Design Time Support for Adaptive Behavior in Web Sites

Sven Casteleyn
Vrije Universiteit Brussel,
Department of Computer Science, WISE
Pleinlaan 2
1050 Etterbeek, Belgium
+32 2 629 37 54
Sven.Casteleyn@vub.ac.be

Olga De Troyer
Vrije Universiteit Brussel,
Department of Computer Science, WISE
Pleinlaan 2
1050 Etterbeek, Belgium
+32 2 629 35 04
Olga.Detroyer@vub.ac.be

Saar Brockmans
Vrije Universiteit Brussel,
Department of Computer Science, WISE
Pleinlaan 2
1050 Etterbeek, Belgium
+32 2 629 35 04
Saar.Brockmans@vub.ac.be

ABSTRACT
Adaptive web sites are sites that automatically improve their internal organization and/or presentation by observing user-browsing behavior. In this paper we argue that adaptive behavior of websites should be controlled in order to keep the website manageable. We believe that adaptive behavior may be a useful complement to a good website design method on the condition that the adaptations are limited and according to the modeling approach followed during design. Therefore, we allow a website designer to specify at design time the adaptive behavior that will be allowed at runtime. To accomplish this goal, an Adaptation Specification Language is defined that allows designers to specify at the level of the navigational model, which adaptations could be performed at runtime. The language is event based, i.e. a collection of rules is used to specify the adaptation operations to be carried out if certain conditions are satisfied. The approach proposed is elaborated in the context of WSDM, an audience driven website design method, but is generally applicable to other design methodologies.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques; H.5.4 [**Information Systems**]: Information Interfaces and Presentations - *Hypertext/Hypermedia - Architectures, Navigation, User Issues*

General Terms
Design, Languages, Theory

Keywords
Web design method, adaptive behavior, web site adaptation, adaptation language

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC 2003, Melbourne, Florida, USA
© 2003 ACM 1-58113-624-2/03/03...$5.00

1. INTRODUCTION
The World Wide Web has become a very popular medium for sharing data: companies, organizations, governments, individuals … they all found benefits in exchanging and sharing their data online. With this explosive amount of information becoming available, two important problems with the current WWW are manifesting themselves. First of all, the vast majority of web sites become increasingly unmanageable for the web site administrator, usually due to the lack of any systematic underlying design, and secondly, web site visitors find it increasingly difficult to find the information they need in the huge, mostly poorly structured and ordered information available.

To tackle the first problem, several web site design methods have been described in the literature. The most known ones include HDM and its successors HDM2 [12] and OOHDM [21] [22], RMM [15], W3DT [1] [2], WEBML [3] [5] and WSDM [9] [4] [23]. While most of these methods differ in approach, all methods do provide some basic mechanisms for describing, ordering and structuring the data contained on the website, thus making it (partly automatically) manageable for web site administrators.

The second problem is addressed to a different extent by the different design methods, according to their philosophy. Organization driven design methods structure the information according to the availability of the data within the structure of the organization, data driven methods are lead solely by the available data, and user centered or audience driven methods are driven by the requirements of the intended users of the web site. Taking into account the different requirements for the different target audiences, and ordering the information accordingly, audience driven design methods help in offering the visitors less but more relevant information, thus effectively decreasing the time visitors have to search for the information they require. The Web Site Design Method (WSDM), which will be used in this paper as a framework to present our work, is an example of an audience driven approach.

Another attempt to make web site's information more readily available to the visitor is the use of adaptive behavior to re-arrange and re-organize the content and/or structure of the web site. Adaptive websites are characterized as sites that automatically improve their organization and presentation by

learning from visitor access patterns [17]. A typical example would be moving a link that is frequently accessed to a higher level in the page hierarchy, so that visitors are able to find that particular information quicker.

Extensive research has been done in adapting websites to facilitate an individual user or a group of users, using both content-based and access-based adaptation [19]. Until now, the main research issue has been techniques such as information retrieval, classification, page clustering and semantics matching for content-based adaptation; and data mining, association rule discovery, page clustering, … techniques for access-based adaptation [13], [18], [20], [25]. For a more extensive list of references, see e.g. [19]. In the Hypermedia community, some attempts have been made to formulate adaptation models [6], [7], often integrated within the DEXTER reference model [13]. None of these models targets websites specifically though, building on standard hypermedia systems functionality, or they are very implementation oriented.

The approach to adaptive websites taken in this paper is to complement a website design method with adaptive behavior at runtime. We believe that the use of a website design method is essential for a website to be maintainable and usable. On the other hand, it is impossible, even in an audience driven approach and especially for public websites, to accommodate all the needs of all website users at design time. Therefore it may be useful to allow for adaptive behavior during runtime to enhance the usability of the website. However, if the adaptation is done in an uncontrolled way, we may end with a website whose structure is no longer conform the philosophy of the design method used and is not manageable because the structure is no longer transparent for the designer. Therefore, we will allow the designer to specify the kinds of adaptation that may be executed at runtime, during the design of the website. We present the approach in the context of WSDM (Web Site Design Method), but it is also applicable for other website methods. WSDM is an audience driven website design method, which also makes a clear distinction between the conceptual design (free from any presentation or implementation issue) and the implementation design. Following this distinction, the specification of the adaptive behavior will be done during the conceptual design. This specification of the allowed adaptive behavior of the website is done by means of an event based specification language that allows to express changes to the structure of the website.

In this paper, we mainly focus on the adaptation for a group of users by adapting the structure of and the navigation possibilities in the website. In the literature, this is often called "transformation", as opposed to "customization" [19], where a site is adapted for an individual user (either by the user himself, or by the system using knowledge about the user) which usually involves profiling in some way (for example, ATG's CRM software). The latter, customization, is not discussed in this paper. Also, adaptation of presentation issues (page layout, fonts, etc.) and adaptation of actual content of the website is outside the scope of the paper.

The paper is organized as follows. In the second section, we will give a brief introduction to WSDM. In the third section, we give a formalization of the navigation model of WSDM to allow to define basic operations upon this model in the fourth section. We present an event-driven adaptation language in section five. Conclusions and future work can be found in section six.

2. WSDM: AN OVERVIEW

Here, we give a short overview of WSDM, more information can be found in [4], [8], [9], [10], [11], [23]. WSDM puts the emphasis on the different kind of users, and their different requirements. The first step is to define the Mission Statement. The Mission Statement should express the purpose and the subject of the web site and declare the target audience. Based on this Mission Statement a two-step Audience Modeling phase is performed. In the first step, Audience Classification, the different kinds of users are identified and classified. Members of the same Audience Class have the same information and functional requirements. In the next step, Audience Class Characterization, the characteristics of the different Audience Classes are given. The result of the Audience Modeling is a set of Audience Classes, ordered into an Audience Class Hierarchy, together with an informal description of all their requirements: information- and functional - as well as navigational- and the usability requirements, and their characteristics. For a formal definition of Audience Classes, and a method to automatically derive them, we refer to [4].

Next, we perform a Conceptual Design of the site. The Conceptual Design phase is divided in three steps: Information Modeling, Functional Modeling and Navigational Design. During Information & Functional Modeling, the information & functional requirements of the different Audience Classes are modeled. For each requirement a Chunk is created that models the information needs described in the information requirement or the functionality needed by a functional requirement. To control the redundancy that we may introduce in this way, all Chunks are linked together in a single information model, called the Business Information Model.

During Navigation Design we design the (conceptual) structure of the website and model how the members of the different Audience Classes will be able to navigate through the site. For each Audience Class a Navigation Track is created. Navigational requirements are taken into consideration in this step. All Navigation Tracks together form the Navigation Model of the site. A Navigational Model is composed of nodes. Nodes represent units of information or functionality. The information or functionality represented by a node is denoted by connecting the node to one or more chunks. Nodes themselves are connected by means of links. Links are used to model the structure of the website as well as to indicate the need for navigation. We can put conditions on links to indicate that the availability of the link is dependent on the truth-value of the condition. The main structure of the Navigation Model can be easily derived from the Audience Class Hierarchies. How this is done is explained in [10]. The integration of the Information Chunks and Functional Chunks in the Navigation Model is called the Conceptual Model of the web site. An example of a Navigational Model fragment is given in figure 1. The example is a website for a telephone company and shows the main structure of the website and the navigation structure for the Personal User Track. The other tracks are not elaborate to not overload the figure. Nodes are represented by rectangles, chunks by rounded rectangles, links by arrows and the connections between chunks and nodes by lines. A double lined

rectangle is used for the root of a track or the root of the website (in this example the node 'Visitor Track' is the root of the website)

During Implementation Design we essentially design the page structure as well as the 'look and feel' of the web site. The aim is to create a consistent, pleasing and efficient look and feel for the conceptual design made in the previous phase by taking into consideration the usability requirements and characteristics of the Audience Classes.

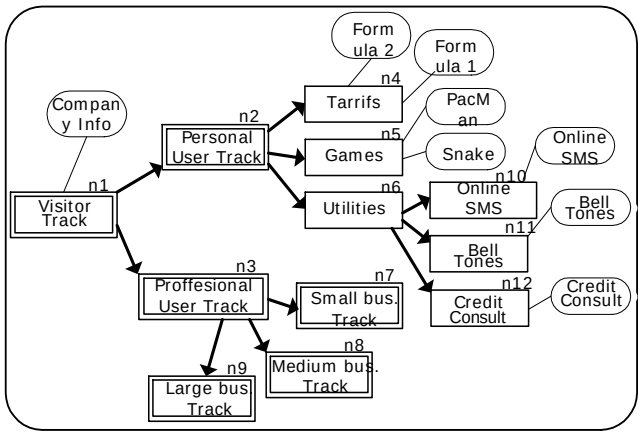

Figure 1 Simplified Navigational Model for Phone Company

The design of the page structure starts from the Navigation Model. Nodes may be grouped into or distributed among pages. This depends on the amount of information represented by a node: the amount of information on a page should not overwhelm the user, but too little information on a page may force the user to "click" too much.

The Implementation design should also provide the specification for the logical data design. This could be a logical database design, an XML DTD, RDF definitions, or any other suitable data definition format. Whatever format chosen, the data design can be derived from the Business Information Model.

The last phase, Implementation, is the actual realization of the web site using the chosen implementation environment.

3. FORMALIZATION OF THE NAVIGATIONAL MODEL

In this section, we formally define the concepts used during the Navigational Design in WSDM. During Navigational Design, a web site is modeled as a graph of nodes. Every node may group a number of chunks, which are pieces of information or functionality and defined during Information Modeling and Functional Modeling. Chunks model elementary information and functional requirements of the users. Please note that nodes do not necessarily correspond with the actual pages in a website. It is only in a later phase (during Page Design) that nodes are assigned to pages. Depending on the size of a node, different nodes can be grouped on one page or a node can be distributed among pages.

Definition

- N is a finite non-empty set of nodes
- H is a finite non-empty set of chunks
- $L \subseteq N \times N$, is the set of links
- $C \subseteq N \times H$ is the set of connections between nodes and chunks

The first node of a link l is called the source, denoted *source(l)*; the second node is called the target, denoted *target(l)*. The node of a connection c is denoted *node(c)*; the chunk is denoted *chunk(c)*.

The purpose of the Navigational Model as defined here, is to define the main structure of the website. This is done by defining nodes for the different elementary informational and functional requirements (represented by means of chunks) and by linking these nodes according to a certain structure[1] (e.g. a hierarchical structure or a linear structure).

We define a path (loop-free) between two nodes in the usual way:

Definition Let n_1 and n_2 be two nodes of a Navigational Model M= (N, H, L, C).A *path* p from n_1 to n_2 is a sequence of links l_1, ..., l_n, $\in$ L where:

- source (l_1) = n_1
- target (l_n) = n_2
- $\forall$ k where $1 < k \leq n$: target(l_{k-1}) = source(l_k)
- $\forall$ i where $1 \leq i \leq n$, $\forall$ j where $1 \leq j \leq n$, $i \neq j$: source(l_i) $\neq$ source(l_j) $\wedge$ source(l_i) $\neq n_2$

The notation *path(l_1, ..., l_n,)* is used to denote the path. If no confusion is possible also the notation *path(n_1 ,n_2)* can be used. The length of path p, denoted by *length(p)*, is n, the number of links in p. n_1 is called the top of the path p, noted *pathTop(p)*.

We call a node n_2 *reachable* from another node n_1 if there is a path from n_1 to n_2.

The set of all nodes in a path p is denoted by *pathNodes(p)*. We also define the notion of *inLink* and *outLin*k for a node in a path:

Definition Let n $\in$ pathNodes(p) where p = (l_1, ..., l_n).
*inLink(*n,p) is defined as l where l $\in$ {l_1, ..., l_n} and target(l) = n
*outLink(*n,p) is defined as l where l $\in$ {l_1, ..., l_n} and source(l) = n

We define the level of a node in a path as the number of links from the top of the path to the node:

Definition Let p be a path, n $\in$ pathNodes(p).
*level(*n,p) is defined as

0	if n = pathTop(p)
k+1	if level(source(inLink(n,p),p) = k

Due to lack of space, we do not go into the following defintions, they are defined in the usual way: *parent(*n,p) & isParent(n, n'), *grandparent(*n,p) & isGrandParent(n, n'), *sibling(*n_1, n_2) & isSibling(n_1,n_2)
Websites usually provide a single access point that will act as a direct or indirect gateway to all the information and functionality offered on the site. During the conceptual design, this access point corresponds with a node. We call this node the *root*.

[1] In WSDM this structure is defined by the structure of the Audience Class Hierarchy and is therefore a hierarchical structure.

Definition Given a Navigational Model M = (N, H, L, C), the *root*, denoted *Root(M)* is the node ($\in$ N) that was chosen to be the primary access point to the site. Every other node n from N is reachable from Root(M).

Within a website, very often we can distinguish parts that constitute a whole (also called sub sites). E.g. in WSDM, each audience class will have its own sub site (called audience track). We formally define a sub model of a Navigational Model as follows:

Definition Consider a Navigational Model M= (N, H, L, C). The Navigational Model $M_1=(N_1, H_1, L_1, C_1)$ is a (Navigational) *sub model* of M if:

$N_1 \subset N, N_1 \neq \emptyset$
$H1 \subset H, H_1 \neq \emptyset$
$L_1 \subset L,$
$C1 \subset C,$

where

$\forall l \in L_1$: source (l) $\in N_1$ and target (l) $\in N_1$ and
$\forall c \in C_1$: node (c) $\in N_1$ and chunk (c) $\in H_1$

By definition, the following lemma holds:

Lemma A Navigational sub model is a Navigational Model

We are now ready to define an Audience Track. n Audience Track can be seen as a collection of paths that a group of visitors (an Audience Class) can follow from a single access point to reach all the information and functionality they need.

Definition Given a website W with a set of Audience Classes $A=\{A_1, ..., A_n\}$ and a Navigational Model M= (N, H, L, C), an *Audience Track* for an Audience Class $A_m \in A$ is a sub model $T_{Am} = (N_m, H_m, L_m, C_m)$ of M with the following properties:

- H_m contains all and only the chunks that resulted from the information and functional requirements of A_m
- Root(T_{Am}) is defined
- $\forall h \in H_m$: $\exists n \in N_m \wedge \exists c \in C_m$: node(c) = n $\wedge$ chunk(c) = h

The last condition in this definition states that each chunk in the sub model must be connected to a node of the sub model.

We are now able to define basic operations on the navigational model. With these operations, it will be possible to manipulate the navigational model of a website. This will be used to describe the allowed adaptive behavior of the website.

4. OPERATIONS ON THE NAVIGATIONAL MODEL

To allow manipulation of the navigational model, we first define the elementary operations. There are three different kinds of elementary operations on the navigational model: operations on nodes, operations on chunk and operations on links. In the definitions of the operations that follow, we use the following conventions: M = (N, H, L, C) will be the input, and M' = (N', H', L', C') will be the output Navigational Model. If N', respectively H', L' and C', are not defined explicitly, we have N'=N, respectively H'=H, L'=L and C'=C. Based on the elementary operations we will define a number of more high level operations like promotion and demotion, linking and unlinking and clustering.

4.1 Elementary Operations

4.1.1 Operations on Nodes

We only need two operations on nodes: addNode and deleteNode. The move operation on nodes can be realized by multiple link operations.

Definition

- *deleteNode(n)*, where n $\in$ N
  N' = N \ {n}
  L' = L \ {$\forall$ l $\in$ L: source(l) = n or target(l) = n }
  C' = C \ {$\forall$ c $\in$ C: node(c) = n }
- *addNode(n*), where n $\notin$ N
  N' = N $\cup$ {n}

4.1.2 Operations on Chunks

We do not allow to add or to delete chunks in adaptive behavior. Chunks are the result of a modeling activity and therefore it makes no sense to add a new chunk during adaptive behavior. Also deleting is not allowed because if the chunk is removed it cannot anymore be used later on. However it is possible to disconnect from a certain node and connect it to another node.

Definition Let n $\in$ N and h $\in$ C:

- *connectChunk(h, n)*:
  C' = C $\cup$ {(n, h)}
- *disconnectChunk(h, n)*:
  C' = C \ {(n, h)}

4.1.3 Operations on Links

Next to connecting and disconnecting chunks to nodes, we also define operations to connect or disconnect the nodes themselves by means of links:

Definition

- *deleteLink (l)* where l $\in$ L:
  L' = L \ { l }
- *addLink (l)* where l = (n, n') and n, n' $\in$ N and l $\notin$ L:
  L' = L $\cup$ { l }

4.2 Conceptual Navigation Transformations

In this section we define a number of more high-level adaptation operations on the navigational model that we call conceptual navigational transformation. These are promotion and demotion, linking and unlinking, and clustering. We will discuss each of these in the following subsections. Other conceptual navigational transformations are also possible

4.2.1 Promotion and Demotion

Promotion makes a node easier to find by moving it closer to the root of the web site. Demotion on the contrary moves the node further away from the root. Doing promotion and demotion will usually be based on the popularity of the node, and possibly also on the access path used to find the node (a node may be reachable by means of more than one path). Usually the rule "the more popular the node, the closer to the root" is followed. Here, we

define promotion and demotion as moving the node one link closer or further away from the top of a path. Promotion is realized by adding a link to the node from a grandparent of this node, while demotion means adding a link from a sibling to the node and removing the original link to the node. Promotion is shown in figure 2, demotion in figure 3. Note that in this version of promotion, the original link is preserved.

Definition Let n ∈ N, p path where n ∈ pathNodes(p) and level(n,p) ≥ 2

- *promoteNode(n, p)* stands for
  addLink(grandParent(n,p)),n);

Definition Let n ∈ N, p path where n ∈ pathNodes(p), n_p = parent(n,p) and ∃ n' ∈ N: sibling(n, n') ∧ n_p = isParent(n_p , n'), then

- *demoteNode(n, p, n')* stands for
  addLink(n',n);
  deleteLink((n_p , n))

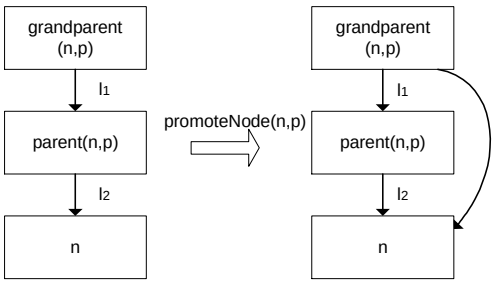

Figure 2: Promotion of a node n along a path path(l_1 , l_2)

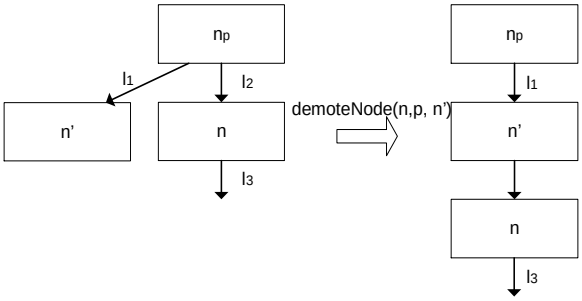

Figure 3: Demotion of a node n via a path p to a sibling n'

Linking connects two nodes that were not (or not directly) connected by adding new links between them. Even though the designer did not model a link explicitly, for some nodes it turn out at runtime that they are conceptually related in the users' minds. Similarly, unlinking is based on observing a lack of correlation; if a link between two nodes is never followed, we might infer that they are unrelated in the users' minds, even though the designer connected them for some reason. Removing them may result in a simpler and more transparent site structure. In the literature, many algorithms have been described to determine whether pages are related, based on (restricted) clustering and user access patterns within one user session. These may be useful to determine if nodes are related (node will be contained in pages in the website)

Definition Let n, n' ∈ N:

- *linkNodes(n, n')*
  addLink((n',n)); addLink((n,n'))
- *unLinkNodes(n, n')*
  deleteLink((n',n)); deleteLink((n,n'))

4.2.2 Clustering

In the literature, clustering associates a collection of related pages and makes them accessible as a group on a newly created page. The system recognizes a collection of similar documents that are not grouped together anywhere at the site, creates a new page for them and adds a reference to the new page. Documents may be considered related based on their filenames, their locations in the site hierarchy, their correlation in visitor paths, etc. A similar transformation can be defined on our Navigational Model: if two or more nodes are related, but not yet grouped we can add a node and the necessarily links to make them accessible as a group. Clustering of chunks can be done by connecting them to a single node.

Definition Let n ∈ N, n' ∉ N, h, h' ∈ H

- *clusterChunks(n, h, h')*
  addNode(n');
  connectChunk(h, n'); connectChunk(h', n');
  addLink((n, n')); addLink((n', n))

Definition Let n, n_1, n_2 ∈ N, n' ∉ N

- *clusterNodes(n, n_1, n_2)*
  addNode(n');
  addLink((n',n_1)); addLink((n',n_2));
  addLink((n, n')); addLink((n', n))

5. ADAPTATION SPECIFICATION LANGUAGE

Now that we have defined the navigational model, the basic operations upon that model and some conceptual navigational transformations, we are ready to define a language that allows specifying at design time certain kinds of runtime adaptive behavior. Using this language, the designer can specify what type of adaptation is permitted during the life of the website. In this way, he can prohibit that the structure of the website completely runs out of his control due to the unlimited use of adaptation. The language can be characterized as event based: conditions (based on user access patterns) will trigger rules (the adaptive behavior). Frequency of re-applying the rules is not specified here. Also the way the user access patterns are determined is not specified withthe language. These issues are left to (the implementation of) the adaptation engine, and are not described in this paper.

We suppose that the following functions are available to monitor the user access to the website:

Let M = (N, H, L, C) be the Navigational Model for the website W, n, n_1, n_2 ∈ N, l ∈ L, h_1, h_2 ∈ C, and p a path in M:

- numberOfVisits (n)
  Returns the number of visits to the nod
- totalNumberOfVisits
  Returns the number of visits to the website
- numberOfVisits (n, p)
  Returns the number of visits to the node n via the path p
- numberOfTraversings (l)
  Returns the number of time link l is traversed
- relatedNodes (n_1, n_2)

Returns true if the nodes n_1 and n_2 are related, false otherwise

- relatedChunks (h_1, h_2)
  Returns true if the chunks h_1 and h_2 are related, false otherwise
- grouped (h_1, h_2)
  Returns true if the chunks h_1 and h_2 are connected to the same node, false otherwise.
- inAudienceTrack(x, T)
  Returns true if the concept x (e.g. a node or a chunk) is present in the Audience Track T.

The designer can define rules on one single element (a node, a link, or a chunk) to anticipate adaptive behavior on one particular element, but in this way the possibility are rather limited. Therefore, the designer is also able to specify adaptive behavior that is not hooked to one particular element, but to a group of elements, e.g., all nodes of a particular Audience Track, or nodes that have a similar access pattern. To accomplish this functionality, the forEach constructor can be used. The forEach constructor allows to apply a rule on each element of a set.

We will use BNF notation to define the syntax of the Adaptation Specification Language (ASL). Due to space considerations it is impossible to fully specify ASL; only a compact version of the syntax is given here. Words in italic in the syntax denote identifiers that need to be substituted. Words or symbols in bold are keyword. The symbol '!' is used to denote empty (nothing). <condition> is not specified, but the syntax is the usual one making use of boolean operators, function calls, the relation operators $<, \lesssim, >, \gtrsim, =, \tilde{=}, \neq$, and the set relational operations $\supset, \supseteq, \not\subset, \not\subset, \subseteq, \in, \notin$.

<rule> ::= (<expression> (**;** <expression>) * |
forEach (<set> (**,** <set>)* **:**
<expression>(**;** <expression>)* **)**)

<set> ::= <variable> (**,** <variable>)* **in** <elements>
(! |**with** <property> (**and** <property>) *)

<elements> ::= (**Nodes** | **{** *node* (**,** *node*)* **}** |
Chunks | **{** *chunk* (**,** *chunk*)* **}** |
Links | **{** *link* (**,** *link*)* **}**)

<property> ::= <condition>

<expression>::= **if** <condition>
then <operation> (**;** <operation>)*

<operation> ::= (deleteNode (*node*) | addNode (*node*) |
connect(*chunk*,*node*) | disconnect(*chunk*,*node*)|
deleteLink (*link*) | addLink (*link*) |
promoteNode(*node*, *path*) |
demoteNode(*node1*, *path*, *node2*) |
linkNodes(*node1*, *node2*) |unLinkNodes(*node1*, *node2*) |
clusterChunks(*node*, *chunk1*, *chunk2*) |
clusterNodes(*node*, *node1*, *node2*))

To illustrate the use of the Adaptation Specification Language, we will give a few examples, using the Navigational Model of the telephone company of figure 1. As a first example of adaptation, consider the following situation: a lot of users might want to check the tariff formula after they've consulted their credit, to find out why a certain call was billed in a particular way. Beforehand, we do not know if people will act in this way, and more-over, the telephone company would like to avoid linking these two pages if it's not strictly necessary, fearing people will discover their tariff is not that beneficial after all. As a designer, specifying the following rule would be the ideal solution:

if related(n12, n4) then linkNodes(n12,n4)

Another interesting and more complex adaptation we could consider is the following: we might want to anticipate that certain utilities offered by the website will be extremely popular, while others will not be. However, beforehand we are unable to predict which utilities will be the popular ones, and which the less popular ones. Therefore, we'll use the forEach rule:

forEach node **in** {n10, n11, n12} :
if numberOfVisits (node, ((n1,n2),(n2,n6),(n6,node))) / numberOfVisits (node) ≥ 0,9 **and**
(numberOfVisits(node,((n1,n2),(n2,n6),(n6,node))) / totalNumberOfVisits ≥ 0.05
then promoteNode(node, ((n1,n2),(n2,n6),(n6,node)))

Note that both rules are specified at design time, thus effectively allowing to model adaptive behavior before implementation should even be considered. Given an implementation framework, the runtime adaptation rules could even be automatically generated.

6. CONCLUSION & FUTURE WORK

In this paper we present an approach to design adaptive websites. This is done by extending an existing web site design method WSDM with an Adaptation Specification Language that allows a designer to specify the desired adaptive behavior. Following the WSDM approach, the specification is done at the conceptual level rather than at the implementation (or page) level. In this way the specification is independent of the actual implementation and can be carried over to different implementations. The benefit of the approach is that the adaptive behavior is under control of the website designer. In this way he can avoid that the structure of the website runs out of his control by unlimited adaptation and that the website becomes unmanageable. At this point, it is still up to the designer not to specify any rules that would cause the resulting navigational model to be inconsistent, or violate the design philosophy. In the audience driven approach, for example, one might easily imagine a rule triggering a series of linking and unlinking operations, causing a node to become unreachable from its Audience Track. Although possible, we have not defined any constraints that allow controlling this. If a tool supports the method, this tool can guarantee that the adaptation rules formulated in the language will not violate the chosen organizational approach.

We are also fully aware of the current limitations of the Adaptation Specification Language and that further study of different kinds of adaptive behavior is necessary and may lead to additional rules being specified in the Adaptation Specification language.

Future work also includes the actual mapping of the conceptual and implementation design to the actual implementation, after which further experiments with the adaptation language will be possible.

Complementing the work on adapting the navigation and structural issues of web sites, a specification mechanism for

presentational, content and semantic adaptation could be provided.

7. REFERENCES

[1] Bichler, M., and Nusser, S. W3DT - The Structured Way of Developing WWW-Sites, in Proceedings of the ECIS 1996, Lisbon, Portugal (1996)

[2] Bieber, M., and Isakowitz T. Designing Hypermedia Application, in Communications of the ACM, Volume 38, S. 26 - 29.

[3] Bongio, A., Ceri, S., Fraternali, P., and Maurino, A. Modeling Data Entry and Operations, in WebML in Proceedings WebDB 2000 p. 201-214, Dallas (2000)

[4] Casteleyn, S., and De Troyer, O. Structuring Web Sites Using Audience Class Hierarchies, in Proceedings of DASWIS 2001 workshop of the ER 2001 conference, Yokohama, Japan (2001)

[5] Ceri, S., Fraternali, P., and Bongio, A. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites, in Proceedings of the WWW9 Conference, Amsterdam (2000)

[6] Coenen, F., Swinnen, G, Vanhoof, K., and Wets, G. A Framework for Self Adaptive Websites: Tactical versus Strategic Changes, Workshop on Web Mining for E-Commerce, Boston (2000)

[7] De Bra, P., Houben, G.J., and Wu, H. AHAM, A Dexter-based Reference Model for Adaptive Hypermedia, in Proceedings of the 10th ACM Conference on Hypertext and Hypermedia, pp. 147-156, Darmstadt, Germany (1999)

[8] De Troyer, O.M.F. Designing Well-Structured Web Site: Lessons to be Learned from Database Schema Methodology, in Proceedings of the ER'98 Conference, Lecture Notes in Computer Science (LNCS), Springer-Verlag (1998)

[9] De Troyer, O. Audience-driven web design in Information modelling in the new millennium, IDEA GroupPublishing, ISBN 1-878289-77-2 (2001)

[10] De Troyer, O., and Casteleyn, S. The Conference Review System with WSDM, IWWOST 2001, http://www.dsic.upv.es/~west2001/iwwost01/, Valencia (2001)

[11] De Troyer, O., and Leune, C. WSDM: A User-Centered Design Method for Web Sites, in Computer Networks and ISDN systems, in Proceedings of the 7th International World Wide Web Conference, Elsevier (1998) 85 - 94

[12] Garzotto, F., Paolini, P., and Baresi, L. Supporting Reusable Web Design with HDM-Edit, in Proceedings of the 34th Hawaii International Conference on System Sciences, Maui USA (2001)

[13] Halasz, F., and Schwartz, M. The Dexter hypertext reference model, in Communications of the ACM, 37(2): 30 - 39, ISSN 0001-0782 (1994)

[14] Han, E.H., Jain, N, Mobasher, B., and Srivastava, J. Web Mining: Pattern Discovery from World Wide Web Transactions, Technical Report 96-050, Departement of Computer Science, University of Minnesota, Minneapolis, USA (1996)

[15] Isakowitz, T., Stohr, E.A., and Balasubramanian RMM: A methodology for structured hypermedia design, in Communications of the ACM 38, 8 (1995), 34-44

[16] Marcotty, M., and Ledgard, H. The World of Programming Languages, Springer-Verlag, Berlin, pp. 41 and following (1986)

[17] Perkowitz, M., and Etzioni, O. Adaptive web sites: an AI challenge, in Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan (1997)

[18] Perkowitz, M., and Etzioni, O. Adaptive web sites: Automatically Synthesizing Web Pages, in Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI98), Wisconsin (1998)

[19] Perkowitz, M., and Etzioni, O. Towards Adaptive Web Sites: Conceptual Framework and Case Study, in Artificial Intelligence 118 (1-2) (2000)

[20] Rasmussen, E. Clustering algorithms, W.B. Frakes, R. Baeza-Yates (Eds.), Information Retrieval, Prentice Hall, Englewood Cliffs, NJ, pp. 419-442 (1992)

[21] Schwabe, D., Guell, N., and Vilain, P. Modeling Interactions and Navigation in Web Applications, in Proceedings of the World Wild Web and Conceptual Modeling 2000 Workshop, ER Conference, Springer, Salt Lake City (2000)

[22] Schwabe, D., and Rossi G. An Object Oriented Approach to Web-Based Application Design, in Theory and Practice of Object Systems 4(4). Wiley and Sons, New York (1998)

[23] WSDM website. http://wsdm.vub.ac.be/

[24] Wu, H., De Kort, E., and De Bra, P. Design Issues for General-Purpose Adaptive Hypermedia Systems, in Proceedings of the 12th ACM Conference on Hypertext and Hypermedia, pp. 141-150, Aarhus, Danmark (2001)

[25] Yan, T., Jacobsen, H., Garcia-Molina, H., and Daval, U. From user access patterns to dynamic hypertext linking, in Proceedings of the 5th International WWW Conference, Paris, France (1996)