Vrije Universiteit Brussel
Faculteit Wetenschappen
Departement Informatica

XML and related standards for representing conceptual website schemas

Bert Lefever
Academiejaar
1999-2000

Eindwerk voorgedragen tot het behalen van de graad van
Licentiaat in de Toegepaste Informatica

Promotor : Prof. Olga De Troyer

Table of Contents

Abstract

The ever-growing Internet connects the world through simple and well-known protocols. A simple markup language like HTML has, through its simplicity, given a great boost to this network. However, the markup of HTML primarily formats the data in the documents. This does not bring about an efficient maintenance of the documents. Therefore XML was introduced. This is, like HTML, a subset of SGML. Though more complex than HTML, it introduces the flexibility wanted in the third generation of Internet applications.

XML, with its extensibility, allows definition of semantical markup to enhance the data, but does not include formatting or structuring markup in its documents. This gives us documents, which are self-documenting and efficient to reuse for different media types or for exchange of information between multiple partners.

XML in itself has no representation mechanism. External tools are needed for this, hence the creation of new style sheet languages which can use the full power of XML. We scout possible representation mechanisms and put the emphasis on XSL, which seems to be the best way to handle the documents. We discuss the different emerging or fixed standards of the W3C (CSS, XML namespaces, XPath and XSL) and implement them in a case study.

The success of Internet has as an effect that a lot of data becomes available, but the main issue is also that this data is not structured enough to efficiently find the right information. Therefore search engines were introduced to find information based on full text indexing and the use of meta-information. Different techniques exist to guide users within a web site to the wanted information. An extension of these two facilities is making a conceptual schema of a web site available for both user and search engine. Based on the work of Decruyenaere and De Troyer we can make XML documents that represent the conceptual schema of a web site to transfer meta-information to the user.

Using the three schemas (content, structure and mapping schema) of this conceptual schema we use the discussed standards to suggest two style sheets. These style sheets will, each from a different viewing angle, represent the conceptual schema to the user. This way the user can explore the web site in what conceptual elements are available, what pages and link structure is available and where the conceptual elements can be found on the web site. Through extensive use of hyperlinking the user can explore the web site within the conceptual schema or explore the web site itself using the conceptual schema as a point of entrance.
The style sheets can be applied to any XML document representing the conceptual schema and which conforms to the given DTD.

This way we have shown the viability of the new standards and proven the usability of the conceptual schema as a medium to convey meta-information of web sites to people who want to search the web in an efficient way.

Samenvatting

Het Internet, dat nog voortdurend groeit, verbindt de wereld door middel van enkele eenvoudige en goed gekende protocols. Een eenvoudige opmaaktaal als HTML heeft, dankzij haar eenvoud, voor een enorme stuwkracht gezorgd. Nochtans geeft HTML enkel de data in de documenten een grafische opmaak. Dit zorgt voor moeilijke onderhoudbaarheid van deze documenten. Daarom werd XML, net als HTML een subset van SGML, ingevoerd. Hoewel XML complexer is dan HTML, heeft het de flexibiliteit nodig voor derde generatie Internet toepassingen. XML laat door haar uitbreidbaarheid (extensibiliteit) toe semantische – met een betekenis - opmaak te definiëren. Er wordt echter geen grafische of structurele opmaak meegegeven in de documenten. Dit geeft ons documenten die zelfdocumenterend zijn en efficiënt herbruikt kunnen worden voor verschillende media types of voor uitwisseling met verschillende communicatiepartners.

XML heeft op zichzelf geen manier van presentatie. Hiervoor zijn externe tools nodig, vandaar de creatie van nieuwe style sheet talen, waardoor de volle kracht van XML kan benut worden. Wij onderzoeken verschillende presentatiemogelijkheden en benadrukken XSL, welke de beste manier lijkt te zijn om XML documenten te behandelen. We bespreken de verschillende standaarden (CSS, XML namespaces, XPath en XSL) van het W3C die vastliggen of in aanmaak zijn en we implementeren deze in een toepassing.

Het succes van het Internet heeft als effect dat veel data beschikbaar wordt, maar belangrijk hierbij is deze data niet genoeg gestructureerd is om snel de correcte informatie terug te vinden.
Daarom werden al vroeg zoekmachines geprogrammeerd die informatie opzochten op basis van de indexering van volledige tekst en metainformatie (zoals sleutelwoorden).
Er bestaan ook verschillende technieken om de gebruiker snel doorheen een web site naar de geschikte informatie te loodsen.
Een uitbreiding van o.a. deze twee mogelijkheden is het aanmaken van een conceptueel schema van een web site en deze beschikbaar te stellen voor zowel eindgebruiker als zoekmachine. We baseerden ons op het werk van Decruyenaere en De Troyer om een XML document aan te maken dat het conceptueel schema van een web site voorstelt om op deze manier metainformatie aan de gebruiker door te spelen.
Gebruik makend van de drie schema's (inhoud, structuur en connecterend) van dit conceptueel schema gebruiken we de besproken standaarden om twee style sheets aan te brengen. Deze style sheets zullen, elk vanuit een andere invalshoek, het conceptuele schema aan de gebruiker aanbieden. Zo kan de gebruiker de web site verkennen aan de hand van de conceptuele elementen beschikbaar, de pagina's en bijhorende linkstructuur en waar de conceptuele elementen terug te vinden zijn. We maken gebruik van uitgebreide hyperlinks zodat de gebruiker de web site zowel binnenin het conceptuele schema kan verkennen als van het schema gebruik kan maken om snel in de web site door te dringen.
De style sheets kunnen toegepast worden op elk XML document dat het conceptueel schema van een web site voorstelt en dat voldoet aan de beperkingen opgelegd door het opgenomen DTD.

Op deze manier hebben we de levensvatbaarheid van de nieuwe standaarden aangetoond. Hieraan gekoppeld hebben we ook de bruikbaarheid aangetoond van het conceptuele schema als een medium om metainformatie over web sites door te geven aan gebruikers die het World Wide Web op een efficiënte manier wensen te doorzoeken.

Dankwoord

Graag had ik nog enkele mensen bedankt voor hun steun en raadgevingen bij het tot stand komen van dit werk.

Mijn promotor, Prof. Olga De Troyer, heeft mij uitstekend geholpen bij het afbakenen van het onderwerp en heeft mij waardevol begeleiding gegeven bij het uitwerken ervan.
Tom Decruyenaere, die ik pas leerde kennen in de VUB, wil ik bedanken voor de vele vruchtbare discussies over allerhande informaticaonderwerpen. Ik vond in het onderwerp van zijn thesis een prima toetssteen voor de toepassing in mijn thesis.
Mijn vriendin, An Descheemaeker, wil ik bedanken voor het doorstaan van sommige bovenvermelde discussies en vooral voor het stellen van het goede voorbeeld en de steun bij het maken van dit werk.
Verder dank ik vooral nog mijn ouders, familie en vrienden voor de steun die zij mij leverden bij het afwerken van mijn avondstudies.

Bert

Introduction

The Internet

The Internet grew from an academic network to a distributed library connecting millions of users worldwide. The use of relatively easy protocols and languages like HTML (Hypertext Markup Language) provided for this growth. Companies who have discovered the Internet as a medium to efficiently reach other business partners as well as consumers find that maintaining the information using plain HTML pages requires a lot of resources. Therefore, new and more efficient ways to convey information to communication partners are searched to support the growth of this World Wide Web. XML (eXtensible Markup Language) is one of these evolutionary languages. It allows to separate formatting and structuring of data, so the formatting can be changed without having to alter the structure and vice versa.

Due to this separation of structure and formatting, new techniques are needed to visualize and handle this structured data. Therefore new standards are emerging to be able to use the full potential of XML.

In the expansion of the web a lot of data is put on servers worldwide without any structure, whatsoever. This created the need for possibilities to search information in this distributed library. Search engines, which use full indexing and keyword search, enable this. However, this method of searching not always proofs to be efficient for a user. When one enters a query, a lot of redundant information is supplied by the search engines.
Decruyenaere and De Troyer bring forward a technique to supply both search engines and users with an XML document representing the conceptual schema of a web site. This conceptual schema, based on modeling techniques of Database theory, provides an oversight of the contents and structure of a web site. Search engines could search this XML document to efficiently answer queries and end users could use this conceptual schema to quickly get an overview of the web site involved.

Objectives

We will make a survey of different techniques emerging from the introduction of XML. This will primarily be based upon the recommendations of the World Wide Web Consortium, which plays the role of arbitrator for the different propositions made by industry players.

After a brief introduction to XML, the standards to visualise this XML are discussed. Cascading Style Sheets are a way to display XML without editing the content or order of it. XPath and XSL can help visualise but also change order and content of the original XML document. We say that the source tree, from the original XML document, is transformed into a result tree. This result tree can contain formatting information. In the case of this thesis we transformed the source tree into a tree with HTML formatting.

The XSL-Transformation is applied to the XML representation of a conceptual schema. We defined two style sheets, which approach the data from different viewing angles. These style sheets and their result can be found in Appendix 7 till 10. We extensively probed the data to compose again the conceptual schema from the elements in the XLM document. This way, we offer the user a starting point to explore the web site the conceptual schema is based upon. He/she can browse the schema itself by using the extensive hyperlinking, but can also use the schema to go to the correct web page without traversing all the other pages.

1. Markup languages

1.1. Markup: Structure, content and formatting

In writing this text a lot of formatting has taken place - fonts have been bolded, italicised, enlarged, etc – to produce a clearly structured text with emphasis on certain parts of it. This has been done by selecting pieces of text and by use of menu items to declare what formatting should be applied to the selected text. This formatting has been saved with the text in some kind of binary format, probably proprietary to the producer of the word processor. So we have now a text saved in some form with formatting included. The codes for the formatting are not, at least not in every word processor, visible, and they shouldn't be because of the burden it imposes to the user. "What You See Is What You Get" (WYSIWYG) is the paradigm that relieves the user from looking at formatting codes and trying to figure out how a text will look eventually.

An alternative would have been to declare in human readable form what the text should look like and use some kind of processing to produce a document that displays what the text will look like in published form. The advantage here is that we can save the text in non-proprietary ASCII-format and it remains human readable. The disadvantage is that processing is needed to "see what you will get". This principle of declaring how a text should look like is called *markup*.

Markup – literally meaning "increase" - languages will enhance the text by adding information to the text. This information can then be interpreted by a processor to produce the resulting document. Depending on the particular language the text is surrounded by instructions called *tags*. One tag signals the start of the text being enhanced with information, another tag signals the end of the text fragment. The enhancement of the information is to be found in the tags themselves.

Different kinds of markup are:

- Stylistic or formatting markup: this indicates how the document is to be presented. In HTML for example, the tags <I> for italicising and <B> for bolding are well known tags.
- Structural markup: this informs us of how the document is to be structured. <P> for a paragraph and <DIV> for grouping into block-level flow objects are again examples from HTML.
- Semantic markup: this informs us of the content of the data. <TITLE>, <BODY> are again examples from HTML.

The use of markup languages is mainly textual interchange, but they can also be used with graphics, video, sound and other multimedia elements.

1.2. Some history of WWW markup languages

The first markup languages only included rendition or stylistic markup to the text. An important remark here is that there was a mix of text with rendition information. This prevents the reuse for different media, because rendition for computer screens is different from rendition for paper documents.

Therefore generalised markup languages were introduced. These types of markup languages only describe the structure of the text to make it possible to process the text and the recorded information. The best known example is SGML (Standard Generalised Markup Language).

SGML is a platform-neutral standard for creating documents and information archives. It is a series of rules that everyone can follow in order to make their documents publishable in different media. It also defines a structure for storing information that eases information management and manipulation: it allows very powerful searching, and allows large information repositories to be re-purposed, broken down, and rearranged intelligently into individual documents.

SGML allows the author to define any tag that is needed and this way actually lets authors define their own markup vocabularies (this is the set of tags used). The author can define a formal Document Type Definition (DTD) to check a document on conformity to the defined vocabulary. This way, one can be sure no unknown tags are used. Also the order can be constrained by the DTD.

Because of its inherent flexibility but complexity SGML has been only used in certain application domains. When in 1989 Tim Berners-Lee, a researcher of CERN, was looking for a way information could be shared within the institute using hyper linked text documents he adopted the SGML standard. However, because of SGML's complexity, only a subset of the possibilities was used, by defining a limited set of tags that could be used when marking up documents. This was the development of HTML, which is actually an application of SGML.

The predefined markup vocabulary of HTML contains stylistic, structural and semantic tags and because of good use of SGML the tags were descriptive. Only when the proliferation of HTML pages started and non-conforming vocabularies started to appear a DTD was introduced.
The simplicity of HTML was its greatest strength and attributed to the ever-growing importance of the World Wide Web. Now simple programs were made which could interpret the HTML and produce a formatted document on screen.

The simplicity of HTML, however, prevents authors to make abstractions. One can only say that certain parts should be italic, or that a text fragment is a paragraph. But one cannot tell a browser that a certain paragraph is a chapter of a book or that there is a difference between the italicised book title and the italicised footnote. These abstractions would allow to style documents in a more consistent way: should book titles be bold and italic you do not need to find every book title and put the bold tags around it.

Therefore an SGML concept was introduced to HTML resulting into Cascading Style Sheets (discussed further on). These Style Sheets allow authors to format with greater granularity the

HTML markup. Now a paragraph for example can be classified, and styling can be applied to each class of paragraph.

The growth of the WWW soon also called for the possibility to exchange information and process it without human intervention. The abstractions introduced into HTML were not sufficient to support this exchange, more semantically abstractions were needed. This need for exchange caused the W3C to re-introduce the major strength of SGML: extensibility. This can be viewed as a classical pendulum movement.
Because of the wish to avoid SGML's complexity, the new standard "eXtensible Markup Language" was introduced, which is easier to learn.

2. Extensible Markup Language (XML)

As already explained, markup languages add extra information to data through grouping of the data with surrounding tags. While some markup languages add any of the three kinds of markup information – stylistic, structural or semantic - to the data, XML only defines semantic markup.

2.1.   XML Documents

XML documents must follow certain constraining rules, discussed further on. A document must be "well-formed" and can be validated against a predefined template called Document Type Definition. The different aspects of an XML document are explained in this chapter.

An example of an XML document:

```
<?xml version="1.0"?>
<Person ID="BeLef" >
   <Name>Lefever</Name>
   <First_Name>Bert</First_Name>
<Address>
      <Type_Address >Work</Type_Address >
      <Street>Grote Steenweg</Street>
      <Number>15  </Number>
      <Postal_Code>9840</Postal_Code>
      <City>De Pinte</City>
      <Country>Belgium</Country>
       <Phone/>
</Address>
</Person>
```

2.1.1. The XML Prolog

The prolog is made up out of an XML declaration and a document type declaration, both optional.
The XML declaration is a processing instruction, a statement commencing with ‘<?’ and ending with ‘?>’. This processing instruction tells the browser how to interpret the document being browsed. In the case of XML it also states the version the recommendation of XML that’s being used.

```
<?xml version=”1.0”?>
```

The Document Type Declaration (DTD) is a formalisation of the structure of a document type. It lists the element types available to structure the document and can put constraints on the occurrence and the content of elements and other details of the document structure. This DTD can be explicitly included in the XML document or a reference can point out the place where the definition can be found.

2.1.2. Elements

These are the basic constructs of an XML document. They are built up of a start-tag/end-tag pair and content in between. The extensibility of XML allows one to define own tags with a name describing the content of the element, hence the meta language aspect of XML..

Tags used to markup data must always be used in pairs, start- and end-tag, unless the tag defines an empty element. For a pair to match each other, the case must be respected, because of the case sensitivity of XML.

Correct use:

```
<Name>Lefever</Name>
or
<Empty_element/>
```

Elements can be nested within each other, this way complex constructs can be built. An element can contain other elements. Because of this nesting an XML document can be considered as a tree.

An XML document must be “well formed” according to three simple rules, which must be followed by the defined tags:

- The document must contain one or more elements
- It must contain a uniquely named element, no part of which appears in the content of any other element, known as the root element.
- All other elements within the root element must be correctly nested.

2.1.3. Attributes

Attributes, like an identification (ID), can be used within elements to supply additional information, in the form of properties, about the information included in the tags. The attributes are a part of the start tag of an element.

```
        <Person ID="BeLef" >
```

2.2. *Valid documents*

Because of the extensibility of XML, there is a need to be able to validate the XML document to make sure that sender and receiver use the same structure.

A Document Type Definition (DTD) defines the allowed elements and attributes of an XML document and how they should be used in relation to each other. The DTD is written in a formal notation, Extended Backus-Naur Form, and is used to check the document. If the document conforms to the DTD, which serves as a syntactical and logical schema, the document is said to be valid.

In XML one can define DTD's to be used to check documents . This is in contrast to HTML where the DTD is fixed by the World Wide Web Consortium and therefore the usable tags are fixed. A browser will read an HTML document, as it can do for XML if the DTD is linked, and will validate the HTML document to the standard DTD.

2.3. *Namespaces in XML*

The use of tags to identify elements or attributes works fine when using one or more documents from one author who defined a vocabulary (DTD) consisting of elements and attributes, and uses these consistently. However, the picture gets more complicated in a distributed environment like the Web. When you want to be sure that two tags in two different XML-documents refer to the same type of element (semantically speaking), you cannot assume that both will eventually use the same name for the same tag.
For example, the element <Title> in one document can mean the title of a person, and in another document mean the title of a book.

Therefore, namespaces have been introduced. The purpose of these namespaces is threefold:
1. It enables editors to uniquely identify a tag, using a repository. Anyone referring to the same repository and using the same tag is talking about the same thing.
2. It is not necessary to reinvent a markup vocabulary (the elements and attributes used) when it already exists.
3. The use of namespaces can also educate User Agents, whether human or computer agents. The repository could contain information on how to treat or interpret certain elements or attributes.

Namespaces in XML are identified by *Uniform Resource Identifiers*. This allows each namespace to be universally unique, as it must be if we are looking at element and attribute names from all over the Web.

Declaration of a namespace:

```
        xmlns:[prefix] = “[URI of namespace]”
             example xmlns:xsl='http://www.w3.org/XSL/Transform/1.0'
```

where:

- xmlns tells the XML-processor this statement is about namespaces
- prefix can be any XML-tag and will be used in the document as an alias for the URI of the namespace. When no prefix is supplied, all tags in the document without explicit prefixes will by default map to this namespace.
- URI is the location of the namespace

Referral to a namespace:

```
        [prefix]:[local part]
```

where:

- prefix is an optional prefix to be mapped to the URI. When a prefix is used, it should be declared as a namespace.
- Local part is an element or attribute used in the document

We call a *qualified name* the use of a tag with or without a declared prefix.

An *expanded name* is either

- the type of element, followed by its namespace
- the name of the attribute together with the type of the element, followed by its namespace.

This expanded name is used by the XML-processor to make comparisons.

The use of namespaces is needed for the eXtensible Stylesheet Language (explained later on).

2.4. Visualising XML

XML is about structuring data through use of markup, without adding rendition-oriented markup to the data. This enables reuse of documents for different media and easy maintenance.

An XML document in itself can not be visualised using a browser. Because of the extensibility of tags inherent to XML, the browser cannot predict how an element should be displayed. Therefore one has to externally tell the browser how to display the different elements included in the XML document.

Visualising XML can be done as follows:

- by replacing XML tags by HTML tags manually or semi-automatically using a tool
- with style sheets (e.g. CSS): telling the browser how to display the elements in the document
- by converting the XML into HTML using style sheets (XSLT) or scripting in HTML pages (e.g. VB Script or Java Script).
- by formatting the XML with style sheets (XSL) into formatted objects ready for display.

This thesis will experiment on the display of XML documents using style sheets XSLT. The fourth possibility, using XSL, is at the time of writing still work in progress.

3. Cascading Style Sheets (CSS)

3.1. CSS and HTML

An HTML author has limited control over the rendering of his documents by the browser. There are a number of attributes and elements offering control over alignment, font size, and text colour. Tables can also be used as a means for laying out pages.

A flow object is an object being created when displaying any text or graphic. This object will be "flowed" on-screen and starts, for text, from a single character being grouped together into increasingly larger flow objects, such as a phrase, a paragraph, a page or even a whole document. The HTML author can use the HTML elements to define styling for each flow object.

Style sheets, however, add extra functionality of styling and rendering to the HTML documents. Instead of defining for each flow object how it should look, style sheets allow to style all flow objects in once or with finer granularity to style flow objects of a certain class.

By using Style Sheets, authors can also render a set of documents consistently.

The separation of structural and presentational aspects allows to reduce costs in serving a wide range of platforms and media (text, paper, projector…) and easy change of the looks of a document (for example, when company style changes).

For example, in HTML the tags for headings are limited to a set of predefined headings<H1> till <H6>, varying from most important to least important. Before the advent of style sheets the representation of these tags were fixed, according to the browser being used. Style sheets can alter this, by letting the developer or the user defining styling rules for each header according to platform or media or according to company style guides.

A Style Sheet is a series of styling rules. Each of these rules select an element and declares what the style properties (font, size, colour, flow type…) are for this element. However, the sheet can only add styling to the document, and nothing else. This means that no extra functionality can be added through scripting, for example.

3.1.1. Some important characteristics of Cascading Style Sheets:

Structure of a styling rule:

```
[object(s) to be styled] {([property name]:[property value];)+}

example:
H1, NAME{
      Font:Arial;
      Font-size:16pt;
      Color:red}
```

Thus you can apply a styling rule to several objects at once, and you can define several styling properties in one rule.

Types of selectors:

The first part of a styling rule is the selector, which can vary in form.

1. the simple selector: selects an element based on the element type and/or attribute
2. class selector: increases granularity control over elements by subdividing an element type into several classes. Only the elements within a certain class, defined as an attribute in the element, will be styled according to the defined rule. An example is provided, where element H1 is divided in a class “green_title” and “normal”. Class “normal” has no colour attached to it, so it will be shown in the normal colour for this element, whereas the class “green_title” will be coloured green.

```
<HTML>
      <HEAD>
          <TITLE>Title</TITLE>
          <STYLE TYPE="text/css">
              H1.green_title { color: green }
              H1.normal {}
          </STYLE>
      </HEAD>
      <BODY>
          <H1 CLASS=green_title>Green colour</H1>
          <H1 CLASS=normal>Normal colour</H1>
      </BODY>
</HTML>
```

3. ID selector: using a unique ID to style one individual element would be the same as adding inline styling attributes to this element.

4. Contextual selectors: match an element relative to its parents. In the example, only the *<EM>* elements within *<H1>* elements will be styled according to the defined styling rule. Other *<EM>* elements, not nested in *<H1>* elements can be styled differently.

```
        <STYLE TYPE=”text/css”>
            H1 EM {…}
        </STYLE>
```

Inheritance:

If a styling property is not defined for an element, the value for this property of the parent element can be used.

Not all styling properties are inherited, for an exhaustive list of properties that inherit from the parent, refer to [CSS1].

Formatting model:

CSS1 starts from a simple formatting model; an element will appear either as an inline or as a block-level formatting object. Both are constructed as boxes, placed on the screen, and can be nested.

The main difference between the two is that a block-level object starts and ends with a line break, whereas the inline formatting object can share space with other elements on the same line.

For example, a character is translated into a small box placed on the screen and can be placed in a larger box being a word and so on till you get a paragraph. The character and grouping objects are inline-formatting objects because they share space on the same line(s) whereas the paragraphs do not use the same space. These paragraphs will then be block-level objects.

Conflict resolution = Cascading:

Multiple style sheets can be applied to a document in order to obtain
- modularity, and thus reducing redundancy by combining multiple (partial) style sheets
- author/reader balance, since the user can define a proper style sheet (e.g. for visually impaired persons).

When conflicts arise between multiple style sheets, because they define a styling for the same elements, a cascading order is defined – hence the name - to decide upon the styling [CSS1]:

A general rule of thumb is available:
- If a style rule has been declared as IMPORTANT, it takes precedence over non-important style rule.
  Important declaration:

```
        P{font-weight: bold !important}
```

- For equal style rules, the latter wins or in other words, the rule cascades from top to bottom.
- Remark that a user defined style rule takes precedence over author's style rules if both are of the same importance. User defined style rules are rules included in style sheets, which can be referred at in the browser. It enables visually invalid persons to define a style which allows them to access the information adapted to their handicap, or just lets user define their own preference style. Author style rules, on the other hand, are referenced by or included in the document to be displayed in the browser.

3.2.1. Coupling a Style Sheet to an HTML document

There are a few possibilities to link a Style Sheet to an HTML document:

1. One could include inline styling as an attribute in an element, but then the styling applies only to this element, which does not use the full power of style sheet languages.

```
<P style="font-size: 12pt; color: fuchsia">
```

2. Including it in the document itself through use of the *<STYLE>* tag inside the *<HEAD>* tag.

```
<HTML>
    <HEAD>
    <STYLE>
        H1, NAME{
    Font:Arial;
    Font-size:16pt;
    Color:red}
    </STYLE>
    …
    </HEAD>
    …
</HTML>
```

3. Linking an (several) external style sheet(s) through use of *<LINK>* tag(s).

```
<HTML>
    <HEAD>
    <LINK href="mystyle.css" type="text/css">
    …
    </HEAD>
    …
</HTML>
```

Where:
- *href* is the URL of the style sheet
- *type* indicates the used styling language. User agents which are not supportive of certain styling languages can prevent downloading an unusable style sheet.
- A *media* attribute can be included to signal user agents which style sheet should be applied to which media type.

3.3. CSS and XML

As for adding styling to HTML tags, one can also add styling to XML tags. This way a browser will know how to display an XML document. If no style sheet is supplied to the XML document, the browser can use a default style sheet to display the XML document.

In using a Cascading Style Sheet, one cannot alter the order or the structure of the output of the data included in the XML document. Only in leaving out styling for certain elements, one can slightly change the representation of the document, since the elements without styling will not be displayed. This will be the greatest difference with Extensible Style Sheets, where we can change the structure of the representation at will.

3.3.1. Coupling a Style Sheet to an XML document

1. Inline styling of an individual element can be attained through use of the *style* attribute. But since this includes representational aspects in a document where only semantic data is in place, this is not a good use of technology.

```
        <Postal_Code style="font-size: 22pt; color: fuchsia">9000</Postal_Code>
```

2. One can link a style sheet to an XML document through the use of following processing instruction, where show_xml.css should be replaced by a Unique Resource Identifier:

```
        <?xml:stylesheet href=”show_xml.css” type=”text/css”?>
```

4. XPath

XPath is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. Only XSLT will be explained further on in this document. XPointer allows to address parts of an XML document with finer granularity than the HTML pointer/anchors system.

XPath operates on the abstract, logical structure of an XML document. It models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes.
XPath will use this tree of nodes to match a pattern in an expression supplied by the user. This matching will especially be used in XSLT.

An expression will be evaluated and yield an object, which can be one of four basic types:
- Node-set (an unordered collection of nodes without duplicates)
- Boolean (true or false)
- Number (a floating-point number)
- String (a sequence of characters)

Expression evaluation occurs with respect to a context, for example a specific node (named context node) in the tree.

Expressions can be functions or location paths. The latter will be extremely important in XSLT and is therefore explained more elaborately.

4.1. Location paths

The result of evaluating an expression that is a location path is the node-set containing the nodes selected by the location path. Location paths can recursively contain expressions that are used to filter sets of nodes.

A location path is expressed in a, rather verbose, non-XML syntax, but the syntax is very straightforward and for the more common cases there exist syntactic abbreviations. The location path can be either an absolute path in the document or a path relative to a context node.

An absolute path consists of “/” optionally followed by a relative location path. The “/“ selects the root node of the document as the context node, thus starts always from one absolute point.

The relative location path contains several location steps separated by “/”.
Each location steps selects a set of nodes relative to the context node. The steps are composed together from left to right. Each node in the selection set will be used as a context node for the following location step. The sets of the location step for each context node in the previous selection will be joined together to serve as context nodes for the next location step in a recursive fashion.

4.1.1. Location steps

A location step has three parts:

- An axis, which specifies the tree relationship between the nodes selected by the location step and the context node.
- A node test, which specifies the node type and expanded-name of the node selected by the location step
- Zero or more predicates, which use arbitrary expressions to further refine the set of nodes selected by the location step.

A location step is composed of these parts in the following fashion:

```
        Axis::node test[predicate(s)]
```

The node-set is resulting from the node-set generated from the axis and node-test and then filtered by each of the predicates.

4.1.2. Axes

In XPath the relations between nodes in the tree are used to nominate axes containing nodes meeting the used relation relative to the context node. The terms parent, child, descendant, ancestor, sibling are used as axes describing relationships.

For an exhaustive list of the possible axes, see the W3C Recommendation of XPath [2].

Some important examples:

- The *child* axis contains the children of the context node

- The *descendant* axis contains the descendants of the context node; this is a recursive definition for all children of the children … of the context node.

- The *parent* axis contains the parent of the context node, if there is one.

- The *attribute* axis selects all attributes of the context node.

- The *namespace* axis selects all nodes belonging to the same namespace as the context node if the context node is an element. Else this axis contains the empty set.

4.1.3. Node tests

When an axis contains all nodes meeting the relationship described by the axis relative to the context node, node tests serve as a way to limit the nodes in the axis.

Every axis has a principal node type: element, attribute or namespace.
A node test that is a Qualified name is true if and only if the type of the node is the principal node type and has an expanded name equal to the expanded name specified by the Qualified name.

For example, *child::para* will select all the *para* element children of the context node. *Child* is the axis, with element as principal node type. The node test will limit the elements in the node set to the *para* elements.

The expansion of the Qualified name uses only the namespace declaration of the expression context, without using the default namespace.

The node test * is true for any node of the principal node type.

4.1.4. Predicates

A predicate filters a node-set, generated by axis and node test, and creates a new node-set, which will be the result of the location path.

A predicate can, for example, test on the content of an element or attribute, the position of the node in the node-set. If the predicate expression evaluates to true for a node in the node-set, the node is included in the new node-set; otherwise, it is not included.

The predicate can include functions from the core function library of XPath.

4.2. Core Function Library

Only some of the core functions are explained, most of the explained functions will be used later on in this thesis. For a full description of the core function library, please consult [CD 99].

Node set functions

Some necessary definitions:

- Context size: is the number of nodes in the node set generated by evaluating a location path or a part of it
- Context position: is the position of a node in a node set, ordering of the node set taken into account.
  The first node in a node set has position=1.

Function: *number* last() ; returns a number equal to the context size from the expression evaluation context or, in other words, the position of the last node in the node set generated by the evaluation of the expression.

Function: *number* position() ; returns a number equal to the context position from the expression evaluation context.

Function: *number* count(*node set*) ; returns the number of nodes in the argument node-set

Function: *string* name(*node set?*) ; returns a string containing a Qualified name representing the Expanded name of the node in the argument node set that is first. It will use the namespace declarations in effect on the node to formulate the Qualified name.

Also String, Boolean and Number functions are available.

4.3. Some examples of location paths and possible abbreviations

- *Child::para* selects the *para* element children of the context node (equals *para* because *child* is the default axis)

- *Attribute::name* selects the *name* attribute of the context node (equals *@name*)

- *Child::/para[position()!=last()]* selects all *para* element children of the root node, except for the last *para* element (equals */para[position()!=last()]*)

- *Child::*/child::para* selects all *para* grandchildren of the context node (equals **/para*)

- *Self* selects the context node (equals .)

- *Parent::attribute::lang* selects the *lang* attribute of the *parent* node of the context node (equals *../@lang*)

- *Following-sibling::*/children::chapter[attribute::type=”Introduction”]* selects all *following siblings* which have *chapter* element children which have an *attribute* type equal to *Introduction*
  (equals to Following-sibling::*/chapter[@type=”Introduction”])

5. Extensible Style Sheet Language (XSL): double standard

XSL is a monumental effort to define a language to translate any class of arbitrarily structured XML documents into a formatted, paginated, styled and laid out document for any sort of presentation medium, either electronical user agent as browser or PDA or physical in paper format.
Since the translation is done in two major steps, the World Wide Web Consortium has defined two separate standards, which address a different topic:

- XSLT which is a language for transforming XML documents
- XSL which is an XML vocabulary for specifying formatting semantics.

5.1. Extensible Style Sheet Language Transformation (XSLT)

This standard is a W3C recommendation, and so it can already be used for transformation of XML documents, since implementations already exist at time of writing.

XSLT enables an author to transform a given XML document, called the source tree, into another XML document, called the result tree. During this process the author has the possibility to enhance the XML tree with generated constructions like text or another XML fragment and add functionality through scripting or styling information. The latter can be ordinary HTML markup or in the future XSL formatting markup.

5.1.1. XSLT namespace

XSL makes full use of XML namespaces for use of uniquely defined elements. The namespace to be used is http://www.w3.org/1999/XSL/Transform.
Each XSLT element will be preceded by the prefix coupled to this namespace. In the W3C Recommendation and in this thesis, the prefix *xsl* is used for the sake of clarity.

5.1.2. Stylesheet element

An XSL style sheet is an XML document that must be represented by a <*stylesheet*> element. For the full syntax and an exhaustive list of allowed top-level elements, refer to [Cl 99]. Only the most common top-level elements will be described here.

```
<xsl:stylesheet
version= number that allows compatibility testing, for now only version 1.0 exists
xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”>
<!-- any number of allowed top-level elements -- >
</xsl:stylesheet>
```

5.1.3. Linking the XSL style sheet to an XML document

The linking occurs through a processing instruction in the XML document, which links to an external or internal style sheet. If an embedded style sheet is used, the reference should be to an anchor identified by an *id* in the *<stylesheet>* element embedded in the XML document.

```
<?xml-stylesheet type="text/xsl" href="my_style.xsl" ?>
or
<?xml-stylesheet type="text/xsl" href="#my_style " ?>
<xsl:stylesheet id=”my_style”
            version=”1.0”
            xmlns:xsl=”http://www.w3.org/1999/XSL/Transform”>
```

5.1.4. Combining style sheets

XSLT provides two mechanisms to combine style sheets:

- An inclusion mechanism that allows style sheets to be combined without changing the semantics of the style sheets being combined
- An import mechanism that allows style sheets to override each other.

Inclusion

The construct:

```
<xsl:include
href = URI-reference/>
```

The include-element will be replaced by the templates defined in the style sheet referenced by the URI. The way the template rules are processed is not changed, when included.

Importing

The construct:

```
<xsl:import
href = URI-reference/>
```

The import-element will be replaced, as for the include-element, by the templates defined in the style sheet referenced by the URI. However the template rules in the importing style sheet take precedence over the template rules and definitions in the imported style sheets.

5.1.5. Templates: the basic construct

XSLT describes rules for transforming a source tree into a result tree using template constructs. The first halve of this construct is the selection of elements and attributes using XPath-patterns

The second halve dictates the XSL-processor how to act on the selected item. This can range from adding styling to manipulating, like copying child, elements. Since the XSL-type of styling is still in draft version, the styling used to display the use of XSLT will be HTML-conversion.

An example:

```
<xsl:template match=”/Person”>
    <H1>This is a person element</H1>
</xsl:template>
```

/Person is the XPath pattern to select all *Person* elements contained directly under the root-node of the XML document. This is a simplified example in that only a fixed text with HTML markup will be shown for each *Person* element.

The template applies to all elements matching the pattern relative to the context node, which can be selected by another template rule. This way nesting of template rules is one way to handle all child elements of the context node.

```
<xsl:template match=”/”>
    <xsl:template match=”Person”>
        <H1>This is a person element</H1>
    </xsl:template>
    <xsl:template match=”Person/Name”>
        <H2>that contains a name</H2>
    </xsl:template>
</xsl:template>
```

This example matches first the *root* element (always included in the XML document) and will handle all *Person* child elements within the root element. The template for *Person/Name* does act the same as a nested template for *Name* within the *Person* template.
Remark that the template will only be applied if elements match the pattern.

The construct:

```
<xsl:template
    match = pattern
    name = Qualified name
    priority = number
    mode =Qualified name>
    <! -- Content -- >
</xsl:template>
```

The *name* attribute allows to assign a name to the template, so this template can be called.
Priority will establish an order for conflicting rules.
Mode allows several templates to be defined for a similar pattern, but by supplying a mode when applying templates allows to apply the correct one.

5.1.6. Applying templates

An alternative to nesting templates to handle the child elements of the context node, is to use the construct *apply-templates* within a certain context.

The construct:

```
<xsl:apply-templates
      select = node-set-expression
      mode = Qualified name>
      <! -- Content -- >
</xsl:apply-templates>
```

When no select attribute is supplied, simply all templates applying to child elements will be invoked.
The *select* attribute enables the author to only apply the templates for this specific node-set. The node-set is specified using XPath location paths and can select any element addressable through a location path relative to the context node, thus more than just the descendant elements.

The templates to handle the elements in the node-set do not have to be nested in any other template, but are top-level elements of the <*stylesheet*> node.

Conflict resolution

The author is able to supply a priority to a specific template.
If a template does not have the priority attribute filled in, one is calculated from the specificity of the pattern match. The priority can be either a positive or a negative number. The most common kind of pattern (a pattern that tests for a node with a particular type and a particular expanded-name) has priority 0. Less specific patterns receive negative priorities, more specific patterns receive positive priorities. [Cl 99]

Modes

A mode defined on a template allows an element to be processed multiple times, each time producing a different result.

The <*template*> element has an optional mode attribute, which can be used by the <*apply-templates*>. This way only the template for the *select* pattern that has the same mode as defined in the apply-templates construct will be applied.

5.1.7. Calling templates

Templates are invoked only for the elements matching the pattern. By calling a template however, one can invoke this template without matching of the pattern is done. More, only one of match and name attribute is required for a template.

The construct:

```
<xsl:call-template
      name = Qualified name>
      <! -- Content -- >
</xsl:call-template>
```

5.1.8. Creating the Result Tree

XSLT allows to transform an XML document, regarded as source tree, into a new XML document, regarded as result tree, where the result tree can differ from the source tree in order or elements, number of elements, scripting, styling etc.

Literal result elements

An element that appears in a template and that does not belong to the XSLT namespace or a namespace defined as an extension namespace [Cl 99], will be copied entirely to the result tree, including content, namespace and attributes other than XSLT namespace attributes.

Creating a new element

One can create a new element in the result tree, using following construct:

```
<xsl:element
      name = {Qualified name}
      namespace = {URI-reference}
      use-attribute-sets = Qualified name>
      <! -- Content -- >
</xsl:element>
```

The curly braces around Qualified name and URI-reference signal an attribute value template which indicates that one can use a literal string here or an expression that evaluates to a string. In practice, this means that we can use the value of a variable or parameter for the attribute value of name and namespace. An alternative for a variable is the contents of an element through use of a location path.

Creating a new attribute

One can add attributes to elements in the result tree, either created by literal result element or by the *<xsl:element>* construct.

```
<xsl:attribute
      name = {Qualified name}
      namespace = {URI-reference}>
<! -- Content -- >
</xsl:attribute>
```

Adding an attribute to an element replaces any existing attribute of that element with the same expanded-name.

Named Attribute Sets

An attribute set can be used to create a series elements, which all contain the same set of attributes. The attribute set can also be used to create other attribute sets.

The construct:

```
<xsl:attribute-set
      name = {Qualified name}
      use-attribute-sets = {Qualified names}>
      <! -- Content xsl:attribute* -- >
</xsl:attribute-set>
```

Creating Text

A template can also contain text nodes, but this can be replaced by just the literal value of the contents of the text node.

```
<xsl:text
      disable-output-escaping=”yes”|”no”>
      <! -- Content -- >
</xsl:text>
```

Two adjacent text nodes are merged into one text node.

Creating Processing Instructions

```
<xsl:processing-instruction
      name = {ncname}>
      <! – Content -- >
</xsl:processing-instruction>
```

The attribute value template for ncname must be instantiated to a string that is a Processing Instruction Target.

Copying

The *copy*-element provides an easy way of copying the current node, including namespace nodes of the current node. However, the attributes and children of the node are not automatically copied. The content of the *copy* element is a template for the attributes and children of the created node.

```
<xsl:copy
      use-attribute-sets = Qualified names>
      <! – Content: template -- >
</xsl:copy>
```

Computing Generated Text

The value-of element creates a text node in the result tree. The string-expression in the select attribute evaluates to an object that is converted to a string.

```
<xsl:value-of
      select = string-expression
      disable-output-escaping = “yes”|”no”/>
```

An example of the evaluation of a string expression is the extraction of a variable value:

```
<xsl:variable name=”a_variable”>value of variable</xsl:variable>
<xsl:value-of select=”$a_variable”/>
```

This example returns text node “value of variable” in the result tree.

Attribute Value Templates

An expression in an Attribute Value Template is surrounded by curly braces ({}). The expression together with surrounding curly braces is replaced by the result of evaluation the expression and converting the resulting object to a string as if by a call to the string function. Multiple curly braces in an expression are replaced by one pair.

The expression can extract the value of a variable or even select the contents of an element using an XPath location path.

Numbering

XSLT allows inserting a formatted number into the result. The number can be specified by an expression, for example the function *position()* to number each element in a given node-set. The format of the number can be supplied in an example, e.g. format = “A.1” will number following this pattern.

5.1.9. Repetition

Repetition can be obtained by using following construct, the template will be applied to each node in the node-set.

```
<xsl:for-each select = node-set-expression>
      <! -- Content -->
</xsl:for-each>
```

5.1.10. Conditional processing

Two constructs support conditional processing in a template

- xsl:if
- xsl:choose

```
<xsl:if
      test = boolean-expression>
      <!-- Content -->
</xsl:if>
```

```
<xsl:choose>
<! -- Content: (xsl:when+, xsl,otherwise?) -- >
</xsl:choose>
```

```
<xsl:when
      test = boolean-expression>
      <!-- Content -->
</xsl:when>
```

```
<xsl:otherwise
      <!-- Content -->
</xsl:otherwise>
```

5.1.11. Sorting

Sorting can be applied to node-sets by adding *xsl:sort* elements as children to an *xsl:apply-templates* or *xsl:for-each* element. Each element defines a sort key, with decreasing priority.

Example:

```
<xsl:for-each select=”Person”>
      <xsl:sort select=”Person/Name”/>
      <! -- Content: template -- >
</xsl:for-each>
```

This will apply the template to all Person elements in the selection, ordered on the primary key: the Name of the Person.

5.1.12. Variables and Parameters

Variables and parameters are names that may be bound to a value. This value can be an object of any of the types that can be returned by expressions.
The difference between the two possibilities is that a parameter can have a default value for the binding, while a variable not.
A parameter may be passed to other templates and can in this manner override the default value within a template.

Construct:

```
<xsl:variable name = Qualified name
      select = expression>
      <! --  Content -->
</xsl:variable>
```

```
<xsl:param name = Qualified name
      select = expression>
      <! --  Content -->
</xsl: param>
```

The select binds the value to the name, but the binding can also be done with the content of the variable or parameter element.

Passing parameters:

```
<xsl:with-param name = Qualified name />
```

The binding of a variable or parameter has a scope within the template in which it has been defined. If a variable or parameter has been declared as top-level element, this is as child of the stylesheet element, then the value is known within the entire style sheet.

5.2. Extensible Style Sheet Language

The formatting part of XSL is at the time of writing in Draft version, and is still susceptible to changes. Therefore, only the conceptual model will be discussed.

The XSL formatting part is a monumental effort to enhance the capabilities of CSS level 1 and the added features of level 2. It has been developed with XML in mind, while CSS is applicable to XML but was developed with HTML in mind.

It tries to address several problems other than mere layout:

- Through use of the XSL Transformation and XPath, a powerful control on how portions of the content should be presented and what properties are associated with these portions, is attained.
- Internationalisation: different modes of writing, for example Japanese and Arabic writing. To enable different writing-modes, XSL leaves the concept of above, under, left and right to indicate position and introduces the concepts before, after, start and end. Furthermore, the line-progression and block-progression mode is introduced. Line-progression states how one line evolves (Western writing: from left to right, Japanese top to bottom) and block-progression how lines evolve (Western: top to bottom, Japanese right to left).
- Pagination: The model for pagination extends what is currently available in CSS2, and in turn is extendable to page structures beyond the simple page models. XSL tries to address layout issues for both scrollable windows and pagination in an analogous fashion. This is possible because there is a correspondence between a page with multiple regions, such as a body, header, footer, and left and right sidebars, and a Web presentation using "frames".
- Linking: XML has no built in notion of hypertext linking. XSL addresses the problem of displaying the link information (including changing of state, when followed) and the following of the link itself.
- Hyphenation over lines when a word has to be broken up
- Aural markup for speech-enabled browsers.

5.2.1. Steps in XSL formatting

1. During the XSL Transformation of source XML document to result XML tree, the XSL style sheet can include formatting semantics

2. The formatting step creates from the XML tree with formatting semantics an area tree, this is a tree with geometric areas. The geometric areas are positioned on a sequence of one or more pages (a browser typically uses a single page). Each geometric area has a position on the page, a specification of what to display in that area and may have a background, padding, and borders. This formatting contains three substeps:

   2.1. The result tree is objectified into formatting object nodes. The classes of formatting objects include typographic abstractions such as page, paragraph, table etc. Finer control over these abstractions is provided by a set of formatting properties, such as for indentation, word- and letter-spacing, hyphenation… In XSL, formatting objects and properties provide the vocabulary for expressing formatting intent of the author.

   2.2. Refinement of the formatting object tree consists of mapping of the formatting properties into traits. It makes sure that all the properties are translated into characteristics more closely connected to the geometric areas. If shorthand notation was used for the properties or inheritance from containing formatting objects was implied, the refinement will make this more explicit.

   2.3. Generation of the geometric areas in a tree based upon the formatting objects. The traits for each formatting object control how the areas are generated.

3. Rendering takes this area tree and represents this on the relevant medium, such as a browser window or sheets of paper.

5.2.2. Formatting objects

The author of an XSL style sheet will state there the transformation of the source tree and add the wanted formatting, using elements and attributes of the XSL namespace. These will be converted by the formatting-engine into formatting objects when reading and parsing the result tree.

Formatting objects exist for pages, page sequence and layout within a page, using regions. Other formatting objects handle tables, lists, links, …
Formatting properties exist for each formatting object and for common characteristics.

5.2.3. Geometric areas

A formatting object can generate on or more area's in the area tree. Each area is associated with a rectangular portion of the output medium, whether the formatting object is a table or a character.

Each area has a set of traits, this is a mapping of names to values. These traits can either express formatting purposes or defining constraints, then they are called **formatting traits** or they can be used for rendering, thus **rendering traits**.

5.3. CSS contra XSL

Both styling sheets can be used to visualise XML. However, XSL has the invaluable characteristic of being able to transform the source tree into a totally different result tree. Next to this it adds a lot of non-styling possibilities, and thus extends the capabilities of CSS.

Since not all HTML pages will be replaced by XML pages, each has its own use and economic justification, CSS is not obsolete. Hence, the W3C development of CSS3, the third version of the styling mechanism.

Since XSL has not been implemented in full yet, CSS can be used to style the result tree after transformation. This can be done either by styling the HTML included in the result tree or by styling the resulting XML tags.

6. Conceptual modelling of web sites

6.1. The need for meta-information on the web

The World Wide Web more and more plays an important role in our information society. The collection of information is spread over thousands of servers all over the world. This pile of data is connected by hyperlinks making it possible to go from one web site to another, jumping from one subject to the next one.

Most of this data is being represented in documents in HTML available for the public. However, while HTML is easy to use for data representation, the possibilities in creating structure for this data is limited. This is the main reason why the WWW is, next to this gigantic resource of information, a chaotic pile where the “lost in hyperspace” syndrome frequently occurs. Since people *hyperjump* from one location to another in their quest for information, it remains less clear which source of information they are looking at or where it is kept.

To search this distributed library engines have been created, called search engines. These will propose resources meeting search criteria demanded by the user. These search engines (e.g. Yahoo, Altavista) use full indexing and meta-information for selecting the resources that should meet the criteria specified by the users. The problem today, is that one is overwhelmed with sites were the selection subject is mentioned and it is difficult to select the site which meets the criteria best. The introduction of meta-information, primarily keywords included in the HTML headers, already helped with this problem
Still, it is only by looking at the titles of sites that one can vaguely classify the sites into less or more interesting, and the ultimate decision can only be taken by actually visiting the site and browsing through it.

Browsing a web site means that one walks through the web site making use of the hyperlinks spread throughout the site linking one page with the other. These pages can be contained within the web site itself, these are internal links, or can remain outside the web site, thus being external links. Each page can deal with the central subject of the site, and will probably handle subtopics. This means that the user must spend a lot of time before he has a comprehensive overview of the content and structure of the web site. It would be much more efficient to supply the end-user with the structure and a survey of a web site.

This overview on structure and contents can be supplied by many possible techniques ranging from menus in a frame (mostly on the left side of the web site), sitemaps, topic maps to structured maps [Dec 99], [DD2000]. These techniques are either intelligently or rather laboriously and provide from mere structure till relationships between topics.
Also interesting in the area of improvement of search results is the development of Resource Description Framework (RDF) which provides for a framework to include meta-information in resources. This meta-information is machine-readable and can therefore be understood by search engines.

6.2. A conceptual Web site Schema for End-users

One of the major objectives of this thesis lies in the representation of the structure and survey of a web site based on the conceptual modelling of a web site provided by the work of Decruyenaere and De Troyer [Dec99], [DD2000]. This overview 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.

The overview is based on the technique of *Conceptual Schemas*, much used in the area of database design. The conceptual schema of a web site was defined by means of a conceptual "meta" schema, which defines the concepts and their relationships to be used in the definition of the conceptual schemas of web sites. This conceptual "meta" schema has been modelled in a Binary Role Model [Hal 95], which allows for a graphical representation. This BR-schema has then been translated into an XML Document Type Definition (DTD), which is a more suitable representation technique for the WWW. Now, this DTD can be used to represent the conceptual schema of any web site as an XML document.

Search engines can inquire this XML document to suggest more precisely web sites to a user searching the web. This can be done because the engines can go beyond mere indexing and using keywords, but query the concepts of a web site and its relationships.

End-users visiting a web site can use this XML document, in any representation, to efficiently obtain an overview of the structure and concepts of the web site. We will suggest some useful style sheets, which will represent the conceptual schema.

Decruyenaere and De Troyer divided their conceptual web site into three main parts
The first part of the schema – the *content schema* - describes what kind of information is available in the web site. The second – the *structure schema* - describes the page and link structure of the web site, while the third part describes 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, hence being the *mapping schema*.

The process for arriving at a conceptual schema is discussed extensively in [Dec99] and [DD2000]. Hereafter, you will find the description of the conceptual meta schemas because they define the concepts used in the conceptual schemas and are needed to understand what the overview involves. These concepts were translated into the DTD following an algorithm defined by Decruyenaere and will return in the XML documents as the entities.

6.2.1. From Conceptual Meta Schema to DTD

The basic concepts for the three schema parts (content schema, structure schema and mapping schema) of a conceptual web site schema can be defined using 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*. This meta schema has been represented with an information modelling technique from the database community, the Binary Role Modelling approach. The basic building blocks of the BR Model are Object Types (OT), graphically represented as circles, and binary relationships composed of two roles. These relationships are 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.

Because the conceptual web site schema is composed of 3 sub-schemas 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.

6.2.1.1. The Content Meta Schema

The different conceptual 'elements' and their (conceptual) relations to each other are defined in this part, and they will be used to model the content of a web site (figure 1).

The conceptual schema will primarily show the object types and the relationships between these object types, but can also contain 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 relationships between object types. E.g. a company's web site can contain information about products (object types) but probably also information like a mission statement, an organisation diagram, a route description. These last are single objects not belonging to any object type.
A conceptual schema can also include some instances of object types explicitly. For example, it could be useful to include the names of painters in a schema of a web site on painters, but less useful to include all names of paintings of these painters. The level of abstraction should be a trade-off between completeness from user view point and ease of overview.

Building blocks of the content schema being represented by object types in the content meta schema:

- Object types
- Relation types
- Object type instances
- Relation type instances
- Objects
- Relations

ObjectType and *RelationType* are modelled as subtypes of *ConceptType*, as *Object* and *Relation* are subtypes of *Concept*. These supertypes are useful in the Mapping Meta Schema. *ObjectInstance* is a subtype of *Object*. However, not all objects are instances of *ObjetType*.

Relations are constructed out of objects playing a role in the relation. Therefore relation (type)s exist out of role (type)s. For example, the object type 'Painter' plays the role of 'paints' in the relationship with painting, which plays the role of 'is painted by'. A *RoleType* between two *ObjectTypes* creates a *RelationType*. Two *ObjectInstances* correspondingly share a *RelationInstance*. Also *Objects* can share a *Relation*. Remark that for all *RelationType*, *RelationInstance* and *Relation* there are only two 'players' because of the binary character of the 'relation'.

Objects can be structured (*StructuredObject*), constructed out of other objects or just simply elementary (*ElementaryObject*). There are also basic classifications of objects into Text, Graphic and Video, but these could still be extended.

6.2.1.2. The Structure Meta Schema

This part defines the different conceptual ‘elements’ and their relations to each other needed to model (at a conceptual level) the structure of a web site (figure 2).

A web site is built up of pages with links between them, therefore *Page* and *Link* are two of the conceptual building blocks of a structure schema and are two OTs in 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 ot belonging to the web site is an *ExternalPage*.

Larger web sites can contain a collection of pages all resembling in structure or representation, but above all handling the same type of information. Therefore a generic structure for these collection of pages can be defined called *PageProtoType* with corresponding *LinkProtoTypes*.
A page can then be an instance of a page prototype. For example, the site on painters could contain for each painter a page with all the paintings of the respective painter. The links between page prototypes are then called *LinkProtoTypes.* The actual instantiation of *LinkProtoTypes* and *PageProtoTypes* are respectively *LinkInstances*, subtypes of *Link*, and *PageInstances*, subtypes of *Pages.*

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. Their corresponding page prototypes are tailored page prototypes. In dynamic pages the content may change depending on the navigation, e.g. a shopping basket page.

6.2.1.3. The Mapping Meta Schema

Describing what content and what structure a web site has, leaves out a third major issue, which is where to find the content in the structure. Therefore, a mapping meta schema (figure 3) has been proposed specifying which *Concept(Type)s* can be found on what *Page(ProtoType)s.*

Three relations between content and structure have been defined:
- between *Concept* and *Page* to indicate the pages on which a particular concept can be found.
- between *ConceptType* and *PageProtoType* to indicate that a page prototype is built for a certain concept type.
- between *ConceptType* and *Page* to indicate the pages that provide an overview/index of the population of a concept type.

6.2.1.4. The Conceptual Schema DTD

The binary relationship model, describing the schema, is not convenient for transporting over HTTP. Therefore it has to be transformed into XML for the schema and for the meta schema into a DTD. Like the meta schema is defining the concepts and the structure to be used in the schema, the DTD defines the allowed elements and attributes and their relative order for the XML document.

The transformation algorithm is described in [Dec99]. The main idea is that the OTs in the schema are defined as elements in the DTD. Also roles are defined as elements, whereas links, to for example relations, are translated into IDs in attributes of these elements.

7. Case Study: DINF homepage

7.1. Introduction

We use a web site of the VUB to demonstrate the capabilities of XSLT to represent the conceptual schema discussed in previous chapter. The site contains links and information available for students of the department of Computer Science at the VUB. This site contains some excellent examples of dynamic PageProtoTypes and static Pages.

Since the conceptual schema of this web site will contain three parts, these three parts will be discussed in this chapter. First the content will be pointed out, then the structure together with the mapping will be described. Next the conceptual schema will be transformed into an XML document using the algorithm developed in [Dec99].

7.2. Content

The web site primarily functions as a reference for co-ordinates (telephone, fax, email, location, and homepage) of department, research groups and of staff members of the research groups. Since these objects all share the same properties we will model them as ObjectTypes.

To model the individual telephone, fax, etc we would need ObjectInstances of these ObjectTypes. However, these ObjectInstances will not be contained within the conceptual content schema to prevent a clogging of the schema. This is clearly a design choice, because we could do this for subgroups. For example, we could record the ObjectInstances of the chairs of the research groups but not of all members of the research groups. This would enable the user to search with faster result the pages on which the chairs have information on.

Next to the co-ordinates, we can also find the courses of the different academic years and different disciplines, and who teaches them. For some of these courses, there are also on-line textbooks and tutorials.

Also information for working students can be found and information on a masters in science. And finally, some general links are included to guide the student looking for more information.

In figure 5, you will find the content schema.

7.3. Structure of the web site

In this chapter you will find the discussion of the different pages with the mapping to their content and linking to other information.

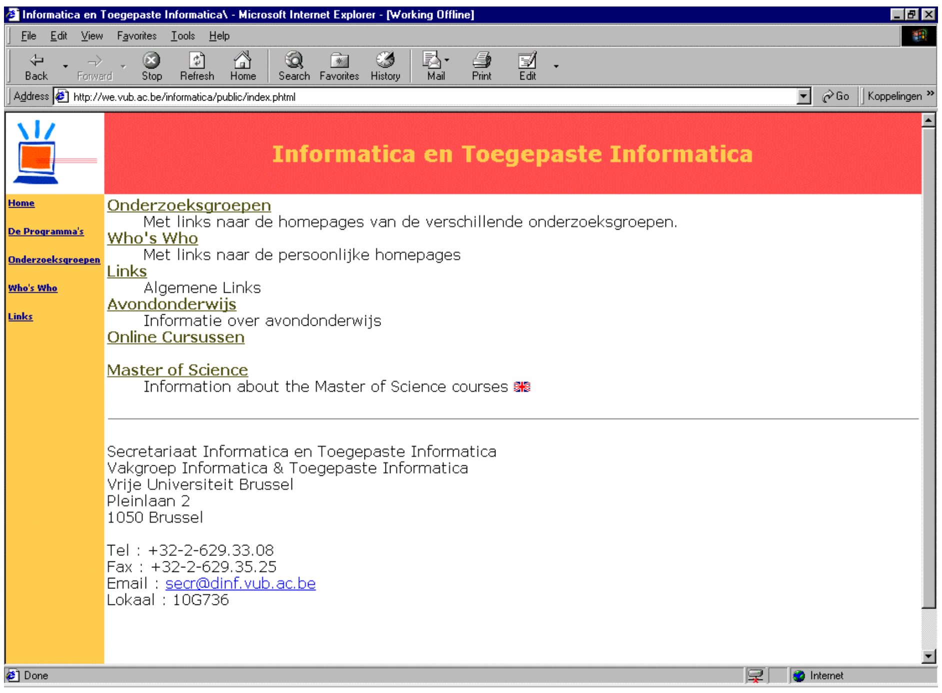
Pages on the web site:
- Computer Science
- Research groups overview
- Research group
- Who’s who

- Staff member
- Links
- Evening students
- On-line tutorials
- Masters of Computer Science
- Course page
- Academic year/discipline page

The navigational links in the left frame of each page will be neglected in the structure schema. We consider these links not as part of the individual pages but as a structural part of the web site.

7.3.1. Computer Science

The start page (http://we.vub.ac.be/informatica/english) which we will call the Computer science page contains links to the different subparts of the site. While we consider the Computer Science Department as an ObjectInstance, we will not regard the page as a PageInstance of a Department-PageProtoType. This is an arbitrary decision.


Informatica en Toegepaste Informatica\ - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://we.vub.ac.be/informatica/public/index.phtml
Go Koppelingen
Home
De Programma's
Onderzoeksgroepen
Who's Who
Links
Done
Internet

Informatica en Toegepaste Informatica

Onderzoeksgroepen
Met links naar de homepages van de verschillende onderzoeksgroepen.
Who's Who
Met links naar de persoonlijke homepages
Links
Algemene Links
Avondonderwijs
Informatie over avondonderwijs
Online Cursussen

Master of Science
Information about the Master of Science courses

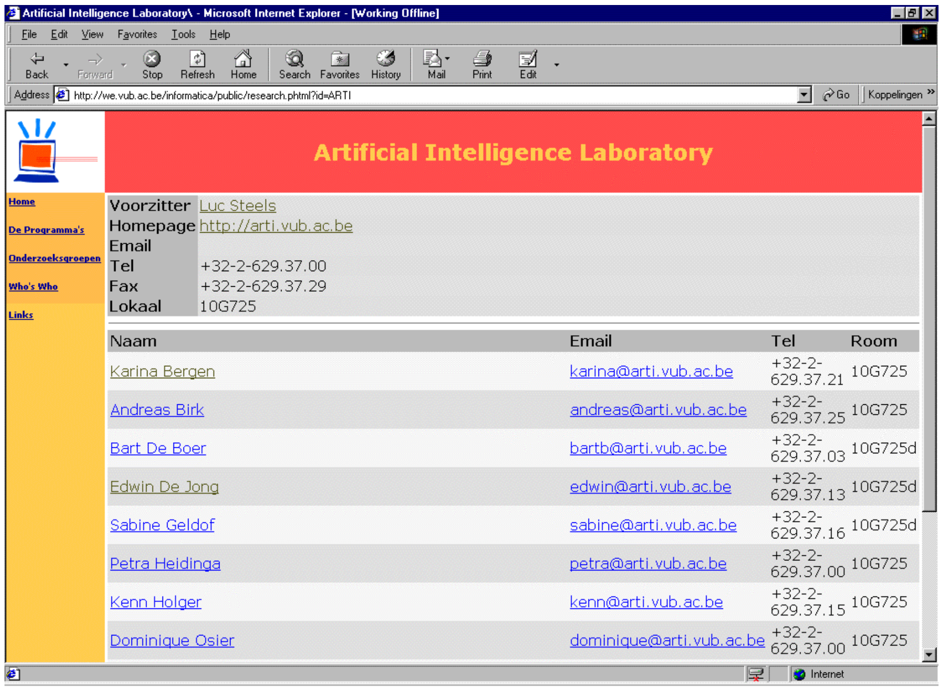
Secretariaat Informatica en Toegepaste Informatica
Vakgroep Informatica & Toegepaste Informatica
Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussel

Tel : +32-2-629.33.08
Fax : +32-2-629.35.25
Email : secr@dinf.vub.ac.be
Lokaal : 10G736

Page contains :
- Department
- Links to other pages
- Co-ordinates of department

7.3.2. Research groups: overview and detail

The Research groups overview page only contains links to all Research groups within the Department. The research group pages contain the co-ordinates of all Staff members of this research group, including the chair of the research group.


Artificial Intelligence Laboratory\ - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://we.vub.ac.be/informatica/public/research.phtml?id=ARTI
Go Koppelingen
Home
De Programma's
Onderzoeksgroepen
Who's Who
Links
Internet

Artificial Intelligence Laboratory

Voorzitter	Luc Steels
Homepage	http://arti.vub.ac.be
Email	
Tel	+32-2-629.37.00
Fax	+32-2-629.37.29
Lokaal	10G725

Naam	Email	Tel	Room
Karina Bergen	karina@arti.vub.ac.be	+32-2-629.37.21	10G725
Andreas Birk	andreas@arti.vub.ac.be	+32-2-629.37.25	10G725
Bart De Boer	bartb@arti.vub.ac.be	+32-2-629.37.03	10G725d
Edwin De Jong	edwin@arti.vub.ac.be	+32-2-629.37.13	10G725d
Sabine Geldof	sabine@arti.vub.ac.be	+32-2-629.37.16	10G725d
Petra Heidinga	petra@arti.vub.ac.be	+32-2-629.37.00	10G725
Kenn Holger	kenn@arti.vub.ac.be	+32-2-629.37.15	10G725
Dominique Osier	dominique@arti.vub.ac.be	+32-2-629.37.00	10G725

Page contains :

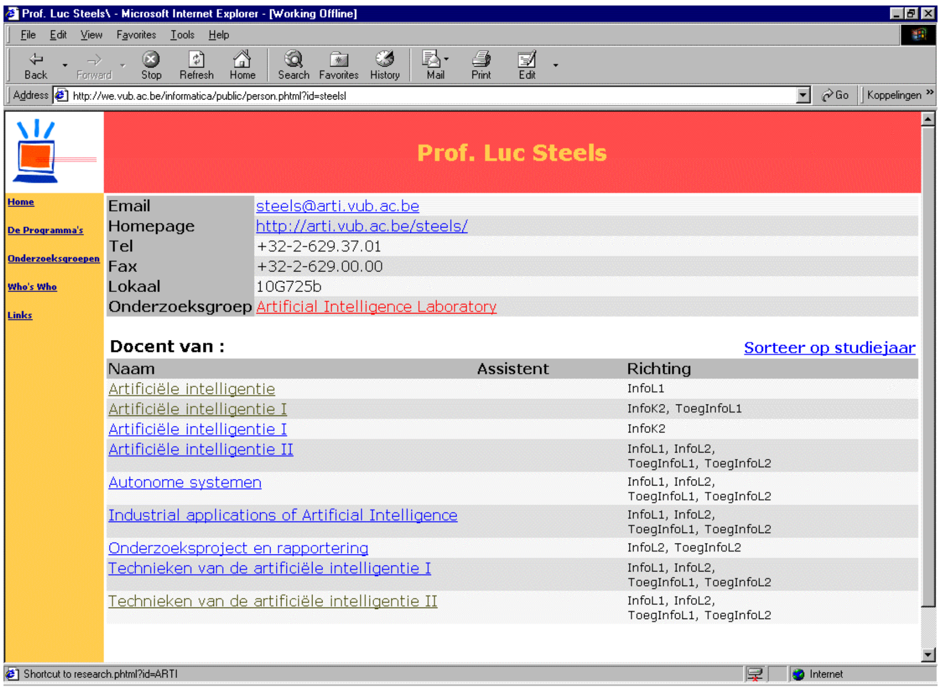
- Research group
- Co-ordinates of research group
- Chair
- Link to homepage of research group
- Staff members
- Co-ordinates of Staff members
- Links to Staff member pages

The page contains links to the homepage of the research group and to the Staff member pages of all members of the research group. Also links to the email addresses are included. We will not consider the link of an email address differently than a link to another page, because clicking on the address has the action of opening the default email client with the email address as addressee. We see this as an OutSideLink with the tailored external Page being the email client window.

7.3.3. Who’s who and Staff member pages

The who’s who is a dynamic – with information from a small database - Page including all Staff members of the department. These Staff members are not grouped according to the research group but to their function, this is lecturer, assistant, researcher or administrative or technical worker. For each Staff member the phone, email with link and a link to their personal page is included.

The Staff member page is a page detailing the co-ordinates for the Staff member and including all courses the Staff member co-operates on, either lecturing or assisting.



Prof. Luc Steels\ - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://we.vub.ac.be/informatica/public/person.phtml?id=steelsl
Go Koppelingen
Home
De Programma's
Onderzoeksgroepen
Who's Who
Links
Shortcut to research.phtml?id=ARTI
Internet

Prof. Luc Steels

Email	steels@arti.vub.ac.be
Homepage	http://arti.vub.ac.be/steels/
Tel	+32-2-629.37.01
Fax	+32-2-629.00.00
Lokaal	10G725b
Onderzoeksgroep	Artificial Intelligence Laboratory

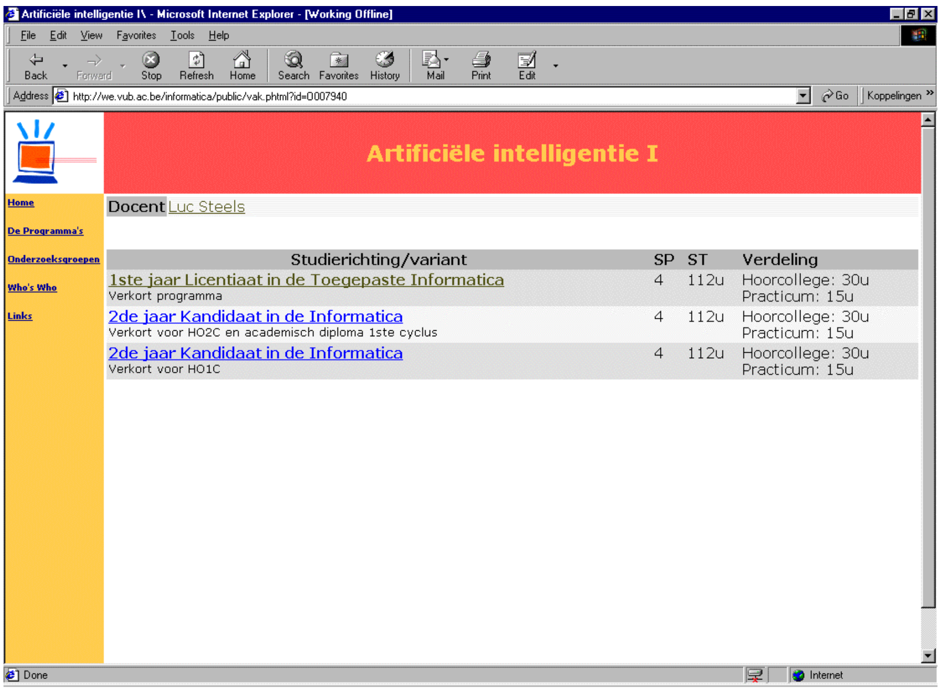
Docent van : Sorteer op studiejaar

Naam	Assistent	Richting
Artificiële intelligentie		InfoL1
Artificiële intelligentie I		InfoK2, ToegInfoL1
Artificiële intelligentie I		InfoK2
Artificiële intelligentie II		InfoL1, InfoL2, ToegInfoL1, ToegInfoL2
Autonome systemen		InfoL1, InfoL2, ToegInfoL1, ToegInfoL2
Industrial applications of Artificial Intelligence		InfoL1, InfoL2, ToegInfoL1, ToegInfoL2
Onderzoeksproject en rapportering		InfoL2, ToegInfoL2
Technieken van de artificiële intelligentie I		InfoL1, InfoL2, ToegInfoL1, ToegInfoL2
Technieken van de artificiële intelligentie II		InfoL1, InfoL2, ToegInfoL1, ToegInfoL2

Page contains :

- Staff member
- Co-ordinates of Staff member
- Link to personal homepage
- Link to research group page
- Courses with links
- Assistents for the courses
- Academic years/disciplines

7.3.4. Course page and academic year page


Artificiële intelligentie I\ - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://we.vub.ac.be/informatica/public/vak.phtml?id=0007940
Go Koppelingen
Home
De Programma's
Onderzoeksgroepen
Who's Who
Links
Done
Internet

Artificiële intelligentie I

Docent Luc Steels

Studierichting/variant	SP	ST	Verdeling
1ste jaar Licentiaat in de Toegepaste Informatica Verkort programma	4	112u	Hoorcollege: 30u Practicum: 15u
2de jaar Kandidaat in de Informatica Verkort voor HO2C en academisch diploma 1ste cyclus	4	112u	Hoorcollege: 30u Practicum: 15u
2de jaar Kandidaat in de Informatica Verkort voor HO1C	4	112u	Hoorcollege: 30u Practicum: 15u

Page contains :
- Course
- Lecturer with link
- Academic year with links
- Weight
- Assimilation time
- Division key

The course page contains the academic years, with links to the respective academic year page, it is being taught in, together with the weights for the students' marks it has been given. Also the estimated time to assimilate the course and the division between theoretical and practical tutoring is marked. We also find the lecturer, with link to the Staff member page.

The academic year page contains all compulsory and optional courses to be taken in the academic year, with links to each course page. Also the lecturer is marked, together with a link to the Staff member page.

7.3.5. Static pages

Following pages are static pages and are therefore discussed less thoroughly:

- Links: containing links to several interesting outside pages and to forms to apply for different accounts.
- Evening students: page with information and Frequently Asked Questions about the organisation of courses for evening students. These will not be taken into account for the conceptual schema, only the link to this page is included.
- On-line tutorials: links to tutorials for some of the courses
- Masters of Computer Science: information – among others curriculum – of the masters degree. This page contains still links to other pages, which will not be discussed. Only the link to this page is included in the structure schema.

8. Generating HTML pages reflecting the conceptual schema as an application of XSLT

The transformation of the conceptual schema to an XML document is outside the scope of this thesis. In appendix figure 6 the full XML document for the case study is included. This XML document is now suitable for transportation over the Internet, using standard HTTP. The user can manipulate this document to suite his needs using style sheets as discussed in chapters 3 and 5. At the time of writing, most browsers do not implement the full power of XSLT and certainly not XSL because there is no full standard yet. Therefore, tools developed especially for transformation are needed to use the possibilities of XSLT, which is already a standard, and the suggested possibilities of XSL.

The server containing the web site and the XML document can also suggest some representations for the conceptual schema. The web site administrators can provide for a couple of style sheets. Since the conceptual schema of a web site and thus the XML document are expected to be stable in a production environment, the web site administrator can, for efficiency of use, provide for HTML documents ready for display.

We discuss in this chapter two possible style sheets, each using a different viewing angle and thus highlighting different topics of the conceptual schema.

Both of the style sheets will include also a link to a Cascading Style Sheet, so this style sheet can be used to style the HTML in the wanted way (for example, colour all ObjectTypes in a consistent way).
Remark that we used a different namespace than the standard namespace, because the used tool “XSL Editor”, an IBM Alphaworks product, uses this namespace.
We encountered one major problem in using a predicate on multiple elements and had to find a, not performant, work-around. It is more elaborately described in the description of the style sheets. This work-around is possible thanks to the redundancy included into the XML document. This way we can approach the information from different viewing angles. The redundancy also makes the production and maintenance of the XML document very laborious, hence the need of automated production and maintenance.

We show the relation between Objects for structuring and ObjectTypes for sub/supertyping through enumerating the related Objects or ObjectTypes and supply them with hyperlinks. An alternative could be to show all Role(Type)s of the related Object(Type)s.

We found that, although the rules are very highlevel and easy to learn, the templates are very error-prone. Debugging the style sheets is possible with the used tool, but as soon as the sheet grows substantially, the debugging becomes almost impossible. We can find the error then only by commenting out large blocks of codes and each time parsing, this way binary searching to the troublesome part.
New tools should guide one better through this process.

8.1. Overview of Pages and PageProtoTypes

This style sheet proposes an overview of all information on a Page(ProtoType), including Object(Type)s with their respective Role(Type)s and the links on the Page(ProtoType).
The information is presented in tables because this offers a natural overview of the information.

The style sheet starts with extracting an index of all PageProtoTypes available on the site, with for each PageProtoType the possibly included PageInstances. Also the stand-alone Pages are included in the index. For each Page and PageProtoType, the number of Object(Type)s and Link(ProtoType)s are shown, not counting the instances of the ObjectTypes and of LinkProtoTypes.
Through extensive use of hyperlinks, we can manoeuvre through the whole document to find the needed information.

For each PageProtoType following information is shown:

- The type of PageProtoType: choice of Static, Tailored or Dynamic PageProtoType
- The ObjectTypes on the page, with their RoleTypes and the counterpart in the Relation. This information is supplied using tables. Since the DTD is based on a Binary Role diagram, a RelationType can only have two participants.
  A major difficulty encountered here was to show only the RelationTypes on the current PageProtoType and not all RelationTypes of the current ObjectType, whether on the same PageProtoType or not. The tools used did not support the selection of RelationTypes which are found on multiple PageProtoTypes, which was expected to be possible after studying the Recommendations. A work-around is therefore used, with testing every RelationType the ObjectType plays a RoleType in, to make sure the RelationType is used on the PageProtoType.
- The LinkProtoTypes on the page

For each Page following information is shown:

- The URL where the page can be found.
- The PageProtoType, if the Page is actually a PageInstance.
- ExternalPage, if the page does not belong to the discussed web site but is part of a different one.
- The type of Page : choice of Static, Tailored or Dynamic Page
- The ObjectTypes on the page, with their RoleTypes and the counterpart in the RelationType. This information is supplied using tables. Problem as described above had to be worked around.
- The Objects on the page, with their Roles and the counterpart in the Relation. We encountered the same problems as with ObjectTypes.
- The LinkProtoTypes on the page in a table.
- The Links on the page in a table.

8.2. Objects and ObjectTypes

This style sheet provides for an overview, starting from the Object(Type)s on a web site.

Again, the style sheet starts with showing the user an index of all ObjectTypes and Objects. The number of Page(ProtoType)s the Object(Type)s are used on together with the number of Relations(Type)s are also shown.

For each ObjectType we then show:
- The name if this is supplied.
- If the Object is a SubType or SuperType of another ObjectType, this information is listed.
- The PageProtoTypes and the Pages the ObjectType is used on, in a table.
- The RelationTypes the ObjectType plays a role in together with the counterpart, in a table.

For each Object we show:
- The name if supplied
- If the Object is Elementary, we show the type of content with a choice of Text, Video or Graphics.
- If the Object is Structured, we show the composing parts of the object. If the Object is part of a Structured Object, we show the composed objects.
- The Pages the Object is used on, in a table.
- The Relations the Object plays a role in, together with the counterpart.

9. Future research

9.1. Suggestion for further style sheets.

The suggested style sheets only allow simple interaction from the user, through navigational links on each topic in the HTML page. Through using on-demand-transformation with parameterisation, one could produce more customised HTML pages.

The user could fill in a form, to suggest the topics of interest.

Objects	**Include?**	**ObjectTypes**	**Include?**
Commitment	☐	Staff member	☐
	☐	Tel	☐
	☐	Email	☐

Show conceptual schema grouped around:	
Pages	☐
Object(Type)s	☐

The choices in the form should be used to send a query, preferably based on XQL or another query standard based on XML (these are still in early stadium). This query would in turn produce an XML document containing only information about the chosen Objects and ObjectTypes. For example, only Page(ProtoType)s with all the checked Objects and ObjectTypes are included, together with all information about the checked Object(Type)s – roles, relations including with other Object(Type)s – and the Link(ProtoType)s to be found on these Page(ProtoType)s. This XML document is then to be styled with the style sheets chosen. This style sheet will then group the information.

Also, the mutual parts of the suggested style sheets could be filtered out in a new style sheet containing only the common elements and this style sheet could then be included in the two style sheets. This would prove a more efficient use of the style sheets, but at the time of writing the used tools did not support the xsl:import/include elements yet.

One could set an order on Objects, ObjectTypes, Pages and PageProtoTypes to give a more meaningful ordering than the alphabetical ordering used.

9.2. Extension of conceptual schema

The conceptual schema should be searchable for an search agent. In this matter, the Resource Description Framework could play a role. RDF could be used to define the possible elements and the relations between the elements of the conceptual meta schema as a replacement of the DTD.
In combination with RDF, a default namespace could be declared for the elements and attributes of the DTD so search engines could be sure about the semantics of the XML tags.
Next to automating the production of the XML document describing the conceptual schema, the question remains how the XML document should be linked with a web site so this document can be found by an automated process.

Conclusion

While the Internet grew from a couple of linked computers sharing research information to a global village connecting young and old, standardisation is needed to guide the millions using this distributed library to quickly find and access the needed information. The ease of use of HTML made both amateur and professional users enthusiastic. The chaotic amount of information provided by these authors needs to be structured and presented in an orderly manner. HTML documents are easy to produce but difficult to maintain. In production environments this requires too much resources. Also ways to provide a more efficient way to search this heap of information is needed.

XML and its use in conceptual schema enables to provide search engines concise and relevant information about a web site and with some representation mechanism the user can be offered an elaborate sketch of the structure and content of the web site.
XSL provides easy to use syntax to produce in a template way a formatted representation of XML, which in itself has no visualisation in a browser. XSLT, fixed part of the growing XSL, uses XPath to walk the tree-like structure of XML. We illustrated this using XSLT to produce an HTML document representing and investigating the XML document of a conceptual schema of a web site. This XML document is based on a DTD, which is the translation of a conceptual meta schema, containing all elements to be used in the conceptual schema. This DTD allows approaching the information from different viewing angles through redundant information. The meta schema provides content, structure and mapping schema. The first two were used as viewing angles for two elaborate style sheets, while the third, being the connection between the first two, was used to link items from the first two schemas in the style sheets.

XML and XSL, through separation of content/structure and representation, are the sure ways to produce efficient and easy to maintain documents. The example XSLT templates can be used with any XML document, following the DTD, to present the structure and content of any web site.
The recommendations receive a lot of support from major players and there is a growing community working on it. Other standards are emerging on XML to produce a totally new framework of tools to produce a more efficient, distributed library of information.

Appendix A : The conceptual meta schema

Figure 1 The Content Meta Schema

As modelled by [DD 2000].

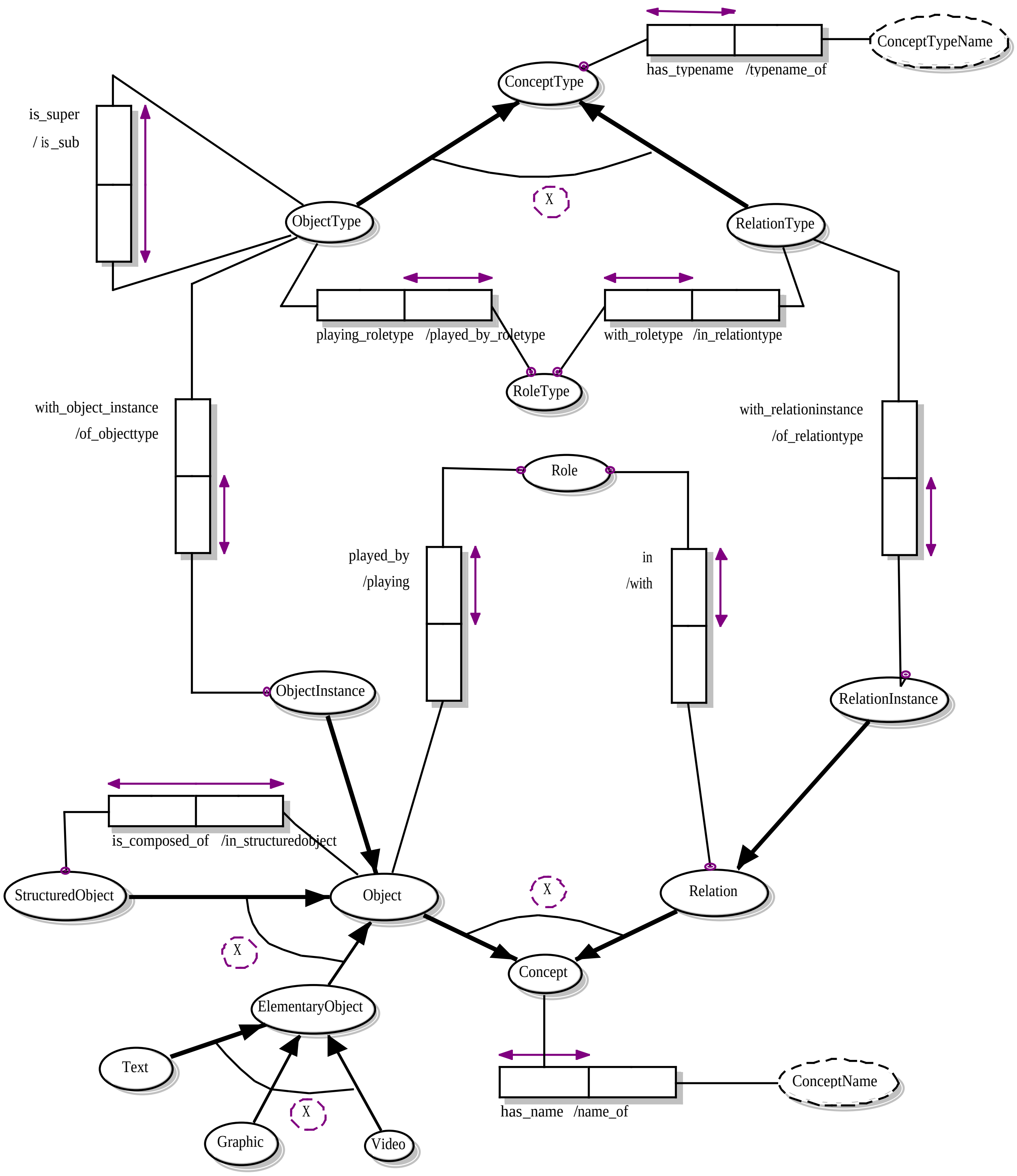


Figure 2 The Structure Meta Schema

As modelled by [DD 2000].

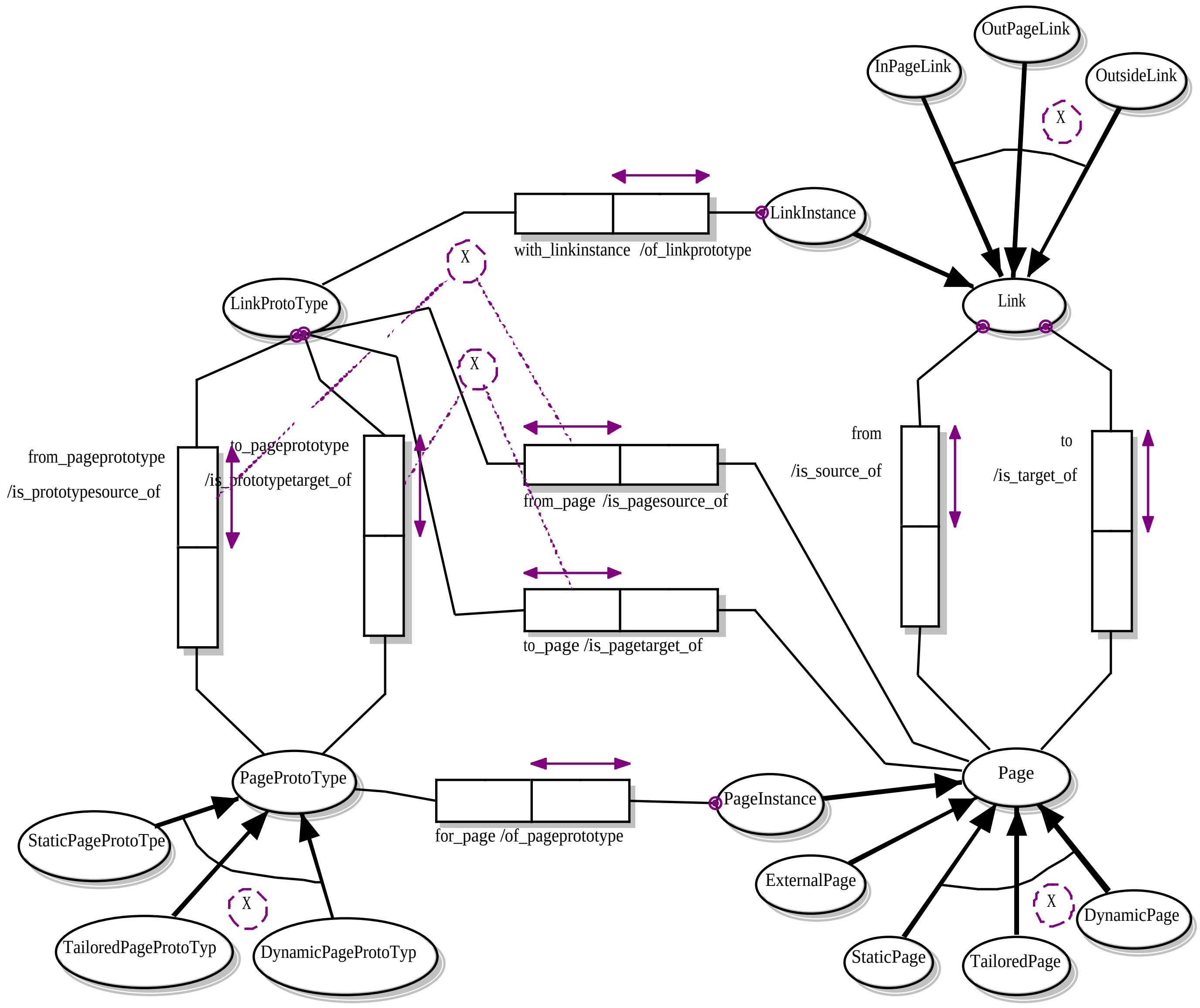


Figure 3 The Mapping Meta Schema

As modelled by [DD 2000].

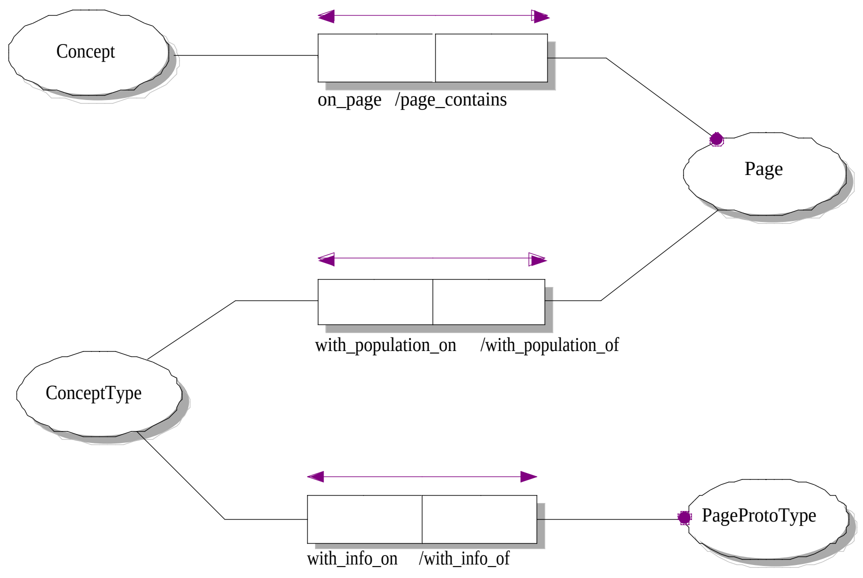


Figure 4 The complete Conceptual Schema DTD
As modelled by [DD 2000].

```
<?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 (URL ?,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 URL (#PCDATA)>
            <!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_pageprototype 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>
```

Appendix B: The Case study: DINF

Figure 5 The Conceptual Schema of DINF web site

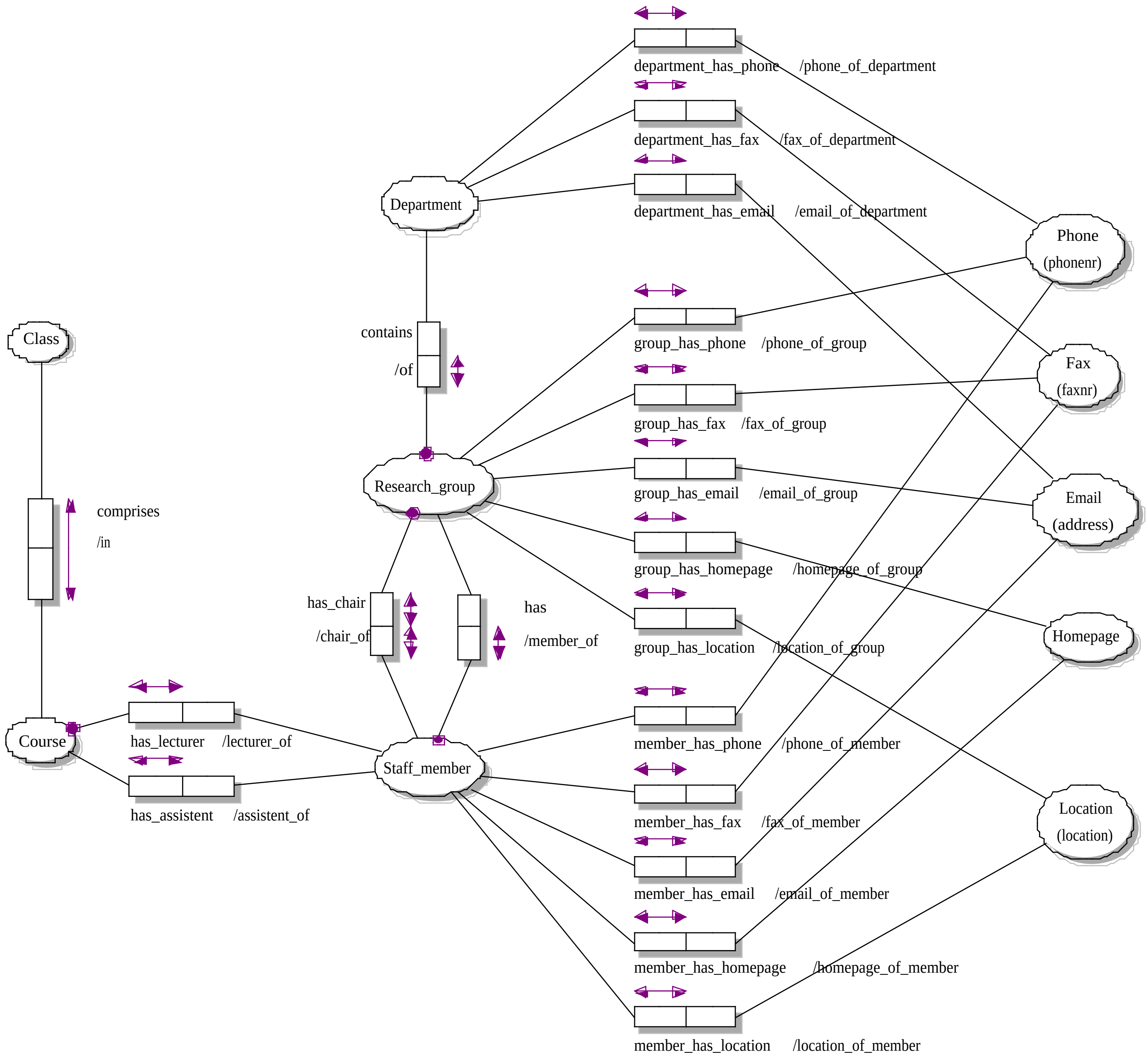


Figure 6 Full XML document of DINF web site

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

<!DOCTYPE Website SYSTEM "website_OD.dtd" >
<!--                ConceptType                  -->
<Website WebsiteID="Computer Science">

<ConceptType ConceptTypeID="C_Department">
    <ObjectType ObjectTypeID="Department">
        <with_objectinstance ObjectInstanceID="Computer_Science"/>
        <playing_roletype RoleTypeID="contains"/>
        <playing_roletype RoleTypeID="department_has_phone"/>
        <playing_roletype RoleTypeID="department_has_fax"/>
        <playing_roletype RoleTypeID="department_has_email"/>
    </ObjectType>
    <with_population_on PageID="Computer_SciencePage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Research_group">
    <ObjectType ObjectTypeID="Research_group">
        <playing_roletype RoleTypeID="of"/>
        <playing_roletype RoleTypeID="has_chair"/>
        <playing_roletype RoleTypeID="has"/>
        <playing_roletype RoleTypeID="group_has_phone"/>
        <playing_roletype RoleTypeID="group_has_fax"/>
        <playing_roletype RoleTypeID="group_has_email"/>
        <playing_roletype RoleTypeID="group_has_homepage"/>
        <playing_roletype RoleTypeID="group_has_location"/>
    </ObjectType>
    <with_population_on PageID="Research_groups_overview"/>
    <with_info_on PageProtoTypeID="Research_grouppage"/>
    <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Staff_member">
    <ObjectType ObjectTypeID="Staff_member">
        <playing_roletype RoleTypeID="chair_of"/>
        <playing_roletype RoleTypeID="member_of"/>
        <playing_roletype RoleTypeID="lecturer_of"/>
        <playing_roletype RoleTypeID="assistant_of"/>
        <playing_roletype RoleTypeID="member_has_phone"/>
        <playing_roletype RoleTypeID="member_has_fax"/>
        <playing_roletype RoleTypeID="member_has_email"/>
        <playing_roletype RoleTypeID="member_has_homepage"/>
        <playing_roletype RoleTypeID="member_has_location"/>
    </ObjectType>
    <with_population_on PageID="Who's_who"/>
    <with_info_on PageProtoTypeID="Research_grouppage"/>
    <with_info_on PageProtoTypeID="Staff_memberpage"/>
    <with_info_on PageProtoTypeID="Coursepage"/>
    <with_info_on PageProtoTypeID="Classpage"/>
</ConceptType>
```

```
<ConceptType ConceptTypeID="C_Phone">
      <ObjectType ObjectTypeID="Phone">
          <playing_roletype RoleTypeID="phone_of_department"/>
          <playing_roletype RoleTypeID="phone_of_group"/>
          <playing_roletype RoleTypeID="phone_of_member"/>
      </ObjectType>
      <with_population_on PageID="Computer_SciencePage"/>
      <with_population_on PageID="Who's_who"/>
      <with_info_on PageProtoTypeID="Research_grouppage"/>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Email">
      <ObjectType ObjectTypeID="Email">
          <playing_roletype RoleTypeID="email_of_department"/>
          <playing_roletype RoleTypeID="email_of_group"/>
          <playing_roletype RoleTypeID="email_of_member"/>
      </ObjectType>
      <with_population_on PageID="Computer_SciencePage"/>
      <with_population_on PageID="Who's_who"/>
      <with_info_on PageProtoTypeID="Research_grouppage"/>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Fax">
      <ObjectType ObjectTypeID="Fax">
          <playing_roletype RoleTypeID="fax_of_department"/>
          <playing_roletype RoleTypeID="fax_of_group"/>
          <playing_roletype RoleTypeID="fax_of_member"/>
      </ObjectType>
      <with_population_on PageID="Computer_SciencePage"/>
      <with_info_on PageProtoTypeID="Research_grouppage"/>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Homepage">
      <ObjectType ObjectTypeID="Homepage">
          <playing_roletype RoleTypeID="homepage_of_group"/>
          <playing_roletype RoleTypeID="homepage_of_member"/>
      </ObjectType>
      <with_info_on PageProtoTypeID="Research_grouppage"/>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Location">
      <ObjectType ObjectTypeID="Location">
          <playing_roletype RoleTypeID="location_of_group"/>
          <playing_roletype RoleTypeID="location_of_member"/>
      </ObjectType>
      <with_info_on PageProtoTypeID="Research_grouppage"/>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
       <with_population_on PageID="Computer_SciencePage"/>
</ConceptType>
```

```
<ConceptType ConceptTypeID="C_Course">
      <ObjectType ObjectTypeID="Course">
          <playing_roletype RoleTypeID="in"/>
          <playing_roletype RoleTypeID="has_lecturer"/>
          <playing_roletype RoleTypeID="has_assistant"/>
      </ObjectType>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
      <with_info_on PageProtoTypeID="Coursepage"/>
      <with_info_on PageProtoTypeID="Classpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Class">
      <ObjectType ObjectTypeID="Class">
          <playing_roletype RoleTypeID="comprises"/>
      </ObjectType>
      <with_info_on PageProtoTypeID="Staff_memberpage"/>
      <with_info_on PageProtoTypeID="Coursepage"/>
      <with_info_on PageProtoTypeID="Classpage"/>
</ConceptType>

<!-- End ConceptTypes -->

<!-- RelationTypes    -->

<ConceptType ConceptTypeID="C_Department_Research_group">
  <RelationType RelationTypeID="R_Department_Research_group">
      <with_roletype RoleTypeID="contains"/>
      <with_roletype RoleTypeID="of"/>
  </RelationType>
</ConceptType>
<ConceptType ConceptTypeID="C_Research_group_chair_Staff_member">
  <RelationType RelationTypeID="R_Research_group_chair_Staff_member">
      <with_roletype RoleTypeID="chair_of"/>
      <with_roletype RoleTypeID="has_chair"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Research group"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Research_group_Staff_member">
  <RelationType RelationTypeID="R_Research_group_Staff_member">
      <with_roletype RoleTypeID="has"/>
      <with_roletype RoleTypeID="member_of"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
  <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_Class_Course">
  <RelationType RelationTypeID="R_Class_Course">
      <with_roletype RoleTypeID="comprises"/>
      <with_roletype RoleTypeID="in"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Classpage"/>
```

```
    <with_info_on PageProtoTypeID="Coursepage"/>
    <with_info_on PageProtoTypeID="Staff_memberpage"/>
  </ConceptType>
  <ConceptType ConceptTypeID="C_Course_lecturer_Staff_member">
    <RelationType RelationTypeID="R_Course_lecturer_Staff_member">
        <with_roletype RoleTypeID="has_lecturer"/>
        <with_roletype RoleTypeID="lecturer_of"/>
    </RelationType>
    <with_info_on PageProtoTypeID="Staff_memberpage"/>
    <with_info_on PageProtoTypeID="Coursepage"/>
    <with_info_on PageProtoTypeID="Classpage"/>
  </ConceptType>
  <ConceptType ConceptTypeID="C_Course_assistant_Staff_member">
    <RelationType RelationTypeID="R_Course_assistant_Staff_member">
        <with_roletype RoleTypeID="has_assistant"/>
        <with_roletype RoleTypeID="assistant_of"/>
    </RelationType>
    <with_info_on PageProtoTypeID="Staff_memberpage"/>
  </ConceptType>

  <ConceptType ConceptTypeID="C_department_phone">
    <RelationType RelationTypeID="R_department_phone">
        <with_roletype RoleTypeID="department_has_phone"/>
        <with_roletype RoleTypeID="phone_of_department"/>
    </RelationType>
    <with_population_on PageID="Computer_SciencePage"/>
  </ConceptType>
  <ConceptType ConceptTypeID="C_department_fax">
    <RelationType RelationTypeID="R_department_fax">
        <with_roletype RoleTypeID="department_has_fax"/>
        <with_roletype RoleTypeID="fax_of_department"/>
    </RelationType>
    <with_population_on PageID="Computer_SciencePage"/>
  </ConceptType>
  <ConceptType ConceptTypeID="C_department_email">
    <RelationType RelationTypeID="R_department_email">
        <with_roletype RoleTypeID="department_has_email"/>
        <with_roletype RoleTypeID="email_of_department"/>
    </RelationType>
    <with_population_on PageID="Computer_SciencePage"/>
  </ConceptType>
  <ConceptType ConceptTypeID="C_group_phone">
    <RelationType RelationTypeID="R_group_phone">
        <with_roletype RoleTypeID="group_has_phone"/>
        <with_roletype RoleTypeID="phone_of_group"/>
    </RelationType>
    <with_info_on PageProtoTypeID="Research_grouppage"/>
  </ConceptType>
  <ConceptType ConceptTypeID="C_group_fax">
    <RelationType RelationTypeID="R_group_fax">
```

```
        <with_roletype RoleTypeID="group_has_fax"/>
        <with_roletype RoleTypeID="fax_of_group"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_group_email">
  <RelationType RelationTypeID="R_group_email">
        <with_roletype RoleTypeID="group_has_email"/>
        <with_roletype RoleTypeID="email_of_group"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_group_homepage">
  <RelationType RelationTypeID="R_group_homepage">
        <with_roletype RoleTypeID="group_has_homepage"/>
        <with_roletype RoleTypeID="homepage_of_group"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_group_location">
  <RelationType RelationTypeID="R_group_location">
        <with_roletype RoleTypeID="group_has_location"/>
        <with_roletype RoleTypeID="location_of_group"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_member_phone">
  <RelationType RelationTypeID="R_member_phone">
        <with_roletype RoleTypeID="member_has_phone"/>
        <with_roletype RoleTypeID="phone_of_member"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Staff_memberpage"/>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
  <with_population_on PageID="Who's_who"/>
</ConceptType>
<ConceptType ConceptTypeID="C_member_fax">
  <RelationType RelationTypeID="R_member_fax">
        <with_roletype RoleTypeID="member_has_fax"/>
        <with_roletype RoleTypeID="fax_of_member"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Staff_memberpage"/>
</ConceptType>
<ConceptType ConceptTypeID="C_member_email">
  <RelationType RelationTypeID="R_member_email">
        <with_roletype RoleTypeID="member_has_email"/>
        <with_roletype RoleTypeID="email_of_member"/>
  </RelationType>
  <with_info_on PageProtoTypeID="Staff_memberpage"/>
  <with_info_on PageProtoTypeID="Research_grouppage"/>
  <with_population_on PageID="Who's_who"/>
```

```
        </ConceptType>
        <ConceptType ConceptTypeID="C_member_homepage">
          <RelationType RelationTypeID="R_member_homepage">
                <with_roletype RoleTypeID="member_has_homepage"/>
                <with_roletype RoleTypeID="homepage_of_member"/>
          </RelationType>
          <with_info_on PageProtoType="Staff_memberpage"/>
        </ConceptType>
        <ConceptType ConceptTypeID="C_member_location">
          <RelationType RelationTypeID="R_member_location">
                <with_roletype RoleTypeID="member_has_location"/>
                <with_roletype RoleTypeID="location_of_member"/>
          </RelationType>
          <with_info_on PageProtoTypeID="Staff_memberpage"/>
          <with_info_on PageProtoTypeID="Research_grouppage"/>
        </ConceptType>

        <!-- End RelationTypes -->

        <!-- RoleTypes          -->

        <RoleType RoleTypeID="contains">
                <played_by_objecttype ObjectTypeID="Department"/>
                <in_relationtype RelationTypeID="R_Department_Research_group"/>
        </RoleType>
        <RoleType RoleTypeID="of">
                <played_by_objecttype ObjectTypeID="Research_group"/>
                <in_relationtype RelationTypeID="R_Department_Research_group"/>
        </RoleType>
        <RoleType RoleTypeID="has_chair">
                <played_by_objecttype ObjectTypeID="Research_group"/>
                <in_relationtype RelationTypeID="R_Research_group_chair_Staff_member"/>
        </RoleType>
        <RoleType RoleTypeID="chair_of">
                <played_by_objecttype ObjectTypeID="Staff_member"/>
                <in_relationtype RelationTypeID="R_Research_group_chair_Staff_member"/>
        </RoleType>
        <RoleType RoleTypeID="has">
                <played_by_objecttype ObjectTypeID="Research_group"/>
                <in_relationtype RelationTypeID="R_Research_group_Staff_member"/>
        </RoleType>
        <RoleType RoleTypeID="member_of">
                <played_by_objecttype ObjectTypeID="Staff_member"/>
                <in_relationtype RelationTypeID="R_Research_group_Staff_member"/>
        </RoleType>

        <RoleType RoleTypeID="lecturer_of">
                <played_by_objecttype ObjectTypeID="Staff_member"/>
                <in_relationtype RelationTypeID="R_Course_lecturer_Staff_member"/>
        </RoleType>
```

```
	<RoleType RoleTypeID="has_lecturer">
		<played_by_objecttype ObjectTypeID="Course"/>
		<in_relationtype RelationTypeID="R_Course_lecturer_Staff_member"/>
	</RoleType>
	<RoleType RoleTypeID="assistant_of">
		<played_by_objecttype ObjectTypeID="Staff_member"/>
		<in_relationtype RelationTypeID="R_Course_assistant_Staff_member"/>
	</RoleType>
	<RoleType RoleTypeID="has_assistant">
		<played_by_objecttype ObjectTypeID="Course"/>
		<in_relationtype RelationTypeID="R_Course_assistant_Staff_member"/>
	</RoleType>
	<RoleType RoleTypeID="comprises">
		<played_by_objecttype ObjectTypeID="Class"/>
		<in_relationtype RelationTypeID="R_Class_Course"/>
	</RoleType>
	<RoleType RoleTypeID="in">
		<played_by_objecttype ObjectTypeID="Course"/>
		<in_relationtype RelationTypeID="R_Class_Course"/>
	</RoleType>

	<RoleType RoleTypeID="member_has_phone">
		<played_by_objecttype ObjectTypeID="Staff_member"/>
		<in_relationtype RelationTypeID="R_member_phone"/>
	</RoleType>
	<RoleType RoleTypeID="member_has_fax">
		<played_by_objecttype ObjectTypeID="Staff_member"/>
		<in_relationtype RelationTypeID="R_member_fax"/>
	</RoleType>
	<RoleType RoleTypeID="member_has_email">
		<played_by_objecttype ObjectTypeID="Staff_member"/>
		<in_relationtype RelationTypeID="R_member_email"/>
	</RoleType>
	<RoleType RoleTypeID="member_has_homepage">
		<played_by_objecttype ObjectTypeID="Staff_member"/>
		<in_relationtype RelationTypeID="R_member_homepage"/>
	</RoleType>
	<RoleType RoleTypeID="member_has_location">
		<played_by_objecttype ObjectTypeID="Staff_member"/>
		<in_relationtype RelationTypeID="R_member_location"/>
	</RoleType>
	<RoleType RoleTypeID="group_has_phone">
		<played_by_objecttype ObjectTypeID="Research_group"/>
		<in_relationtype RelationTypeID="R_group_phone"/>
	</RoleType>
	<RoleType RoleTypeID="group_has_fax">
		<played_by_objecttype ObjectTypeID="Research_group"/>
		<in_relationtype RelationTypeID="R_group_fax"/>
	</RoleType>
	<RoleType RoleTypeID="group_has_email">
```

```
        <played_by_objecttype ObjectTypeID="Research_group"/>
        <in_relationtype RelationTypeID="R_group_email"/>
    </RoleType>
    <RoleType RoleTypeID="group_has_homepage">
        <played_by_objecttype ObjectTypeID="Research_group"/>
        <in_relationtype RelationTypeID="R_group_homepage"/>
    </RoleType>
    <RoleType RoleTypeID="group_has_location">
        <played_by_objecttype ObjectTypeID="Research_group"/>
        <in_relationtype RelationTypeID="R_group_location"/>
    </RoleType>
    <RoleType RoleTypeID="phone_of_group">
        <played_by_objecttype ObjectTypeID="Phone"/>
        <in_relationtype RelationTypeID="R_group_phone"/>
    </RoleType>
    <RoleType RoleTypeID="fax_of_group">
        <played_by_objecttype ObjectTypeID="Fax"/>
        <in_relationtype RelationTypeID="R_group_fax"/>
    </RoleType>
    <RoleType RoleTypeID="email_of_group">
        <played_by_objecttype ObjectTypeID="Email"/>
        <in_relationtype RelationTypeID="R_group_email"/>
    </RoleType>
    <RoleType RoleTypeID="homepage_of_group">
        <played_by_objecttype ObjectTypeID="Homepage"/>
        <in_relationtype RelationTypeID="R_group_homepage"/>
    </RoleType>
    <RoleType RoleTypeID="location_of_group">
        <played_by_objecttype ObjectTypeID="Location"/>
        <in_relationtype RelationTypeID="R_group_location"/>
    </RoleType>

    <RoleType RoleTypeID="department_has_phone">
        <played_by_objecttype ObjectTypeID="Department"/>
        <in_relationtype RelationTypeID="R_department_phone"/>
    </RoleType>
    <RoleType RoleTypeID="department_has_fax">
        <played_by_objecttype ObjectTypeID="Department"/>
        <in_relationtype RelationTypeID="R_department_fax"/>
    </RoleType>
    <RoleType RoleTypeID="department_has_email">
        <played_by_objecttype ObjectTypeID="Department"/>
        <in_relationtype RelationTypeID="R_department_email"/>
    </RoleType>
    <RoleType RoleTypeID="phone_of_department">
        <played_by_objecttype ObjectTypeID="Phone"/>
        <in_relationtype RelationTypeID="R_department_phone"/>
    </RoleType>
    <RoleType RoleTypeID="fax_of_department">
        <played_by_objecttype ObjectTypeID="Fax"/>
        <in_relationtype RelationTypeID="R_department_fax"/>
    </RoleType>
    <RoleType RoleTypeID="email_of_department">
        <played_by_objecttype ObjectTypeID="Email"/>
        <in_relationtype RelationTypeID="R_department_email"/>
    </RoleType>
    <RoleType RoleTypeID="phone_of_member">
        <played_by_objecttype ObjectTypeID="Phone"/>
        <in_relationtype RelationTypeID="R_member_phone"/>
    </RoleType>
    <RoleType RoleTypeID="fax_of_member">
        <played_by_objecttype ObjectTypeID="Fax"/>
        <in_relationtype RelationTypeID="R_member_fax"/>
    </RoleType>
    <RoleType RoleTypeID="email_of_member">
        <played_by_objecttype ObjectTypeID="Email"/>
        <in_relationtype RelationTypeID="R_member_email"/>
    </RoleType>
    <RoleType RoleTypeID="homepage_of_member">
        <played_by_objecttype ObjectTypeID="Homepage"/>
        <in_relationtype RelationTypeID="R_member_homepage"/>
    </RoleType>
    <RoleType RoleTypeID="location_of_member">
        <played_by_objecttype ObjectTypeID="Location"/>
        <in_relationtype RelationTypeID="R_member_location"/>
    </RoleType>

    <!-- End RoleTypes -->

    <!-- Concept -->
    <Concept ConceptID="C_Computer_Science">
        <Object ObjectID="O_Computer_Science">
            <ObjectInstance ObjectInstanceID="Computer_Science">
                <of_objecttype ObjectTypeID="Department"/>
            </ObjectInstance>
        </Object>
        <on_page PageID="Computer_Science"/>
        <has_name>Computer Science Department</has_name>
    </Concept>
    <!-- End Concept -->

    <!-- Relations -->
    <!-- End Relations -->

    <!-- Roles -->
    <!-- End Roles -->
```

```
<!-- PageProtoType -->

<PageProtoType PageProtoTypeID="Homepage_Research_group">
      <is_prototypetarget_of LinkProtoTypeID="Homepage_Research_group_Link"/>
</PageProtoType>
<PageProtoType PageProtoTypeID="Homepage">
      <is_prototypetarget_of LinkProtoTypeID="Homepage_Link"/>
</PageProtoType>
<PageProtoType PageProtoTypeID="Email">
      <is_prototypetarget_of LinkProtoTypeID="Email_Staff_member_Link"/>
      <is_prototypetarget_of LinkProtoTypeID="Email_Research_group_Link"/>
      <is_prototypetarget_of LinkProtoTypeID="Email_Computer_Science_Link"/>
</PageProtoType>
<PageProtoType PageProtoTypeID="Research_grouppage">
      <DynamicPageProtoType/>
      <with_info_of ConceptTypeID="C_Research_group"/>
      <with_info_of ConceptTypeID="C_Staff_member"/>
      <with_info_of ConceptTypeID="C_Phone"/>
      <with_info_of ConceptTypeID="C_Fax"/>
      <with_info_of ConceptTypeID="C_Location"/>
      <with_info_of ConceptTypeID="C_Email"/>
      <with_info_of ConceptTypeID="C_Homepage"/>
      <is_prototypetarget_of LinkProtoTypeID="Research_group_Link"/>
      <is_prototypesource_of LinkProtoTypeID="Staff_member_Link"/>
      <is_prototypesource_of LinkProtoTypeID="Homepage_Research_group_Link"/>
      <is_prototypetarget_of LinkProtoTypeID=
                                                 "Research_group_Staff_member_Link"/>
      <is_prototypesource_of LinkProtoTypeID="Email_Research_group_Link"/>
</PageProtoType>

<PageProtoType PageProtoTypeID="Staff_memberpage">
      <DynamicPageProtoType/>
      <with_info_of ConceptTypeID="C_Research_group"/>
      <with_info_of ConceptTypeID="C_Staff_member"/>
      <with_info_of ConceptTypeID="C_Phone"/>
      <with_info_of ConceptTypeID="C_Fax"/>
      <with_info_of ConceptTypeID="C_Location"/>
      <with_info_of ConceptTypeID="C_Email"/>
      <with_info_of ConceptTypeID="C_Homepage"/>
      <with_info_of ConceptTypeID="C_Course"/>
      <with_info_of ConceptTypeID="C_Class"/>
      <is_prototypetarget_of LinkProtoTypeID="Staff_member_Link"/>
      <is_prototypetarget_of LinkProtoTypeID="Staff_member_Course_Link"/>
      <is_prototypetarget_of LinkProtoTypeID="Staff_member_Class_Link"/>
      <is_prototypesource_of LinkProtoTypeID="Homepage_Link"/>
      <is_prototypesource_of LinkProtoTypeID="Coursepage_Link"/>
      <is_prototypesource_of LinkProtoTypeID=
                                                 "Research_group_Staff_member_Link"/>
      <is_prototypesource_of LinkProtoTypeID="Email_Staff_member_Link"/>
       <is_prototypetarget_of LinkProtoTypeID="Staff_member_Who's_who_Link"/>
```

```
          <is_prototypetarget_of LinkProtoTypeID="Email_Who's_who_Link"/>
      </PageProtoType>
      <PageProtoType PageProtoTypeID="Coursepage">
          <DynamicPageProtoType/>
          <with_info_of ConceptTypeID="C_Staff_member"/>
          <with_info_of ConceptTypeID="C_Course"/>
          <with_info_of ConceptTypeID="C_Class"/>
          <is_prototypetarget_of LinkProtoTypeID="Coursepage_Link"/>
          <is_prototypetarget_of LinkProtoTypeID="Coursepage_Classpage_Link"/>
          <is_prototypesource_of LinkProtoTypeID="Classpage_Link"/>
          <is_prototypesource_of LinkProtoTypeID="Staff_member_Coursepage_Link"/>
      </PageProtoType>
      <PageProtoType PageProtoTypeID="Classpage">
          <DynamicPageProtoType/>
          <with_info_of ConceptTypeID="C_Staff_member"/>
          <with_info_of ConceptTypeID="C_Course"/>
          <with_info_of ConceptTypeID="C_Class"/>
          <is_prototypetarget_of LinkProtoTypeID="Classpage_Link"/>
          <is_prototypesource_of LinkProtoTypeID="Coursepage_Classpage_Link"/>
          <is_prototypesource_of LinkProtoTypeID="Staff_member_Class_Link"/>
      </PageProtoType>

      <!-- End PageProtoType -->

      <!-- Page -->

      <Page PageID="Computer_SciencePage">
          <URL>http://we.vub.ac.be/Informatica</URL>
          <StaticPage/>
          <page_contains ConceptID="C_Computer_Science"/>
          <with_population_of ConceptTypeID="C_Phone"/>
          <with_population_of ConceptTypeID="C_Fax"/>
          <with_population_of ConceptTypeID="C_Email"/>
          <with_population_of ConceptTypeID="C_Department"/>
          <is_source_of LinkID="Research_groups_overview_Link"/>
          <is_source_of LinkID="Linkpage_Link"/>
          <is_source_of LinkID="Who's_who_Link"/>
          <is_source_of LinkID="Evening_students_Link"/>
          <is_source_of LinkID="On-line_tutorials_Link"/>
          <is_source_of LinkID="Master_of_Computer_Science_Link"/>
          <is_pagesource_of LinkProtoTypeID="Email_Computer_Science_Link"/>
      </Page>

      <Page PageID="Research_groups_overview">
          <URL>http://we.vub.ac.be/informatica/public/research.phtml</URL>
          <StaticPage/>
          <with_population_of ConceptTypeID="C_Research_group"/>
          <is_target_of LinkID="Research_groups_overview_Link"/>
          <is_pagesource_of LinkProtoTypeID="Research_group_Link"/>
      </Page>
```

```
<Page PageID="Who's_who">
     <URL>http://we.vub.ac.be/informatica/public/who.phtml</URL>
     <DynamicPage/>
     <with_population_of ConceptTypeID="C_Phone"/>
     <with_population_of ConceptTypeID="C_Email"/>
     <with_population_of ConceptTypeID="C_Staff_member"/>
     <is_target_of LinkID="Who's_who_Link"/>
     <is_pagesource_of LinkProtoTypeID="Staff_member_Who's_who_Link"/>
     <is_pagesource_of LinkProtoTypeID="Email_Who's_who_Link"/>
</Page>
<Page PageID="LinkPage">
     <URL>http://we.vub.ac.be/informatica/public/links.phtml</URL>
     <StaticPage/>
     <is_target_of LinkID="Linkpage_Link"/>
</Page>
<Page PageID="Evening_Students">
     <URL>http://we.vub.ac.be/informatica/public/avond/</URL>
     <StaticPage/>
     <is_target_of LinkID="Evening_students_Link"/>
</Page>
<Page PageID="On-line_tutorials">
     <URL>http://we.vub.ac.be/informatica/public/courses/</URL>
     <StaticPage/>
     <is_target_of LinkID="On-line_tutorials_Link"/>
</Page>
<Page PageID="Master_of_Computer_Science">
     <URL>http://we.vub.ac.be/informatica/public/masters/</URL>
     <StaticPage/>
     <is_target_of LinkID="Master_of_Computer_Science_Link"/>
</Page>

<!-- End Page -->

<!-- Links    -->

<Link LinkID="Research_groups_overview_Link">
    <OutPageLink/>
     <from PageID="Computer_SciencePage"/>
     <to PageID="Research_groups_overview"/>
</Link>
<Link LinkID="Linkpage_Link">
    <OutPageLink/>
     <from PageID="Computer_SciencePage"/>
     <to PageID="LinkPage"/>
</Link>
<Link LinkID="Who's_who_Link">
    <OutPageLink/>
     <from PageID="Computer_SciencePage"/>
     <to PageID="Who's_who"/>
</Link>
```

```
        <Link LinkID="Evening_students_Link">
            <OutPageLink/>
             <from PageID="Computer_SciencePage"/>
             <to PageID="Evening_Students"/>
        </Link>
        <Link LinkID="On-line_tutorials_Link">
            <OutPageLink/>
             <from PageID="Computer_SciencePage"/>
             <to PageID="On-line_tutorials"/>
        </Link>
        <Link LinkID="Master_of_Computer_Science_Link">
            <OutPageLink/>
             <from PageID="Computer_SciencePage"/>
             <to PageID="Master_of_Computer_Science"/>
        </Link>

        <!-- End Links -->

        <!-- LinkProtoType -->
        <LinkProtoType LinkProtoTypeID="Research_group_Link">
             <from_page PageID="Research_groups_overview"/>
             <to_pageprototype PageProtoTypeID="Research_grouppage"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Staff_member_Link">
             <from_pageprototype PageProtoTypeID="Research_grouppage"/>
             <to_pageprototype PageProtoTypeID="Staff_memberpage"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Staff_member_Coursepage_Link">
             <from_pageprototype PageProtoTypeID="Coursepage"/>
             <to_pageprototype PageProtoTypeID="Staff_memberpage"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Staff_member_Classpage_Link">
             <from_pageprototype PageProtoTypeID="Classpage"/>
             <to_pageprototype PageProtoTypeID="Staff_memberpage"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Homepage_Research_group_Link">
             <from_pageprototype PageProtoTypeID="Research_grouppage"/>
             <to_pageprototype PageProtoTypeID="Homepage_Research_group"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Homepage_Link">
             <from_pageprototype PageProtoTypeID="Staff_memberpage"/>
             <to_pageprototype PageProtoTypeID="Homepage"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Coursepage_Link">
             <from_pageprototype PageProtoTypeID="Staff_memberpage"/>
             <to_pageprototype PageProtoTypeID="Coursepage"/>
        </LinkProtoType>
        <LinkProtoType LinkProtoTypeID="Research_group_Staff_member_Link">
             <from_pageprototype PageProtoTypeID="Staff_memberpage"/>
             <to_pageprototype PageProtoTypeID="Resource_group"/>
```

```
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Email_Staff_member_Link">
        <from_pageprototype PageProtoTypeID="Staff_memberpage"/>
        <to_pageprototype PageProtoTypeID="Email"/>
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Email_Research_group_Link">
        <from_pageprototype PageProtoTypeID="Research_grouppage"/>
        <to_pageprototype PageProtoTypeID="Email"/>
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Email_Department_Link">
        <from_pageprototype PageProtoTypeID="Department"/>
        <to_pageprototype PageProtoTypeID="Email"/>
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Classpage_Link">
        <from_pageprototype PageProtoTypeID="Coursepage"/>
        <to_pageprototype PageProtoTypeID="Classpage"/>
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Coursepage_Classpage_Link">
        <from_pageprototype PageProtoTypeID="Classpage"/>
        <to_pageprototype PageProtoTypeID="Coursepage"/>
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Email_Who's_who_Link">
        <from_page PageID="Who's_who"/>
        <to_pageprototype PageProtoTypeID="Email"/>
    </LinkProtoType>
    <LinkProtoType LinkProtoTypeID="Staff_member_Who's_who_Link">
        <from_page PageID="Who's_who"/>
        <to_pageprototype PageProtoTypeID="Staff_memberpage"/>
    </LinkProtoType>

    <!-- End LinkProtoType -->

    </Website>
```

Figure 7 Generated HTML-document: Pages and PageProtoTypes

We will only highlight some specific parts of the HTML document, since this is quite a large document to include in this thesis.
What will you find:

- Index
- Staff_memberpage which is a PageProtoType
- Who's_who which is a Dynamic Page

Conceptual schema of Website : Computer Science

Grouped around Pages and PageProtoTypes (version of 31/5/2000)

Available on this site :

PageProtoTypes	Number of ObjectTypes	Number of LinkProtoTypes to	Number of LinkProtoTypes from	PageInstances
Homepage_Research_group	0	1	0	
Homepage	0	1	0	
Email	0	3	0	
Research_grouppage	7	2	3	
Staff_memberpage	9	5	4	
Coursepage	3	2	2	
Classpage	3	1	2	

Page	Number of Object(Type)s	Number of Link(ProtoType)s to	Number of Link(ProtoType)s from
Computer_SciencePage	4	0	7
Research_groups_overview	1	1	1
Who's_who	3	1	2
LinkPage	0	1	0
Evening_Students	0	1	0
On-line_tutorials	0	1	0
Master_of_Computer_Science	0	1	0

PageProtoType : Staff_memberpage

- **DynamicPageProtoType**

ObjectTypes :

ObjectTypes	RoleTypes	CounterObjectType
Class		
	comprises/in	Course
Course		
	has_assistant/assistant_of	Staff_member
	has_lecturer/lecturer_of	Staff_member
	in/comprises	Class
Email		
	email_of_member/member_has_email	Staff_member
Fax		
	fax_of_member/member_has_fax	Staff_member
Homepage		
Location		
	location_of_member/member_has_location	Staff_member
Phone		
	phone_of_member/member_has_phone	Staff_member
Research_group		
	has/member_of	Staff_member
Staff_member		
	assistant_of/has_assistant	Course
	lecturer_of/has_lecturer	Course
	member_has_email/email_of_member	Email
	member_has_fax/fax_of_member	Fax
	member_has_location/location_of_member	Location
	member_has_phone/phone_of_member	Phone
	member_of/has	Research_group

LinkProtoTypes :

Linkprototype	From pageprototype	From page	To pageprototype	To page
Coursepage_Link	Staff_memberpage		Coursepage	
Email_Staff_member_Link	Staff_memberpage		Email	
Homepage_Link	Staff_memberpage		Homepage	
Research_group_Staff_member_Link	Staff_memberpage		Resource_group	
Staff_member_Classpage_Link	Classpage		Staff_memberpage	
Staff_member_Coursepage_Link	Coursepage		Staff_memberpage	

Staff_member_Link	Research_grouppage		Staff_memberpage	
Staff_member_Who's_who_Link		Who's_who	Staff_memberpage	

Page : Who's_who

- **at : http://we.vub.ac.be/informatica/public/who.phtml**
- **DynamicPage**

ObjectTypes :

ObjectTypes	RoleTypes	CounterObjectType
Email		
	email_of_member/member_has_email	Staff_member
Phone		
	phone_of_member/member_has_phone	Staff_member
Staff_member		
	member_has_email/email_of_member	Email
	member_has_phone/phone_of_member	Phone

LinkProtoTypes :

Linkprototype	From pageprototype	From page	To pageprototype	To page
Email_Who's_who_Link		Who's_who	Email	
Staff_member_Who's_who_Link		Who's_who	Staff_memberpage	

Links :

Link	Link prototype	From page	To page	Type of Link
Who's_who_Link		Computer_SciencePage	Who's_who	OutPageLink

Figure 8 Style sheet: Pages and PageProtoTypes

```
<?xml version='1.0'?>
 <!-- NOT THE OFFICIAL NAMESPACE, ONLY TO BE USED WITH IBM XSL Editor -->
 <xsl:stylesheet xmlns:xsl='http://www.w3.org/XSL/Transform/1.0'>

<xsl:output method="html"/>
<xsl:template match="/Website">
<HTML>
 <HEAD>
       <LINK rel="stylesheet" href="website.css" type="text/css"/>
 </HEAD>
 <BODY>
  <H1>Conceptual schema of Website :
     <xsl:value-of select="@WebsiteID"/>
  </H1>
  <EM>Grouped around Pages and PageProtoTypes (version of 31/5/2000)</EM>
  <BR/>
  <H2>Available on this site :</H2>
  <TABLE Border = "2">
     <xsl:apply-templates select="//PageProtoType" mode="simple"/>
  </TABLE>
  <TABLE Border ="2">
     <xsl:apply-templates select="//Page[count(PageInstance)=0]" mode="simple"/>
  </TABLE>

  <xsl:apply-templates select="PageProtoType" mode="extended">
    <xsl:sort select="@PageProtoTypeID"/>
  </xsl:apply-templates>

  <xsl:apply-templates select="Page" mode="extended">
     <xsl:sort select="@PageID"/>
  </xsl:apply-templates>

 </BODY>
</HTML>
</xsl:template>

<xsl:template match="Page" mode="extended">
  <xsl:param name="Page"><xsl:value-of select="@PageID"/></xsl:param>
  <xsl:param name="mode">Page</xsl:param>
  <H2>Page :
   <a><xsl:attribute name="name"><xsl:value-of select="@PageID"/></xsl:attribute>
     <xsl:value-of select="@PageID"/>
   </a>
  </H2>
  <H3>
  <xsl:apply-templates select="URL"/>
```

```
<xsl:apply-templates select="PageInstance"/>

<xsl:apply-templates select="ExternalPage | StaticPage | TailoredPage |
       DynamicPage"/>
</H3>

<TABLE border="1">
     <xsl:for-each select="with_population_of">
       <xsl:if test="position()=1">
       <CAPTION><H3>ObjectTypes : </H3></CAPTION>
       <TR>
         <TH>ObjectTypes</TH>
         <TH>RoleTypes</TH>
         <TH>CounterObjectType</TH>
       </TR>
       </xsl:if>
        <xsl:sort select="@ConceptTypeID"/>
        <xsl:apply-templates select=
        "//ConceptType[@ConceptTypeID=current()/@ConceptTypeID]/ObjectType">
            <xsl:with-param name="Resource" select="$Page"/>
         <!--   Pass mode as parameter instead of mode itself to prevent double
                templates -->
            <xsl:with-param name="mode" select="$mode"/>
        </xsl:apply-templates>
     </xsl:for-each>
</TABLE>

<TABLE border="1">
     <xsl:for-each select="page_contains">
        <xsl:if test="position()=1">
           <CAPTION><H3>Objects :  </H3></CAPTION>
           <TH>Objects</TH>
           <TH>ObjectType</TH>
           <TH>Roles</TH>
           <TH>CounterObject</TH>
        </xsl:if>
        <xsl:sort select="@ConceptID"/>
        <xsl:apply-templates select=
                                  "//Concept[@ConceptID=current()/@ConceptID]/Object">
            <xsl:sort select="@ObjectID"/>
            <xsl:with-param name="Resource" select="$Page"/>
        </xsl:apply-templates>
     </xsl:for-each>
</TABLE>
<BR/>

<TABLE Border="1">
     <xsl:apply-templates select="//LinkProtoType[from_page/@PageID=$Page or
                                                           to_page/@PageID=$Page]">
            <xsl:sort select="@LinkProtoTypeID"/>
```

```
                <xsl:with-param name="PageProtoType">
                        <xsl:value-of select=
                                            "PageInstance/of_pageprototype/@PageProtoTypeID"/>
                </xsl:with-param>
           </xsl:apply-templates>
      </TABLE>
      <BR/>

      <TABLE Border="1">
           <xsl:apply-templates select="//Link[from/@PageID=$Page or
                                                                         to/@PageID=$Page]">
             <xsl:sort select="@LinkID"/>
             <xsl:with-param name="PageProtoType" select=
                                             "PageInstance/of_pageprototype/@PageProtoTypeID"/>
           </xsl:apply-templates>
      </TABLE>

    </xsl:template>

    <xsl:template match="URL">
      <li>
      at : <a><xsl:attribute name="href"><xsl:value-of select="current()"/></xsl:attribute>
              <xsl:value-of select="current()"/>
          </a>
      </li>
    </xsl:template>

    <xsl:template match="ExternalPage | StaticPage | TailoredPage | DynamicPage">
      <li>
        <xsl:value-of select="name()"/>
      </li>
    </xsl:template>

    <xsl:template match="PageProtoType" mode="extended">
     <xsl:param name="PageProtoType">
              <xsl:value-of select="@PageProtoTypeID"/>
     </xsl:param>

     <H2>PageProtoType :
      <a><xsl:attribute name="name"><xsl:value-of select=
                                                       "@PageProtoTypeID"/></xsl:attribute>
        <xsl:value-of select="@PageProtoTypeID"/>
      </a>
     </H2>

     <H3>
      <ul>
           <xsl:apply-templates select="StaticPageProtoType | TailoredPageProtoType |
                                                                  DynamicPageProtoType"/>
      </ul>
```

```
             </H3>

             <TABLE border="1">
                    <xsl:for-each select="with_info_of">
                      <xsl:if test="position()=1">
                      <CAPTION><H3>ObjectTypes : </H3></CAPTION>
                      <TR>
                        <TH>ObjectTypes</TH>
                        <TH>RoleTypes</TH>
                        <TH>CounterObjectType</TH>
                      </TR>
                      </xsl:if>
                       <xsl:sort select="@ConceptTypeID"/>
                       <xsl:apply-templates select=
                       "//ConceptType[@ConceptTypeID=current()/@ConceptTypeID]/ObjectType">
                            <xsl:with-param name="Resource" select="$PageProtoType"/>
                            <xsl:with-param name="mode" select="'PageProtoType'"/>
                       </xsl:apply-templates>
                    </xsl:for-each>
             </TABLE>
             <BR/>
             <TABLE Border="1">
                    <xsl:apply-templates select=
                      "//LinkProtoType[from_pageprototype/@PageProtoTypeID=$PagePrototype or
                                        to_pageprototype/@PageProtoTypeID=$PageProtoType]">
                            <xsl:sort select="@LinkProtoTypeID"/>
                            <xsl:with-param name="PageProtoType" select="$PageProtoType"/>
                    </xsl:apply-templates>
             </TABLE>
            </xsl:template>

            <xsl:template match="StaticPageProtoType | TailoredPageProtoType |
                                                                      DynamicPageProtoType">
              <li>
                <xsl:value-of select="name()"/>
              </li>
            </xsl:template>

            <!-- ObjectTypes -->

            <xsl:template match="ObjectType">
              <TR>
               <TD CLASS="ObjectType">
                 <a><xsl:attribute name="name"><xsl:value-of select="$Resource"/>-
                                                 <xsl:value-of select="@ObjectTypeID"/>
                       </xsl:attribute>
                        <xsl:value-of select="@ObjectTypeID"/>
                 </a>
               </TD>
              </TR>
```

```
       <xsl:variable name="ObjectTypeID"><xsl:value-of select=
                                                              "@ObjectTypeID"/></xsl:variable>

       <xsl:for-each select="playing_roletype">
          <xsl:sort select="@RoleTypeID"/>
          <!-- Fetch the RelationType for this RoleType -->
          <xsl:variable name="RelationType">
             <xsl:value-of select=
      "//RoleType[@RoleTypeID=current()/@RoleTypeID]/in_relationtype/@RelationType
                                                                                   ID"/>
          </xsl:variable>
   <!-- Predicate to select only those RelationTypes on this PageProtoType does not work
        since the "with_info_on"-element occurs multiple times on RelationType, therefore
         the test is done in RelationType -->
          <xsl:apply-templates select=
         "//ConceptType[RelationType/@RelationTypeID=$RelationType]/RelationType">
                <xsl:with-param name="Resource" select="$Resource"/>
                <xsl:with-param name="RoleType" select="@RoleTypeID"/>
                <xsl:with-param name="mode" select="$mode"/>
          </xsl:apply-templates>
       </xsl:for-each>

      </xsl:template>

      <xsl:template match="RelationType">
         <xsl:if test="$mode='PageProtoType'">
            <xsl:for-each select="../with_info_on">
            <!-- if-test to include only those Relation/RoleTypes on this PageProtoType  -->
               <xsl:if test="@PageProtoTypeID=$Resource">
                  <xsl:apply-templates select="//RoleType[@RoleTypeID=$RoleType]">
                     <xsl:with-param name="Resource" select="$Resource"/>
                  </xsl:apply-templates>
               </xsl:if>
            </xsl:for-each>
         </xsl:if>

         <xsl:if test="$mode='Page'">
           <xsl:for-each select="../with_population_on">
           <!-- if-test to include only those Relation/RoleTypes on this Page  -->
               <xsl:if test="@PageID=$Resource">
                  <xsl:apply-templates select="//RoleType[@RoleTypeID=$RoleType]">
                     <xsl:with-param name="Resource" select="$Resource"/>
                  </xsl:apply-templates>
               </xsl:if>
           </xsl:for-each>
         </xsl:if>

      </xsl:template>
```

```
        <xsl:template match="RoleType">
          <xsl:variable name="CounterRoleType">
              <xsl:value-of select=
"//RoleType[in_relationtype/@RelationTypeID=current()/in_relationtype/@RelationTypeID
                                                    and played_by_objecttype/@ObjectTypeID!=
                                current()/played_by_objecttype/@ObjectTypeID]/@RoleTypeID"/>
          </xsl:variable>
          <xsl:variable name="CounterObjectType">
             <xsl:value-of select=
"//RoleType[@RoleTypeID=$CounterRoleType]/played_by_objecttype/@ObjectTypeID"/>
          </xsl:variable>
          <TR>
          <!-- We need a new line for each RoleType, in case multiple RoleTypes exist for one
               ObjectType -->
           <TD/>
           <TD>
            <xsl:value-of select="@RoleTypeID"/>
            <xsl:text>/</xsl:text>
            <xsl:value-of select="$CounterRoleType"/>
           </TD>
           <TD>
             <a><xsl:attribute name="href">#<xsl:value-of select="$Resource"/>-
                                                    <xsl:value-of select="$CounterObjectType"/>
              </xsl:attribute>
              <xsl:value-of select="$CounterObjectType"/>
             </a>
           </TD>
          </TR>
        </xsl:template>

        <!-- End ObjectTypes -->

        <!-- Objects -->

        <xsl:template match="Object">
         <TR>
            <TD CLASS="Object">
              <a><xsl:attribute name="name">
                    <xsl:value-of select="$Resource"/>-<xsl:value-of select="@ObjectID"/>
                 </xsl:attribute>
                 <xsl:value-of select="@ObjectID"/>
              </a>
            </TD>
            <TD CLASS="ObjectType">
              <xsl:value-of select="ObjectInstance/of_objecttype/@ObjectTypeID"/>
            </TD>
          </TR>

          <xsl:for-each select="playing">
```

```
            <xsl:sort select="@RoleID"/>
            <!-- Fetch the Relation for this Role   -->
            <xsl:variable name="Relation">
                <xsl:value-of select=
                                        "//Role[@RoleID=current()/@RoleID]/in/@RelationID"/>
            </xsl:variable>
            <!-- Predicate to select only those Relations on this Page does not work since the
  "with_population_on"-element occurs multiple times on Relation, therefore the
   test is done in Relation -->
            <xsl:apply-templates select=
                                         "//Concept[Relation/@RelationID=$Relation]/Relation">
                  <xsl:with-param name="Resource" select="$Resource"/>
                  <xsl:with-param name="Role" select="@RoleID"/>
            </xsl:apply-templates>
        </xsl:for-each>
      </xsl:template>

      <xsl:template match="Relation">
         <xsl:for-each select="../on_page">
         <!-- if-test to include only those Relation/Role on this Page  -->
            <xsl:if test="@PageID=$Resource">
               <xsl:apply-templates select="//Role[@RoleID=$Role]">
                  <xsl:with-param name="Resource" select="$Resource"/>
               </xsl:apply-templates>
            </xsl:if>
         </xsl:for-each>
      </xsl:template>

      <xsl:template match="Role">
        <xsl:variable name="Relation">
          <xsl:value-of select="//Role[@RoleID=current()/@RoleID]/in/@RelationID"/>
        </xsl:variable>
        <xsl:variable name="CounterRole">
          <xsl:value-of select="//Role[in/@RelationID=$Relation and
                                      @RoleID!=current()/@RoleID]/@RoleID"/>
        </xsl:variable>
        <xsl:variable name="CounterObject">
          <xsl:value-of select="//Role[@RoleID=$CounterRole]/played_by/@ObjectID"/>
        </xsl:variable>
        <TR>
          <TD/><TD/>
          <TD>
           <xsl:value-of select="@RoleID"/>
           <xsl:text>/</xsl:text>
          <xsl:value-of select="$CounterRole"/>
          </TD>
          <TD>
          <a>
            <xsl:attribute name="href">#<xsl:value-of select="$Resource"/>-
                                            <xsl:value-of select="$CounterObject"/>
                </xsl:attribute>
                <xsl:value-of select="$CounterObject"/>
              </a>
            </TD>
          </TR>
        </xsl:template>

        <xsl:template match="ElementaryObject">
          <xsl:apply-templates select="Text|Video|Graphics"/>
        </xsl:template>
        <xsl:template match="Text|Video|Graphics">
          <xsl:value-of select="name()"/>
        </xsl:template>
        <xsl:template match="StructuredObject">
          <xsl:for-each select="is_composed_of">
            <xsl:value-of select="@ObjectID"/><BR/>
          </xsl:for-each>
        </xsl:template>

        <!-- End Objects -->

        <!-- Page -->
        <xsl:template match="Page" mode="simple">
           <xsl:if test="position()=1">
            <TH>
                Page
            </TH>
            <TH>
                Number of Object(Type)s
            </TH>
            <TH>
                Number of Link(ProtoType)s to
            </TH>
            <TH>
                Number of Link(ProtoType)s from
            </TH>
           </xsl:if>
          <TR>
           <TD>
             <xsl:apply-templates select="current()" mode="link"/>
           </TD>
           <TD>
             <xsl:value-of select=
"count(//Concept[on_page/@PageID=current()/@PageID][Object][count(ObjectInstance)=0])
                                                          + count(with_population_of)"/>
           </TD>
           <TD>
             <xsl:value-of select="count(is_target_of) + count(is_pagetarget_of)"/>
           </TD>
           <TD>
```

```
    <xsl:value-of select="count(is_source_of) + count(is_pagesource_of)"/>
   </TD>
  </TR>
</xsl:template>

<xsl:template match="PageProtoType" mode="simple">
  <xsl:if test="position()=1">
    <TH>
        PageProtoTypes
    </TH>
    <TH>
        Number of ObjectTypes
    </TH>
    <TH>
        Number of LinkProtoTypes to
    </TH>
    <TH>
        Number of LinkProtoTypes from
    </TH>
    <TH>
        PageInstances
    </TH>
  </xsl:if>
  <xsl:param name="PageProtoType"><xsl:value-of select=
                                                  "@PageProtoTypeID"/></xsl:param>
  <TR>
    <TD>
     <a><xsl:attribute name="href">#<xsl:value-of
                                     select="@PageProtoTypeID"/></xsl:attribute>
      <xsl:value-of select="@PageProtoTypeID"/>
     </a>
   </TD>
   <TD>
     <xsl:value-of select="count(with_info_of)"/>
   </TD>
   <TD>
     <xsl:value-of select="count(is_prototypetarget_of)"/>
   </TD>
   <TD>
     <xsl:value-of select="count(is_prototypesource_of)"/>
   </TD>
   <TD>
    <xsl:apply-templates select=
         "//Page[PageInstance/of_pageprototype/@PageProtoTypeID=$PageProtoType]"
                                                                   mode="link"/>
   </TD>
 </TR>
</xsl:template>

<xsl:template match="PageInstance">
```

```
     <li>
     Instance of PageProtoType :
     <a><xsl:attribute name="href">#<xsl:value-of select=
                                                   "of_pageprototype/@PageProtoTypeID"/>
        </xsl:attribute>
        <xsl:value-of select="of_pageprototype/@PageProtoTypeID"/>
     </a>
     </li>
    </xsl:template>
    <xsl:template match="Page" mode="link">
       <a><xsl:attribute name="href">#<xsl:value-of select="@PageID"/>
          </xsl:attribute>
          <xsl:value-of select="@PageID"/>
       </a>
    </xsl:template>

    <!-- End Page -->

    <!-- Links -->

    <xsl:template match="Link">
     <xsl:if test="position()=1">
      <CAPTION><H3>Links : </H3></CAPTION>
      <TR>
       <TH>Link</TH>
       <TH>Link prototype</TH>
       <TH>From page</TH>
       <TH>To page</TH>
       <TH>Type of Link</TH>
      </TR>
     </xsl:if>
      <TR>
       <TD>
        <a>
          <xsl:attribute name="name">
              <xsl:value-of select="@LinkID"/>
          </xsl:attribute>
        </a>
        <xsl:value-of select="@LinkID"/>
       </TD>
       <TD>
        <xsl:apply-templates select="LinkInstance/of_linkprototype">
                <xsl:with-param name="PageProtoType" select="$PageProtoType"/>
        </xsl:apply-templates>
       </TD>
       <TD>
        <xsl:apply-templates select="from"/>
       </TD>
       <TD>
        <xsl:apply-templates select="to"/>
      </TD>
      <TD>
        <xsl:apply-templates select="InPageLink | OutPageLink | OutSideLink"/>
      </TD>
     </TR>
 </xsl:template>

 <xsl:template match="LinkInstance">
        <xsl:value-of select="@LinkInstanceID"/>
 </xsl:template>
 <xsl:template match="of_linkprototype">
        <a>
          <xsl:attribute name="href">
                   #<xsl:value-of select="$PageProtoType"/>-<xsl:value-of
                                                              select="@LinkProtoTypeID"/>
          </xsl:attribute>
          <xsl:value-of select="@LinkProtoTypeID"/>
        </a>
 </xsl:template>
 <xsl:template match="from">
        <a>
          <xsl:attribute name="href">#<xsl:value-of select="@PageID"/>
          </xsl:attribute>
          <xsl:value-of select="@PageID"/>
        </a>
 </xsl:template>
 <xsl:template match="to">
        <a>
          <xsl:attribute name="href">#<xsl:value-of select="@PageID"/>
          </xsl:attribute>
          <xsl:value-of select="@PageID"/>
        </a>
 </xsl:template>
 <xsl:template match="InPageLink | OutPageLink | OutSideLink">
        <xsl:value-of select="name()"/>
 </xsl:template>

 <!-- End Links -->

 <!-- LinkProtoType -->
 <xsl:template match="LinkProtoType">
  <xsl:if test="position()=1">
   <CAPTION><H3>LinkProtoTypes : </H3></CAPTION>
   <TR>
    <TH>Linkprototype</TH>
    <TH>From pageprototype</TH>
    <TH>From page</TH>
    <TH>To pageprototype</TH>
    <TH>To page</TH>
   </TR>
```

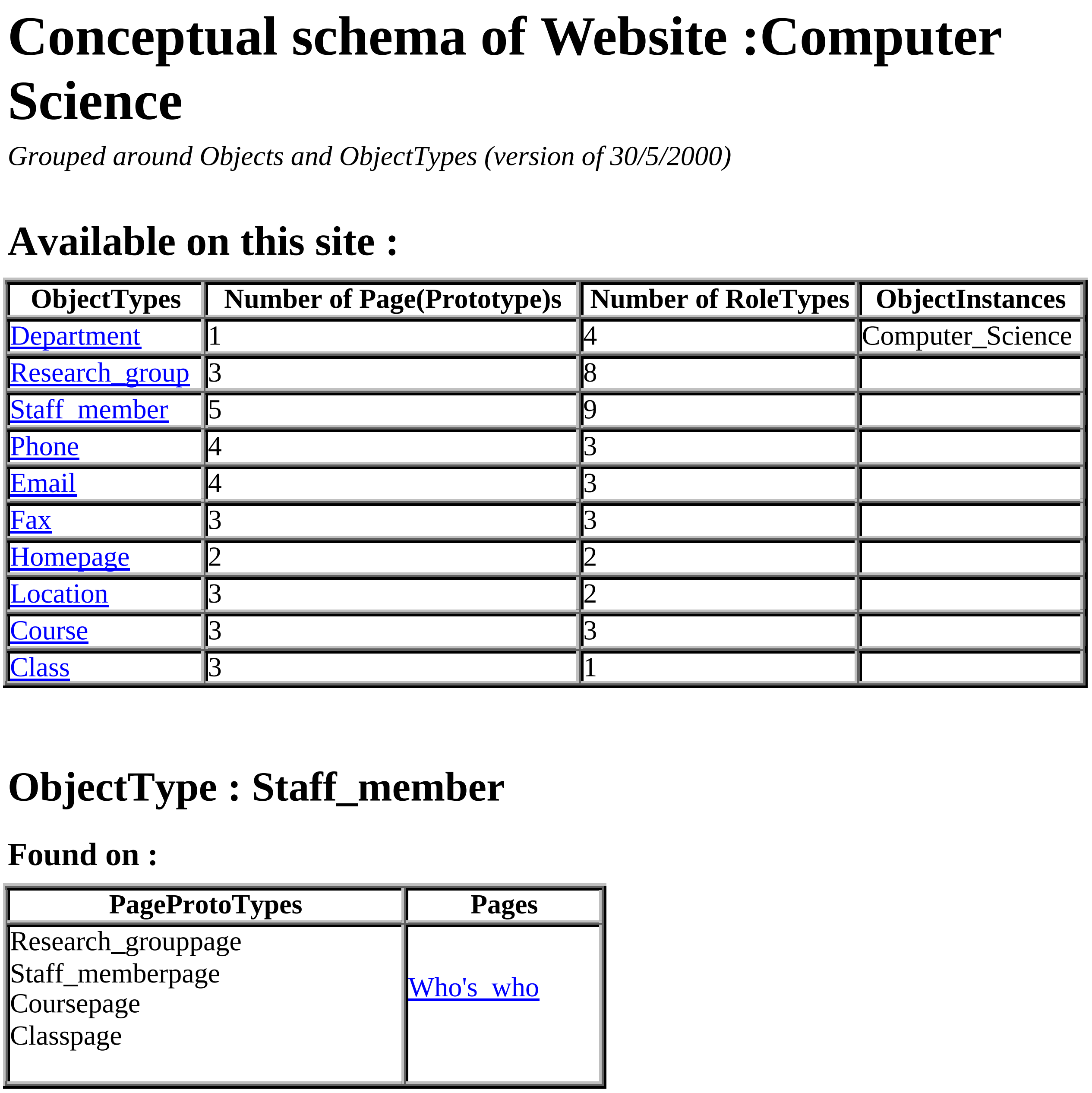
```
             </xsl:if>
             <TR>
               <TD>
                  <a>
                     <xsl:attribute name="name">
                          <xsl:value-of select="$PageProtoType"/>-<xsl:value-of
                                                                      select="@LinkProtoTypeID"/>
                     </xsl:attribute>
                     <xsl:value-of select="@LinkProtoTypeID"/>
                   </a>
               </TD>
               <TD>
                  <xsl:apply-templates select="from_pageprototype"/>
               </TD>
               <TD>
                  <xsl:apply-templates select="from_page"/>
               </TD>
               <TD>
                  <xsl:apply-templates select="to_pageprototype"/>
               </TD>
               <TD>
                  <xsl:apply-templates select="to_page"/>
               </TD>
             </TR>
            </xsl:template>

            <xsl:template match="from_pageprototype">
                     <a>
                       <xsl:attribute name="href">#<xsl:value-of select="@PageProtoTypeID"/>
                       </xsl:attribute>
                       <xsl:value-of select="@PageProtoTypeID"/>
                    </a>
            </xsl:template>
            <xsl:template match="to_pageprototype">
                    <a>
                       <xsl:attribute name="href">#<xsl:value-of select="@PageProtoTypeID"/>
                       </xsl:attribute>
                       <xsl:value-of select="@PageProtoTypeID"/>
                    </a>
            </xsl:template>
            <xsl:template match="from_page">
                    <a>
                       <xsl:attribute name="href">#<xsl:value-of select="@PageID"/>
                       </xsl:attribute>
                       <xsl:value-of select="@PageID"/>
                    </a>
            </xsl:template>
```

```
<xsl:template match="to_page">
        <a>
           <xsl:attribute name="href">#<xsl:value-of select="@PageID"/>
           </xsl:attribute>
           <xsl:value-of select="@PageID"/>
        </a>
</xsl:template>

<!-- End LinkProtoType -->

</xsl:stylesheet>
```

Figure 9 Generated HTML-document: Objects and ObjectTypes

We will only highlight some specific parts of the HTML document, since this is quite a large document to include in this thesis.
What will you find:

- Index
- Staff_member which is an ObjectType
- Since there aren't any Objects in the model and we did not mention the ObjectInstances explicitely one will not find an example of an Object.



Conceptual schema of Website :Computer Science

Grouped around Objects and ObjectTypes (version of 30/5/2000)

Available on this site :

ObjectTypes	Number of Page(Prototype)s	Number of RoleTypes	ObjectInstances
Department	1	4	Computer_Science
Research_group	3	8	
Staff_member	5	9	
Phone	4	3	
Email	4	3	
Fax	3	3	
Homepage	2	2	
Location	3	2	
Course	3	3	
Class	3	1	

ObjectType : Staff_member

Found on :

PageProtoTypes	Pages
Research_grouppage Staff_memberpage Coursepage Classpage	Who's_who

In RelationTypes :

RoleTypes	CounterObjectType
chair_of/has_chair	Research_group
member_of/has	Research_group
lecturer_of/has_lecturer	Course
assistant_of/has_assistant	Course
member_has_phone/phone_of_member	Phone
member_has_fax/fax_of_member	Fax
member_has_email/email_of_member	Email
member_has_homepage/homepage_of_member	Homepage
member_has_location/location_of_member	Location

Figure 10 Style sheet: Objects and ObjectTypes

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/XSL/Transform/1.0'>
<!-- NOT THE OFFICIAL NAMESPACE, ONLY TO BE USED WITH XSL EDITOR -->

<xsl:output method="html"/>
<xsl:template match="/Website">
<HTML>
 <HEAD>
       <LINK rel="stylesheet" href="website.css" type="text/css"/>
 </HEAD>
 <BODY>
 <H1><xsl:text>Conceptual schema of Website :</xsl:text>
      <xsl:value-of select="@WebsiteID"/>
 </H1>
 <EM><xsl:text>Grouped around Objects and ObjectTypes (version of
                                                     30/5/2000)</xsl:text></EM>
 <BR/>  <BR/>
 <H2>Available on this site :</H2>
 <TABLE Border = "2">
    <xsl:apply-templates select="//ObjectType" mode="simple"/>
    <xsl:apply-templates select="//Object[count(ObjectInstance)=0]"
                                                              mode="simple"/>
                  <!-- Excluding ObjectInstances -->
 </TABLE>
 <xsl:apply-templates select="//ObjectType" mode="extended">
       <xsl:sort select="@ObjectTypeID"/>
 </xsl:apply-templates>
 <xsl:apply-templates select="//Object[count(ObjectInstance)=0]" mode="extended">
                  <!-- Excluding ObjectInstances -->
       <xsl:sort select="@ObjectID"/>
 </xsl:apply-templates>
 </BODY>
</HTML>
</xsl:template>
<!-- End Main Template -->

<!-- OBJECTS -->
<xsl:template match="Object" mode="extended">
  <H2>Object :
   <a><xsl:attribute name="name"><xsl:value-of select="@ObjectID"/>
         </xsl:attribute>
         <xsl:value-of select="@ObjectID"/>
   </a>
  </H2>
  <H3>
  <ul>
       <xsl:apply-templates select="..//has_name"/>
```

```
            <xsl:apply-templates
                  select="StructuredObject|ElementaryObject"/>
   </ul>
   </H3>
   <BR/>
   <TABLE BORDER="2">
     <xsl:apply-templates select="../on_page"/>
   </TABLE>
   <BR/>
   <TABLE BORDER="2">
      <xsl:apply-templates select="playing"/>
   </TABLE>
 </xsl:template>

 <xsl:template match="playing">
   <xsl:if test="position()=1">
     <CAPTION><H3>In Relations : </H3></CAPTION>
     <TH>Role</TH>
     <TH>CounterObject</TH>
   </xsl:if>
   <TR>
     <TD CLASS="Role">
       <xsl:value-of select="@RoleID"/>
       <xsl:variable name="RoleID"><xsl:value-of
                         select="@RoleID"/></xsl:variable>
       <xsl:text>/</xsl:text>
       <xsl:variable name="RelationID">
              <xsl:value-of select=
                                    "//Role[@RoleID=current()/@RoleID]/in/@RelationID"/>
       </xsl:variable>
       <xsl:variable name="CounterRoleID">
              <xsl:value-of select="//Role[in/@RelationID=$RelationID
                                                and @RoleID!=current()/@RoleID]/@RoleID"/>
       </xsl:variable>
       <xsl:value-of select="$CounterRoleID"/>
     </TD>
     <TD CLASS="Object">
       <xsl:variable name="CounterObjectID">
            <xsl:value-of
   select="//Role[@RoleID=$CounterRoleID]/played_by/@ObjectID"/>
       </xsl:variable>
       <a><xsl:attribute name="href">#<xsl:value-of select="$CounterObjectID"/>
         </xsl:attribute>
         <xsl:value-of select="$CounterObjectID"/>
       </a>
     </TD>
    </TR>
 </xsl:template>
```

```
<xsl:template match="on_page">
  <xsl:if test="position()=1">
    <CAPTION><H3>Found on : </H3></CAPTION>
    <TH>Page</TH>
  </xsl:if>
  <TR>
    <TD CLASS="Page">
      <xsl:value-of select="@PageID"/>
    </TD>
  </TR>
</xsl:template>

<xsl:template match="has_name">
  <li>
    has name : <xsl:value-of select="current()"/>
  </li>
</xsl:template>
<xsl:template match="ElementaryObject|StructuredObject">
  <li>
    consists of : <xsl:apply-templates select="Text|Graphics|Video"/>
                <xsl:apply-templates select="is_composed_of"/>
  </li>
</xsl:template>
<xsl:template match="Text|Graphics|Video">
  <xsl:value-of select="name()"/>
</xsl:template>
<xsl:template match="is_composed_of">
  <a>
    <xsl:attribute name="href">#<xsl:value-of
                        select="@ObjectID"/></xsl:attribute>
    <xsl:value-of select="@ObjectID"/>
  </a>
  <xsl:if test="position()!=last()"><xsl:text>, </xsl:text></xsl:if>
</xsl:template>

<!-- END OBJECTS -->

<!-- OBJECTTYPES -->
<xsl:template match="ObjectType" mode="extended">
  <H2>ObjectType :
   <a><xsl:attribute name="name"><xsl:value-of
                     select="@ObjectTypeID"/></xsl:attribute>
      <xsl:value-of select="@ObjectTypeID"/>
   </a>
  </H2>
  <H3>
  <ul>
       <xsl:apply-templates select="../has_typename"/>
       <xsl:apply-templates select="is_sub|is_super"/>
  </ul>
```

```
            </H3>
            <BR/>
            <xsl:if test="(count(../with_population_on) + count(../with_info_on)) &gt; 0">
            <TABLE BORDER="2">
              <CAPTION><H3>Found on : </H3></CAPTION>
              <TH>PageProtoTypes</TH>
              <TH>Pages</TH>
              <TR>
                <TD CLASS="PageProtoType">
                  <xsl:apply-templates select="../with_info_on"/>
                </TD>
                <TD CLASS="Page">
                  <xsl:apply-templates select="../with_population_on"/>
                </TD>
              </TR>
            </TABLE>
           </xsl:if>
           <BR/>

           <TABLE BORDER="2">
               <xsl:apply-templates select="playing_roletype"/>
           </TABLE>
          </xsl:template>

          <xsl:template match="playing_roletype">
            <xsl:if test="position()=1">
             <CAPTION><H3>In RelationTypes : </H3></CAPTION>
             <TH>RoleTypes</TH>
             <TH>CounterObjectType</TH>
            </xsl:if>
            <TR>
               <TD CLASS="RoleType">
                   <xsl:value-of select="@RoleTypeID"/>
                      <xsl:text>/</xsl:text>
                    <xsl:variable name="RelationTypeID">
                            <xsl:value-of select=
"//RoleType[@RoleTypeID=current()/@RoleTypeID]/in_relationtype/@RelationTypeID"/>
                    </xsl:variable>
                    <xsl:variable name="CounterRoleID">
                            <xsl:value-of select=
                                          "//RoleType[in_relationtype/@RelationTypeID=$RelationTypeID
                                            and @RoleTypeID!=current()/@RoleTypeID]/@RoleTypeID"/>
                    </xsl:variable>
                    <xsl:value-of select="$CounterRoleID"/>
               </TD>
               <TD CLASS="ObjectType">
                    <xsl:variable name="CounterObjectTypeID">
                         <xsl:value-of select=
      "//RoleType[@RoleTypeID=$CounterRoleID]/played_by_objecttype/@ObjectTypeID"/>
                    </xsl:variable>
          <a><xsl:attribute name="href">#<xsl:value-of
                              select="$CounterObjectTypeID"/>
             </xsl:attribute>
               <xsl:value-of select="$CounterObjectTypeID"/>
          </a>
      </TD>
    </TR>
   </xsl:template>

   <xsl:template match="with_population_on">
       <a><xsl:attribute name="href"><xsl:value-of
              select="//Page[@PageID=current()/@PageID]/URL"/>
         </xsl:attribute>
         <xsl:value-of select="@PageID"/>
       </a>
       <BR/>
   </xsl:template>
   <xsl:template match="with_info_on">
       <xsl:value-of select="@PageProtoTypeID"/>
       <BR/>
   </xsl:template>
   <xsl:template match="has_typename">
     <li>
      has name : <xsl:value-of select="current()"/>
     </li>
   </xsl:template>
   <xsl:template match="is_sub">
     <li>
      ObjectType is subtype of :
           <a><xsl:attribute name="href">#<xsl:value-of select="@ObjectTypeID"/>
              </xsl:attribute>
             <xsl:value-of select="@ObjectTypeID"/>
           </a>
     </li>
   </xsl:template>
   <xsl:template match="is_super">
     <li>
      ObjectType is supertype of :
           <a><xsl:attribute name="href">#<xsl:value-of select="@ObjectTypeID"/>
              </xsl:attribute>
             <xsl:value-of select="@ObjectTypeID"/>
           </a>
     </li>
   </xsl:template>
   <xsl:template match="ObjectType" mode="simple">
    <xsl:if test="position()=1">
    <TR>
      <TH>
        ObjectTypes
      </TH>
    <TH>
       Number of Page(Prototype)s
    </TH>
    <TH>
       Number of RoleTypes
    </TH>
    <TH>
       ObjectInstances
    </TH>
  </TR>
  </xsl:if>
  <TR>
   <TD CLASS="ObjectType">
    <a><xsl:attribute name="href">#<xsl:value-of select="@ObjectTypeID"/>
      </xsl:attribute>
      <xsl:value-of select="@ObjectTypeID"/>
    </a>
   </TD>
   <TD>
    <xsl:value-of select="count(../with_info_on) + count(../with_population_on)"/>
   </TD>
   <TD>
    <xsl:value-of select="count(playing_roletype)"/>
   </TD>
   <TD CLASS="ObjectInstance">
    <xsl:for-each select="with_objectinstance">
      <xsl:value-of select="@ObjectInstanceID"/>
    </xsl:for-each>
   </TD>
  </TR>
 </xsl:template>

 <!-- END OBJECTTYPES -->

 <xsl:template match="Object" mode="simple">
   <xsl:if test="position()=1">
    <TH>
       Objects
    </TH>
    <TH>
       Number of Pages
    </TH>
    <TH>
       Number of Roles
    </TH>
   </xsl:if>
    <TR>
     <TD CLASS="Object">
      <a><xsl:attribute name="href">#<xsl:value-of select="@ObjectID"/>
           </xsl:attribute>
```

```
          <xsl:value-of select="@ObjectID"/>
      </a>
     </TD>
     <TD>
      <xsl:value-of select="count(../on_page)"/>
     </TD>
     <TD>
      <xsl:value-of select="count(playing)"/>
     </TD>
    </TR>
</xsl:template>

</xsl:stylesheet>
```

References

[BP 98] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen
“Extensible Markup Language (XML) Version 1.0”, W3C Recommendation
10 February 1998
http://www.w3.org/TR/REC-xml

[CD 99] James Clark, Steven DeRose
“XML Path Language (XPath) Version 1.0”, W3C Recommendation
16 November 1999
http://www.w3.org/TR/xpath

[AB 2000] Sharon Adler, Anders Berglund…
“Extensible Style Sheet Language (XSL) Version 1.0”, W3C Working Draft
1 March 2000
http://www.w3.org/TR/xsl/

[Cl 99] James Clark
"XSL Transformations (XSLT) Version 1.0", W3C Recommendation
16 Novembre 1999
http://www.w3.org/TR/WD-xslt

[BH 99] Tim Bray, Dave Hollander, Andrew Layman
"Namespaces in XML", W3C Recommendation 14 January 1999
http://www.w3.org/TR/REC-xml-names

[LS 99] Ora Lassila, Ralph R. Swick
"Resource Description Framework (RDF) Model and Syntax Specification",
W3C Recommendation 22 February 1999
http://www.w3.org/TR/REC-rdf-syntax

[RL 98] Dave Raggett, Arnaud Le Hors, Ian Jacobs
"HTML 4.0 Specification", W3C Recommendation 24 April 1998
http://www.w3.org/TR/REC-html40

[LB 99] Hakon Wium Lie, Bert Bos
"Cascading Style Sheets, level 1", W3C Recommendation
revised 11 January 1999
http://www.w3.org/TR/REC-CSS1

[BL 98] Bert Bos, Hakon Wium Lie, Chris Lilley, Ian Jacobs
"Cascading Style Sheets, level2", W3C Recommendation
12 May 1998
http://www.w3.org/TR/REC-CSS2

[BD 98] Frank Boumphrey, Olivia Direnzo, …
"Programmer to Programmer XML Applications"
Wrox Press Ltd. 1998

[Dec 99] T. Decruyenaere
"The use of XML in modeling Conceptual Schemas in web site design",
Master Thesis, Vrije Universiteit Brussel, 1999.

[DD 2000] O. De Troyer, T. Decruyenaere
"Conceptual Modelling of Web Sites for End-Users"

[Hal 95] Terry Halpin
"Conceptual Schema & Relational Database Design" (second edition)
Prentice Hall Australia Pty Ltd ,1995

Fora on the world wide web

http://msdnnews.microsoft.com/
http://www.w3.org/
http://www.xslt.com/xslt_tools.htm

Used tools

√ Microsoft Internet Explorer 5 with msxml3 technology preview extension
Is said to be a full implementation of XSLT, but we still encountered many problems. The list of known bugs can be found on http://msdn.microsoft.com/xml/c-frame.htm?/workshop/xml/general/msxml_buglist.asp.

√ IBM Alphaworks XSL Editor: http://www.alphaworks.ibm.com/tech/xsleditor
We used this reliable, early implementer of the full XSLT recommendation, tool to edit the style sheets. The tool enables to debug the style sheets, by walking through the templates step wise.
Remark that the namespace used for XSLT-elements is not the official one.
Though the debug facilities have proven great value when starting with XSLT, debugging soon gets too slow when the style sheets grow.
Another flaw is selection with a predicate on the attribute of an element appearing multiple times. Though this is in the recommendation, it doesn’t work in the tool.

√ XSL Tester 1.1.7 allows to view formatted output in a browser window but is still showing many errors.

√ Microsoft Notepad

√ Kazushi Pc Editor 2