

# MODELLING AND IMPLEMENTING MACRO WEB NAVIGATION STRUCTURES

**Christian Brelage**

Department of Information Systems  
University of Muenster, Germany  
Leonardo-Campus 3, 48149 Münster  
phone: +49 251 / 83 - 38075 fax: - 38109  
ischbr@wi.uni-muenster.de

**Lars H. Ehlers**

(see above)  
phone: +49 251 / 83 - 38084 fax: - 28084  
islaeh@wi.uni-muenster.de

**Jörg Becker**

(see above)  
phone: +49 251 / 83 - 38100 fax: - 38109  
isjobe@wi.uni-muenster.de

## ABSTRACT

*From the three existing dimensions - content objects, navigation and presentation - of web pages, the navigation dimension is analysed. It consists of macro, meso and micro navigation. A conceptual approach to model and implement the macro navigation structure of web sites is presented. We start by enhancing a function decomposition diagram to model the macro navigation structure. The resulting graph is analysed and transformed to find its crucial components. In order to store these components in a database table, an appropriate entity-relationship model is developed. Afterwards we identify three navigation styles that can be produced from the underlying database and give the PHP source code for one of them. Finally, an example web page using our navigation styles is depicted. It is concluded that using our ideas may lead to reduced development time and avoids link errors in the process of creating the site navigation.*

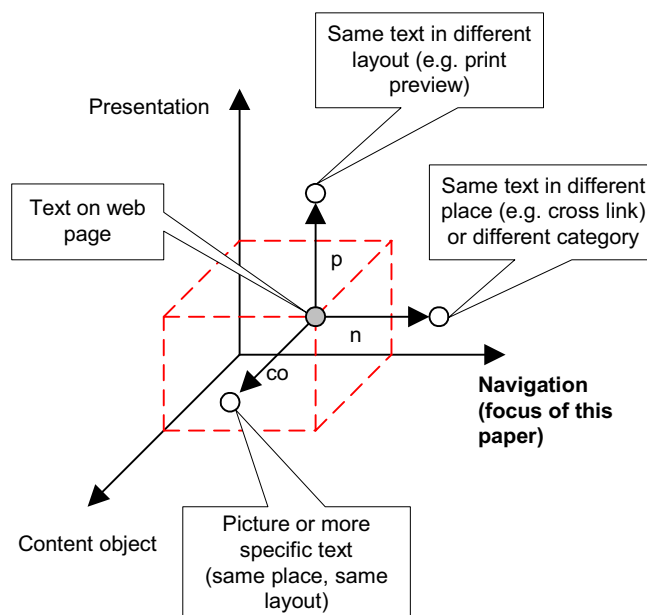
## 1. MOTIVATION

Web application development or the implementation of a large web site with a Content Management System is a very important and difficult task for companies and organizations (Myers et al. (1996) and Lerner (2000)). Web based information systems become increasingly complex and are therefore more difficult to implement. Additionally, more and more business processes are supported by web based applications to reduce costs and processing times.

There are several approaches and techniques to describe web applications at the conceptual level, e. g. HDM, Dexter and WebML. All these techniques have one thing in common: They separate different views (or dimensions) of web applications. The Dexter Reference Model concentrates on naming conventions and was introduced by Halasz and Schwartz (1990). HDM was described by Garzotto et al. (1993). Ceri et al. (2000) propose the four main views *structure model*, *presentation model*, *hypertext*

*model* and *personalization model* in WebML. For the purpose of this paper we ignore the personalization because it is not our focus. We will distinguish the three dimensions *content objects*, *presentation* and *navigation* only. They correspond (in the same order) in general to the previously named views from Ceri et al. (2000) (without personalization). The three orthogonal dimensions are shown in figure 1 as an example. The *content object* dimension describes the media itself, which is presented to the user (e. g. texts, pictures, videos). The *presentation* dimension deals with the layout and appropriate presentations of the content objects (e. g. layout rules to support corporate identity or templates to support handicapped people with greater text size). Finally, the *navigation* dimension describes the general structure and categories of the website.

These three dimensions are the "building blocks" of HTML-pages and have to be defined and described properly to meet the demands of a modern software engineering process (e. g. differentiation into phases, proper documentation). The circles are symbols for parts of web pages that consist of a content object with a layout (= presentation) in a special category (= navigation). The grey circle is our starting point and represents a text on a web page. The arrow *co* shows a change of content object (while staying in the same presentation and navigation). This means that the text could be replaced or accompanied by a picture (e.g. the picture of an island instead of the text "island") or by a more detailed description. The arrow *p* corresponds to a move on the presentation dimension. The same text is displayed for example as a print preview. The *n* arrow from the grey circle to the white circle represents two different options. Either the same content object is used in a different place of the web site or a change of category takes place (categories are described later in this paper). Note that categories belong to the navigation and are not content objects.



**Figure 1: Dimensions of web pages**

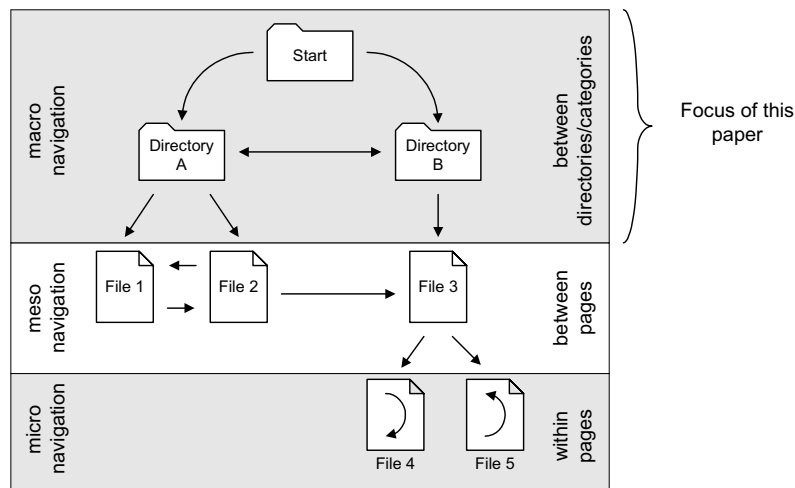
In this paper we will present an approach to model the navigation dimension of websites on the conceptual level. To clarify our approach, some examples will be given. These examples are drawn from the travel industry. According to Bichler (2001) and the WTTC (1996) the travel and tourism industry are growing industries and by 2007 approximately 11 % of the worlds gross domestic product (GDP) will be achieved by this industry. Out of the dimensions mentioned above, we will discuss the navigation dimension in detail and present an approach to differentiate various navigation structures.

The work from Niederacher and Wahler (1999) started with ideas on this subject, but lacks some important features. Their conceptual structure model does not include cross links or the important sorting of navigation elements.

## 2. THE NAVIGATION DIMENSION: MACRO-, MESO AND MICRO NAVIGATION

Usually, the whole navigation structure (including macro and meso aspects) is modelled within a single model, using a symbol for nodes (representing a category or a HTML-page) and arrows for links. The result of this approach is a complex graph, which is difficult to communicate to others who lack the technical skills to understand the complexity. Furthermore, there is a semantic difference between the various types of navigation structures.

On the one hand, most website use some kind of hierarchy to organize content (e. g. categories in the Travelocity screenshot below in figure 3). On the other hand, web pages are linked based on semantic dependencies, e. g. to order the different steps of a business process (booking process at Travelocity). We therefore propose the following differentiation of navigation structures (see figure 2). We have taken Travelocity as example for an internet travel agency as Standing et. al (1999) predict from their research that these agencies will supersede some of the high street retail travel agents, Anckar and Walden (2000) take a more critical perspective, as their finding shows that using "e-intermediaries" lead in some cases to higher prices and lower quality of products.



**Figure 2: Three different navigation structures**

- *Macro navigation:* The macro navigation builds up the general navigation structure of the website. A node in the macro navigation generally corresponds to a category of the website. A category is a container that is used to collect content objects under a common theme (e. g. “lodging”, “cars/rail”) or topic. This navigation structure mainly defines the way users are able to find the desired information. If the web pages are stored as static HTML-documents, the macro navigation structure can correspond to the directories in the file system. Note that a node of the macro navigation structure does not have to correspond to a directory in the file system. The macro navigation links collections (categories or directories) of content objects (HTML-pages).
- *Meso navigation:* The meso navigation links content objects or HTML-pages. As stated above, pages in complex web applications are linked based on their semantic dependencies and relations. If a business process like “booking a flight” should be supported by the web application, several steps will be necessary to collect all the information, that is needed. This navigation structure is not discussed in this paper.
- *Micro navigation:* Finally the micro navigation defines a navigation structure within a content object. Sometimes, a content object cannot be displayed on the screen at once (especially large texts). The usability of the web page can be increased by using micro navigation structures (these are corresponding to anchors in a HTML-document). This navigation structure is not discussed here either.

The definition of the macro navigation of a website is one of the key development steps. The following examples may underline the importance of a well-defined macro navigation structure (compare Zafiris (2001)):

- As noted above, the macro navigation mainly defines the way information can be retrieved by users. Usually, visitors who enter a website for the first time try to understand the main navigation structure. Hence, a well-defined navigation structure is indispensable in enabling users to find their information efficiently. Conceptual shortcomings will result in confused users and reduced traffic.
- Usually, the macro navigation of a website is defined once and does not change over time. Visitors tend to use bookmarks to find their favourite pages. If the macro navigation of a website is altered, users are not able to use their old bookmarks anymore.
- Nevertheless, the macro navigation has to be flexible and adaptable. A company or organization may decide to offer a new product or service. Therefore, the conceptual definition of the navigation structure has to allow these changes to fit in smoothly with the existing navigation structure. An example may illustrate this. Figure 3 shows the homepage of Travelocity (<http://www.travelocity.com/>). In this case, the macro navigation corresponds to categories ("lodging", "cars/rail" etc.) of the website. If Travelocity would decide to add other services to their website, these would fit in smoothly with the existing navigation structure.

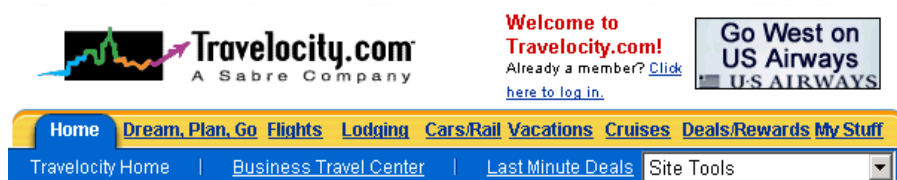


Figure 3: Macro navigation structures (respectively categories) at Travelocity

### 3. MODELLING NAVIGATION STRUCTURES

In order to model the navigation structure on a conceptual level we use the ARIS Business Suite. Although this tool was originally not designed to model web navigation structures, it can be used to describe the navigation with a simple graphic notation. This notation can be easily communicated to non technical members of the project group, that are as well involved in developing the web application (e. g. designers, marketing people). In contrast to tools like Visio, ARIS is a *modelling* tool. All models and objects are *defined only once* and stored in a repository. An object definition may then be *represented multiple times* in different models. The most important ARIS feature is the possibility to generate reports (in a Visual Basic like scripting language). A report can be used to generate a database input or a flat file that contains all the information needed to fill the database tables described in the next chapter. By this, it is possible to automate parts of the web application process in respect to defining the macro navigation structure of a website. Another feature of the ARIS Toolset is exploited in this paper. Objects can be arranged with the “auto layout” function. We propose a layout with the first level being horizontally formatted and all consecutive levels vertical. This will offer a clear picture for many site navigation trees.

We adapted the modelling method “function decomposition diagram” to fit our needs. Each node (shown as a green rectangle) represents a category of the website (as in figure 4). The root node with the object identification 1 (OID) represents the website itself. Note that we are modelling the macro navigation structure of the website only. In our case we do not have a strong hierarchy, because the node “Airport codes” is used twice in our model. One of these nodes links to the other: The node with object ID 13 cross links to the node with object ID 10 (and they refer to the same HTML-page). In the figure this is shown by the additional “X: 10” to the right of the OID 13. Although OID 10 and OID 13 carry the same name, the objects are not identical (hence they have a different OID). This allows that

the node with OID 13 may be named differently (e. g. “Explanation of Airport codes”). The object identification for each node can be generated automatically. Figure 4 shows a screenshot of an example travel agency macro navigation structure modelled in ARIS.

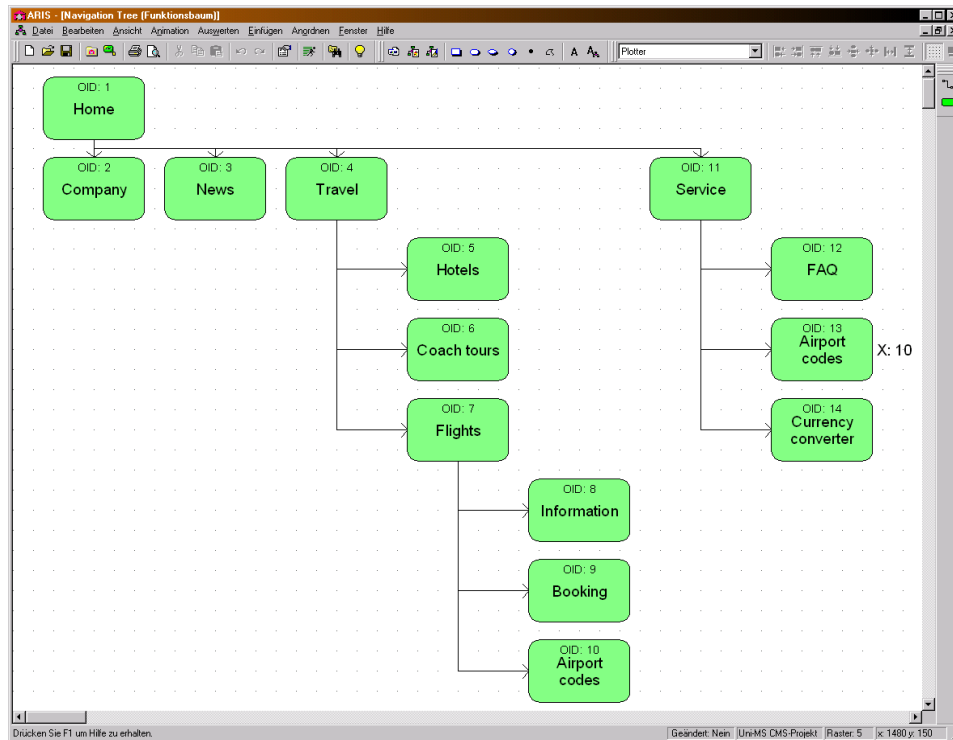


Figure 4: Screenshot depicting enhanced function decomposition diagram in ARIS Toolset

## 4. STORING NAVIGATION STRUCTURES

In the previous chapter we adopted a graphical notation for the navigation structure of web sites. This structure needs to be stored in an appropriate way in a database. We will create a conceptual data model in this paragraph, which is able to fulfil the task of storing hierarchical trees with cross links. When the scripting facilities of ARIS are used as mentioned above, no manual interaction is needed in the process as the script runs automated and updates the database table on a regular basis. In a scenario where no modelling tool is employed, a manual transfer from the graphical design to the database tables would be necessary. Our conceptual data model will be an entity relationship model (ERM) according to Chen's (1976) notation with some minor extensions in respect to the placing and naming of elements as proposed by Becker and Schütte (1996). Relationship types will have their cardinalities printed in the (min, max) notation.

### 4.1. Conceptual model

We are faced with the task to develop an ERM for hierarchical navigation trees with ordered elements and cross links between the leaves. Figure 5 displays the simple structure of a site navigation tree in step one. As the order of the elements is only shown *implicitly* (from left to right between elements on the same level), we add *explicit* arrows to illustrate sorting of elements in step two. Only explicit information can be stored in the tables defined by the ERM later on. Looking closely at the tree in step two, the reader will recognize that some redundant information are captured. The relational concept does not need arcs between *all* elements with relations to each other because transitive queries are supported. So the arc between H and N can transitively be resolved by using H to C and then C to N and must consequently be deleted. This deletion can be repeated for many other arcs in the site naviga-

tion tree shown in step two. Due to this, the positioning of elements is rearranged in step three to give a clearer picture of a tree and NUL pointers are added.

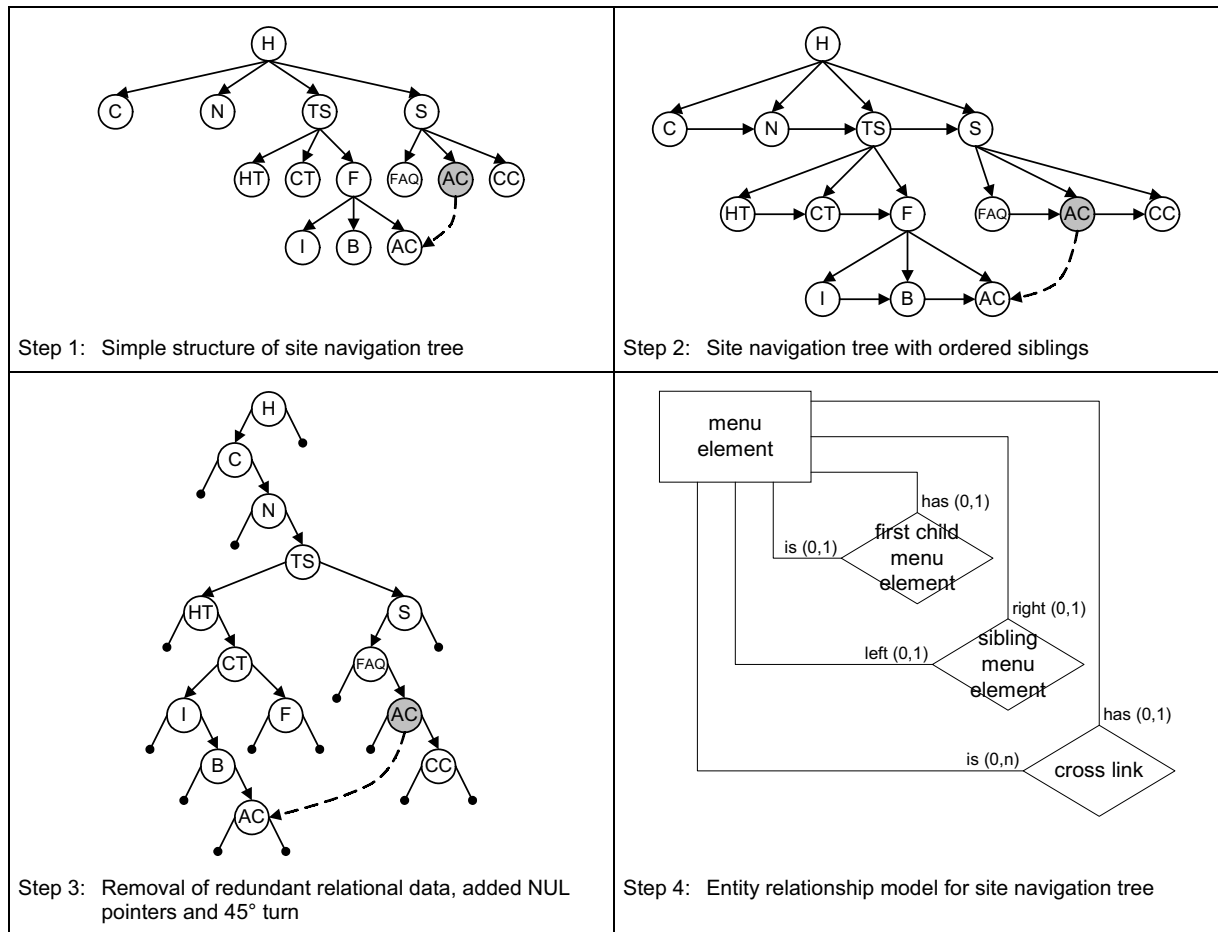


Figure 5: From simple site navigation tree to ERM

In the last step an ERM is derived which describes the conceptual level of how to store the site navigation tree of the previous step. Chen (1976) differs between entities and relationships on one hand and entity sets and relationship sets on the other hand. While entities and relationships are *individual* elements the sets are *collections* of similar elements. In literature the term *set* is often replaced by *type* as stated in Becker and Schütte (1996). In the following sections we will therefore use the term entity type and relationship type. So H, C, N, etc. are individual *entities* while talking about them as a group leads us to the term *entity type*.

The basic element of our ERM is the entity type *menu element*. It corresponds to the categories shown in step three (H, C, N, etc). Every menu element *has* either none or at most one *first child menu element*. Looking at this relationship from the perspective of a child, a menu element *is* either a first child or not. The relationship type *first child menu element* covers the left arrows of the elements in the site navigation tree. The right arrows are handled by the *sibling menu element*. Siblings can have respectively at most one "family member" to their *right* and *left* side. Menu elements can only have ("*has (0,1)*") one *cross link* while they can be ("*is (0,n)*") the target or several of those links. The cross link relationship type is responsible for storing the information from the dotted arrows.

It could be argued to merge the two relationship types *first child menu element* and *sibling menu element* into one relationship type. But by doing so, one would have to add a binary attribute to the new relationship type that has either the value "first child" or "sibling". For the sake of clarity, we neglect this possibility. Our ERM is understandable without looking at the attributes of entity types or relationship types.

## 4.2. Design specification

The rules for transforming ER diagrams into tables as defined by Chen (1976) lead to one single table for our site navigation tree. The relationship types *first child menu element* and *sibling menu element* are each only (0,1):(0,1) relations and can thus be appended to the table "menu". The same applies to the cross link relationship type with its (0,1):(0,n) cardinality. For pronouncing reasons we have not called the database attribute for the cross link "cid" but "xid" instead to differentiate it from the siblings identification ("sid"). The final table may need more attributes (e.g. for presentation) and we have given a picture attribute as an example (which will be used in a later section).

Table: menu

meid	text	filepath	fcid	sid	xid	normal_pic	...
1	Home	/	2				...
2	Company	company.html		3		compn.gif	...
3	News	news.html		4		newsn.gif	...
4	Travel	travel/	5	11		traveln.gif	...
5	Hotels	hotels.html		6		hotelsn.gif	...
6	Coach tours	coach.html		7		coachn.gif	...
7	Flights	flights/	8			flightsn.gif	...
8	Information	information.html		9		infor.gif	...
9	Booking	booking_start.html		10		bookingn.gif	...
10	Airport codes	aircode.html				aircoden.gif	...
11	Service	service/	12			servicen.gif	...
12	FAQ	faq.html		13		faqn.gif	...
13	Airport codes			14	10	aircoden.gif	...
14	Currency converter	currency.html				currencyn.gif	...

**Table 1: Combined table "menu"**

It is important to mention that the data in the table can automatically be filled from the design tool, which is a great improvement in maintaining a site navigation tree. We have taken the ARIS Toolset as an example but any other modelling tool will be fine, as long as it supports scripting and multiple, user defined attributes. The former is used for reading the elements in the repository and outputting them into the database (or a flat file) and the later is needed for manually adding cross links, filenames of pictures, pathnames for the navigation, etc. All additional attributes of table 1 are mapped to user-defined attributes in the function decomposition diagram. While the OID is filled automatically (or at least semi-automatically), adding a cross link is a manual task. For those links, we take a user-defined attribute as seen in figure 4. If a modelling tool is not used for some reason, the transfer has to be done manually.

## 5. NAVIGATION STYLES

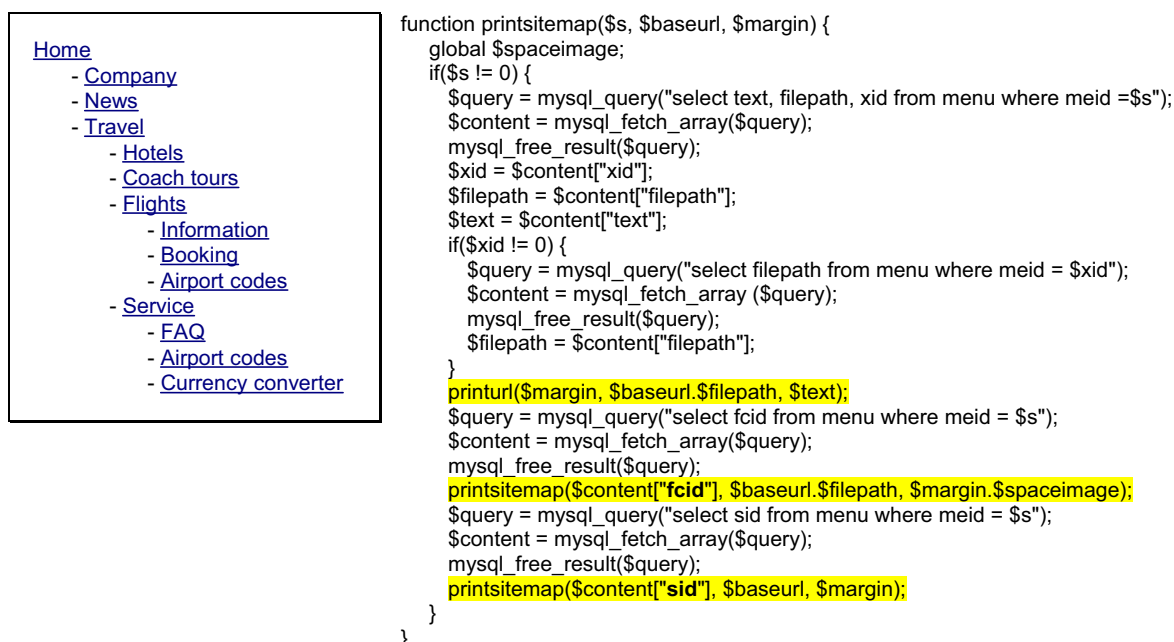
Having defined the conceptual model and the design specification for storing navigation structures, we now give different styles that create a navigation structure to the visitor of a site. In the following paragraph we will identify three navigation styles. We will introduce very basic formatting examples for each of the three styles. A more sophisticated presentation is possible by modifying underlying procedures according to the design requirements. The navigation styles can be implemented with dynamic Internet technologies like PHP or JSP. For more information about recursion in PHP see Bakken et al. (2001). For real-life implementations we propose to call the PHP code and save the result into a static file (SHTML or PHP). This is a commonly used caching technique in the web to present database results that change on a low frequency only. Wills and Mikhailov (1999) show basic concepts of web caching in their paper. The resulting file can then be integrated using server side includes (SSI) in other web pages. The file must only be modified when the database is updated.

In the following sections we will talk about styles, modules and categories. A navigation *style* is a synonym for the graphical representation to the user. Styles can be modified to fit into corporate design by modifying the program code in the *module*. A module is therefore the lines of code building up the specific design and the functionality. In our example we use basic formatting instructions only. *Categories* are, as mentioned before, the points reachable by navigating through the tree.

## 5.1. Sitemap

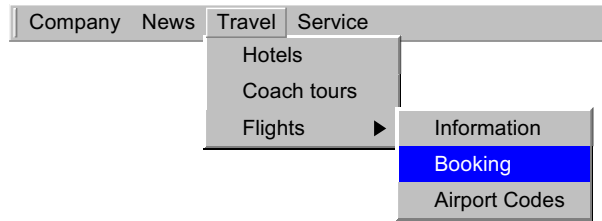
A very common output that can be achieved by querying the database tables is a static sitemap (Inder et al. (1996)). Figure 6 depicts the output of such a sitemap on the left side. On the right the algorithm of a PHP function is printed. The function call `printsitemap(1, "", "")` returns the full static sitemap. In the algorithm three lines are highlighted. The first highlighted line shows the call of the `printurl` function with three arguments. This function takes its first argument to determine the margin on the left side. The second argument is used for the anchor hypertext reference ("a href"). The text in the third argument is then printed as a clickable output and links to the hypertext reference. The second and third highlighted line of code use recursive function calls. The former prints the sitemap for the first child (hence "fcid" as first argument of `printsitemap`), the later takes care of the siblings by using the "sid" variable.

A static sitemap shows all accessible categories of the web site to the user. To offer a higher flexibility, we allow the creation of *partial* static sitemaps. A partial sitemap does not start with the root of the tree, but with any other valid category from a lower level. The same algorithm from figure 6 can be used to create the partial sitemap for the category "Travel". The code must then be called with the corresponding "meid" as starting argument. The proposed caching technique can still be used. For the partial static sitemap a new cache file is created.



**Figure 6: Static sitemap and example recursive PHP code**

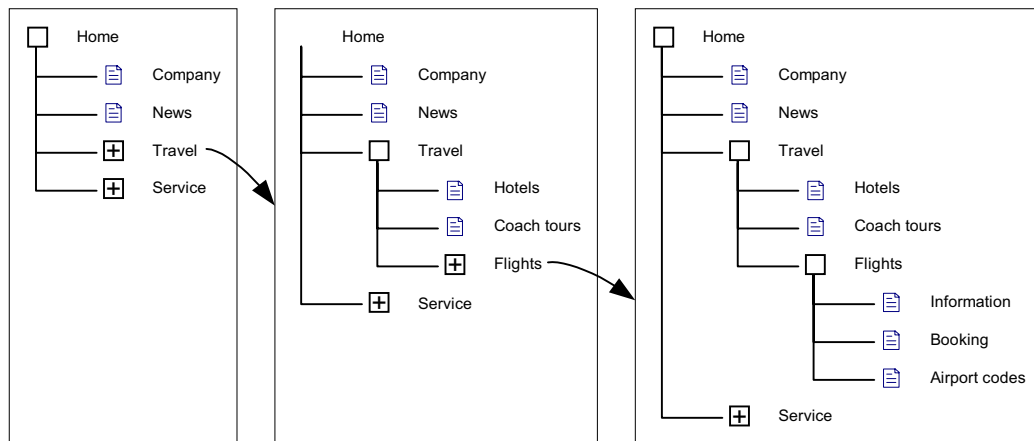
For a smoother interaction with the user of a web site, java script in combination with dynamic HTML (DHTML) is often used. A dynamic (java script and DHTML based) sitemap offers the same possibilities of navigating the web site, but it does not display all deeper links at once. Instead the scripting possibilities of modern browsers are utilized. For each main category the deeper categories are only displayed when the mouse moves over the entry. The concept of dynamically showing other links to the user does not distract the visitor with too many other entries. We have depicted in figure 7 a dynamic sitemap where the user selected "Travel" first and "Flights" afterwards. The menu can be displayed using pictures for every element. The name and location of the picture files can be retrieved from the "menu" table as mentioned above (see table 1 on page 1553).



**Figure 7: Dynamic sitemap**

The way of implementing the dynamic sitemap module is similar to the static sitemap. A recursive algorithm selects all entries from the database and integrates them in the correct way into a valid java script code. The resulting code includes all navigation entries and shows the categories by displaying their pictures. By automatically executing the code in a web page, the dynamic menu is generated.

The third possibility of presenting a sitemap applies drill down techniques. Whenever a visitor clicks on a category that has subfolders, they are opened. The appropriate opening can technically be achieved by forcing a page reload with new parameters or by using java script and DHTML as in the previous example. In figure 8 the visitor selects the same way of navigation from "travel" over "flights" to "booking".



**Figure 8: Drill-down sitemap**

All three styles of presenting a navigation are based on a sitemap and can show *all* reachable categories of a site. In the next two paragraphs we will present special modules for the navigation, which will only show *some* categories at once.

## 5.2. Path list

In a path list the web visitor sees a clickable list of previously visited categories. Those categories are more detailed to the right and show the navigation path back home on the left. This navigation style is employed on many web sites and became popular due to Yahoos and Lycos usage of this technique to present their web directories (for an illustration see Yahoo! Inc (2001) and Lycos Inc. (2001)).

For our example travel agency figure 9 depicts the output of a path list for a visitor who navigated from the homepage via travel and flights to the bookings category. The navigation module forming a path list can be used in conjunction with other navigation modules. The most common combination is together with a siblings list.

## 5.3. Siblings list

The "siblings list" displays all navigation categories which are positioned on the same navigational depth. The first element of a siblings list is always a "first child" (from the parents view) and the next

entry can be found by following the right arrow (see step 3 in figure 5) of each element. The arrow corresponds to the relationship type "sibling menu element" in the ERM (step 4 in figure 5).

[Home](#) > [Travel](#) > [Flights](#) > [Booking](#)

Figure 9: Path list

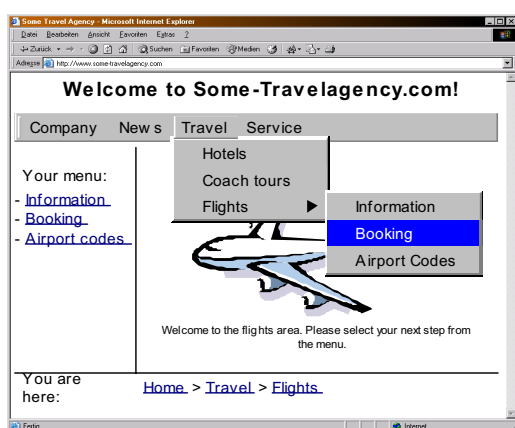
- [Information](#)
- [Booking](#)
- [Airport codes](#)

Figure 10: Siblings list

The techniques to pre-calculate the different siblings lists can be used here, too. Each sibling list belongs to only one parent and therefore the amount of different lists is finite. The site of our example travel agency offers four siblings lists. The first one is home's list with the elements company, news, travel and service; the second list is travel's list with hotels, coach tours and flights. The third list is shown in figure 10 and the fourth list belongs to the service category with the entries FAQ, airport codes (which is itself only a cross link) and currency converter.

## 5.4. Example site

The previous navigation styles can be deployed to build an example web site. By using the SSI technique (Webopedia (2001)) we show a basic site (figure 11, left) and the corresponding source code (figure 11, right) that uses our navigation modules. The content of the page is currently not modelled (see final chapter for further research in this area) and we include it only by using a standard HTML-file ("flight\_content.php" in the example).



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<html>
<head>
<title>Some Travelagency.com</title>
</head>
<body>
<h3>Welcome to Some-Travelagency.com!</h3>
<!--#include file="dynamic_sitemap.php" -->
<table cellspacing="2" cellpadding="2" border="0" width="100%">
<tr>
<td>Your menu:<!--#include file="flights_siblings.php" --></td>
<td><!--#include file="flight_content.php" --></td>
</tr>
<tr>
<td colspan="2"><hr>
You are here:<!--#include file="flights_path.php" --></td>
</tr>
</table>
</body>
</html>
```

Figure 11: Example site with the three presented navigation styles

The first highlighted include takes the code with the dynamic sitemap and inserts it into the web page. The second highlighted line reads the siblings list for the flight category and positions it to the left of the page. As mentioned before, we neglect in this paper the modelling of content and view it as an encapsulated element, hence we just include all of it in one large chunk. The last highlighted line adds the path list for the category flights.

## 6. CONCLUSION AND OUTLOOK

Based on a function decomposition diagram and the conceptual ERM, a storage structure was defined. It supports an automated process of web navigation development. We defined three different styles (site map, path list and siblings list) that are used on many web sites. For each of the styles, a module with an algorithm exists and the one for the static sitemap was presented in detail. The modules described form small SSI files that can be inserted into the code of a web page. The content of the page

still has to be manually maintained (although it is encapsulated in a separate SSI file) but all work related to updating and modifying the navigation can be handled with the proposed ideas.

Our approach helps in the initial creation of a web site as well as in the maintenance phase. It will save time due to automatic generation of navigation structures and reduce the possibilities of linking errors in the navigation tree ("dead links" are avoided). Furthermore it supports standard software engineering processes. The graphical notation and the included user defined attributes per element serve as documentation for the navigation structure of the web application.

We are currently working on extending the concept to include meso- and micro-navigation. A second line of research looks into developing similar approaches for the other two dimensions *presentation* and *content object*. The result could be an integrated architecture describing on the one hand single web pages and on the other hand complete web applications. This architecture will be open to other modelling tools (not only ARIS) due to an XML compliant interface.

Our findings will help travel agencies and other mediators in the industries to build their web site in a cost effective way. Due to growing competition in the industry and the intensive use of information technology (Borbely and Vasudavan (1999)) an easy method of creating web sites is needed by the industry.

## REFERENCES

- Anckar, B and Walden, P. (2000). Becoming Your Own Travel Agent: A Web of Potentials and Pitfalls. In Proceedings of the 33rd Hawaii International Conference on System Science, p. 1-10.
- Bakken, S. et al. (2001). PHP Manual. <http://www.php.net/manual/en/>
- Becker, J. and Schütte, R. (1996). Handelsinformationssysteme. verlag moderne industrie. Landsberg/ Lech, 75-82.
- Bichler, M. (2001). An Application of Multi-Attribute Auction Markets in Tourism. In Information Age Economy. Fünfte Internationale Tagung Wirtschaftsinformatik 2001. Physica-Verlag, Heidelberg.
- Borbely, S and Vasudavan, T. (1999). A Study of Web Diffusion in Travel Agencies. In Proceedings of the 32th Hawaii International Conference on System Sciences, p. 1-9.
- Ceri, S., Fraternali, P., Paraboschi, S. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. In Proceedings of the 9th International World Wide Web Conference. Elsevier, p 137-157.
- Chen, P. P. S. (1976). The Entity-Relationship Model - Toward a Unified View of Data. In ACM Transactions on Database Systems, 9-36.
- Garzotto, F., Paolini, P. and Schwabe, D. (1993). HDM - An model-based approach to hypertext application design. In ACM Transactions of Information Systems 11. p. 1- 26.
- Halasz, F. and Schwartz, M. (1990). The Dexter Hypertext Reference Model. In Proceedings of the Hypertext Workshop. National Institute of Standards (NIST) Special Publication. Gaithersburg. p. 95-133.
- Inder, R., Kilgour, J. and Lee, J. (1998). Automatic generation of diagrammatic Web site maps. In Proceedings of the 1998 ACM symposium on Applied Computing, p. 719-725.
- Lerner, R. M. (2000). At the Forge: Content Management. Linux Journal 77, no. 14.
- Lycos Inc (2001). Lycos Internet Software. <http://computers.lycos.com/software/swinternet.asp>
- Myers, B., Holland, J. and Cruz (1996). Strategic Directions in Human Computer Interaction. In ACM Computing Surveys 28(4), p. 794-809.
- Scheer, A. W. (2000). ARIS: Business Process Modeling. Springer Verlag, Berlin
- Standing, C, Borbely, S. and Vasudavan, T. (1999). A Study of Web Diffusion in Travel Agencies. In Proceedings of the 32nd Hawaii International Conference on System Science, p. 1-9.
- Webopedia (2001). Server Side Includes. <http://webopedia.internet.com/TERM/S/SSI.html>
- Wills, C. and Mikhailov M. (1999). Towards a better understanding of web resources and server responses for improved caching. In Eighth International World Wide Web Conference, Toronto, Canada.
- World Travel and Tourism Council (1996). The WTTC 1996/7 World Travel & Tourism Report. WTTC, London.
- Yahoo! Inc. (2001). Yahoo! Computers and Internet. [http://dir.yahoo.com/Computers\\_and\\_Internet/Software/](http://dir.yahoo.com/Computers_and_Internet/Software/)
- Zafirris, P. A. et al. (2001). A Practitioner's Approach to Evolving and Remodelling Large-Scale WWW Sites. In Proceedings of the 34th Hawaii International Conference on System Sciences, p. 1-10.