Updated version Published in Proceedings of the ER'98 Conference, Lecture Notes in Computer Science (LNCS), Springer-Verlag, 1998.

Designing Well-Structured Websites: Lessons to be Learned from Database Schema Methodology.

Olga De Troyer

Tilburg University, INFOLAB, Tilburg, The Netherlands
detroyer@kub.nl

Abstract. In this paper we argue that many of the problems one may experience while visiting websites today may be avoided if their builders adopt a proper methodology for designing and implementing the site. More specifically, introducing a systematic conceptual design phase for websites, similar in purpose and technique to the conceptual design phase in database systems, proves to be effective and efficient. However, certain differences such as adopting a *user-centered* view are essential for this. Existing database design techniques such as ER, ORM, OMT are found to be an adequate basis for this approach. We show how they can be extended to make them appropriate for website design. We also indicate how conceptual schemes may be usefully deployed in future automation of site creation and upkeep. Furthermore, by including parts of such a conceptual schema inside the site, a new generation of search engines may emerge.

1 Introduction

The World Wide Web (WWW) offers a totally revolutionary medium for asynchronous computer-based communication among humans, and among their institutions. As its primary use evolves towards commercial purposes, competition for the browser's attention, often split-second, is now a dominating issue. This has forced the focus of website design towards visual sophistication. Websites must be 'cool, hip, killer'. Most of the literature on website design therefore appears to deal with graphics, sound, animation, or implementation aspects. The content almost seems to be of less importance. Most 'web designers' have certainly never been schooled in traditional design principles nor in fundamental communication techniques. They have 'learned' to design webs by looking at other websites and by following a 'trial-and-error' principle. In addition, the Web is constantly in evolution, outdating itself nearly daily. The combination of all these factors for an individual website easily leads to problems of maintenance but also of elementary usability.

Indeed, as any database designer knows, if the represented information is not structured properly, *maintenance problems* occur which are very similar to those in databases: redundancy, inconsistency, incompleteness and obsolescence. This is not surprising as websites as well as databases may provide (large) amounts of information which need to be maintained. The same aspects also lead to *usability problems.* These are particularly obnoxious as they are problems experienced by the target audience of the website:

- *Redundancy.* Information which is needlessly repeated during navigation is annoying to most users.
- *Inconsistency.* If information on the site is found to be inconsistent, the user will probably distrust the whole site.
- *Incompleteness.* Stale and broken links fall in this category, but incompleteness is also experienced by users who cannot find the information which they expect to be available on a site.
- *Actuality*. Organizations and information are often changing so quickly that the information provided on websites soon becomes out of date. If a website has visibly not been updated for a while, confidence of users in the information provided is likely not to be very high.

Other usability problems are caused by:

- *Lack of a mission statement.* If the website has no declared goal, that goal, quite simply, can not be reached. The key question, therefore, that must be answered by its owner first is "What do I want to get out of my site?". This mission statement is the basis for any evaluation of the effectiveness of the site.
- *Lack of a clearly identified target audience.* The target audience is the audience which will be interested in the site. If one does not have a clear understanding of one's target audience, it is quite difficult to create a compelling and effective site.
- *Information overload.* Users typically are not interested in wading through pages and pages of spurious "information". Also, attention spans tend to be short.
- *The lost-in-hyperspace syndrome* [11]. Hypertext requires users to navigate through the provided information. If this navigation process is not well structured or guided, users may easily get lost. This makes it more difficult and time-consuming to locate the desired information.

The use of a proper design method could help solve some of these problems. A number of researchers have already recognized the lack of a design method for websites, or more in general for web-based information systems, and have proposed methods: HDM [7] and its successors HDM2 [6] and OOHDM [13], RMM [9], W3DT [17], the method for analysis and design of websites in [15], SOHDM [10]. Older methods (HDM, OOHDM, RMM) were originally designed for hypertext or hypermedia applications and do not deal comfortably with web-specific issues. In addition, these methods are very much data-driven or implementation oriented. Some have their origin in database design methods like the E-R method [1] or object oriented (OO) methods such as OMT [12]. These methods may be able to solve maintenance problems to some extent but they do not address the other usability problems mentioned above.

In [4], we have proposed a website design method, called WSDM, which is 'user centered' rather than 'data-driven'. In a data-driven method the data available in the organization is the starting point of the modeling approach. In our approach, however, the starting point is the target audience of the website. The issues related to this target audience run through the method like a continuous thread. We will explain the differences between *data-driven* and *user-centered* in more detail in section 3.1. We argue that our approach results in websites which are more tailored to their users and therefore have a higher usability and greater satisfaction coefficient. WSDM also makes a clear distinction between the conceptual design and the design of the actual presentation. The conceptual design, as in database design, is free from implementation details and concentrates on the content and the structuring of the website. The design of the presentation takes into consideration the implementation language used, the grouping in pages, and the actual 'look and feel' of the website. This distinction is comparable to the distinction made in database design between the conceptual design (e.g. an E-R schema [1]) and the logical design (e.g. a relational schema).

The purpose of this paper is to explain the concept of a conceptual schema within the context of a website design method (section 3) and to identify the different roles it plays in the life cycle of the website (section 4). In section 2 we give a short overview of the different phases of our WebSite Design Method. Section 5 concludes the paper.

2 The WebSite Design Method (WSDM)

We only present a brief overview of WSDM; a more detailed description can be found in [4] and [5]. The method currently concentrates on *kiosk websites*. A kiosk website [9] mainly provides information and allows users to navigate through that information. An *application website* is a kind of interactive information system where the user interface is formed by a set of web pages.

The core of the method consists of the following phases: User Modeling, Conceptual Design, Implementation Design and the actual Implementation (see figure 1 for an overview).

We suppose that the *mission statement* for the website has been formulated before the start of the User Modeling phase. The mission statement should describe the subject and the purpose of the website as well as the target audience. Without giving due consideration to these issues, there is no proper basis for decision making, or for the evaluation of the effectiveness of the website. As an example we consider the mission statement of a typical university department website. It can be formulated as follows: "Provide information about the available educational programmes and the ongoing research to attract more students, researchers and companies, and enhance the internal communication between students and staff members".

The *User Modeling* phase consists of two sub-phases: User Classification and User Class Description. In the *User Classification* we identify the future users or visitors of the website and classify them into *user classes*. The mission statement will give an

indication of the target audience, but this has to be refined. One way of doing this is by looking at the organization or the business process which the website should support. Each organization or business process can be divided into a number of *activities.* Each activity involves people. These people are potential users/visitors of the site. In our method, a user class is a subset of the all potential users who are similar in terms of their information requirements. Users from the same user class have the same information requirements. As an example the user classes of our university example are: `Candidate Students`, `Enrolled Students`, `Researchers`, `Staff Members` and `Companies`. User classes need not be disjoint. The same person may be in different user classes depending on the different roles he plays in the organizational environment. For example, a person can be an enrolled student as well as a staff member.

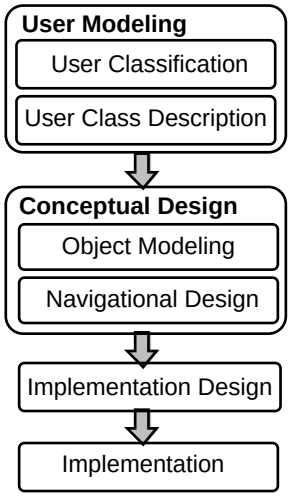

Fig. 1: Overview of the WSDM phases

In the *User Class Description,* the identified user classes are analyzed in more detail. We not only describe (informally) the *information requirements* of the different users classes, but also their *usability requirements* and *characteristics*. Some examples of user's characteristics are: levels of experience with websites in general, language issues, education/intellectual abilities, age. Some of the characteristics may be translated into usability requirements while others may be used later on in the implementation phase to guide the design of the 'look and feel' of the website, e.g. younger people tend to be more visually oriented than older people.

Although, all users from a single user class potentially have the same information requirements, they may diverge with respect to their characteristics and usability requirements. For example, within the user class `Enrolled Students` we may distinguish between *local students* and *exchange students.* They have the same information requirements (detailed information on courses) but have different characteristics and usability requirements. Local students are young (between 18 and 28), are familiar with the university jargon, the university rules and customs. They have a good level

of experience with the WWW. They prefer the local language for communication, but have in general a good understanding of English. On the other hand, all communication with exchange students is done in English. We may not presume that they are familiar with the university jargon and customs, or with the WWW.

To support different characteristics and usability requirements within a single user class, we use *perspectives*. A perspective is a kind of user subclass. We define a perspective as all users in a user class with the same characteristics and usability requirements. For the user class `Enrolled Students` we may distinguish two perspectives: `Local Students` and `Exchange Students`.

The *Conceptual Design* phase also consists of two sub-phases: the Object Modeling and the Navigational Design. During *Object Modeling* the information requirements of the different user classes and their perspectives are formally described in a number of conceptual schemes. How this is done is described in section 3. During the *Navigational Design* we described how the different users will be able to navigate through the website. For each perspective a separated *navigation track* will be designed. It is precisely the approach taken in the Object Modeling and Navigational Design based on user classes and perspectives that constitute the user-centered approach of WSDM and its departure from purely classic information system modeling.

In the *Implementation Design* we essentially design the 'look and feel' of the website. The aim is to create a consistent, pleasing and efficient look and feel for the conceptual design made in the previous phase. If information provided by the website will be maintained by a database then the implementation design phase will also include the logical design of this database. The last phase, *Implementation*, is the actual realization of the website using the chosen implementation environment, e.g. HTML.

3 The Conceptual Design of a Website.

During the User Modeling phase, the requirements and the characteristics of the users are identified and different user classes and perspectives are recognized. The aim of the Conceptual Design phase is to turn these requirements into a high level, formal description which can be used later on to generate (automatically or semi-automatically) effective websites.

During Conceptual Design, we concentrate on the conceptual 'what and how' rather than on the visual 'what and how'. This means that like in database design we describe what kind of information will be presented (object types and relationships; the conceptual 'what'), but unlike in database design we also describe how it will be able to navigate through the information (the conceptual 'how'). This is needed because navigating through the information space is an essential characteristic of websites. If the navigation is not (well) designed or not adapted to the target audience, serious usability problems occur. The conceptual 'what' is covered by the Object Modeling step, the conceptual 'how' by the Navigational Design.

3.1 Object Modeling in a User-Centered Approach

In WSDM, the Conceptual Object Modeling results in several different conceptual schemes, rather then in a single one as in classical database design. This is because we have opted for a user-centered approach. In a data-driven approach, as used in database design, the starting point of a conceptual design is the information available in the organization: designers first model the application domain, and subsequently they associate information with each class of users (e.g. by means of views or external schemes). However, the data and the way it is organized in the application domain may not reflect the user's requirements. A good example of such a mismatch can be found in the current website of our university[1]. The structure of this website completely reflects the internal organizing structure of our university. This structure is completely irrelevant for, and unknown to most users of this site. As an example, if you want to look at the web for the products offered by the Computer-shop of our university (called PC-shop) you must know that the PC-shop is part of the Computer Center (actually it is one of the 'External Services' of the Computer Center), which itself is a 'Service Department' of the University. You will not find it under 'Facilities' like the Restaurant, the Copy-shop or the Branch Bank.

In our user-centered approach we start by modeling the information requirements of the different types of users (user classes). Note that we make a distinction between a *user-centered* approach and a *user-driven* approach. In a user-driven approach the users are actively involved in the design process, e.g. through interviews, during scenario analysis, prototyping and evaluation. This is not possible for kiosk websites on the internet because most of the users are unknown and cannot be interviewed in advance or be involved in the design process. However, we can fairly well identify the different types of users and investigate their requirements. After all, the main goal of a kiosk site is to provide information. Therefore, for each user class a conceptual schema is developed expressing the information needs of that type of user. We call these conceptual schemes *user object models.* Like an "ordinary" conceptual schema, a user object model (UOM) is expressed in terms of the business objects of the organization.

[1] http://www.kub.nl/ (in Dutch).

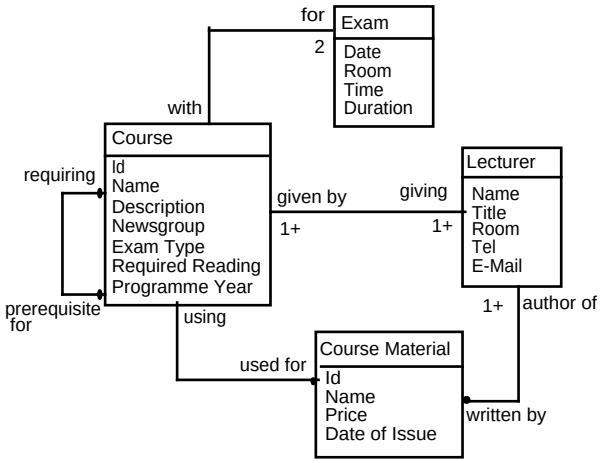

Fig. 2: User object model for `Enrolled Students`

In [5] we explain how a user object model is constructed from the information requirements expressed in a user class description. For each requirement a so-called *object chunk* is constructed. Next, the object chunks of one user class are merged into a single model.

In conceptual modeling in general, object models describe the different object types (OTs), the relationships between these OTs, and rules or constraints. OO models also describe behavior. For our purpose (modeling kiosk websites), modeling behavior is not (yet) needed. The traditional conceptual modeling methods like E-R [1], the Object-Role Model [8], [16], [2], or "true" OO methods like OMT [12] are therefore all suitable. Figure 2 shows the UOM (in OMT notation) developed for the user class `Enrolled Students` of our university department example.

3.2 Object Type Variants

As explained, the same user class may include different perspectives expressing different usability requirements and characteristics. It is possible that this also results in slightly different information requirements. In WSDM, we model this by means of *variants* for OTs. A variant of some OT corresponds largely with the original OT but has some small differences (*variations*). Consider as an example the OT `Course` for the user class `Enrolled Students`. See figure 2 for a graphical representation. About a course, enrolled students in general need the following information: the identification number of the course, the name of the course, a description of the content of the course, the prerequisites for the course, specification of the required reading, the type of exam of the course, the name of the newsgroup of the course and the programme year in which the course may be followed.

However, for the subgroup (perspective) `Local Students` we want to offer this information in the local language, while for the subgroup `Exchange Students`

the information must be provided in English. Also the programme year is not relevant for exchange students and (the actual value of) the prerequisites, the required reading and the exam type may differ between exchange students and local students. Indeed, local students may have required reading written in the local language while for the exchange students the required reading must be written in English. In implementation terms, this means that for most (but not all) attributes of the OT `Course` we will need to maintain two *variants*; an English one and a local language one. The recognition of these differences is essential for a user-centered approach and therefore they should be modeled in an early phase. Some people may argue that the language is a representation issue and therefore it should not be considered in the conceptual phase but left to the implementation design. However, in this example, the language issue is an important user requirement which also influences the actual information that will be provided. If we do not recognize this during conceptual design, the information provided for a course, except for the language, would be the same for local students and exchange students.

To model the differences we introduce two variants for the OT `Course`: `Course/Local Students` and `Course/Exchange Students`.

`Course/Local Students`:

- the identification number of the course;
- the local language name of the course;
- a description of the content of the course in the local language;
- the prerequisites for the course for the local students in the local language;
- the specification of the required reading for the local students in the local language;
- the type of exam of the course for the local students in the local language;
- the name of the newsgroup of the course;
- the programme year in which the course may be followed.

`Course/Exchange-Students`:

- the identification number of the course;
- the English name of the course;
- a description of the content of the course in English;
- the prerequisites for the course for the exchange students in English;
- the specification of the required reading for the exchange students in English;
- the type of exam of the course for the exchange students in English ;
- the name of the newsgroup of the course.

Graphically, we use a parent-child notation to represent variants (see figure 3). The parent OT is variant independent, each child OT is a *variant* of the parent OT. The name of a variant OT is composed of the name of the parent OT followed by the variant identification, e.g. `Course/Exchange Students`.

A variant OT can have less attributes that its parent OT. Semantically, this means that the omitted attributes are not meaningful for the variant. E.g. the `Programme Year` attribute is omitted in the `Course/Exchange Students` because it is not meaningful for exchange students. Note that in this respect variants are clearly differ-

ent from the notion of subtype. Subtypes can in general not be used to model variants.

Also attributes may have variants. `Name/English Name` and `Name/Dutch Name` are two variants of the attribute `Name`. To relate the attribute variant to the original attribute in the parent OT, the name of the original attribute is preceding the name of the attribute variant. In some cases, it is possible that the original attribute never will have an own value, but only serves as a means to indicate that the underlying variant attributes have the same semantics. This is comparable to the concept of abstract OT in object-oriented modeling. By analogy, we call this an *abstract attribute*. In the OT `Course`, the attributes `Name`, `Description`, `Exam Type` and `Required Reading` are abstract attributes.

A variant OT cannot include or refer to attributes which are not defined in the parent OT. This is to prohibit addition of completely new information (attributes) to a variant, in which case it will not be a variant anymore.

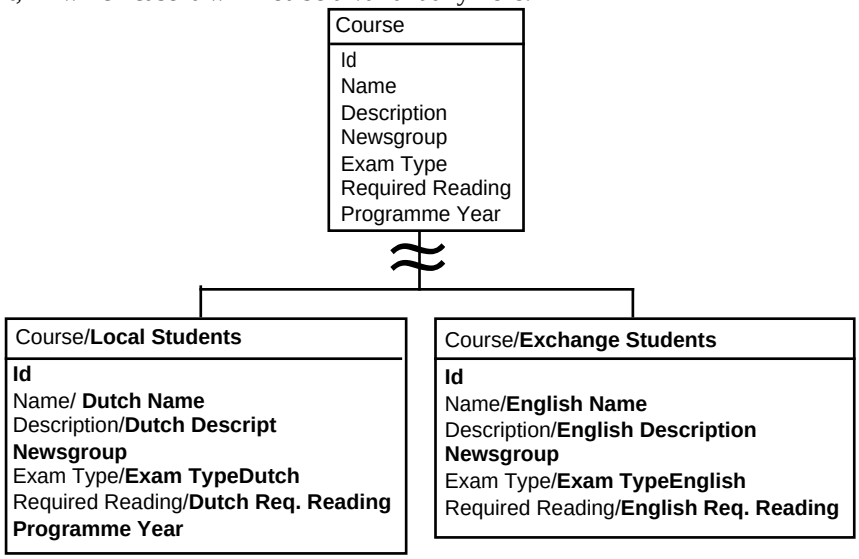

Fig. 3: Variants for the OT `Course`

In WSDM, information differences between the perspectives of a single user class are modeled by means of OT variants. For each OT in the UOM of a user class, and for each perspective of this user class, a variant may be defined to reflect the possible information differences. To derive the conceptual schema for a perspective, called a *perspective object model* (POM), it suffices to replace the OTs in the corresponding UOM by the corresponding perspective variants. If an OT has no variant for the perspective, the OT is kept as it is.

3.3 Linking the Conceptual Models

As explained, the Object Modeling starts by building the user object models, one for each user class. Subsequently, these models are refined using perspective variants to derive the perspective object models (if a user class has no perspectives then the user object model acts as perspective object model). In what follows we call OTs from a perspective object model *perspective OTs* (POTs).

Perspective object models of a single user class are related by means of their user object model. However, the different user object models are (still) independent. This is not desirable, especially not when several user classes share the same information. It would result in an uncontrollable redundancy. Therefore, the different user object models must be related. To do this we use an overall object model, the *business object model* (BOM). This model is a conceptual description of the information (business objects) available in the organization. It is independent of any type of user. Such a business object model may already have been developed for the organization or the application domain. If not, or if it is not available in a shape usable for our purpose, it must be (re-)developed. The classical information analysis methods mentioned earlier may be used for this. For this model, a data-driven approach is not a problem, on the contrary: it is preferred.

Next, the different user object models are expressed as (possibly complex) views on the BOM. Note that it is possible that during this step it turns out that the (existing) BOM is incomplete. This is the case if information modeled in a user object model cannot be expressed as information modeled in the BOM. In such a case it is necessary to re-engineer the BOM. Figure 4 illustrates how the different types of conceptual schemes developed during Object Modeling relate to each other.

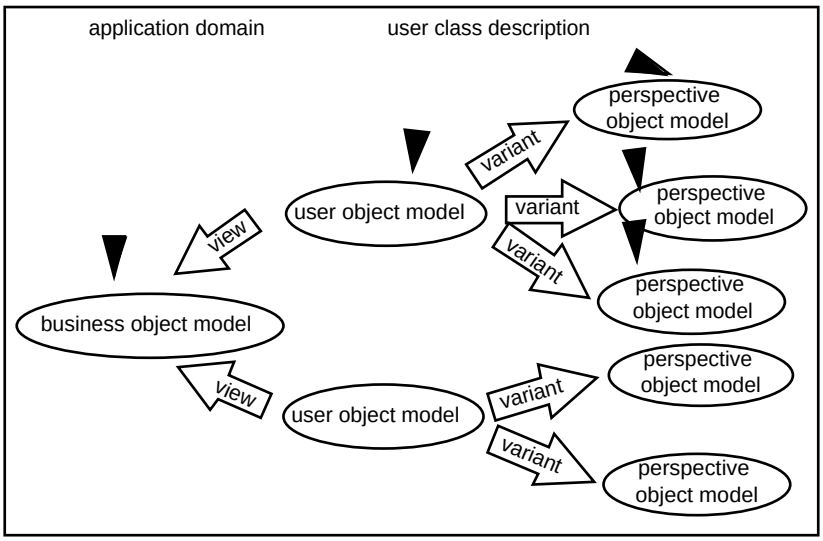

Fig. 4: Relationship between the different types of object models

3.4 Navigational Design

Once the Object Modeling is done, a *conceptual navigation model* is constructed. The navigation model expresses how the different user types will be able to navigate through the available information. Navigational models are usually described in terms of *components* and *links*. We distinguish between *information components*, *navigation components* and *external components* (see figure 5). Information components represent information. An information component may be linked to other components to allow navigation. A navigation component can be seen as a grouping of links, and so contains no real information but allows the user to navigate. An external component is actually a reference to a component in another site.

Following our user-centered approach, we design an independent *navigation track* for each perspective. To derive the navigation model, it is sufficient to connect the different navigation tracks by a navigation component. In a nutshell, a navigation track for a perspective may be constructed as follows: information components are derived from the POTs and links are used to represent the relationships between POTs. This forms the *information layer* of the navigation track. Next, a *navigation layer*, built up of navigation components, is designed to provide different access paths to the information components in the information layer. The top of a navigation track is a single navigation component which provides access to the different navigation components in the navigational layer. When the different navigation tracks are composed, these top level components form the *context layer* of the navigation model. Figure 6 shows the navigation track for the POM `Exchange Students`. Figure 7 shows how the different navigation tracks are composed to make up the navigation model.

Fig. 5: Graphical representation of the navigation model concepts

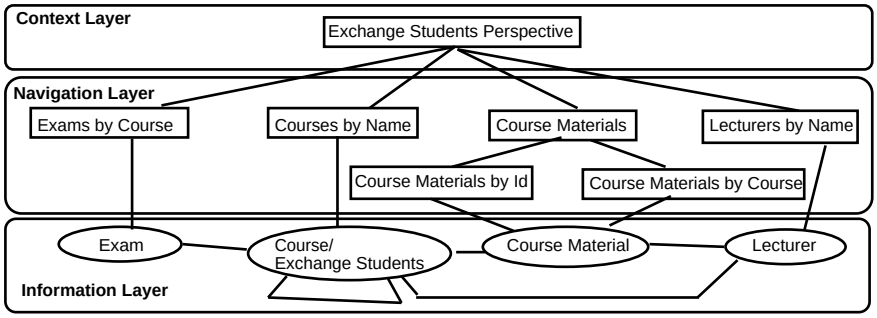

Fig. 6. Navigation track for the perspective `Exchange Students`

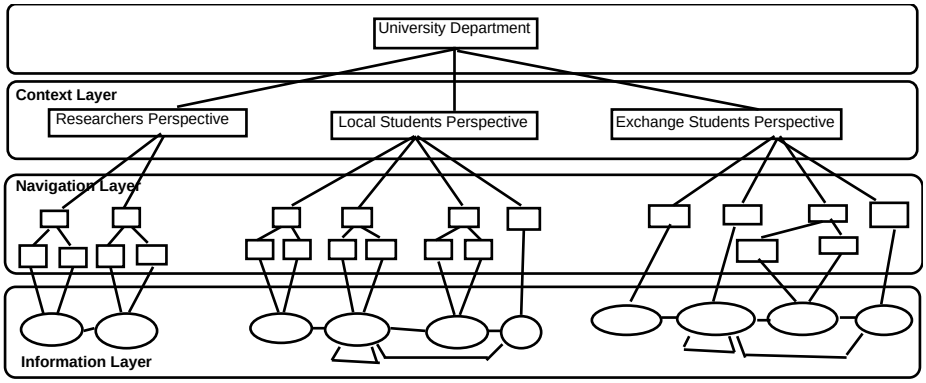

Fig. 7: Composition of navigation tracks into a navigation model

In the rest of this paper we will use the term *conceptual schema* (CS) to denote the result of the Conceptual Design: the UOMs, POMs, BOM and the navigation model.

4 Roles of the CS in the Website Life Cycle

The life cycle of a website contains many of the phases of a traditional Information System (IS) life cycle, such as planning, analysis, design and implementation, but also phases which are specific for web systems. The development process of a website is more open-ended because a website is often not as permanently fixed as a traditional IS. Designing a website is an ongoing process. Maintenance includes activities such as monitoring new technologies, monitoring users, and adapting the website accordingly. It is a continuous process of improvement. To emphasize this distinction, the maintenance phase is sometimes called *Innovation* [3]. The typical Installation phase is replaced by a *Promotion* phase in which the existence of the website is made public (by publicity, references from other websites, etc.).

In this section we explain what role the CS may play in the Implementation phase, the Promotion phase and the Innovation phase, and we explain how the CS may be exploited even more inside the website.

During Implementation Design, the 'look and feel' of the website is developed. Starting point for this is the navigation model. Through use of graphical design principles and visual communication techniques, taking into account the characteristics of the different perspectives, the navigation model will be translated into a *presentation model.* (content of pages and their layout). Again, this is in some respect similar to the mapping of a conceptual data schema into a logical data schema (e.g. a relational one). Indeed, during Implementation Design one may decide to group information components and links (from the navigational model) together and to present them to the user as single packages of information. (In fact, we are developing algorithms and tools to support this.)

Separating the conceptual and the implementation design for websites has the same advantage as in database design. It offers the flexibility needed for designing large websites. As explained, designing a website is an continuous process. By separating the conceptual design from the implementation design, we yield the flexibility required to support this incremental and evolving design process. Different implementation designs may be built (e.g. as prototypes) and evaluated. Changes and additions to the content are localized to the conceptual level, and the impact on the implementation design can easily be traced. Adding a new user class only involves adding a new UOM with its associated perspectives and navigation tracks. Changes to the presentation only influence the implementation design. The actual implementation can be automated using available tools and environments for assisting in e.g. HTML implementations.

Because different perspectives may offer the same information (possibly presented differently)[2], we need to provide means to maintain this information and keep it consistent. The obvious way of doing this is by maintaining the underlying information (or parts of it) in a database. This need not be a full-fledged database, but in any case a single storage place for information shared between different perspectives. As all information presented in the website is ultimately related to the business object model (BOM) (by means of the POMs and UOMs), this BOM provides the conceptual schema for the underlying database. From this BOM a logical database schema is then generated (using appropriate database development tools) or manually built. The queries needed to extract the information for building the pages can then be derived from the POMs because they are already expressed as views on the BOM.

To reduce the lost-in-hyperspace syndrome, many sites contain an index page or site map. This index page or site map gives a (hierarchical) overview of the website and provides a central point for the user to locate a page in the website. We may consider instead to replace it by a representation of (parts of) the conceptual schema which is much richer in information than an index page. Each navigation track could contain a suitable representation of its corresponding POM. This will not only allow the user to locate information directly but will also help him/her to build a mental model of the site and ultimately provide an on-line repository of meta-information which may be queried. The availability of the CS literally 'in-site' may also be exploited by the many different types of search engines to enhance their search effectiveness. In this way promotion benefits as well.

5 Conclusions

In this paper we have explained the need for a conceptual design phase in website design similar to the conceptual design phase in database systems. Based on early

[2] Note that this does not lead to the redundancy mentioned as a usability problem in the introduction, because a user only follows one perspective and within one perspective, redundancy is avoided.

experience with our method WSDM, we argued that a user-centered approach is more appropriate for websites than the traditional data-centered approach used for database design. As a consequence, the conceptual schema of a website cannot be seen as a single schema but as a collection of schemes; each user perspective has its own conceptual schema. To relate the different schemes and to control the redundancy possibly introduced in this way, a business object model is used. To capture variations between perspectives schemes, so-called OT variants are introduced. Because navigation is an essential characteristic of websites, the conceptual schema also includes a navigation model which describes how users will be able to navigate through the website, being a collection of navigation tracks, one for each user perspective.

We have also shown that separation of the conceptual and the implementation design for websites has the same advantages as in database design. As for database design it is possible to deploy the conceptual schema technology in the future automation of site creation and upkeep. CASE-type tools generating well-structured websites from user requirements and business domain models are the next logical step. In addition, (parts of) the conceptual schema may be represented and queried inside the website to reduce the lost-in-hyperspace syndrome. New generations of search engines may exploit such additional structural knowledge, e.g. by allowing them to interpret the meta-information present in a website, and act on its semantics.

Acknowledgments: Many thanks go to Wim Goedefroy and Robert Meersman for the interesting discussions on and the contributions to this research work.

References

1. P.P. Chen, The Entity-Relationship Model: Towards a Unified View of Data, ACM Transactions on Database Systems, Vol 1 no 1, 1976, 471-522.
2. O. M.F. De Troyer, A formalization of the Binary Object-Role Model based on Logic. In: Data & Knowledge Engineering 19, pp. 1-37, 1996.
3. J. December, M. Ginsberg, HTML & CGI Unleased, Sams.net Publishing, 1995.
4. O.M.F. De Troyer, C.J. Leune, WSDM: a User-Centered Design Method for Web Sites, in proceedings of the WWW7 Conference, Brisbane, April 1997.
5. W. Goedefroy, R. Meersman, O. De Troyer, UR-WSDM: Adding User Requirement Granularity to Model Web Based Information Systems. Proceedings of 1st Workshop on Hypermedia Development, Pittsburgh, USA, June 20-24, 1998.
6. F. Garzotto, P. Paolini, L. Mainetti, Navigation patterns in hypermedia databases, Proceedings of the 26th Hawaii International Conference on System Science, IEEE Computer Society Press, pp. 370-379, 1993.
7. F. Garzotto, P. Paolini, D. Schwabe, HDM - A Model-Based Approach to Hypertext Application Design, ACM Transactions on Information Systems, Vol 11, No 1, pp. 1-26, 1993.
8. T. Halpin, Conceptual Schema and Relational Database Design, second edition, Prentice Hall Australia, 1995.
9. T. Isakowitz, E. A. Stohr, P. Balasubramanian, RMM: A Methodology for Structured Hypermedia Design, Communications of the ACM, Vol 38, No 8, pp. 34-43, 1995.
10. H. Lee, C Lee, C. Yoo, A Scenario-Based Object-Oriented Methodology for Developing Hypermedia Information Systems, Proc. of HICSS '98.

11. H. Maurer, Hyper-G - The Next Generation Web Solution, Addison-Wesley 1996.
12. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, Object Oriented Modeling and Design, Prentice Hall Inc., 1991.
13. D. Schwabe, G. Rossi, The Object-Oriented Hypermedia Design Model, Communications of the ACM, Vol 38, No 8, 1995.
14. D. Schwabe, G. Rossi, S.D.J. Barbosa, Systematic Hypermedia Application Design with OOHDM, http://www.cs.unc.edu/barman/HT96/P52/section1.html.
15. K. Takahashi, E. Liang, Analysis and Design of Web-based Information Systems, Sixth International World Wide Web Conference, 1997, http://www6.nttlabs.com/papers/PAPER245/Paper245.html.
16. J.J. Wintraecken, The NIAM Information Analysis Method - Theory and Practice, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
17. M. Bichler, S. Nusser, W3DT - The Structural Way of Developing WWW-sites, Proceedings of ECIS'96, 1996.