Updated version published in: WWW Journal, Volume 3, Issue 1, Baltzer Science Publishers, 2000.

Conceptual Modelling of Web Sites for End-Users.

Olga De Troyer
Vrije Universiteit Brussel
WISE Research group, Department of Computer Science
Pleinlaan 2
B-1050 Brussel, Belgium
Olga.DeTroyer@vub.ac.be

Tom Decruyenaere
Volvo IT Belgium
Eksaardserijweg 188
B-9041 Gent, Belgium
Tom.Decruyenaere@volvo.be

Abstract

Internet and the WWW more and more play an important role in our information society. It is now one of the major sources of information in every rank of our society. The overwhelming accessibility to data, on a global scale, does not necessarily translate to widespread utility of data. We often find that we are drowning in data, with few tools to help managing relevant data for our various activities. In this paper, we argue that the WWW and its end-users could benefit from the existence of a *conceptual web site schema*. We propose such a conceptual web site schema that describes what information is available in a web site and how this information is structured into pages and links. To allow to communicate this information through the web, we developed an XML DTD for this conceptual web site schema. We also illustrate the feasibility of the approach by a simple application program developed using XML DOM.

1 Introduction

The WWW is a gigantic collection of information spread over millions of sites located on thousands of servers all over the world. This pile of data is connected together by hyperlinks making it possible to go from one web site to another, jumping from one subject to the next one. This makes the WWW the most dynamic accumulation of information but perhaps also the most disordered one. The WWW creation relies on the simple concept of Hypertext, currently realised using the HTML language. This is one of the main reasons why it became as big as it is today. But because HTML doesn’t have many possibilities in creating structured data presentations it also makes the WWW hard to explore.

The WWW has evolved from a private experiment for some scientists towards a medium that reaches millions of people all over the world. Companies have taken over the techniques and use them as a professional tool to distribute their information towards customers. Today almost anyone who is a bit familiar with the use of a PC can go on the WWW and use a browser program to look for information. To help people in their search for information search engines (e.g. YAHOO, ALTAVISTA) have been developed. It is hard to find a subject that won’t give you a result on those search engines. The problem today is more of the kind that you will be overwhelmed with sites were your subject is mentioned and you will not see the wood for the trees. It has become very difficult to use such an enormous collection of data in an efficient way.

The overwhelming accessibility to data does not necessarily translate to widespread utility of data. We often find that we are drowning in data. The lost-in-hyper-space syndrome is a phenomenon that occurs when surfing on the WWW. By frequently use of hyperlinks in web sites people loose track of where they are and what they are looking for. You can compare it with finding the way in a strange city without having a city map at your disposal. The availability of the structure and a survey of a web site become more and more important to help end-users to deal with their queries in an efficient way.

The objective of this paper is to provide means to create an overview of a web site that permits end-users to see in an efficient way what information is available inside a web site and how this information is structured and reachable. We will appeal to the technique of *Conceptual Schemas* to describe the type of information present in a web site and how this information is structured.

We first define the concept of a Conceptual Schema (CS) for a web site. We do this by means of a conceptual “meta” schema. This conceptual meta schema defines the concepts and their relationships that will be used to define the conceptual schemas of whatever instance of a web site. To model the conceptual meta schema we use the Binary Role (BR) Model [Wintraecken 1990] that allows for a graphical representation. Next we translate this BR-schema into an XML document type definition (DTD), which is a more suitable representation technique for the WWW. Then, this DTD can be used to represent the conceptual schema of any web site as an XML document. Next, web tools may be developed to explore the knowledge available in such a conceptual schema (XML) document, e.g. to give a visual representation of (the CS of) the web site to the end-user.

The rest of this paper is structured as follows: In section 2 we describe current techniques and related work to reveal the information in a web site. In section 3 we will explain our concept of conceptual web site schema and the objectives of such a schema. In sections 4, a more formal description is given by means of the conceptual meta schema. Section 5 gives an XML Document Type Definition (DTD) for a conceptual web site schema. The conceptual web site schema and the corresponding DTD are illustrated and explained into more detail by means of a case study in section 6. In section 7 a simple example application is given to illustrate the feasibility of our approach. Section 8 draws conclusions.

2 Providing structure and overview of a web site

Today’s most used technique to present the structure and the information available in a web site to the end-user is by means of a (hierarchical) menu. A menu (mostly on the left side of the page) is used to depict the subjects (connected with links to the related pages) accessible from that page. This only gives some minor information about the information available and about the structure of the web site. Using this technique, it is important to structure the menu’s well, making sure that each label used as a menu item is a good representation of what information can be found underneath it.

There are also some tools that can help users to expose the structure of a web site they want to explore. *Site maps* are some kind of visual description of a web site generated by Java applications. Sitemap [Sit] and Mapuccino [IBM 2000] are two examples. Through a WebCrawler, Sitemap first traverses every link of the web site, collects statistical data, and indexes all the words and pages of the site. Based on the statistical and the indexing, it converts each page of the site into a vector, and uses these vectors to train a neural network. As the outcome, the trained neural network presents the site in an organized map: subject areas are identified and labeled; their sizes and locations are determined by relationships among the subjects and their occurrence and co-occurrence frequencies. Links are clustered and located within their respective subject areas, represented by colored dots. Some patterns of the site are clearly revealed. To help users interact with the map, Sitemap provides various interactive tools. For example, areas can be labeled in more/less details through adjusting a scroll bar; links can be selected through clicking or dragging; contents of any selected links can be shown in a separate window, etc.

Mapuccino dynamically constructs visual maps of web sites and enables you to capture and view the overall structure of any web site, including links to other sites, and to navigate visually through the contents of a site. The site maps can be displayed in a variety of ways.

Topic Maps [ISO/IEC JTC 1/SC34 1999] are proposed by the SGML community as an ISO standard to provide multiple document indices and glossaries. The purpose of a Topic Map is to highlight individual topics that appear in a set of documents and also to establish relationships among topics. Topic Maps are intended to support navigation like a general index or glossary in a conventional book, but extended to multiple documents.

Structured Maps [Delcambre et al. 1996] are based on Topic Maps but uses the terms entity and relationship instead of topic and topic relation. Structured Maps can model entity types like Student and Course and relationships like Student-following-Course. Instances of the entity types are connected to elements (often fragments of some larger source) in the underlying universe of information. This gives a three-level model: the Structured Map definition defining the entity types and relationships, the Structured Map instance i.e. the populated instance of the Structured Map definition, and the underlying universe of information. In this way Structured Maps can be used as a navigational guide for the underlying information universe. Note that Structured Maps are not limited to the structure of any particular web like e.g. WWW. But we may consider a web site as being the underlying universe of information.

Resource Description Framework (RDF) [W3C RDF WG 2000] is an activity of the World Wide Web Consortium (W3C) [W3C 2000]. RDF is a declarative language and provides a standard way for using XML to represent metadata in the form of properties and relationships of items on the WWW. Such items, known as resources, can be almost anything, provided it has a Web address. RDF provides a framework in which independent communities can develop vocabularies that suit their specific needs and share vocabularies with other communities. The descriptions of these vocabularies are called RDF Schemas. A schema defines the meaning, characteristics, and relationships of a set of properties. The RDF Schema specification provides a mechanism for defining the vocabulary needed for web site maps and the content of web pages.

3 A conceptual web site schema for end-users

What we are seeking for is a means to describe the content and the structure of a web site in a general and formal way, such that it can be used by browsers to display the structure of the web site and allow end-users to see at a glance what kind of information the web site is offering. Also search engines should be able to explore this information.

A common way in databases to describe the type of information they are dealing with and the way it is structured is a *conceptual schema*. The conceptual schema of a database gives the type of objects (often also called entities) the database contains and the relationships between these object types. As an example, consider a university database. Typical object types are “Student”, “Course”, and “Lecturer”. The relationships between these object types are e.g. “Student follows a Course” and “Lecturer gives a Course”. The conceptual schema of a database further describes the properties of the object types (called attributes, e.g. a “Student” has a “Name” and an “Administration Number”) and the constraints that apply to the attributes and the relationships (e.g. “each Student has a unique Administration Number”)

When we consider a web site as being a forum to deliver information, then there are some similarities with a database. Therefore we may consider using the technique of conceptual schemas from database theory to describe the content and structure of a web site. However a web site is not a database. A web site may contain collections of structured data, but usually it also contains so-called ‘unstructured’ data. In addition the organisation of a web site is different from that of a typical database. A web site is organised into pages and links between pages and pieces of information. This implies that the basic building blocks of a conceptual schema for a database, being object types, relationships and constraints, are not necessarily suitable and probably not sufficient to build a conceptual schema of a web site. Therefore, we first have to establish the basic concepts needed to specify the conceptual schema of a web site. We have to find out what exactly of a web site we want to describe with a conceptual schema. E.g. do we merely describe the (type of) information contained in the web site, or do we describe the page and link structure of the web site or both? In addition, there is the question of what level of abstraction we are looking for. A conceptual schema for a database does not deal with the individual instances contained in the database. However, because in a web site the distinction between instance and type is not as clear as in a database, omitting all information about instances in the conceptual web site schema may not be a good idea. On the other hand including all instances may also not be desirable. E.g. for a university department web site it may be useful to include the names of all lecturers in the conceptual schema of the web site. (This would allow to give the end-user an overview of all lecturers for whom information can be found on the web site.) However including all names of all students may not be desirable because the number of students is too high.

3.1 The objectives of a conceptual web site schema

In database theory, the conceptual schema concentrates on the conceptual issues of the allowable contents and the meaning of the database. Implementation issues such as the grouping into records and the available access paths are not described in the conceptual schema. These are part of the database schema. If we would follow this philosophy for web sites, it would only be possible to derive from the conceptual schema what kind of information is available in the web site. We would not be able to derive where the information can be found in the web site. For databases this is not a problem because a database is used through an application program that hides the database structure from the end-user. For a web site the only existing application is the web browser, which does not hide the pages and links from the user. On the contrary, pages and links are the basic manipulation concepts to be used in a web browser.

Also our conceptual web site schema has a different goal. In the field of databases the conceptual schema is used as an intermediate step in the design of the database. Usually, it is not given to the end-user. The conceptual schema we are seeking for is not a priori intended as a design aid but as an aid for end-user to explore the web site. (However, it is possible that such an end-user conceptual schema can also play a role in a web site design method.)

Therefore to fulfil the original goal, to allow users to see at a glance where and what kind of information can be found in a web site, the conceptual schema of a web site for rnd-users should also describe where the information can be found in the web site. To realise this, we propose that the conceptual schema of a web site be composed of three parts. One part of the schema will describe what kind of information is available in the web site (called the *content schema*), a second part will describe the page and link structure of the web site (called the *structure schema*) and the third part will describe where the information can be found in the web site. This last part describes how the content is mapped on the structure of the web site. Therefore this schema part is called the *mapping schema.* We have opted to clearly identify those different parts rather than intertwine all the information. This will offer more flexibility not only towards application programs that will populate or exploit the content of a conceptual web schema but also towards the future as web technology changes quickly. E.g. if new technology becomes available that allow for a richer web structure, only the structure schema and (parts of) the mapping schema are affected.

In the next section, we describe each of these sub-schemas into more detail by means of a conceptual meta schema.

4 The conceptual meta schema

In database theory, the set of basic concepts used to describe a conceptual schema is usually described by the same technique, i.e. by a conceptual schema. Because this is a conceptual schema of conceptual schemas, this schema is called the *Conceptual Meta Schema.* We will also apply this principle to define the basic concepts of our conceptual web site schema. To represent the conceptual meta schema of a web site we have used an information modelling technique from the database community, the Binary Role Modelling (BRM) approach ([Halpin 1995;

Wintraecken 1990; De Troyer 1996]), but any other information modelling technique such as E-R [Chen 1976], or OMT [Rumbaugh et al. 1991] is also suitable. BRM views the world as objects playing roles. The basic building blocks of the BR Model are Object Types (OT) (graphically represented as circles) and binary relationships composed of two roles (graphically represented as a rectangle composed into two boxes, each box connects with a line to the corresponding OT). Identifiers (represented by arrows on roles) are used to indicate one-to-one, one-to-many or many-to-many relationships. A mandatory role is indicated by a dot.

To keep it simple we will only present the kernel of the conceptual meta schema and deliberated introduced some simplification. We also have omitted most attributes for the different object types.

As already explained our conceptual web site schema is composed of 3 sub-schemas. As a consequence, the conceptual meta schema can also be divided into three parts:

- The *Content Meta Schema*: describing the basic concepts of a content schema.
- The *Structure Meta Schema*: describing the basic concepts of a structure schema.
- The *Mapping Meta Schema*: describing the basic concepts of a mapping schema.

Note that these three schemas together form one single schema. Some OTs may appear in more than one sub-schema. Therefore, we agree that OTs with the same name represent the same OT.

4.1 The content meta schema.

In this part of the meta schema we define the different conceptual ‘elements’ and their (conceptual) relations to each other needed to model the content of a web site.

As already indicated, and opposed to the conceptual schema of a database, the conceptual schema of a web site will also allow containing information about instances of object types and one-of-a-kind objects. This is necessary because not all web sites (completely) deal with collections of data that can be classified into object types and relationship between object types. E.g. a company’s web site usually contains information like the address, the mission statement of the company, an organization diagram, a route description, etc. These are single objects not belonging to any object type. Type level modeling is not appropriate for this. We better model this information as objects and relations between objects.

As already explained it may also be useful to include some instances of object types explicitly in the conceptual schema. E.g. the conceptual schema of a web site on painters is more effective from an end-user's point of view if it also includes the names of the painters described in the web site.

Therefore, to model the information available in a web site, we not only needs object types and relation types, but also object type instances, relation type instances, (one-of-a-kind) objects and relations. These are the building blocks of the content schema. They are represented as object types (OTs) in the Content Meta Schema. A graphical representation of the Content Meta Schema is given in figure 1. We will use italic to refer to names used in the meta schema.

In the Content Meta Schema *ObjectType* and *RelationType* are modeled as subtypes of *ConceptType*. Similar *Object* and *Relation* are subtypes of *Concept*. The reason for having the supertypes *ConceptType* and *Concept* will become clear in the Mapping Meta Schema. *ObjectInstance* is a subtype of *Object*. Object instances are objects that are related to the object type of which they are an instance. Note that not all objects need to be an instance of some object type. There is a similar distinction between *Relation* and *RelationInstance*. To clarify the differences we can use the following examples: 'Student' is an example of a *ConceptType*, more in particular an *ObjectType*, whereas 'the student Tom' can be considered as an example of a *Concept* and more in particular an *ObjectInstance* because it is an instance of the *ObjectType* 'Student'.

Like in ORM objects may play roles creating relations between each other. Therefore relation (type)s exist out of role (type)s. The object type 'Student' has a relation type with an object type 'Course' because the first plays a role on the second and vice versa: a Student follows a Course and a Course is followed by a Student. A *RoleType* between two *ObjectTypes* creates a *RelationType*. When we become specific and indicate that the Student 'Tom' follows the Course 'Software Engineering' we talk about a *RelationInstance* between two *ObjectInstances.* One-of-a-kind objects may also play roles in relations. E.g. "the Route Description - describes how to reach – the Address of the Company” is an example of a *Relation* between two *Objects*, “the Student Tom - is the representative of the students for - the Department Board” is an example of a *Relation* between an *ObjectInstance* and an *Object*.

Objects may be structured (*StructuredObject*), constructed out of other objects or just simply elementary (*ElementaryObject*). E.g. a ‘description of a field of study’ may contain general information like a ‘general description of the field’ and ‘entry requirements’, as well as the ‘study program’ that itself will be composed of ‘descriptions of program years’. We also provide a basic classification of objects into *Text*, *Graphic* and *Video*. More detailed classifications are of course possible.

4.2 The structure meta schema.

In this part of the meta schema we define the different conceptual ‘elements’ and their relations to each other needed to model (at a conceptual level) the structure of a web site.

From the point of view of an end-user, a web site is built up of pages and links. Therefore ‘page’ and ‘link’ are two of the conceptual building blocks of a structure schema. So, *Page* and *Link* are two OTs in the Structure Meta Schema. See figure 2 for a graphical representation of the Structure Meta Schema. Links are classified as *OutPageLink* (from one page to a different page), *InPageLink* (a link within a page) or *OutsideLink* (a link to a page not belonging to the web site). A page not belonging to the web site is an *ExternalPage*.

More and more there is a trend to have a uniform layout for pages containing the same type of information. E.g. all pages on courses of a university department may contain the same type of information presented in the same way. Usually the pages are generated using a kind of template from information stored in a database. Therefore, we also have introduced the concept of *PageProtoType* and *LinkProtoType*. A page prototype allows defining a generic structure for a collection of pages. This makes it possible to define a page as an ‘instance’ of such a page prototype. This means that the page must be conform the prototype. E.g. we can define a page prototype for the object type Course. Each course could than have a page that contains specific information concerning this course in a way specified in the page prototype for Course. (Note that the structure schema does not contain the relationship between the page prototype and the object type, this will be specified in the mapping schema). Page prototypes may specify links; e.g. the Course page prototype may specify that each course page should contain a link to the page of its lecturer. This we call a *LinkProtoType*. The actual instantiation of such a link prototype is called a *LinkInstance*. A *LinkInstance* is considered to be a special kind of *Link.* Similar a *PageInstance* is a special kind of *Page*. Note that not every page or link should have a prototype.

Pages and page prototypes can be classified into *Static, Tailored* or *Dynamic.* The information on a static page (instance) is the same for all users and will not change during navigation. The content of a tailored page may be different for different users. The difference can be based on the identity of the user or the value of one or more input fields. E.g. for each user (student) a page may only show the courses for which the student is enrolled. On another page it may be possible to select a subset of available course by means of some criteria, e.g. the number of study points. These type of pages we call tailored pages. Their corresponding page prototypes are tailored page prototypes. In a dynamic pages the content may change depending on the navigation, e.g. a shopping basket page.

4.3 The mapping meta schema.

The purpose of the mapping schema is to describe where the content (in the form of object (type)s, relation (type)s can be found in the web site, i.e. on which pages and following which links. The mapping schema must allow to indicate how the conceptual content of a web site is reflected on the structure. More specific, it indicates what concept (type)s can be found on what page (prototype)s. Therefore the Mapping Meta Schema contains the following relationship (see figure 3 for a graphical representation of the Mapping Meta Schema):

- A relationship between *Concept* and *Page* (*Concept on_page/page_contains Page*) to indicate the pages on which a particular concept can be found. E.g. the student Tom can be found on his own homepage but also on the home page of the department board.
- A relationship between *ConceptType* and *PageProtoType (ConceptType with_info_on/with_info_of PageProtoType*) to indicate that a page prototype is built for a certain concept type. E.g. the page prototype CoursePage is built for the object type Course,

i.e. info on Course instances can be found on CoursePage instances.

- A relationship between *ConceptType* and *Page (ConceptType with_population_on/with_population_of Page*) to indicate the pages that provide an overview/index of the population of a concept type. E.g. Most department web sites have a page (at least one) that lists all lecturers of the department.

Note that this is a simplified version of the Mapping Meta Schema. It only gives a broad indication (in terms of concept types) of what information can be found on a page. As an example, for the page prototype CoursePage the mapping schema will specifies that such a page will contain information on Courses and about the relationships Lecturer-giving-Course and LectureRoom-of-Course. However it is not possible to specify that a CoursePage instance will only contain information of one single Course instance and only on the Lecturers giving that course and on the LectureRoom of that course. To allow this kind of detailed mapping information, the Mapping Meta Schema should be refined. This can be done by introducing an OT *ConceptCollection* and relations to allow to specify a concept collection, e.g. by enumerating concepts or by construction a path expression [De Troyer 1996] that allows to refer to a subset of a concept type population. Then the mapping can be expressed in term of concept collections instead of in term of concepts.

5 The conceptual schema DTD

The BR Model is not a very convenient formalism to be used on the Internet. The Internet is nowadays dominated by HTML, but the language of the future seems to be XML [W3C 1997]. XML allows to bring more structure and semantics to the web. Therefore a suitable representation for a conceptual web site schema could be XML[1]. What we are aiming for is a standard way to represent the conceptual schema of a web site on the web. This can be accomplished by defining a DTD for describing a conceptual web site schema. Then, each conceptual web site schema can be expressed as an XML document using this DTD. Because this DTD will describe the general structure of any conceptual schema for a web site it is in fact the same as the conceptual meta schema defined in the previous section. What we need to do is to translate the conceptual meta schema given in the BR formalism into an XML DTD.

[1] Note that RDF can be used as an alternative for XML. Also XML data [Layman et al. 1998] is an alternative.

To transform the conceptual meta schema into an XML DTD we have developed an algorithm that is capable to transform any BR-schema into a DTD. This algorithm is outside the scope of this paper, it will be described elsewhere (see also [Decruyenare 1999]). The main idea is that the OTs in the schema is defined as elements in the DTD. We will illustrate the transformation by means of some examples. To avoid confusion we will use italic for names from the conceptual meta schema and bold for names in the DTD.

ConceptType is an OT in the meta schema and therefore declared as an element with the same name in the DTD: **ConceptType**. In the conceptual schema we see that *ObjectType* and *RelationType* are subtypes of *ConceptType.* This is represented by the two subsidiary elements in the content model of **ConceptType**. The separator "|" indicates that it is either an **ObjectType** or a **RelationType** but never both.
The roles played by *ConceptType* are also present in the content model of the element **ConceptType** by elements: **with_population_on** and **with_info_on.** The use of an asterisk * in the content model reflects the non-mandatory and non-unique constraints for these roles.

```
     <!ELEMENT ConceptType ((ObjectType | RelationType ),
with_population_on* , with_info_on*, has_typename )>
```

The attribute declaration of the element **ConceptType** contains the definition of the ID attribute **ConceptTypeID**. This attribute will be used to make references in other elements to this element.

```
<!ATTLIST ConceptType        ConceptTypeID ID #REQUIRED >
```

The declaration of the role elements derived is as follows:

```
    <!ELEMENT with_population_on EMPTY>
    <!ATTLIST with_population_on PageID IDREF #REQUIRED>
    <!ELEMENT with_info_on EMPTY>
    <!ATTLIST with_info_on PageProtoTypeID IDREF #REQUIRED>
    <!ELEMENT has_typename (#PCDATA)>
```

Role elements are empty but refer to the other OT (called CO-OT) in the relation by means of the ID attribute of the OT. An exception is the situation if the CO-OT is a so-called lexical OT. This means that its instances are utterable such as a name or a number. In that case the role element is not empty and its type of content will be in accordance with the lexical type of its CO-OT, e.g. for the role *has_typename* the content is PCDATA because the type of *ConceptTypeName* is String.

If we apply the algorithm on the complete conceptual meta schema given in section 4, we obtain the needed DTD, which we will refer to as the *Conceptual Schema DTD*. The complete Conceptual Schema DTD is given in the appendix and will be explained in more detail in section 6.4 by means of an example.

6 Case study: Volvo IT Belgium, Y2K web site

As a case study we have developed the conceptual schema for the Y2K web site of a company “Volvo Information Technology Belgium”. The web site represents the Y2K status information of soft- and hardware and was necessary to provide a forum to deal with the questions of the company’s customers. First we will give a short informal description of the content and the structure of the web site. Next, we show by means of some examples how the conceptual schema of this web site looks like. Finally, we give an extract from the XML document describing the conceptual schema according to the DTD defined in section 5.

6.1 The content of the web site

The web site is intended to be the official source of information for the customers of Volvo

Information Technology Belgium (VITB) about the Y2K for all types of infrastructure components. It provides information to what is being done, whether the specific item has been cleared for Y2K compliance and if not, when it will be. For each of the items, a division is made in terms of hardware, operating system and software.

For each of the items, the following information is presented:

- Y2K status and dates:
    - not started – specifying planned start date and planned completion date;
    - in progress – specifying start date, planned completion date, indication of the progress (e.g. 60% completed);
    - completed – specifying start date and completion date;
- The possibility to consult more detailed information, if available;
- Contact person for gathering more extensive information.

The items presented were identified as the information needs of the customers.

Further on, an introduction is given as well as a statement stating that the information is official, reliable, up-to-date and binding. There is also a section on general information: what is Y2K; what are the generic responsibilities of VITB towards its customers when it comes to Y2K issues; general information per platform; and related links (e.g. the Swedish Y2K site).

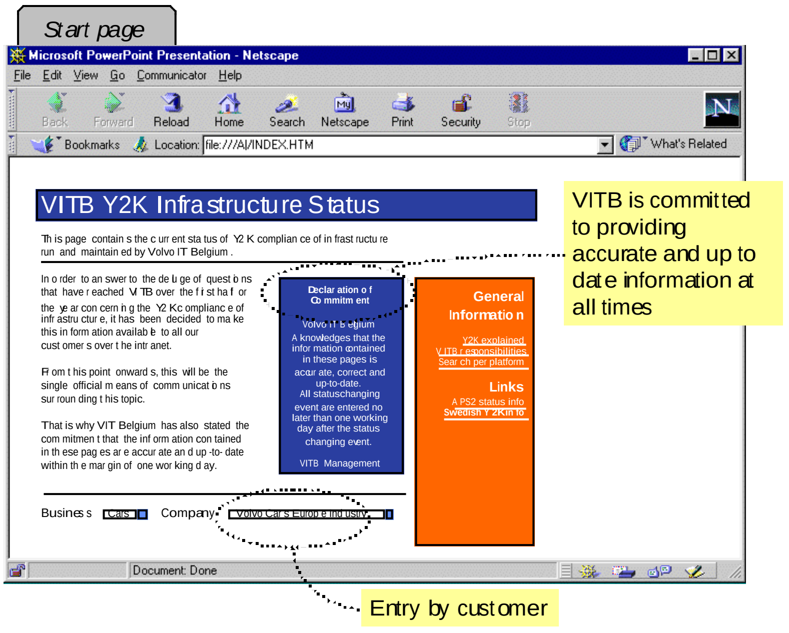
6.2 The structure of the web site and napping of the content

The Y2K web site contains the following pages:

- A Start page
- Customer pages
- Detailed pages

6.2.1 The start page

Start page


Microsoft PowerPoint Presentation - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///A|/INDEX.HTM
What's Related
Document: Done

VITB Y2K Infrastructure Status

This page contains the current status of Y2K compliance of infrastructure run and maintained by Volvo IT Belgium.

In order to answer to the deluge of questions that have reached VITB over the first half or the year concerning the Y2K compliance of infrastructure, it has been decided to make this information available to all our customers over the intranet.

From this point onwards, this will be the single official means of communications surrounding this topic.

That is why VIT Belgium has also stated the commitment that the information contained in these pages are accurate and up-to-date within the margin of one working day.

Declaration of Commitment

Volvo IT Belgium Acknowledges that the information contained in these pages is accurate, correct and up-to-date. All statuschanging event are entered no later than one working day after the status changing event.

VITB Management

General Information

Y2K explained
VITB responsibilities
Search per platform

Links

A PS2 status info
Swedish Y2K info

Business Cars Company Volvo Cars Europe Industry

VITB is committed to providing accurate and up to date information at all times

Entry by customer

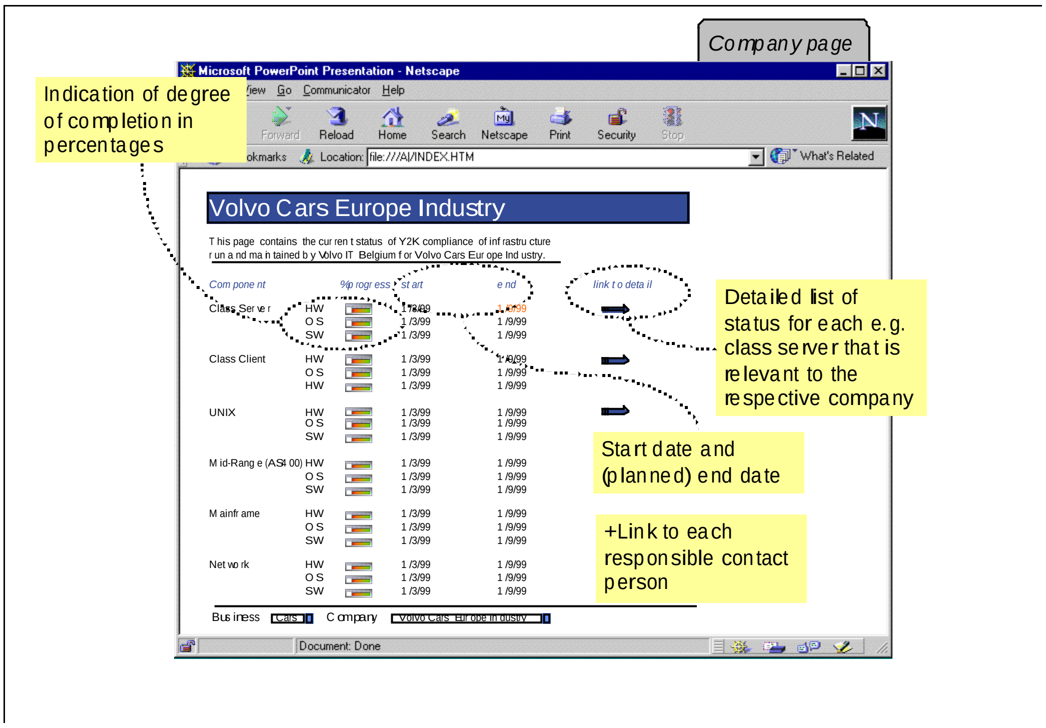
Page contents

- Information grouped by customer
- postulation for site
- single official means of communication
- statement delaring accuracy of info
- general information
- links to related info

Figure1: The start page.

This page is the central start page for every customer in search for information concerning the Y2K status of his infrastructure. The page contains some general information like the statement concerning the commitment made by VITB regarding the statuses provided on this web site. Also the link towards the Swedish Y2K site and other links are presented on this start page. Furthermore, each customer can find an entry (link) to his customer page where he can find specified information regarding his infrastructure.

6.2.2 *The customer pages*


Company page
Indication of degree of completion in percentages
Detailed list of status for each e.g. class server that is relevant to the respective company
Start date and (planned) end date
+Link to each responsible contact person
Microsoft PowerPoint Presentation - Netscape
View Go Communicator Help
Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///A|/INDEX.HTM
What's Related
Business Cars Company Volvo Cars Europe Industry
Document: Done

Volvo Cars Europe Industry

This page contains the current status of Y2K compliance of infrastructure run and maintained by Volvo IT Belgium for Volvo Cars Europe Industry.

Component		%progress	start	end	link to detail
Class Server	HW	▬	1/3/99	1/9/99	➡
	OS	▬	1/3/99	1/9/99	
	SW	▬	1/3/99	1/9/99	
Class Client	HW	▬	1/3/99	1/9/99	➡
	OS	▬	1/3/99	1/9/99	
	HW	▬	1/3/99	1/9/99	
UNIX	HW	▬	1/3/99	1/9/99	➡
	OS	▬	1/3/99	1/9/99	
	SW	▬	1/3/99	1/9/99	
Mid-Range (AS400)	HW	▬	1/3/99	1/9/99	
	OS	▬	1/3/99	1/9/99	
	SW	▬	1/3/99	1/9/99	
Mainframe	HW	▬	1/3/99	1/9/99	
	OS	▬	1/3/99	1/9/99	
	SW	▬	1/3/99	1/9/99	
Network	HW	▬	1/3/99	1/9/99	
	OS	▬	1/3/99	1/9/99	
	SW	▬	1/3/99	1/9/99	

Figure 2: A customer page.

A customer page contains the specific information for a customer. The information is brought together in a table where it is broken down into the different platforms used by the customer. Per platform he can find an indication of the status of the Y2K compliance, divided into operating system, hardware and software. Each platform has a link towards a page (the detailed page) where some detailed information can be found concerning that platform.

6.2.3 *The detailed pages*

We have omitted a picture of a detailed page. On such a page the customer can see the status of all e.g. AS400 servers that are relevant to him with their respective status info. This information should be available for all “components”: class clients, UNIX, mid-range, etc. Further on, there are links to “full report”, usually leading to a PDF file and a “Contact” link, providing the possibility of sending e-mails to an appointed author or contact person where unanswered queries can be addressed.

6.3 The conceptual schema of the Y2K web site

In this section we represent the Y2K web site using the OTs and the roles defined in the conceptual meta schema in section 4. Although it is possible to populate the binary relationships defined in the meta schema, for reasons of compactness, we will represent the information using tables. Also note, that only a fragment of the content of the different tables is given.

The table 6.1, 6.2 and 6.3 correspond with the content schema. Table 6.1 gives object types present in the Y2K web site together with the role types they play in a relation type. Table 6.2 shows some of the supertype/subtype relationships between object types and table 6.3 gives some examples of objects and object instances appearing in the web site.

ObjectType	RoleType	RelationType
Customer	operates_on	R_Customer_Platform
Customer	has_name	R_Customer_Name
CustomerName	name_of	R_Customer_Name
Platform	infrastructure_of	R_Customer_Platform
Platform	has_hw	R_Platform_HW
Platform	has_os	R_Platform_OS
Platform	has_sw	R_Platform_SW
HW	hw_of	R_Platform_HW
OS	os_of	R_Platform_OS
SW	sw_of	R_Platform_SW
Item	has_st	R_Item_Status
Status	status_of	R_Item_Status
Status	has_start	R_Status_StartDate
Status	has_end	R_Status_EndDate
Date	start_of	R_Status_StartDate
Date	end_of	R_Status_EndDate

Table 6.1: Object types and their role types in the Y2K web site.

Super ObjectType	Sub ObjectType
Item	HW
Item	SW
Item	OS
HW	Network
HW	Server
HW	Client

Table 6.2: Super/sub relationship between object types.

Object	Is ObjectInstance of_ObjectType	Kind of
Commitment	-	text
Introduction	-	text
VolvoTrucks	Customer	

Table 6.3: Objects an object type instances in the Y2K web site.

Table 6.4, 6.5, 6.6 and 6.7 cover the structure schema. Table 6.4 gives an overview of page prototypes; table 6.5 gives pages and indicates if they are an instance of some page prototype. Table 6.6 provides an overview of link prototypes and table 6.7 gives an overview of links.

PagePrototype	Type
CustomerPage	Tailored
DetailPage	Tailored

Table 6.4: Page prototypes in the Y2K web site.

Page	Type	PageInstance of_PagePrototype
StartPage	static	-
SwedishY2Page	external	-
VolvoTrucksPage	static	CustomerPage

VolvoTrucksDetail	static	DetailPage

Table 6.5: Pages in the Y2K web site.

LinkProtoType	From Page	From Pageprotoype	To Page	To PageProtoType
CustomerLink	StartPage	-	-	CustomerPage
PlatformLink	-	CustomerPage	-	DetailedPage

Table 6.6: Link prototypes in the Y2K web site.

Link	From Page	To Page	LinkInstance of_LinkProtoType
Swedish Y2K_link	StartPage	SwedishY2KPage	-
Apps2status_link	StartPage	Apps2Page	-
VolvoTruck_Link	StartPage	VolvoTrucksPage	CustomerLink
VolvoTruck_Detail_Link	VolvoTrucksPage	VolvoTrucksDetail	PlatformLink

Table 6.7: Links in the Y2K web site.

We will now show the mapping schema. Table 6.8 shows the concepts that can be found on the different pages. Table 6.9 gives an overview of the pages that provide an overview of the population of some concept type. Table 6.10 indicates the concept types of which a PageProtoType contains information.

Page	Contains Concept
StartPage	Introduction
StartPage	Commitment
VolvoTrucksPage	VolvoTrucks
VolvoTrucksDetail	VolvoTrucks

Table 6.8: Concepts on pages.

Page	With_population_of ConceptType
StartPage	Customer

Table 6.9: Pages providing the population of some concept type.

PageProtoType	With_info_of ConceptType
CustomerPage	Customer
CustomerPage	R_Customer_Platform
CustomerPage	R_Platform_HW
CustomerPage	R_Platform_OS
CustomerPage	R_Platform_SW
CustomerPage	R_Item_Status
CustomerPage	R_Status_StartDate
CustomerPage	R_Status_EndDate
DetailedPage	Client
DetailedPage	Server
DetailedPage	Network

Table 6.10: Page prototypes providing information of concept types.

6.4 The conceptual schema document of the Y2K web site

As explained earlier, to exchange the information in a conceptual web schema it is more convenient to represent it as an XML document. In this section we will discuss the different elements in the document and how they are declared using the conceptual schema DTD given in the appendix. Again, we will use italic for names from the conceptual meta schema and bold for names from the conceptual schema DTD. Instances from the conceptual meta schema are put between single quotes, e.g. ‘Customer’.

We start by declaring the element ‘Customer’ which is, in the conceptual web site schema and instance of *ObjectType* and more in general an instance of *ConceptType.* We need to use different ID values for every element within an XML document; therefore we use a different name for the **ConceptTypeID** and for the **ObjectTypeID**. We adopt the convention to build the **ConceptTypeID** by putting a “C_” in front of the **ObjectTypeID**, which corresponds with the name of the object type.

In the content model of the element **ObjectType** we will find the element **with**, which is a reference towards a **RoleTypeID** and gives the *RoleTypes* the *ObjectType* is playing: ‘Customer‘ plays two different role types in two relation types (see table 6.1): ‘operates_on’ and ‘has_name’. This results in two different sub-elements **with** in the **ObjectType** element “Customer”.

In the content model of **ConceptType** there are three elements, **with_population_on**, **with_info_on** and **has_typename**. The first tindicates on which page an overview of the corresponding concept type population can be found and the second indicates on which page prototype the detail information of an instance can be found. For the *ObjectType* ‘Customer’, the population can be found on the ‘StartPage’ and there is a *PageProtoTypePag*e ‘CustomerPage’ that provide detail information on ‘Customer’. The element **has_typename** gives the name of the *ConcepType*.

```
<ConceptType ConceptTypeID = "C_Customer">
     <ObjectType ObjectTypeID = "Customer">
          <with RoleTypeID = "operates_on"/>
          <with RoleTypeID = "has_name"/>
     </ObjectType>
     <with_population_on PageID = "StartPage"/>
     <with_info_on PageProtoTypeID = "CustomerPage"/>
     <has_typename> Customer </has_typename>
</ConceptType>
```

Next we describe a *RelationType* ‘R_Customer_Platform’. A relation type is also declared as a sub-element of **ConceptType** in the DTD. To construct the **ConceptTypeID**, we replace the “R” in the *RelationType* name by a “C”.

Furthermore there is one sub-element in the content model of **RelationType**: **with** is the element that refers to the roles out of which the relation type is composed. As we can see in table 6.1, ‘R_Customer_Platform’ has two roles ‘operates_on’ and ‘infrastructure_of’**.**

```
<ConceptType ConceptTypeID = "C_Customer_Platform">
     <RelationType RelationTypeID = "R_Customer_Platform">
          <with RoleTypeID = "operates_on"/>
          <with RoleTypeID = "infrastructure_of"/>
     </RelationType>
     <has_typename> R_Customer_Platform </has_typename>
</ConceptType>
```

Next is a **RoleType** element including two sub-elements: **played_by_objecttype** is representing the ID reference of the OT playing this role, **in_relationtype** is the element that refers to the relation type of the role.

```
<RoleType RoleTypeID = "operates_on">
     <played_by_objecttype ObjectTypeID = "Customer"/>
     <in RelationTypeID = "R_Customer_Platform"/>
</RoleType>
```

We will now give an example of a **Concept** element. ‘VolvoTrucks’ is a *Concept*, more in particular an *Object* and also an *ObjectInstance* of the *ObjectType* ‘Customer’ (see table 6.3). The connection with the structure schema is made by the elements **on_page** representing the references towards the pages on which information about ‘VolvoTrucks’ can be found. According to table 6.8 these are the pages VolvoTrucksPage and VolvoTrucksDetail.

```
<Concept ConceptID = "C_VolvoTrucks">
     <Object ObjectID = "O_VolvoTrucks">
```

```
           <ObjectInstance ObjectInstanceID = "VolvoTrucks">
                <of_objecttype ObjectTypeID = "Customer"/>
           </ObjectInstance>
      </Object>
      <on_page PageID = "P_VolvoTrucksPage"/>
      <on_page PageID = "P_VolvoTrucksDetail"/>
      <has_name> VolvoTrucks </has_name>
</Concept>
```

Next we will demonstrate how pages and links are represented in the XML document.
The ‘StartPage’ is a *Page* but not a *PageInstance*. It is a static page and contains the *Concepts* ‘Introduction’ and ‘Commitment’ (see table 6.8), an overview of the population of the *ObjectType* ‘Customer’ (see table 6.9), a *Link* to the ‘Swedish_Y2K’ web site (see table 6.7) and a *LinkProtoType* to the ‘CustomerPage’ PageProtoType (see table 6.6)

```
<Page PageID = "StartPage">
      <StaticPage/>
      <page_contains ConceptID = "C_Introduction"/>
      <page_contains ConceptID = "C_Commitment"/>
      <with_population_of ConceptTypeID="C_Customer"/>
      <is_source_of LinkID = "Swedish_Y2K_Link"/>
      <is_pagesource_of LinkProtoTypeID = "CustomerLink"/>
</Page>
```

The element **PageProtoType** is given below and lists all concept types (by means of the element **with_info_of**) for which information may be found on the page instances of this page prototype (see table 6.10). ‘CustomerPage’ is also the prototype source of the *LinkProtoType* ‘PlatformLink’ (see table 6.6)

```
<PageProtoType PageProtoTypeID = "CustomerPage">
```

```
     <TailoredPagePrototype/>
     <with_info_of ConceptTypeID = "C_Customer"/>
     <with_info_of ConceptTypeID = "C_Customer_Platform"/>
     <with_info_of ConceptTypeID = " C_Platform_HW "/>
     <with_info_of ConceptTypeID = " C_Platform_OS "/>
     <with_info_of ConceptTypeID = " C_Platform_SW "/>
     <with_info_of ConceptTypeID = "C_Item_Status"/>
     <with_info_of ConceptTypeID = "C_Status_StartDate"/>
     <with_info_of ConceptTypeID = "C_Status_EndDate"/>
     <is_prototypesource_of LinkProtoTypeID ="PlatformLink"/>
</PageProtoType>
```

We can also define the pages that are instances of some page prototype; e.g. 'VolvoTrucksPage' is an instance of the prototype page 'CustomerPage'**.**

```
<Page PageID = "P_VolvoTrucksPage">
     <PageInstance PageInstanceID = "VolvoTrucksPage">
     <of_pageprototype PageProtoTypeID = "CustomerPage"/>
     </PageInstance>
     <is_source_of LinkID = "L_VolvoTruck"/>
</Page>
```

Next we give an example of a **LinkProtoType** element:

```
<LinkProtoType LinkProtoTypeID = "CustomerLink">
     <from_page PageID = "StartPage"/>
     <to_pageprototype PageProtoTypeID = "CustomerPage"/>
</LinkProtoType>
```

Finally, we present the 'VolvoTruck_Link' that is a LinkInstance of the previous *LinkProtoType* between the ‘StartPage’ and the ‘VolvoTruckPage’ *PageInstance*.

```
<Link LinkID = "L_VolvoTruck_Link">
     <LinkInstance LinkInstanceID = "VolvoTruck_Link">
     <of_linkprototype LinkProtoTypeID = "CustomerLink"/>
     </LinkInstance>
     <from PageID = "StartPage"/>
     <to PageID = "P_VolvoTrucksPage"/>
</Link>
```

7 An example application

In this section we will illustrate what can be done with the conceptual schema XML document of a web site. Here we have used the XML Document Object Model (DOM) [W3C DOM WG 2000] to extract the information from the XML document and to provide it to a user. The example is kept simple and the information in displayed in the form of tables. It is clear that more sophisticated applications are possible, i.e. to display the conceptual schema is some graphical form and to allow navigation and search facilities from this conceptual schema representation.

7.1 The XML document object model

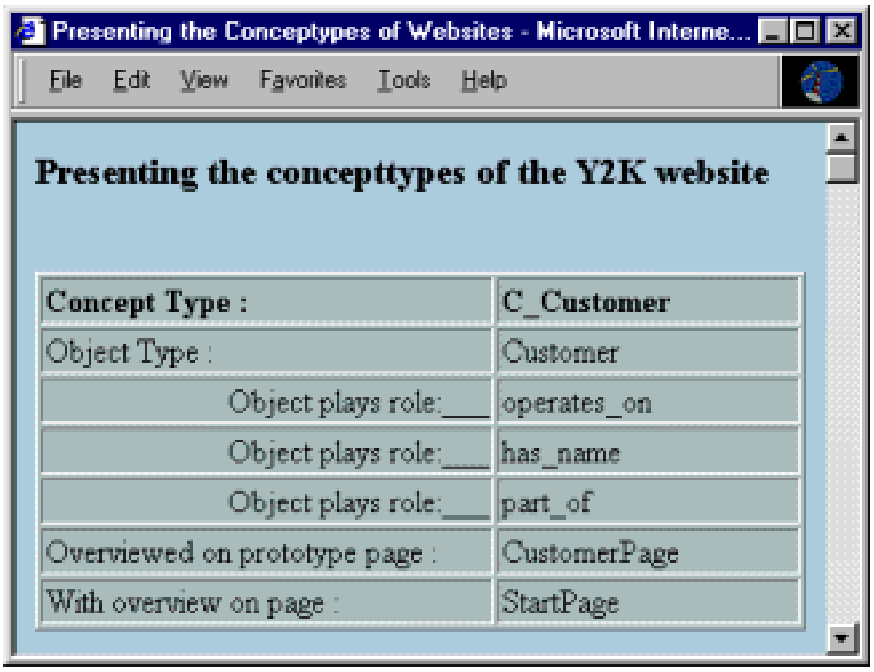
The XML Document Object Model (DOM) proposed by the World Wide Web Consortium (W3C), is potentially one of the most important standards since the XML specification. It gives implementers a common vocabulary to use in manipulating the XML document. No matter which language is applied (C, Java, Python, Visual Basic) exactly the same methods can be used. The core model proposed by the W3C consists of a suggested API (Application Program Interface) for various different applications. The major software vendors have promised to adhere to the W3C DOM interface. Internet Explorer 5 has already implemented the majority of it. We will use XML DOM together with JavaScript [Netscape Communications Corporation 1996] to build some rather simple programs to explore the content of the conceptual schema

XML document and to include the results into HTML pages.

7.2 Reflecting the elements of the conceptual schema

In Fig. 7.1 you see the result of extracting information from the XML document about concept types. Only the first concept type is visible, but by scrolling, all the available concept types can be explored. For the first ***ConceptType*** 'Customer', the role types played by this object type are given as well as the corresponding page prototype 'CustomerPage' and the page on which the population of this object type is given 'StartPage'.

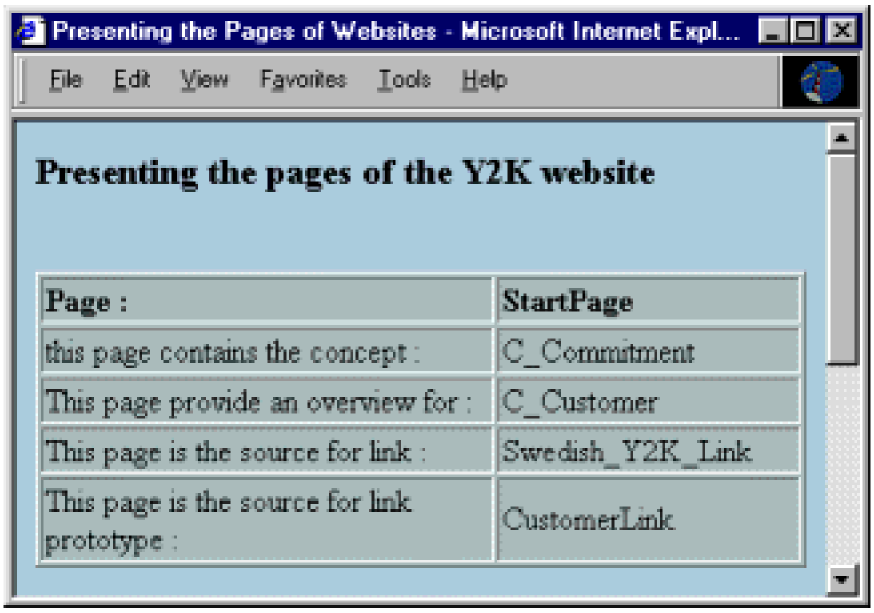
By using another example you can find details concerning the pages and the links, also reflecting the different concepts and the information they contain. Fig. 7.2 presents the result.


Presenting the Conceptypes of Websites - Microsoft Interne...
File Edit View Favorites Tools Help

Presenting the concepttypes of the Y2K website

Concept Type :	**C_Customer**
Object Type :	Customer
Object plays role:___	operates_on
Object plays role:___	has_name
Object plays role:___	part_of
Overviewed on prototype page :	CustomerPage
With overview on page :	StartPage

Figure 7.1: Result of exploring the ConceptTypes of the Y2K web site.


Presenting the Pages of Websites - Microsoft Internet Expl...
File Edit View Favorites Tools Help

Presenting the pages of the Y2K website

Page :	StartPage
this page contains the concept :	C_Commitment
This page provide an overview for :	C_Customer
This page is the source for link :	Swedish_Y2K_Link
This page is the source for link prototype :	CustomerLink

Figure 7.2: Result of exploring the pages of the Y2K web site.

8 Conclusions and future research

WWW is now one of the major sources of information in every rank of our society. However, looking at the evolution of the WWW, it will become more and more important to introduce more structure and standardisation in order to make the available information more accessible. In this paper we have tried to contribute to this by developing a conceptual schema for a web site that describes what information is available in a web site and how this information is structured into pages and links. To allow to communicate this information through the web, we have developed an XML DTD for such a conceptual web site schema. We also illustrated the feasibility of the approach by some simple application programs developed using XML DOM together with JavaScript.

It is clear that such a conceptual web site schema should not be built manually. Tools should exist to assist this process. E.g. such a tool could be a part of a CAWE (Computer Aided Web Engineering) tool generating well-structured web sites from conceptual specifications and based on some solid method for designing web sites. For such methods see e.g. [De Troyer and Leune 1998; Isakowitz et al. 1995; Schwabe and Rossi 1995]. In addition a conceptual web site schema for the end-user must not contain all possible information about a web site. This may be too detailed for its purpose. E.g. it may be limited to the most important concepts and concept types for which information is available in the web site. As a matter of fact it may be seen as a view on a more general type of the conceptual schema that may also be used for designing and implementing the web site. Such a conceptual schema is closer to the conceptual schema of a database. Graphical tools to display the conceptual web site at different levels of detail and from different viewpoint should be developed.

Appendix: The complete conceptual schema DTD

```
<?xml version="1.0"?>

<!ELEMENT Website (ConceptType | Concept | RoleType | Role |
               PageProtoType | Page | LinkProtoType | Link)*>
<!ATTLIST Website   WebsiteID ID #REQUIRED >



<!--                    ConceptType                       -->
  <!ELEMENT ConceptType ((ObjectType | RelationType ),
               with_population_on* ,
                        with_info_on*,
                        has_typename ) >
  <!ATTLIST ConceptType    ConceptTypeID ID #REQUIRED >
               <!ELEMENT with_population_on EMPTY>
               <!ATTLIST with_population_on PageID
                                            IDREF #REQUIRED>
               <!ELEMENT with_info_on EMPTY>
               <!ATTLIST with_info_on PageProtoTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT has_typename (#PCDATA)>



<!--                    ObjectType                        -->
  <!ELEMENT ObjectType (is_super* ,
                        is_sub* ,
                        with_objectinstance*,
                        playing_roletype* ) >
  <!ATTLIST ObjectType  ObjectTypeID ID #REQUIRED >
               <!ELEMENT is_super EMPTY>
               <!ATTLIST is_super ObjectTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT is_sub EMPTY>
               <!ATTLIST is_sub ObjectTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT with_objectinstance EMPTY>
               <!ATTLIST with_objectinstance ObjectInstanceID
                                            IDREF #REQUIRED>
               <!ELEMENT playing_roletype EMPTY>
               <!ATTLIST playing_roletype RoleTypeID
                                            IDREF #REQUIRED>


<!--                    RelationType                      -->
  <!ELEMENT RelationType ( with_roletype+ ,
                           with_relationinstance* ) >
  <!ATTLIST RelationType   RelationTypeID ID #REQUIRED >
               <!ELEMENT with_roletype EMPTY>
               <!ATTLIST with_roletype RoleTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT with_relationinstance EMPTY>
               <!ATTLIST with_relationinstance
                         RelationInstanceID IDREF #REQUIRED>


<!--                    RoleType                          -->
  <!ELEMENT RoleType ( played_by_objecttype ,
                       in_relationtype )>
  <!ATTLIST RoleType    RoleTypeID ID #REQUIRED >
               <!ELEMENT played_by_objecttype EMPTY>
               <!ATTLIST played_by_objecttype ObjectTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT in_relationtype EMPTY>
               <!ATTLIST in_relationtype RelationTypeID
                                            IDREF #REQUIRED>


<!--                    ObjectInstance                    -->
```

```
  <!ELEMENT ObjectInstance  ( of_objecttype ) >
  <!ATTLIST ObjectInstance ObjectInstanceID ID #REQUIRED >
              <!ELEMENT of_objecttype EMPTY>
              <!ATTLIST of_objecttype ObjectTypeID
                                           IDREF #REQUIRED >



<!--                  RelationInstance                   -->
  <!ELEMENT RelationInstance ( of_relationtype ) >
  <!ATTLIST RelationInstance RelationInstanceID
                                           ID #REQUIRED >
              <!ELEMENT of_relationtype EMPTY>
              <!ATTLIST of_relationtype RelationTypeID
                                           IDREF #REQUIRED>



<!--                    Concept                          -->
  <!ELEMENT Concept ( ( Object | Relation ),
                       on_page*,
                       has_name )>
  <!ATTLIST Concept    ConceptID ID #REQUIRED >
              <!ELEMENT on_page EMPTY>
              <!ATTLIST on_page PageID IDREF #REQUIRED>
              <!ELEMENT has_name (#PCDATA) >



<!--                    Object                           -->
  <!ELEMENT Object ((StructuredObject | ElementaryObject)? ,
              ObjectInstance? ) >
  <!ATTLIST Object ObjectID ID #REQUIRED >
              <!ELEMENT StructuredObject (is_composed_of+)>
              <!ATTLIST StructuredObject StructuredObjectID
                                           ID #REQUIRED >
                       <!ELEMENT is_composed_of EMPTY>
                       <!ATTLIST is_composed_of  ObjectID
                                           IDREF #REQUIRED>
```

```
               <!ELEMENT ElementaryObject(
                        ( Text | Graphics | Video)?) >
               <!ATTLIST ElementaryObject   ElementaryObjectID
                                            ID #REQUIRED >
                        <!ELEMENT Text EMPTY)>
                        <!ELEMENT Graphics EMPTY>
                        <!ELEMENT Video EMPTY>


<!--                    Relation                         -->
  <!ELEMENT Relation ( RelationInstance ?, with )>
  <!ATTLIST Relation RelationID ID #REQUIRED >
               <!ELEMENT with EMPTY>
               <!ATTLIST with RoleID IDREF #REQUIRED>


<!--                    Role                             -->
  <!ELEMENT Role ( in, played_by ) >
   <!ATTLIST Role RoleID ID #REQUIRED >
               <!ELEMENT in EMPTY>
               <!ATTLIST in RelationID IDREF #REQUIRED>
               <!ELEMENT played_by EMPTY>
               <!ATTLIST played_by ObjectID IDREF #REQUIRED>


<!--                    Page                      -->
  <!ELEMENT Page (PageInstance?,
               ExternalPage?,
               (StaticPage | TailoredPage | DynamicPage)?,
               is_source_of* ,
               is_target_of* ,
               is_pagesource_of*,
               is_pagetarget_of* ,
               page_contains*,
               with_population_of* ) >
  <!ATTLIST Page PageID ID #REQUIRED >
```

```
               <!ELEMENT ExternalPage EMPTY>
               <!ELEMENT StaticPage EMPTY>
               <!ELEMENT TailoredPage EMPTY>
               <!ELEMENT DynamicPage EMPTY>
               <!ELEMENT is_source_of EMPTY>
               <!ATTLIST is_source_of LinkID IDREF #REQUIRED>
               <!ELEMENT is_target_of EMPTY>
               <!ATTLIST is_target_of LinkID IDREF #REQUIRED>
               <!ELEMENT is_pagesource_of EMPTY>
               <!ATTLIST is_pagesource_of LinkProtoTypeID
                                             IDREF #REQUIRED>
               <!ELEMENT is_pagetarget_of EMPTY>
               <!ATTLIST is_pagetarget_of LinkProtoTypeID
                                             IDREF #REQUIRED>
               <!ELEMENT page_contains EMPTY>
               <!ATTLIST page_contains ConceptID
                                             IDREF #REQUIRED>
               <!ELEMENT with_population_of EMPTY>
               <!ATTLIST with_population_of ConceptTypeID
                                             IDREF #REQUIRED


<!--                    PageInstance                      -->
  <!ELEMENT PageInstance (of_pageprototype)>
  <!ATTLIST PageInstance   PageInstanceID ID #REQUIRED >
               <!ELEMENT of_pageprototype EMPTY>
               <!ATTLIST of_page_prototype  PageProtoTypeID
                                             IDREF #REQUIRED>



<!--                    PageProtoType                     -->
  <!ELEMENT PageProtoType (
               (StaticPagePrototype |
               TailoredPageProtoType |
               DynamicPagePrototype)?,
               with_info_of+ ,
               is_prototypesource_of* ,
               is_prototypetarget_of* ) >
               <!ATTLIST PageProtoType PageProtoTypeID
                                            ID #REQUIRED>
               <!ELEMENT StaticPagePrototype EMPTY>
               <!ELEMENT TailoredPagePrototype EMPTY>
               <!ELEMENT DynamicPagePrototype EMPTY>
               <!ELEMENT with_info_of EMPTY>
               <!ATTLIST with_info_of ConceptTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT is_prototypesource_of EMPTY>
               <!ATTLIST is_prototypesource_of LinkProtoTypeID
                                            IDREF #REQUIRED>
               <!ELEMENT is_prototypetarget_of EMPTY>
               <!ATTLIST is_prototypetarget_of LinkProtoTypeID
                                            IDREF #REQUIRED>


<!--                    Link                          -->
  <!ELEMENT Link  ( LinkInstance?,
               ( InPageLink | OutPageLink | OutSideLink)
               from,
               to ) >
  <!ATTLIST Link LinkID ID #REQUIRED >
               <!ELEMENT InPageLink EMPTY>
               <!ELEMENT OutPageLink EMPTY>
               <!ELEMENT OutsideLink EMPTY>
               <!ELEMENT from EMPTY>
               <!ATTLIST from PageID IDREF #REQUIRED>
               <!ELEMENT to EMPTY>
               <!ATTLIST to PageID IDREF #REQUIRED>


<!--                    LinkInstance                  -->
  <!ELEMENT LinkInstance (of_linkprototype )>
  <!ATTLIST LinkInstance   LinkInstanceID ID #REQUIRED>
```

```
               <!ELEMENT of_linkprototype EMPTY>
               <!ATTLIST of_linkprototype LinkProtoTypeID
                                             IDREF #REQUIRED>


<!--                    LinkProtoType                     -->
  <!ELEMENT LinkProtoType ((from_pageprototype | from_page) ,
               (to_pageprototype | to_page) ) >
  <!ATTLIST LinkProtoType LinkProtoTypeID ID #REQUIRED >
               <!ELEMENT from_pageprototype EMPTY>
               <!ATTLIST from_pageprototype PageProtoTypeID
                                             IDREF #REQUIRED>
               <!ELEMENT to_pageeprototype EMPTY>
               <!ATTLIST to_pageprototype  PageProtoTypeID
                                             IDREF #REQUIRED>
               <!ELEMENT from_page EMPTY>
               <!ATTLIST from_page PageID IDREF #REQUIRED>
               <!ELEMENT to_page EMPTY>
               <!ATTLIST to_page  PageID IDREF #REQUIRED>
```

References

Visual Sitemap (1999) [Sit] http://lislin.gws.uky.edu/Sitemap

Boumphrey, F.and O. Direnzo (1998), *Programmer to Programmer XML Applications,* Wrox Press Ltd.

Chen, P.P. (1976), "The Entity-Relationship Model: Towards a Unified View of Data," *ACM Transactions on Database Systems* 1, 1, 471-522.

Decruyenare, T. (1999), "The use of XML in modeling Conceptual Schemas in web site design," Master Thesis, Department of Computer Science, Vrije Universiteit Brussel.

Delcambre, L.M.L., D. Maier, R. Reddy , L. Anderson (1997), “Structured Maps: modeling explicit semantics over a universe of information,”, *International Journal on Digital Libraries* 1 , 1, 20-35.

De Troyer, O. M.F. (1996), "A formalization of the Binary Object-Role Model based on Logic," *Data & Knowledge Engineering* 19, 1-37.

De Troyer, O.M.F. (1998), "Designing Well-Structured Web Site: Lessons to be Learned from Database Schema Methodology," In *Proceedings of the Entity Relationship Conference* , Lecture Notes in Computer Science 1507, Springer, pp. 51-64.

De Troyer, O.M.F., C. Leune (1998), "WSDM: A User-Centered Design Method for Web Sites," *In Proceedings of the 7th International World Wide Web Conference* ,Computer Networks and ISDN systems, Elsevier , pp. 85 - 94.

Goldfarb, C.F., P. Prescod (1998), *The XML Handbook*, Prentic Hall.

Halpin,T. (1995), *Conceptual Schema and Relational Database Design*, Second Edition, Prentice Hall Australia.

IBM (2000), “Mapuccino,” http://www.ibm.com/java/mapuccino/

Isakowitz, T. , E. A. Stohr, P. Balasubramanian (1995), "RMM: A Methodology for Structured Hypermedia Design," *Communications of the ACM* 38 , 8, 34-43.

ISO/IEC JTC 1/SC34 (1999), “ISO/IEC FCD 13250:1999 - Topic Maps,” http://www.ornl.gov/sgml/sc34/document/0058.htm

Netscape Communications Corporation (1996), Netscape Navigator (version4.0), JavaScript Guide.

Layman et al. (1998), “XML-Data, W3C Note 05 Jan 1998,” http://www.w3.org/TR/1998/NOTE-XML-data/

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991), *Object Oriented Modeling and Design*, Prentice Hall Inc.

Schwabe, D. and G. Rossi (1995), "The Object-Oriented Hypermedia Design Model," *Communications of the ACM* 38 , 8, 45-46.

Wintraecken, J.J. (1990), *The NIAM Information Analysis Method - Theory and Practice*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

W3C (1997), " Extensible Markup Language (XML)," http://www.w3.org/XML/

W3C (2000), "The World Wide Web Consortium," http://www.w3.org/

W3C DOM WG (2000), "Document Object Model (DOM)," http://www.w3.org/DOM/

W3C RDF WG (2000), "Resource Description Framework (RDF)," http://www.w3.org/RDF/

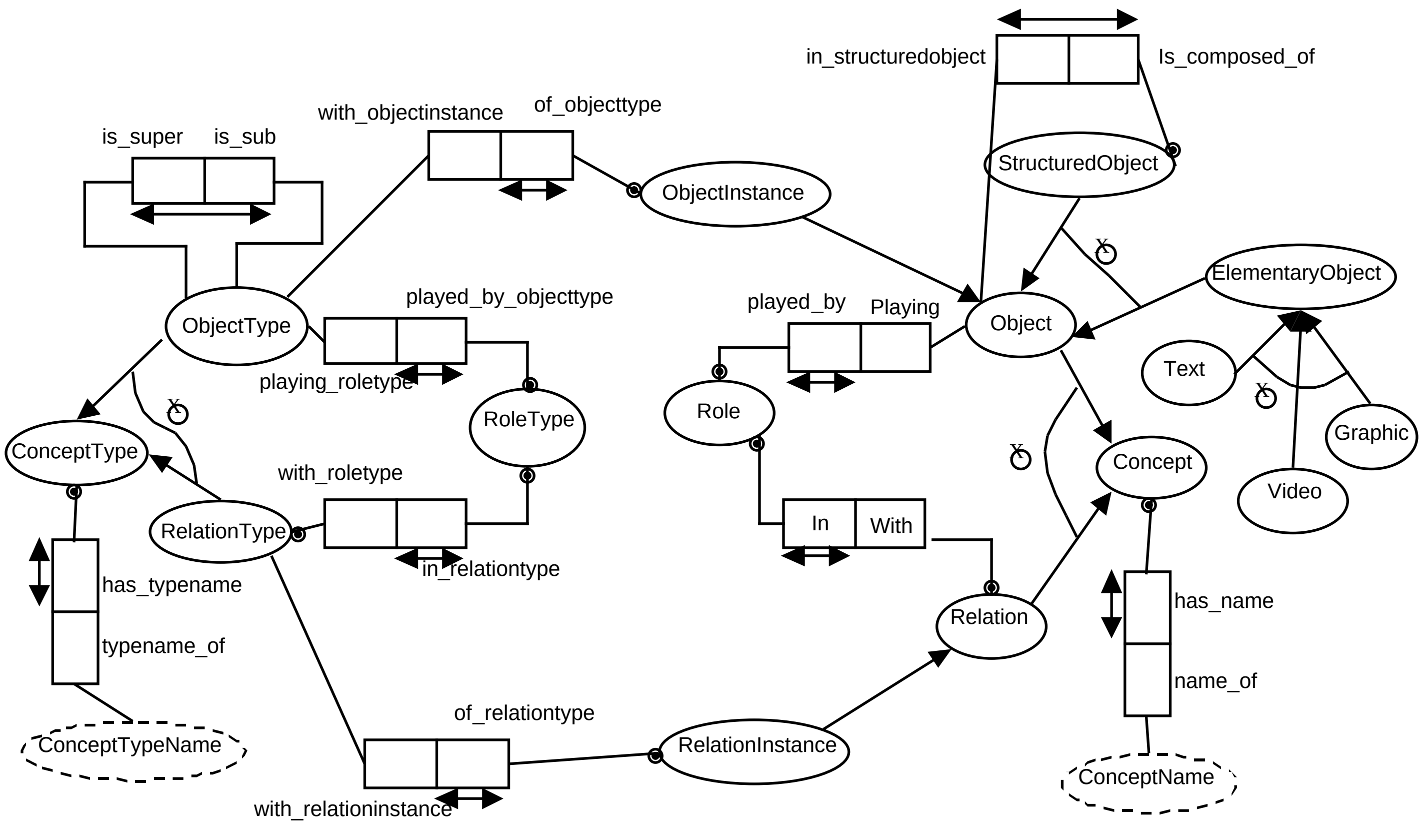


Figure 1: Content meta schema

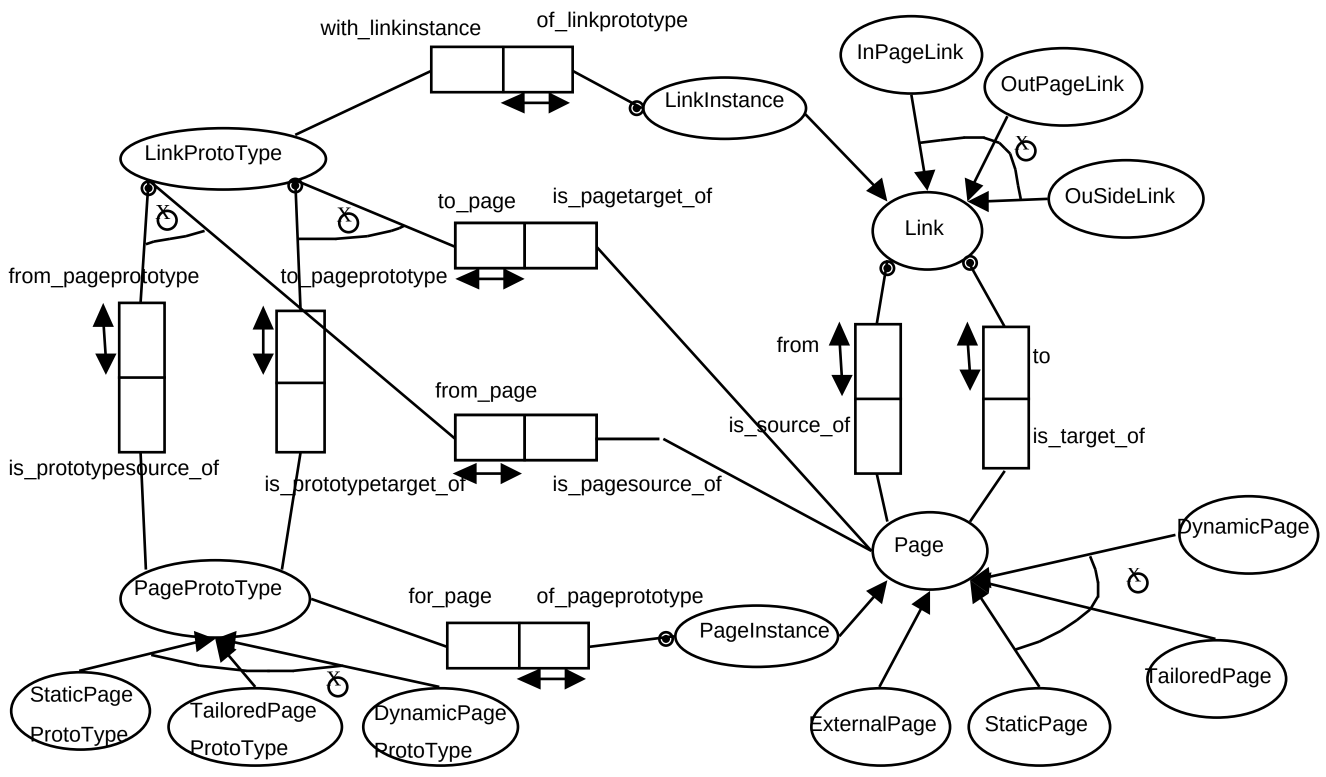


Figure 2: Structure meta schema.

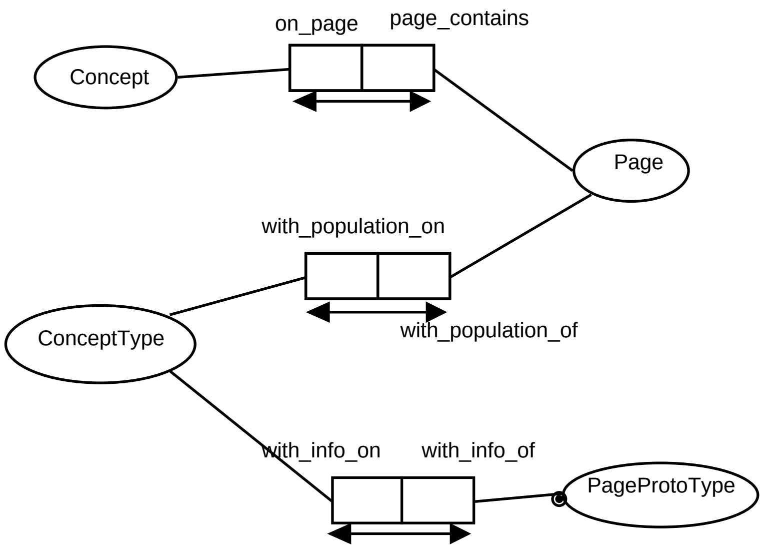


Figure 3: Mapping meta schema