 and T7 in more detail.

4.1 Authoring Annotation Transformation

In this transformation, the mapping rules between modeling concepts defined in the WSDMOntology and authoring concepts from the WAfA ontology are used. To describe these rules, we will prefix the WSDM concepts with 'wsdm' and the WAfA concepts with 'wafa' to avoid confusion. Note that both ontologies were developed independently and therefore a number of concepts exist in both ontologies with the same (or similar) names but with different meanings. An example is the concepts wsdm:ObjectChunk and wafa:Chunk. The former defines a tiny conceptual schema used during information & functional modeling, whereas the latter refers to several page objects grouped together to form a coherent unit.

Due to space limitations, we cannot present all mapping rules. Therefore, we only describe a number of representative examples, though others can be described in a similar way. To describe the mapping rules, we use the following notational convention: first, the WAfA concept is given in bold, followed by its meaning (in italic). Where needed an informal explanation of the mapping rule is given and finally a formal definition using first-order predicate logic is given.

For a website *W*, we define the set *C* as the set of all WSDM modeling concepts and the set *I* as the set of all instances of these concepts.

For some WAfA concepts, the mapping rules are straightforward one-to-one mappings:

- **wafa:Node:** *A single page in a website.* Every generated page can be annotated with this concept.

  `∀ i ∈ I: wsdm:Page`[14]`(i) → wafa:Node(i)`

- **wafa:Figure**: *An image illustrating textual material.*

  `∀ i ∈ I: wsdm:Figure(i) → wafa:Figure(i)`

- **wafa:Footer**: *An area placed at the bottom of a page. Typically contains information about the design, company, copyright note, etc.*

  `∀ i ∈ I: wsdm:Footer(i) → wafa:Footer(i)`

- **wafa:Header**: *An area placed at the top of a page. Typically contains a company logo, page title, etc.*

  `∀ i ∈ I: wsdm:Header(i) → wafa:Header(i)`

- **wafa:TableOfContent**: *A list of available sections and a link to the beginning of each section.*

  ```
  ∀ i ∈ I: wsdm:NavigationTableOfContent(i)
    → wafa:TableOfContent(i)
  ```

- **wafa:FigureCaption**: *A brief description accompanying a figure.*

  ```
  ∀ i ∈ I: wsdm:FigureCaption(i) →
    wafa:FigureCaption(i)
  ```

- **wafa:Advertisement**: *A graphical advertisement element.*

  ```
  ∀ i ∈ I: wsdm:Advertisement(i) →
    wafa:Advertisement(i)
  ```

For a number of WAfA concepts there is no direct corresponding concept in the WSDMOntology, but nevertheless a mapping rule can be defined:

- **wafa:Chunk:** *Several page objects grouped together to form a coherent unit.* In WSDM, an object chunk represents (a unit of) information that describes a single requirement. The wsdm:Grid representing a wsdm:ObjectChunk can be annotated as a wafa:Chunk.[15]

  ```
  ∀ i ∈ I, ∃ x ∈ I: wsdm:Grid(i) ^
    wsdm:representsChunk(i, x) ^
    wsdm:ObjectChunk(x) → wafa:Chunk(i)
  ```

- **wafa:Homepage:** *A single page that represents the homepage of a website.* The annotation can be created for a generated page that is the result of a wsdm:Page that contains an instance of wsdm:Root. The wsdm:Root concept is used in WSDM to indicate the starting point of the navigation structure of a website.

  ```
  ∀ i ∈ I, ∃ x ∈ I: wsdm:Page(i) ^
    wsdm:hasNode(i, x) ^ wsdm:Root(x) →
    wafa:Homepage(i)
  ```

- **wafa:RunningFooter:** *A footer that is present on every page or on a collection of pages.* The annotation can automatically be derived when there is an instance of a wsdm:Footer concept defined in a template. This means that the footer will be placed on every wsdm:Page based on this template.

  ```
  ∀ i ∈ I, ∃ x ∈ I: wsdm:Footer(i) ^
    wsdm:IndependentTemplateConcept(x) ^
    wsdm:hasFooter(x, i) →
    wafa:RunningFooter(i)
  ```

- **wafa:NavigationalList:** *A list of links.* The annotation can be generated for a wsdm:List where all list elements have a wsdm:Link defined upon them.

  ```
  ∀ i ∈ I, ∃ y ∈ I: wsdm:List(i) ^ (∀ x ∈
    I: wsdm:hasChild(a, x) ^
    wsdm:ListItem(x) ^
    wsdm:hasNavigationReference(x, y) ^
      wsdm:NavigationReference(y)) →
    wafa:NavigationalList(i)
  ```

- **wafa:WebDirectory:** *A list that provides links to different websites in a well structured way.* In WSDM, this is represented by a wsdm:List where all list elements (e.g., childs) have a wsdm:Link defined upon them and the target of these wsdm:Links is a wsdm:ExternalNode. The wsdm:ExternalNode concept is used in WSDM to refer to web pages outside the website under design.

  ```
  ∀ i ∈ I, ∃ x, y, z ∈ I: wsdm:List(i) ^
    wsdm:hasChild(i, x) ^ wsdm:ListItem(x) ^
    wsdm:hasNavigationReference(x, y) ^
    wsdm:NavigationReference(y) ^
  ```

[14] The wsdm:Page concept refers to the abstract pages defined during the Site Structure Design (see Section 3).

[15] Note that this rule only covers the (general) wafa:Chunk concept; more specialized subtypes are treated by specialized mapping rules (e.g. wafa:RunningFooter, wafa:WebDirectory, wafa:TitleBanner, etc).

```
wsdm:toNode(y, z) ^ wsdm:ExternalNode(z)
→ wafa:WebDirectory(i)
```

- **wafa:LinkMenu & wafa:DropDownLinkMenu:** *A Linkmenu is a list meant to represent a menu. A DropDownLinkMenu is a menu that appears below an item when the user clicks on it.* Both types of menu correspond to a wsdm:Menu represented as a wsdm:List in WSDM. The difference between the two types of menus can be derived from the behaviour defined for the nodes of a wsdm:List. If no additional behaviour is defined, the menu can be annotated with LinkMenu. If an instance of wsdm:Behaviour is defined with wsdm:Event 'onClick' and wsdm:Action 'dropDown', then the menu can be annotated with DropDownLinkMenu .

```
∀ i, x ∈ I, ∃ y ∈ I: wsdm:Menu(i) ^
  wsdm:representedBy(i, y) ^ wsdm:List(y)
  ^ ¬wsdm:hasBehavior(y, x) →
  wafa:LinkMenu(i)

∀ i ∈ I, ∃ x, y ∈ I: wsdm:Menu(i) ^
  wsdm:representedBy(i, x) ^
  wsdm:List(x)  ^ wsdm:hasBehavior(x, y) ^
  wsdm:Behavior(y) ^
  wsdm:onEvent(y, 'onClick') ^
  wsdm:doAction(y, 'dropDown') →
  wafa:DropDownLinkMenu(i)
```

- **wafa:AdvertisementBanner:** *A banner that is used to advertise a website, product or service.* This annotation can be generated for an instance of wsdm:Banner containing an instance of wsdm:Advertisement.

```
∀ i ∈ I, ∃ x ∈ I: wsdm:Banner(i) ^
  wsdm:Advertisement(x) ^ wsdm:hasChild(i,
  x) → wafa:AdvertisementBanner(i)
```

- **wafa:SectionHeading:** *The heading of a particular section.* The annotation can be generated for the object (string or image) that represents the title of a wsdm:Section.

```
∀ i ∈ I, ∃ x ∈ I: (wsdm:String(i) ∨
  wsdm:Image(i)) ^ wsdm:Section(x) ^
  wsdm:hasTitle(x, i) →
  wafa:SectionHeading(i)
```

- **wafa:Link:** *A link allows the user to navigate from one page object to another web page or page object.* In the WSDM method, a link (defined during navigation design) can be placed on any page object. Therefore, an annotation for wafa:Link can be generated for every WSDM instance that has a reference to a link defined.

```
∀ i ∈ I, ∃ x ∈ I:
  wsdm:hasNavigationReference(i, x) ^
  NavigationReference(x) → wafa:Link(i)
```

- **wafa:referentialLink:** *It provides a link between, on one hand, an item of information and, on the other hand, an elaboration or explanation of that same information [16]. For example a page on a news website showing a summary of the latest news articles. A user can follow a link (e.g., 'read more') to show the complete article.* In WSDM, a wsdm:Link is always defined between two nodes (wsdm:Node) where a wsdm:Node contains one or more wsdm:ObjectChunks. A wsdm:Link can be considered a wafa:ReferentialLink when the information represented by the object chunks of the source node is a subset of the information represented by the object chunks of the target node.

```
∀ i ∈ I, ∃ x, y ∈ I: wsdm:Link(i) ^
  wsdm:hasSource(i, x) ^
  wsdm:hasTarget(i, y) ^ wsdm:Node(x) ^
  wsdm:Node(y) ^
  ∀ u, v ∈ I: wsdm:hasChunk(x, u) ^
    wsdm:hasChunk(y, v) ^

(∀ a, b ∈ C: (wsdm:isComposedOf(u, a) →
  wsdm:isComposedOf(v, b)) ^
  ∃ c ∈ C: wsdm:isComposedOf(v, c) ^
  ¬wsdm:isComposedOf(u, c))
        →

  wafa:referentialLink
```

- **wafa:TitleBanner:** *A banner on a page showing the page title.* This is the same as a wsdm:Banner containing a string with the same value as the title specified in wsdm:Page.

```
∀ i ∈ I, ∃ x, y ∈ I: wsdm:Banner(i) ^
  wsdm:hasChild(i, x) ^ wsdm:String(x) ^
  wsdm:Page(y) ^ isAncestorOf(y, i) ^
  wsdm:hasName(y, x) → wafa:TitleBanner(i)
```

- **wafa:Symbol:** *A set of special characters used to separate one element from the others (e.g., in a menu structure).* Several subtypes of Symbol have been defined in the WAfA ontology to represent a comma, dash, triangle, vertical bar and copyright symbol. We generate the annotation for the Symbol concept when it concerns a wsdm:Separator that is represented as a string. The different subtypes of Symbol can be derived based on the content of the string.

```
∀ i ∈ I, ∃ x, y ∈ I:
  wsdm:CustomSeparator(i) ^
  wsdm:representedBy(i, x) ^
  wsdm:String(x) → wafa:Symbol(i)
```

4.2 Mobility Annotation Transformation

The Dante approach provides a set of mapping rules to extend the authoring annotations with mobility concept annotations. This is done using the existing authoring annotations and analysing the underlying HTML source code. A disadvantage of this technique is that it is implementation dependent.

As the authoring annotations generated automatically by WSDM, are fully compatible with the manual annotations, we could simply use the Dante algorithm to generate the mobility concept annotations. However, as we have the design models available in our approach, we consider it useful to define our own transformation for generating the mobility concept annotation in order to provide implementation independence (i.e., not dependent on HTML as the original Dante approach). Nevertheless, we can re-use the mapping rules provided by Dante, adjusting them to interact with the design models instead of the HTML code.

Consider for example the following mapping rule for mobility annotation taken from [22]:

```
NavigationalList
        ↓
NavigationalList → DecisionPoint ^
        NavigationPoint
TextLink → NavigationPoint ^ TravelMemory
```

```
NavigationalList ^ TextLink → DecisionPoint ^
       NavigationPoint ^ TravelMemory

                    ↓

DecisionPoint ^ NavigationPoint ^ TravelMemory
```

This rule applies to the objects annotated as a "NavigationalList" where all the links in the list are text. This rule is rewritten for our approach as follows:

```
∀ i ∈ I: wsdm:String(i) ∨
  (∃ x, y ∈ C:
    wsdm:ObjectChunkReference(i) ^
    toProperty(i, x) ^ range(x, y) ^
    wsdm:String(y))
  → Text(i)

∀ i ∈ I: wafa:NavigationalList(i) ^
  (∀ x ∈ I, ∃ y ∈ I:
   (wsdm:hasChild(a, x) ^
    wsdm:ListItem(x) ^
    wsdm:hasChild(x, y) ^ Text(y))
  → wafa:DecisionPoint(i) ^
  wafa:NavigationPoint(i) ^
  wafa:TravelMemory(i)
```

The bottom-half of the rule is a direct translation of the original rule, applying 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.

Note that the new rule may seem more complex than the original one, but in fact it is not. However, the original rule does not explicitly specify how the HTML source code is referenced and how properties are extracted from it, while we also formally define the link with the WSDM design models and its content. In this particular example, we formally define when an instance is considered to be text; the original rule doesn't explicitly specify this.

5. ARCHITECTURE & IMPLEMENTATION

To validate the approach, a prototype has been implemented. To generate the actual implementation of a website, a transformation pipeline is used. This pipeline takes the object chunks, navigational model, site structure design, template design and page design as inputs. It consists of five steps:

- *High Level Transformation Mapping (T1): is* responsible for the transformation of the high level template and presentation concepts into the basic set of primitive presentation concepts. This transformation step makes the subsequent transformations simpler.
- *Model Integration (T2):* integrates the navigation, page structure, template & style and page design into one single model. In principle, this transformation can be omitted, but it (again) simplifies the following transformations.
- *Implementation Mapping (T3):* the implementation platform is chosen (e.g., HTML, XHTML[16], WML[17], etc.), and the integrated model derived by the previous transformations is partially transformed towards the chosen platform. References to data (i.e., references to object chunks) are not yet processed. Note that generating a different implementation only requires substituting this transformation (e.g., the other transformations are implementation independent[18]).
- *Data Source Mapping (T4):* the references to attributes in the object chunks are resolved and mapped to their data source. This results into executable queries using the appropriated querying formalism. This mapping can be performed fully automatically, provided that the mapping from object chunks to the data source is available.
- *Query Execution (T5):* finally, the queries are executed and the actual pages can be generated by inserting the actual data. When the query execution phase is performed offline, a static website is created but when it is performed at runtime, a dynamic site is the result.

Figure 2 schematically shows this transformation pipeline, along with the transformations that were added to support semantic annotation for the Dante approach (explained in Section 4).

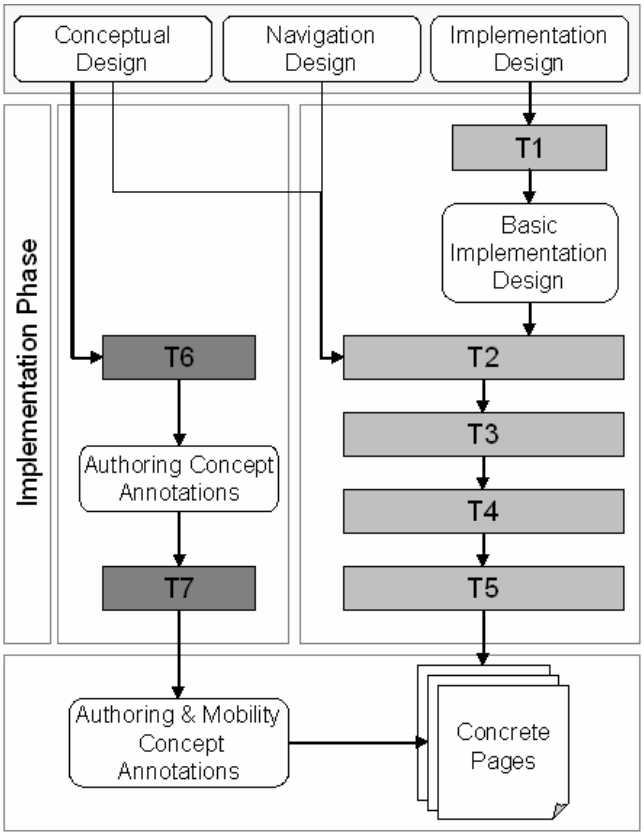

Figure 2 - Transformation Pipeline

The current prototype is based completely on Semantic Web technologies. The WSDMOntology, specified in OWL, is used to create the different design models of WSDM as OWL instance files. As described in Section 3, the WSDM object chunks are also modeled using OWL and thus also give rise to OWL instance files. These OWL specifications are taken as an input for the

[16] See http://www.w3.org/TR/xhtml1/.

[17] See http://www.wapforum.org/.

[18] Note that the data source mapping, described in the next bullet, must, of course, be compatible with the chosen implementation platform. However, it is decoupled from the Implementation Mapping.

transformation pipeline described above. This pipeline is implemented using XSLT[19]. Noteworthy is the *data source mapping*, which takes as input the model generated by the previous transformation (the implementation mapping) and outputs a new transformation in which the queries to the (OWL) object chunk instance files are specified using XPath[20]. Executing this transformation, the OWL data sources (i.e., the object chunks) are used to populate the website with actual data and the actual implementation is output.

Currently, our prototype only supports the generation of (static) HTML pages, but it can be easily extended to support other implementation languages (e.g., XHTML, WML, etc.).

Also the annotation transformations described in Section 4 are implemented using XSLT. The output is a set of XPointer expressions, annotating the website with WAfA authoring concepts.

For a better understanding of the transformation pipeline and the semantic annotation generation for the Dante approach, consider the transformation of a wsdm:Menu, represented as a wsdm:List. To keep the example simple, we have omitted any object chunk references. This has as consequence that the *data source mapping* (T4) and *query execution* (T5) transformations are not applicable. As the example only considers an isolated wsdm:Menu, the *model integration* (T2) transformation is also not needed.

A transcript of the transformation pipeline for the example follows:

(1) Input for transformation pipeline:

```
<wsdm:Menu rdf:id="menu1">
 <wsdm:MenuItem rdf:id="menu1.menuItem1">
   <wsdm:Label>some menu item</wsdm:Label>
   <wsdm:hasNavRef rdf:resource="#nav_ref1"/>
 </wsdm:MenuItem>
 <wsdm:MenuItem id="menu1.menuItem2">
 ...
 </wsdm:MenuItem>
 <wsdm:representedBy rdf:resource="#defaultList"/>
</wsdm:Menu>
```

(2) Result after T1:

```
<wsdm:grid rdf:id="menu1">
 <wsdm:hasChild>
  <wsdm:row rdf:id="menu1.row1">
   <wsdm:hasChild>
    <wsdm:cell rdf:id="menu1.menuItem1">
     <wsdm:hasChild>
      <wsdm:String>some menu item</wsdm:String>
     </wsdm:hasChild>
     <wsdm:hasNavRef rdf:resource="#nav_ref1"/>
    </wsdm:cell>
   </wsdm:hasChild>
  </wsdm:row>
 </wsdm:hasChild>
 <wsdm:hasChild>...</wsdm:hasChild>
</wsdm:grid>
```

(3) Result after T3 (T2, T4 and T5 are omitted):

```
<table id="menu1">
 <tr id="menu1.row1">
  <td id="menu1.menuItem1">
   <a href="a link"> some menu item </a>
  </td>
 </tr>
 <tr id="menu1.row2">...</tr>
</table>
```

(4) Result from T6:

http://augmented.man.ac.uk/ontologies/wafa.owl#linkMenu
http://www.example.com/index.html
#xpointer(id("menu1"))

Note how the unique id's, originating from the WSDMOntology instances, are proliferated through the transformation pipeline and reflected in the final code. This allows us to define the annotation on a conceptual level (i.e., on the WSDMOntology instances), yet effortlessly link the annotation with the actual implementation. In our example, the table structure **(3)** carries the same ID as the high-level presentation concept wsdm:Menu **(1)**, thus denoting that the table structure actually represents a menu.

6. RESULTS AND DISCUSSION

As discussed in [22], annotating web pages with the Dante ontology greatly improves the accessibility of a website. The approach presented in this paper provides the following advantages, compared to the original Dante approach:

- *Automatic:* in contrast to the original Dante approach where annotations done by hand, the WSDM annotation process is fully automatic.
- *No extra effort:* the annotation is a side effect of the WSDM design process, and thus requires no extra effort from the designer.
- *Robust:* the content and/or layout of most web pages change regularly so in the original Dante approach, this may invalidate the performed annotations. In our approach, changes to a website can be done on the conceptual level, after which the implementation, and the annotations, are effortlessly re-generated
- *Implementation independent:* as the annotations are performed on a conceptual level and they are proliferated throughout the WSDM implementation pipeline, the chosen implementation platform (e.g., the chosen implementation mapping) does not influence the annotation.
- *Dynamic pages supported:* the original Dante approach does not deal well with dynamic pages. As the layout of a dynamic page may be dependent on whether or not certain data is available, annotations stored for that page could be invalid. As the annotation is now done in combination with the page generation, dynamic pages are correctly annotated.

An obvious disadvantage of our approach is that it cannot be applied to existing sites. For these legacy sites, the original manual annotation can be an acceptable compromise, while our approach offers a more rigid and systematic solution towards the future.

The results of our approach are very promising. In the WAfA ontology, about 80 different authoring concepts are defined. Currently, we are able to generate annotations for approximately 70% of them fully automatically without any additional effort required from the designers. This percentage could be raised to about 85% by further extending our approach. For example, we could automatically generate a sitemap for a website by taking

[19] See http://www.w3.org/TR/xslt.

[20] See http://www.w3.org/TR/xpath.

into account the navigational design. Also form elements (e.g., text box, check box, etc.) are not yet taken into account.

Annotations for about 15% of the WAfA concepts are impossible to be generated automatically without requiring an extra effort from the designer. An example of such a concept is the WAfA concept 'Abstract' defined as a summary of the main points of an argument or theory. It would require an additional effort from the designer to specify that a given data property defined in a certain object chunk represents an 'Abstract'. Table 1 gives an overview of the results.

Table 1 - Overview of results

	WAfA Concepts	Defined mapping rules	%
Current result	78	55	70,51 %
Best possible result	78	66	84,62 %

7. RELATED WORK

Semantic annotation of web pages is a major research topic within the semantic web community. Two main directions can be distinguished, i.e. manual and automatic annotation approaches. Manual annotation approaches (e.g. [13, 14]) allow users to define annotations by hand and focus on easing the annotation process. Automatic annotation approaches try to extract annotations automatically using machine learning and natural language techniques (e.g. [6, 12]). Most of these approaches focus on the annotation of the actual content of a web page, and do not specifically consider annotation support for visually impaired users.

Some annotation approaches have been developed focusing specifically on semantic annotations for visually impaired users. Their intention is to transcode pages based on these annotations to allow improved audio rendering. A simple, yet effective approach consists of in- and excluding page fragments. In the page *clipping* approach [15], a vocabulary for annotation that includes only "keep" (content should be preserved) and "remove" (content should be removed) statements is proposed. The pages are then filtered based on these annotations.

More advanced approaches propose a more elaborated vocabulary for annotation. This makes it possible to describe the structure of a page and the role page objects fulfil in more detail, and therefore enhance the transcoding possibilities. A similar approach to Dante is described in [2]. Also [20] has defined an approach based on transcoding HTML pages to VoiceXML. They have defined an own annotation language called VXPL to indicate the different structures of a web page. However, in both approaches, the proposed vocabulary is less detailed than the WAfA ontology and does not support the same deep understanding and analysis of pages as the WAfA ontology does.

As in all these approaches the transcoding of pages is based on annotations, an easy and efficient way to obtain these annotations is crucial. Most approaches consider a manual annotation approach which is a time consuming and expensive process. [21] proposes an algorithm to overcome this problem. This algorithm tries to use existing annotations for a page on other pages on that site. Although, it is a promising approach, different annotations are still required for different sites. Another attempt to automate the annotation process is the automatic analysis of HTML pages to identify coherent chunks of information where they are only accessible visually [3, 4, 5]. The approach is mainly focusing on certain types of pages with a certain set of rules. It is, however, difficult to generate rules that are generic enough to apply to different types of pages. They work well on small well-structured web pages but generate almost unusable results for complex pages.

In this paper we integrate the process of semantic annotation for visually impaired users into a website design method, thereby automating this process. To the best knowledge of the authors, the integration of the annotation process in a web engineering approach, as described in this paper, is not yet done.

8. CONCLUSIONS

In this paper, we have presented a web engineering approach to provide accessibility support for visually impaired users. We described how an existing design method, WSDM, can be extended to automatically generate semantic annotations describing support for mobility according to Dante, a stand-alone approach to support web accessibility. WSDM has a conceptual infra-structure that supports web design, instantiated in a series of OWL ontologies. In the combined approach described here, a mapping between the WSDM concepts and the Dante concepts was made and implemented. Currently, +/- 70% of the Dante concepts annotation can be generated automatically, without any extra effort from the designer. This could be increased to 85% by further extending the WSDM-Dante mapping rules and even 100% if the designer is prepared to do some extra effort.

Our annotation approach eliminates the problems of manual annotations: the annotations are generated automatically; no additional effort is required; dynamic pages are supported; and changes to page content and layout do not invalidate the annotations. The process of annotation to support accessibility becomes an implicit part of the web design process. In our future work, we plan to further extend the defined mapping rules to achieve the best possible 84,62% automatic annotation (without additional effort). Additionally, we could extend the WSDMOntology with the missing concepts to support the full WAfA ontology. However, it needs to be investigated to what degree designers are willing to accept the extra effort required.

As a conclusion, this approach is unique in the web accessibility field, as it integrates the requirements of visually impaired users from the early stage of the web design process. Unfortunately at present, the web accessibility is typically an afterthought and this is demonstrated by the number of available evaluation and repair tools for existing pages[21]. Our approach makes accessibility an integral part of web design.

9. ACKNOWLEDGEMENTS

This research is partially performed in the context of the e-VRT Advanced Media project (funded by the Flemish government) which consists of a joint collaboration between VRT, VUB, UG, and IMEC.

10. REFERENCES

[1] Access and inclusion for disabled people. Technical report, Disability Rights Commission (DRC), 2004.

[21] See http://www.w3.org/WAI/ER/existingtools.html.

[2] Asakawa, C., Takagi, H. Annotation-based transcoding for nonvisual web access. In ASSETS'00, pages 172–179, 2000.

[3] Buyukkokten, O., Garcia-Molina, H., Paepcke, A. Accordion summarization for end-game browsing on pdas and cellular phones. In CHI 2001, pages 213–220, 2001.

[4] Chen, Y., Ma, W., Zhang, H. Detecting web page structure for adaptive viewing on small form factor devices. In Proceedings of the Twelfth International World Wide Web Conference, 2003.

[5] Chen, J., Zhou, B., Shi, J., Zhang, H., Wu, Q. Function-based object towards website adaptation. In Proceedings of the Tenth International World Wide Web Conference, Hong Kong, 2001.

[6] Ciravegna, F., Dingli, A., Petrelli, D., Wilks, Y. User-System Cooperation in Document Annotation based on Information Extraction. In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 02), Spain, 2002.

[7] Ceri, S., Fraternali, P., Bongio, A. Web Modeling Language (WebML): a modeling language for designing Web sites, In Proceedings of the Ninth World Wide Web Conference, 2000.

[8] De Troyer, O., Leune, C. WSDM: A User-Centered Design Method for Web Sites, In Proceedings of the Seventh International World Wide Web Conference, pages 85–94, 1998.

[9] De Troyer, O., Casteleyn, S. Designing Localized Web Sites, In Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE2004), pages 547–560, 2004.

[10] Frasincar, F., Houben, G., Vdovjak, R. Specification Framework for Engineering Adaptive Web Applications, In Proceedings of the Eleventh International World Wide Web Conference, 2002.

[11] Gómez, J., Cachero, C., Pastor, O. Extending an Object-Oriented Conceptual Modelling Approach to Web Application Design, In Proceedings of CAiSE'2000, LNCS 1789, Pages 79–93, Stockholm, 2000.

[12] Handschuh, S., Staab, S. Annotation for the Semantic Web, OIS Press, 2003.

[13] Handschuh, S., Staab, S., Maedche, A. CREAM – Creating Relational Metadata with a Component based, Ontology Driven Framework. In Proceedings of K-Cap, Victoria, Canada, 2001.

[14] Heflin, J., Hendler, J. Searching the web with SHOE. Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01, AAAI Press, pages 35–40, 2000.

[15] Hori, M., Ono, K., Koyanagi, T., Abel, M. Annotation by transformation for the automatic generation. In Pervasive 2002, pages 267–281, 2002.

[16] Lowe, D., Hall, W. Hypermedia and the Web: An Engineering Approach. John Wiley and Sons Ltd, 1998.

[17] Paterno, F. Model-Based Design and Evaluation of Interactive Applications, eds. Ray, P., Thomas, P., Kuljis, J., Springer-Verlag Londen Berlin Heidelberg, ISBN 1-85233-155-0, 2000.

[18] Plessers, P., De Troyer, O. Annotation for the Semantic Web during Website Development, In Proceedings of the ICWE 2004 Conference, pages 349–353, 2004.

[19] Schwabe, D., Rossi, G., Barbosa, S. Systematic Hypermedia Application Design with OOHDM, In Proceedings of ACM Hypertext'96 Conference, pages 116–128, 1996.

[20] Shao, Z., Capra, R., Pérez-Quiñones, M. Transcoding HTML to VoiceXML Using Annotation, In Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03) Sacramento, California, USA, 2003.

[21] Takagi, H., Asakawa, C., Fukuda, K., Maeda, J. Site-wide annotation: Reconstructing existing pages to be accessible. In ASSETS'02, pages 81–88, 2002.

[22] Yesilada, Y., Harper, S., Goble, G., Stevens, R. Screen Readers Cannot See (Ontology Based Semantic Annotation for Visually Impaired Web Travellers). In ICWE 2004 Proceedings, pages 445–458, 2004.

[23] Yesilada, Y., Stevens, R., Goble, C. A foundation for tool based mobility support for visually impaired web users. In Proceedings of the Twelfth International World Wide Web Conference, pages 422–430, 2003.