5 IT'S ALL ABOUT MODELS

De Bra et. al outline that an AHS should perform its tasks without requiring any sort of programming by the authors [De Bra+99]. Therefore, the first task towards adaptive behaviour is for the system to learn the content of a domain. Section 5.1 investigates how to develop suitable methods for extracting information from the domain sources in an hypertext context. Next, in section 5.2 we find that Heuristics can ensure quality to the process of abstracting concepts, and we try to identify relations among the concepts found. From this basis, section 5.3 raises issues on how a knowledge representation of the domain can be constructed and how the generation of adaptive documents for each user can be fulfilled.

5.1 Extracting information

It is important to separate the information gathering from the information processing. According to the overall plan outlined in the previous chapter, a crucial step towards a domain model concerns the identification and modelling of the content of each document. After stating the hypothesis of our work, this section discusses the nature of concepts. Also, generic document characteristics and proper techniques for extracting candidate concepts are identified.

5.1.1 Hypothesis of this thesis

As mentioned in the introduction, some models might represent the domain knowledge better than others, and as models play important roles in adaptive systems, the performance of the system depends on the quality of the models. Figure 5–1 illustrates the different roles played in the adaptive system explaining the notion of *high quality*: The author has a mental model of some knowledge, and expresses this knowledge as HTML documents. During analysis of the static documents, the adaptive hypertext system tries to generate a domain model which agrees with the author's view of the domain. The domain model and the user models make up the basis from which the system can generate documents adapted to each user. Since the adaptation depends highly on the content and structure of the domain model, we therefore choose to focus on ensuring high quality to the semi-automatic construction of the domain model, and thereafter use the methods found to build an adaptive hypertext system. Remember from the introductory part that with high quality we mean that the system should capture the domain knowledge well with respect to concepts and relations, so that the domain model is in accordance with the author's view of the domain.

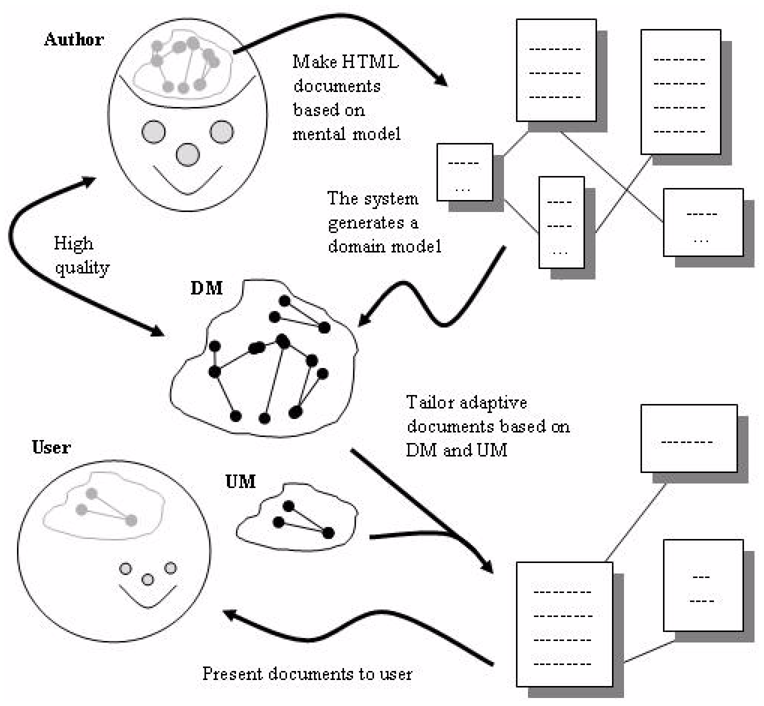

Author
Make HTML documents based on mental model
High quality
The system generates a domain model
DM
Tailor adaptive documents based on DM and UM
User
UM
Present documents to user

Figure 5–1: The concept of high quality explained in terms of the actors.

We hypothesise that the combination of Information Retrieval (IR) techniques and an analysis of document properties and hypertext structure, yields a means to both find and use Heuristical rules for the identification of concepts and relations, and to ensure a resulting domain model of high quality for an adaptive hypertext system.

Table 5–1: Main hypothesis

The combination of *IR-techniques* and an *analysis* of document properties and hypertext structure with respect to content, yields Heuristics that secure the production of an AHS domain model of high quality.

5.1.2 Some possible approaches

How can the content of the documents be abstracted? One way is to require that each document should be structured in a specific manner and tagged with respect to content. Knowing exactly how the documents look like, the adaptive hypertext system can be programmed to perform powerful adaptations. This is clearly a very inflexible approach in that it places a heavy load on the author in the tagging phase. It also conflicts goals of using existing HTML documents as easily as possible, as exemplified in the system KN-AHS, which compose documents based on knowledge about with which hotwords the user is familiar [Kobsa+94]. The hotwords are designed to fit the system in advance of interaction. Systems that manage personalised views of information spaces extend the notion of classical IR-hypermedia systems by not only locating but also organising the information in a way the user wants [Brusilovsky01]. However, running traditional IR-analysis on the documents with vectors representing each section in a document, the adaptive information could be presented by comparing the vector-representation of the document requested for with the user's knowledge represented in a similar vector space. Marinilli et. al propose a case based approach to information filtering that presents HTML documents according to the interests of the users. A filtering component selects relevant documents for each user by means of a vector model. After exposing a text document to stoplist, weighting and separation of words, the resulting array of values (one value for each category) is mapped into stereotypes [Marinilli+99]. Before bringing our approach to light, we refresh on HTML.

5.1.3 What is HTML?

According to the specifications of the World Wide Web Consortium (W3C), HTML (Hyper Text Markup Language) "is the lingua franca for publishing text on the web" [w3]. It is essentially a subset of SGML, but even though a much less complexity in HTML, the language usually offers authors enough flexibility. An HTML-document may consist of elements like text, tables, graphics, sound etc., and these elements are marked up by means of tags. As HTML is fairly easy to learn, anyone can make, structure and format their own Internet pages. The documents may be created either by writing tags and text directly into an HTML-editor or by converting an existing formatted text-document to an HTML-document. In the latter case, a standard text-editor[1] uses advanced techniques like style sheets and the like to extend the expressive power of the HTML-language[2] and make an HTML-document look as similar as possible compared to the original document. A complicating characteristic with such automatic conversion, is that the source of the resulting HTML-document is very difficult to read and edit for a person due to much redundant coding from the automatic process. Programs to "clean up" such auto-generated code, or bad programmed documents, exist, however[3].

An HTML document consists of a header section with meta information and a body section with the text and graphics presented on the screen. The tags mark up the text, so if the author of a document wishes to make a piece of it appear in bold, <B> and </B> tags mark where the bold face-formatting should start and end, respectively. The tags are present in the HTML-file, but only the interpreted result is displayed on the screen by the browser. More advanced features like XML, stylists, Javascript, Java applets and the like can be embedded as necessary to further format and add functionality to the documents.

1. Text editors range from simple to more complex ones, e.g. Notepad, Microsoft Word, Lotus AmiPro, FrameMaker, etc.
2. DTDs and style sheets redefines the standard formatting provided by the tags.
3. The program "Tidy" for Unix is an example of a program that fixes incorrectly tagged documents.

5.1.4 Using tags in the conceptualization process

Brusilovsky emphasises that adaptivity requires knowledge about links and documents, where the documents must be indexed according to the user's goals, knowledge and background [Brusilovsky01]. The process of conceptualization tries to abstract the information of a domain and associate descriptive concepts with this information. For an AHS there is a need to separate the information entities and understand their relative importance as knowledge sources for the user. We therefore question how to identify and conceptualise each chunk of knowledge before constructing the domain model.

There are two interesting observations concerning the nature of online material. First, with the ability to link different documents the author can make a more natural decomposition of large documents into smaller ones, so that the size of an HTML document is on the average relatively small, often limited to knowledge sources dealing with a specific subject. Secondly, writing and formatting a document normally involves emphasising important words, structuring the text through paragraphs, setting up links to related documents, summarising information in tables and bulleted lists, and perhaps of most importance: using descriptive headings. The purpose of formatting is to improve readability and the user’s understanding of the content, and from this we can advantage when aiming for an AHS. Assuming that most concepts, conceptual structures and domain specific terminology appear in the documents of the domain, it should be promising to acquire knowledge from the corresponding nodes [Kietz+00]. Figure 5–2 illustrates how an HTML document is divided into different elements, or sections, in terms of content and presentational form.

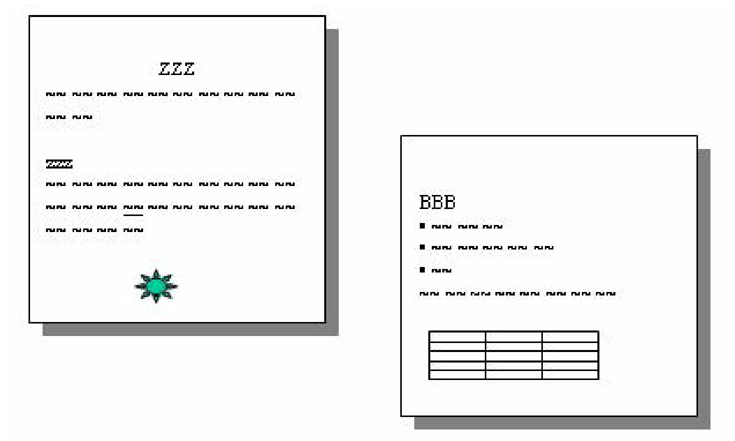


Figure 5–2: Example of HTML documents as they appear for the user on screen, illustrating how the author might organise content into different elements.

In adaptive hypermedia systems, concepts are used both in the domain model and throughout user interaction, and are essential to the system since they are the clues that point to the specific knowledge sources. Davis et. al argues that it is important that a knowledge representation acts well as a medium of communication and expression [Davis+93]. The conceptualization can be fulfilled by picking out the most promising keyword terms from the elements, and concepts can be labelled in a way understandable by a person by using the keywords abstracted. This method therefore agrees with the demand to simplify the manual evaluation and adjustment from a domain expert and facilitates the author’s interaction with the domain model. Moreover, users can better control the content of their user models by simply looking at the concept label and sort out which concepts seem familiar or not. In their simplest form, concepts would consist of only one keyword term. By permitting several concurrent terms, or phrases like “Saying the right thing at the right time” when labelling the conceptual states, the problem of how to compare different concepts arises. On the other hand, if several independent terms are allowed for the concept, similarity measures can be applied. Finally, notice the possible ambiguity in that a term can have different meanings in distinct contexts.

De Bra et. al distinguish three kinds of concepts. Atomic concepts are the smallest information units, pages are composed of atomic concepts, and abstract concepts represent larger units of information [De Bra+99]. In a similar fashion, we assume that regarding content, documents are likely to have a context superior to its sections, and likewise, some documents might be superior to other documents. We therefore find it convenient to separate *document concepts* from *element concepts* in the process of building a domain model.

5.1.5 Importance of the elements

Since the formatting tags are included along with the content in the HTML-files, the system can easily search for concepts in elements regarded important. A tag is embraced in brackets and has a name, but not all tags need to be closed, that is, no end-tag is needed to mark the end of the element in order for the browser to interpret the page correctly. Most tags have some optional attributes that can be assigned values specifying the purpose or layout of elements. Table 5–2 presents the most commonly used tags.

Even though different tags are designed for different purposes in HTML, the author is free to use the tags for other purposes, like layout. There is no guarantee that tags are used coherently among different domains or even within a domain.[1] Particularly, in their survey of the history of hypertext, Ashman et. al call attention to studies which have shown that the set of links varies a lot from author to author [Ashman+99].

The diversity of the use of tags complicates the process of selecting which elements are the best indicators of concepts and choosing the correct level of abstraction. In order to extract the meaning from a document based on its elements, we need to find out and analyse how the average author has used tags to mark up documents, or in other words, identify what are the most important tags in terms of conceptualization for a randomly picked document. As expected, this task is complicated. Therefore, and due to the limited scope of this thesis, the following

1. XML (eXtended Markup Language) is designed with coherence in mind, but since the vast majority of the documents on the web today is HTML-documents, we have chosen to stick to HTML in this research.

discussion is based on a small set of empirical data focusing on the ideal use of tags, the different roles the elements play and potential traps when it comes to extracting concepts based on the elements.

Table 5–2 The most commonly used HTML-tags

Tag name	Short description	Notes	End
<HTML>	Language used is HTML	The document is embraced by this tag	No
<HEAD>	Header information	Not visible to the user, only to the browser	Yes
<TITLE>	A descriptive title of the document	Often used to label the browser window	Yes
<META>	Main purpose is to guide search engines during indexing	The attributes are identified by search engines	No
<BODY>	Indicates the start of the visible portion of a page	Everything contained within the document body is visible to the user	No
<H1>	Heading level 1	Largest font size of the six headings	Yes
<H6>	Heading level 6	Smallest font size of the six headings	Yes
<UL>	Unordered list of LI-elements	Can be nested	Yes
<OL>	Ordered list of LI-elements	Can be nested	Yes
<LI>	List element	Each LI forces a new line	No
<EM>, <I>	Emphasise and Italic, respectively	The <EM> tag usually appears as italic	Yes
<B>, <U>	Bold and Underline, respectively	<U> should not be confused with links	Yes
<FONT>	Provides font formatting	Additional attributes specify size/color	Yes
<P>	Denotes a paragraph	Inserts an empty space before and after	No
<TABLE>	Has rows <TR> and columns <TD>	A table cell can hold any element including another table	Yes
<IMG>	Inserts an image in the document. A description is shown in case the browser has turned images off	<IMG SRC=”name” ALT=”descriptive text goes here”>	No
<!-- -->	Comment	Not interpreted by the browser	Yes
<SCRIPT>	Embeds a script of a specified type	Scripts can be stored in external files	Yes
<A HREF> </A>	Link, i.e. hyper reference	<A HREF =”url”> links to the document with the url specified <A NAME=”location”> creates a target location somewhere *within* a document that can be linked to <A HREF=”url#location”> links to a specific location inside a document	Yes
<BR>	Line break	Inserts a carriage return. No spacing	No
<DIV>	Element used for special purposes	Often contains a class definition	Yes

Meta elements contain information about a document, hidden for the visitor but usable for spiders helping search engines to index Web documents. The three most important meta tags are the description meta tag, the keyword meta tag and the title meta tag. The last one is most commonly used and should identify the content of the document in a fairly wide context. Since the title is a property of the document and not part of the text presented to the user, it can and often is used to label the browser window. Its value is also default as descriptor for both browser history and bookmarked favourite sites. Short and descriptive titles should be used, but there is no guarantee the same title is not used throughout the domain. In general, a drawback in the adaptive setting yields that meta information in the worst case could be equal for every document in a domain. Even worse, this often happens to be true. When creating a bunch of documents an efficient technique is to make a template document with only the structure or form, this acting as the basis for constructing the domain documents in order to ensure the same layout and local structure. Meta information is not visible to the user and the author might therefore more easily forget to change it according to the content of each document.

Headings serve the purpose of describing the content of the sections to follow. Hence headings alone are powerful sources of identifying keywords connected to the distinct elements and the document as a whole. Headings are ordered according to levels ranging from <H1> to <H6>, and by default the font size of <H1> is larger than <H2>, which is larger than <H3> etc., so a natural assumption to make is that the structure of the document agrees with the intended tree-like ordering provided by the HTML specification.

Tags like <B>, <I>, <EM>, <U>, <CITE> and <STRONG> are designed to emphasise text. They often occur inside other elements, and hence serve two purposes: both to outline important keywords or even whole sentences, and to make the text more structured and readable. Even though promising, note that there is no guarantee that emphasizers are used properly or hold important words representative for their parent elements.

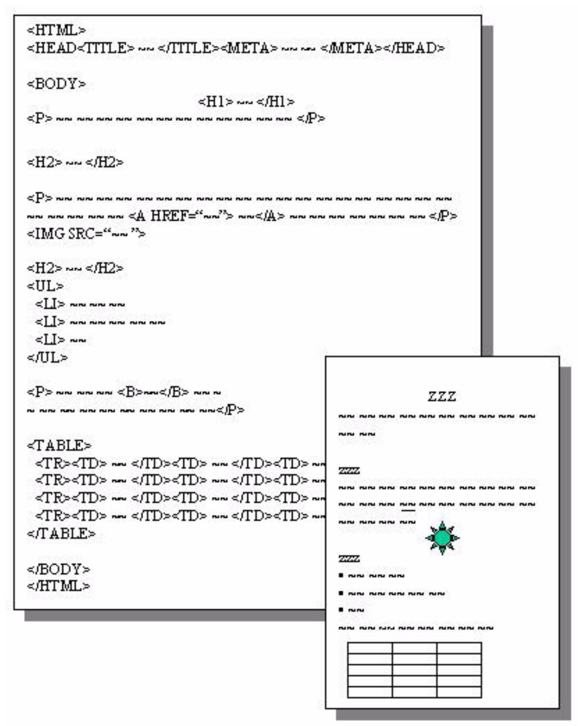
Among indicators of intended groupings are ordered and unordered lists, designated with <OL> and <UL> respectively. Lists consist of several <LI> elements. There is a possibility of nesting lists, that is a <LI> element may actually be replaced by a new list. Another commonly used element for grouping information into paragraphs, is the <P> tag, popular in most documents since it is the easiest way to embody perceived open spaces between textual elements. Other groupings of text are provided by block quotes and tables. The latter is both an interesting and complex matter. <TABLE> can be used to group other HTML elements, summarise information, or even for layout purposes. Many efforts have tried to extract or categorise information into e.g. databases based on analysing table content like in [Cohen00]. It is difficult for the system to find out what purpose a table serves, and hence an analysis of the information exhibited is not too promising.

<A HREF> is the tag indicating a hyper reference, or link, to (somewhere in) a document. Links provide a means to connect related documents, allowing the user to quickly manoeuvre the hyperspace from one place to another. The power of a link is that it can point to anything. Without links, the flexibility of the web disappears. Many styleguides recommend that the text embraced by the link tag should be descriptive of what it is linking to [Berners-Lee95], that is qualified with a clue like “a step-by-step tutorial” rather than “click here”, to allow some people to skip it. What does a link mean to different people? If used properly, even people jumping in on a page from outside the present context would be able to decide whether to explore the link or not. Among common problems are documents

overcrowded with links, and broken links, i.e. links pointing to another destination than intended or to a non-existing destination.

Finally, images should not be forgotten. The phrase “an image says more than a thousand words” is one reason why the web is filled with illustrations. Another reason is that images provide interactive experiences for the user, either as static images or as clickable image maps. The <IMG> tag has an optional ALT attribute that should hold a short textual description of the image. Until recently, this property was needed and widely used since many users instructed their browsers to not show images due to low transfer rates, and authors therefore embedded descriptions in order to allow the users to consider explicitly requesting for the image. Despite increased transfer rates, the ALT-attribute is still widely used today since most browsers show its description whenever the mouse passes over the image. Moreover, images are useful to spice up the content, or for summarising important and difficult subjects in a document. The most commonly used graphical format is jpg- and gif-images.

From the perspective of a the browser software, its task is to interpret the tags in the HTML document in order to present a formatted page to the user. It is our belief that the adaptive hypertext system can make use of the very same tags in order to choose the elements that are important to conceptualise, as illustrated in Figure 5–3. The task to follow concerns how to extract as much information as possible in order to lay the grounds for ensuring quality to the selection of final concepts.



```
<HTML>
<HEAD<TITLE> ~~ </TITLE><META> ~~ ~~ </META></HEAD>

<BODY>
                        <H1> ~~ </H1>
<P> ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ </P>


<H2> ~~ </H2>

<P> ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~
~~ ~~ ~~ ~~ ~~ <A HREF="~~"> ~~</A> ~~ ~~ ~~ ~~ ~~ ~~ ~~ </P>
<IMG SRC="~~ ">

<H2> ~~ </H2>
<UL>
  <LI> ~~ ~~ ~~
  <LI> ~~ ~~ ~~ ~~ ~~
  <LI> ~~
</UL>

<P> ~~ ~~ ~~ <B>~~</B> ~~ ~
~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~</P>

<TABLE>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~
</TABLE>

</BODY>
</HTML>
```

Figure 5–3: Some tags indicate which elements are important to conceptualise, while others assist in the process of conceptualization.

5.1.6 Using IR-techniques within important elements

After identifying important features of the elements above, we need methods to further analyse them in order to extract possible concepts. In the field of Information Retrieval (IR), the main goal deals with automated classification and retrieval of unstructured documents [Frakes+92]. IR-techniques such as *lexical analysis*, *stoplists* and *stemming* provide a way to identify the words of each document with high discrimination values, thus forming the basis of building an index. When a query[1] is being executed the most relevant document(s) matching the query can be retrieved.

An IR system seems useless as an adaptive hypertext system since it provides little or no *understanding* of document quality [Berners-Lee96]. The basic IR-techniques, however, may assist in the adaptive hypertext system, since what is needed in the conceptualization phase is a proposal of important keywords. That is the concept candidates are partly a result of a statistical IR-analysis, whereas the selection of final concepts is made by a more intelligently reasoning upon the candidates.

Lexical analysis is the process of transforming a stream of characters into a list of lower case words or tokens, by removing any character that is not a letter nor a digit. This implies that an HTML-document passing through lexical analysis would loose its brackets around the tags, as the brackets are non-letter characters. Clearly the adaptive system would fail if this important information were lost. Performing the IR-analysis on the text embraced with interesting elements only (not the entire HTML-document) solves this problem.

Frequent occurring words[2] from the spoken language have no indexing value. The use of a stoplist may reduce the number of words since stopwords typically account for 20-30 percent of the tokens in an average document. In the stoplist process, every word from the text is checked against the stoplist and eliminated if found there, thus the survivors are more likely to be of importance.

Stemming is the automatic fusion of term variants, so that after stemming the terms “concerned”, “concerning” and “concerns” all get conflated into the stem “concern”. Although stemming may reduce the size of an index by as much as 50%, some information about the terms going through stemming is clearly lost. The Porter-algorithm is a well-known and fairly small implementation of stemming [Frakes+92].

For instance, we performed a test (using the basic IR-techniques) on an element from a randomly picked HTML-document. The results showed that after being

1. If the user wants to find information on the Internet about a subject, he/she needs to formulate a (boolean) query consisting of key terms.
2. Van Rijsbergen have developed a stoplist of 250 words that is widely used in IR-analysis [Frakes+92]. Common words occurring in many documents may be for example “time”, “any”, “all”, “into”, “very”, “asking”, “where” etc.

exposed to lexical analysis and stoplist only 133 out of 290 terms remained (i.e. 46% of the total number). Removing duplicates *before* stemming lead to a resulting 96 terms (i.e. 33% of the total number), while removing duplicates *after* stemming reduced the text to 87 terms (i.e. 30% of the total number). Even though about 70% compression was obtained after IR-analysis, the test indicated that these procedures alone were not enough to fully extract a few concepts describing the element. The problem is that one paragraph alone is likely to produce far to many potential competing concepts. A more exhaustive approach is needed.

5.1.7 Domain specific list

A concept of an element could be represented by a vector of all the terms in the element accounting for all the terms in the domain. The conceptual space would then consist of many different vectors so that similar elements would have quite similar vector representations. The alternative approach implemented in this research uses human knowledge to ensure quality with respect to the conceptualization. If the author of a domain wants to prepare a collection of documents for the adaptive system, one might require that the first thing to do is constructing a domain-specific list with keywords regarded as being indicators of concepts or important paragraphs of text. Using such a domain specific list[1] or DSL actively when analysing each document, would help the system to identify possible concepts and important elements and later help in building relations among the concepts proposed. As an example, say that the term “star” is listed in the domain specific list[2] and also found in an element. Then the system could select the word “star” as one strong concept candidate for that element or for the document as a whole. Note that constructing a DSL needs only to be done once for each domain.

The performance of the adaptive hypertext system is related to both the quality and complexity of the DSL, since the idea is for the system to actively be using the list in the process of conceptualising the documents. Obviously the content and size of the DSL would influence the selection of concepts. A thoroughly considered list would therefore increase performance, and, from the view of the end user, make the system act more intelligently. Furthermore, using not only one, but several different lists should add flexibility to the system. Remember the DSL is not only of value before extracting information, but also when monitoring the user and adapting documents on the fly. The influence of several lists is best illustrated in an educational setting where the level of knowledge may vary significant among students due to a variation in individual skills and related knowledge obtained from prior courses. If the students were allowed to modify their own DSL and the system could use lists from related courses, the goal of a presentation tailored to each user seems much closer [3].

1. The domain specific list can be thought of as an “anti-stop list”, that is a list acting directly opposite to the IR stoplist method.
2. Allowing for more than one DSL yields a flexible facility to the adaptive hypertext system
3. This is necessary in the case of documents being analysed on the fly. In our approach, we intend to perform the analysis in advance of the user interaction.

Making models explicit to the system adds both flexibility and system performance with respect to intelligence. The system KN-AHS does not integrate the user-modelling component BGP-MS into the application. This leads to adaptivity in the following senses: many types of information about the user can be represented simultaneously, the user-modelling component can receive and answer questions, and accumulation of knowledge takes place more naturally [Kobsa+94]. Embedding the domain specific list in the domain model should help during the analysis of documents. Using the DSL therefore yields a simple technique that strengthens the strategy of conceptualising a document based on the contents of its elements.

5.1.8 Other ways to extract information

Term frequencies (TF) can be used in order to extract concepts [Kietz+00], based on the assumption that terms that occur often, with the exception of stopwords, are of importance. After counting the TF, the resulting information can be sorted so that the most frequent occurring term (TF_{max}), or all terms with a higher TF than a given threshold ($TF_{treshold}$) can be listed.

So far we have found that a domain consists of different documents linked together, each of which has different elements, and we know some simple techniques that can be used to gain statistical information of the text. Keeping this in mind, the work continues on the development of a strategy to guide the conceptualization.

5.2 A domain model in the horizon

It seems convenient to represent the domain knowledge in terms of concepts and relations among the concepts. Before we explore how to build a domain model in the next section, we formalise the notation of the information as provided by the techniques introduced in the previous, and develop Heuristical rules in order to select the most promising concepts and relations. The rules are the backbone of the construction of the model which in turn is quite critical concerning the system's ability to perform adaptations. The following discussion is therefore essential to our work.

5.2.1 The sets of candidate concepts

From the discussion in section 5.1.5 "Importance of the elements", page 37, we see that those elements classified as "important" serve various purposes for both the user and the author. Due to this variety, the system needs to take action according to rules in order to extract concepts. These rules should account for as many aspects as possible, both the global ones concerning the document as a whole, as well as the local ones, like document subsections. Keeping the rules in a rule base yields both power and flexibility to the process of analysing HTML-documents.

New rules may be easily added to the base without affecting other modules of the system. An appealing strategy is to parse each document looking for important elements, extract the information within, and finally, in the search for concepts, analyse it according to rules.

Each method **m** used on an element **i**, produces a set $\mathbf{S}_{\mathrm{im}}$ consisting of **n** candidate concept terms $\mathbf{c}_{\mathrm{j}}$ when run on an important element, i.e. $\mathbf{S}_{\mathrm{im}} = \{\mathbf{c}_1, \mathbf{c}_{2,} \, _{\ldots,} \, \mathbf{c}_{\mathrm{n}}\}$. Therefore we refer to the techniques as information sources from now on. A brief summary of the functionality of each information source is listed in Table 5–3.

Table 5–3: Functionalities of the information sources

Method / Abbreviation	Information source	Functionality
LA	Lexical analysis	Converts the stream of characters in an element to a list of terms, where stopwords are removed
DSL	Domain specific list	Identifies terms from the element that match terms in a domain specific list
TF	Term frequency	Counts number of occurrences for each term in an element
Emp	Emphasizer identification	Lists terms in emphasised elements that occur within an element

Each method produces different sets of terms that all can be more or less suitable as a concept describing the element. In order to visualise, regard the following rather simple element written in HTML:

- `<P>Our solar system is only one of millions of other solar systems. It consists of nine <B>planets</B>, of which the <A HREF=”tellus.html”>earth</A> is the only one with developed life </P>`

Assume a domain specific list (DSL) holding the three terms “Pluto”, “Mars” and “Earth” and a term frequency threshold set to two terms. The list below illustrates the different sets of terms originating from the different information sources. Notice that the terms from the element are stemmed, as are those in the DSL. Finally, since the element is surrounded by a paragraph-tag, we label the sets with a leading “P”. The meaning of the first item in the list is that *the set S of terms from element P when exposed to Lexical Analysis, are “solar”, “system” etc.*

- $S_{P\ LA}$ = {solar, system, million, consist, nine, planet, tellu, earth, develop, life}
- $S_{P\ DSL}$ = {earth}
- $S_{P\ TF}$ = {solar, system}
- $S_{P\ Emp}$ = {planet}

Note that with this notation, it is easy to picture different sets and how their members influence each other.

5.2.2 Values separate the candidates

For an element, the task for the adaptive system is to choose the best concept from the set of all information sources, that is the concept must be chosen from $\mathbf{C}_i = \sum_m \cup \mathbf{S}_{im}$ Notice that the set notion is also true for the whole document, i.e $\mathbf{C}_{DOC}$ is a valid set just as $\mathbf{C}_{TABLE}$, $\mathbf{C}_{IMG}$ and $\mathbf{C}_P$ are valid sets. Apart from *image* and *link* elements, all elements that go through lexical analysis (and stopword removal, both denoted by LA) have a potentially large set $\mathbf{S}_{i\ LA}$. The more the number of competing terms, the more difficult the process of ensuring quality to the final selection. Inspired by Sharma, who outlines the power of a strong Heuristics model for developing a user model in the absence of well known methods [Sharma01], we apply a straightforward approach to the conceptualization process of the document and/or each element by *adding values* to each candidate from $\mathbf{C}_{element}$ based on Heuristical rules. The most important terms can be distinguished from the less important ones simply by judging the final numeric values summed up from all Heuristics, so after all Heuristics have done their job, a set of candidates $\mathbf{c}_i$ ordered by value is the result, where the value of $\mathbf{c}_1 \geq \mathbf{c}_2 \geq \mathbf{c}_3$ etc. To exemplify, say the system produced the candidates “sun”, “rain”, “day” and “wind” for an element, where the candidates had values of 14, 12, 9 and 3, respectively. Then the matter of selecting the concept is simply finding the head of the ordered list, which means that “sun” becomes the concept. The tail of the original list hosts all the candidates that were not selected.

Let us stress the definition of the word *value*. A value is associated with each Heuristic. The terms take on these values as the Heuristics fire, so after the process of analysing an element, all the element terms have various scores. In other words, the *score* of a term is the total of all values added to the term. As an example, say that the following values are associated with three of the Heuristics:

- HeuristicA: positive value of 3
- HeuristicB: positive value of 8
- HeuristicC: negative value of -4

Table 5–4 illustrates the difference between the values of the Heuristics and the total score. Note that the values are fixed, and are used to add up the score of a term. From now on, we refer to the present score of a term as its present value.

Table 5–4: The values as assigned by the Heuristics and the total scores held by the terms

Terms	HeuristicA	HeuristicB	HeuristicC	Total score
agent	Yes	No	No	3
user	No	Yes	Yes	4
model	Yes	Yes	No	11

The following discussion on Heuristics debates how to choose the correct level of abstraction and explores how to guide the process of adding appropriate values to the candidate concepts.

5.2.3 Their fate is in the hands of Heuristics

Three groups of Heuristics were identified. The Heuristics for important elements are listed in Table 5–5 followed by those that focus on aspects for the document as a whole in Table 5–7. An interesting discussion concerning the roles of the elements in total leads to the Heuristics described in Table 5–8.

Consider a fictional piece of text explaining user models with a section debating *user model acquisition*. Within this text the formatted string (written in HTML) `<B>Explicit</B> models gather information by prompting the user` contains the emphasised word “explicit”. The author would prefer the term “acquisition” as the proposed concept, however, so the system needs to account for more than the “emphasizer” Heuristic. Furthermore, when creating the DSL, the author might browse quickly through the pages looking for keywords from headings and emphasizers as a help to create the domain specific list. Higher values should therefore be added to concepts suggested by $\mathbf{S}_{i\ DSL}$ than those in other sets, as stated in the first group of Heuristics below.

Table 5–5: Some Heuristics used for important elements

HC-1 Terms stated explicitly by some domain expert are very likely to be among the most useful concepts, a fact that suggests assigning high values to members of $\mathbf{S}_{i\ DSL}$
HC-2 The more a term occurs, the more important it is. The set $\mathbf{S}_{i\ TF}$ results from counting term frequencies in $\mathbf{S}_{i\ LA}$. Assigning each **c** with values relative to these frequencies therefore seems promising.
HC-3 Members from $\mathbf{S}_{i\ Emp}$ can provide quality for the system in the selection process. Even though the author decides selectively which words to emphasise, emphasizers are not always used properly, and therefore the value should not be too dominant.

Interestingly, $\mathbf{S}_{i\ TF}$ might be quite similar for many documents, if the frequently occurring terms of one document also has a frequent appearance in others. In order to reduce this problem, terms that occur often in many documents could be punished by using a collection frequency and normalising the set [Kietz+00]. Additionally, through the use of stemming the figures are further incremented since terms with the same stem count twice.

Notice the complementary nature of the sets as illustrated in Table 5–6. When their elements coincide, the respective values summed up thus far are incremented or decremented according to Heuristics. The total value, or score of a concept **c** in the set $\mathbf{C}_{element}$ is therefore the sum of the values calculated from one or more

Heuristical rules, so e.g. if $\mathbf{S}_{i\ LA} = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6, \mathbf{c}_7, \mathbf{c}_8\}$, the score of $\mathbf{c}_5$ is the sum of values defined by Heuristics HC-1 and HC-3.

Table 5–6: The complementary role of the Heuristics

Set name	Set content	Associated Heuristic
$\mathbf{S}_{i\ DSL}$	$\{\mathbf{c}_2, \mathbf{c}_5\}$	HC-1
$\mathbf{S}_{i\ TF}$	$\{\mathbf{c}_1, \mathbf{c}_4, \mathbf{c}_6, \mathbf{c}_7, \mathbf{c}_8\}$	HC-2
$\mathbf{S}_{i\ Emp}$	$\{\mathbf{c}_3, \mathbf{c}_5, \mathbf{c}_6\}$	HC-3

The second group of Heuristics focus on aspects for the document as a whole. Since the document can be considered as an element containing children elements (subordinate elements), the information sources (DSL, TF, LA) are valuable and the Heuristics from Table 5–5 can be applied. In addition, the ones listed in Table 5–7 are important contributors to the quality of the Conceptualization of a document.

Table 5–7: Some Heuristics used for the entire document

HC-4 Meta information and the title element ensure quality to the selection since their content most likely are considered thoroughly from the author’s side. Relatively high values can therefore be applied in such cases.
HC-5 A value that reflects the characteristics of the element in which a term lives, should be added to the score. The type of element where a candidate concept occurs matters in the selection process, that is some elements are more important than others.
HC-6 If a document **d** is pointed to from another document in the domain, the description of the link element in **d** should be assigned a very high value, based on the assumption that hyper references are of value for the conceptualization of future documents.

Despite the potential quality provided through meta information and the title element, these sources could in the worst case be equal for every document in a domain, and often is, as noted in the discussion in section 5.1.5 “Importance of the elements”, page 37, thus misleading the system to choose the same concept for every document. The solution is straightforward and involves comparing meta information in every document with that information found this far. If two documents have the same meta information, then Heuristic HC-4 is cancelled for these documents.

Two very interesting features are implicitly contained in Heuristic HC-5. First, if a candidate concept recurs in several different elements within a document, it is likely to be an important keyword for the entire text. Due to the different roles of the elements, their respective candidates should not count equally in the process of adding up values. In particular, headings at the highest level (normally <H1>) should have higher priority than lower level headings, tables and lists. The second feature e.g. reveals the dual role of external hyper references, i.e. links to other documents. As noted in section 5.1.5, the text embraced by the link tag <A HREF> is often (or at least should be) qualified with a clue indicating what it is pointing to, rather than the often used printing block “click here”. Hence, one of the terms *in the clue* is very likely to be a concept (or at least a very strong candidate concept) of the document *pointed to* by the link. The same term is simultaneously destructive as a candidate for the document at hand[1]. In other words, candidates from hyper reference elements should not be accumulated to the set $\mathbf{C}_{doc}$ when searching for a document concept, but should be saved for future use, that is only candidates from hyper references *pointing to* the present document should be accumulated to $\mathbf{C}_{doc}$. Heuristic HC-6 therefore has the implicit assumption that all links in the domain are known in advance of the conceptualization of documents, are represented somewhere (e.g. a matrix or a file) and are used throughout the analysis.

As an example of how Heuristic HC-6 works, assume that the hyper reference `<A HREF=”file3.html”>user</A>` appears in “file1.html”. Then the term `user` is very likely to be the concept of the document “file3.html”, but it should not be the concept of “file1.html”. A high value is added to the score of occurrences of `user` in “file3.html”, and a low value to occurrences of `user` in “file1.html”.

It is interesting to question which rules should fire, that is, are all sets of equal relevance, or should some be omitted under certain conditions? Do the different element types influence the final outcome? Obviously there is a possible conflict between the conceptualization of a document and its elements. By treating the document as one big element enclosed by the <HTML> and </HTML> tags, we see that all sets $\mathbf{C}_{element}$ from a document is contained in $\mathbf{C}_{doc}$ but the reverse does not hold, as illustrated in Figure 5–4. The candidate concepts appear as black dots.

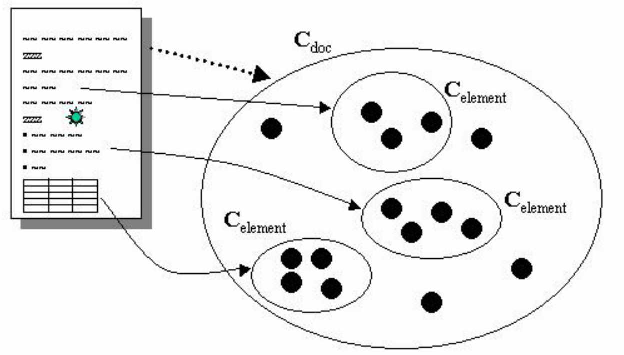


Figure 5–4: Candidate concepts of a document belong to different sets

Why is this observation so important? Remember the purpose of the concepts - they should act as descriptors of the content of each particular knowledge source in the domain, providing a basis for building the domain model. Furthermore the

1. Two documents should not have the same concept according to the requirements of the domain model, but as will be explained in Heuristic HC-7, different *sections* of a document might very well have the same concept due to their complementary roles.

division into document concepts and element concepts was based on the need to acquire knowledge sources at different levels in the adaptation phase. In other words, element concepts should differ from document concepts in order to prevent overlapping information, thus ensuring a better domain model. This conflict is not unique. What happens if the same conceptual description is found for elements of distinct types? Heuristics that solve such problems are listed in Table 5–8 below. Note that most of these Heuristics don't hand out values as did the previous ones (except Heuristic HC-8).

Table 5–8: Some Heuristics used for the ensemble of elements in a document

HC-7 In case the same concept is proposed for two distinct element types, the system can perform more powerful adaptations based on user preferences. Presentational form matters in the adaptive phase, so information on the element types should be preserved.
HC-8 Candidates that equal the document concept found should not be considered when searching for concepts in important elements. In order to prevent their influence, such candidates could be assigned large negative values.
HC-9 The level of abstraction can vary for different documents. Headings indicate the start of new sections and are likely to describe the content that follows.
HC-10 Lists, images and tables should always be considered as single units since they provide variation in the document and often act as complementary sources of information for the surrounding text.
HC-11 Since some elements might be superfluous, an analysis of the document structure should precede that of the elements.

Brusilovsky points out that an adaptive system can choose different types of media with which to present information to the user, according to what is most relevant at the given node (i.e. document) [Brusilovsky01]. Accounting for the fact that different elements act as complementary sources of information for the user provides a means for the system to add variety to the presentation based on context and user preferences, so Heuristic HC-7 suggests that similar conceptual descriptions should be accepted only when the element types differ. Note that this applies for whole documents as well (remember the document-as-HTML-element view), so two documents should not be assigned the same concept. When it comes to the future adaptive generations, the system would be better off if it for elements that equal in terms of both type and proposed conceptual description, tries with another concept. Similarly, candidates that equal the document concept found should not be considered when searching for concepts in the important elements, since they provide no new information to the system. Heuristic HC-8 therefore implies that the search for document concepts should precede that of element concepts, a statement consistent with its fellow Heuristic HC-11.

Heuristic HC-9 claims that the level of abstraction may vary. Moreover, headings should not be regarded as stand-alone elements, as they indicate the start of a new section in the document and are likely to reflect its content well. The scope of a header includes all elements from below it to the next heading at the same level. The problem of choosing the correct granularity concerns which information entities should be regarded as atomic. In the eyes of a natural language expert, the task is to mine text from each sentence. For the adaptive hypertext system, the attempt is to represent the knowledge of a domain in terms of a conceptual domain model: as indicated so far, each document and the different elements are subject to this conceptualization. Some documents are larger than others[1], but if well structured, e.g. assuming a proper use of headings cf. section 5.1.5, the system could try to bring the process of conceptualization to a more fine-grained level. In particular, an examination of which header tags are used should reveal how many levels of the concept hierarchy a single document covers.

Heuristic HC-10 outlines the importance for images, lists and tables to be considered as single units. For instance, in order to conceptualise *list* elements (<OL> or <UL>), it is important to understand how they are used in the document. First, a list can be considered as one entity which implies finding one good candidate concept is desirable. Secondly, an analysis of the content of each <LI> element in a list is somewhat more complex, but provides a means to identify relations between global list concepts and sub concepts within the list. Obviously, if the document only has one element, namely one big list, important information would get lost if the individual <LI> elements were never subject to further analysis. On the other hand, the resulting document representation could get far too complex if all list elements in addition to other elements were analysed. The important and difficult task here is to choose the appropriate level according to the overall document structure. Similar considerations apply for table elements. Finally, due to their value for the user and interactivity, images would probably denote important concepts either for the document or for elements depending of where they occur. The problem of how to conceptualise an image, can be solved by looking in the `ALT` attribute or using the filename of the image.

A summary of the above discussion is exemplified in Figure 5–5. The conceptual descriptions $\mathbf{k}_i$ are document concepts (squares) and element concepts (circles) representing the document and elements of the illustrated types - images, tables, text and lists. Agreeing with Heuristic HC-7, we see that $\mathbf{k}_2$ represents both an image and plain text. The same apply between documents: The system might very

1. An easy measure for size is file size (remember images are imported by reference in HTML), another more demanding measure is by counting words.

well propose the same conceptual description $\mathbf{k}_4$ for the image element of one document as for the table element of another.

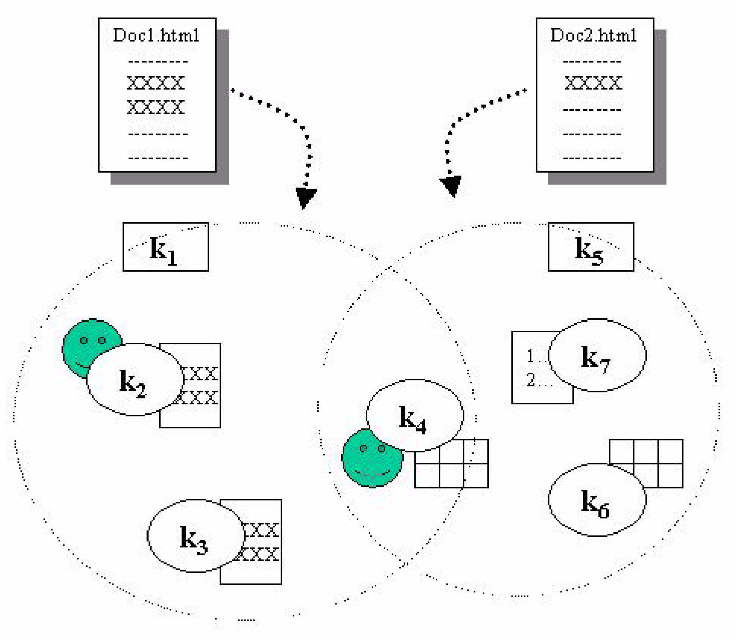


Figure 5–5: Conceptual representation of two documents. Note that an element from the first and one from the second document have been assigned the same concept.

5.2.4 Ensuring flexibility

In order to control the impact from the combination of the different Heuristics, the respective values can be kept in a file and hence adjusted by the domain expert, introducing a means to experiment empirically on the relative importance of the rules. Making a value zero means preventing its rule from influencing the total score of a term. An high negative value would mean that rule to block a candidate from consideration, which would be desirable in cases like Heuristics HC-5 and HC-8. Using ideas from the theory of neural networks (NN), in which values (called weights in NN terminology) are adjusted over time by the system to change performance and knowledge, the adaptive hypertext system could even find means for adjusting the values itself e.g. by tracking the domain expert’s rejection of concepts.

Due to the possibly erroneous outcome of the entire conceptualization process and the future manual adjustment, it is important that the system retains all the information found about every element. If a domain expert is dissatisfied with the suggestions, the system can simply propose new promising concepts selected, and it could even explain why it chose particular concepts by logging and displaying the rules fired and their associated values. If the expert keeps on rejecting the suggestions of the system, it could also, for each specific type of element, track which information sources the expert seems to prefer, and further identify patterns in order to modify or develop new heuristical rules, or the system could learn new rules stated explicitly by the expert. This modification is possible only if the rules are kept external to the system and if some sort of rule inference engine exists. In time and throughout interaction, such a system would possibly improve its performance. Hence the adaptation could be taken further to other levels[1], but such issues are outside the scope of this research.

In the above discussion we tried to choose a document concept and some element concepts for each document. The selection tried to capture the best among several candidates by adding values guided by Heuristics. Regardless of the outcome, the result is a set of concepts describing the content or *knowledge* of the original document and its parts. In other words, $\mathbf{K}_{doc} = \{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_m\}$ where the selected concepts $\mathbf{k}_i \in \mathbf{C}_{doc} \cup \mathbf{C}_{element}$. The next task is to relate the concepts.

5.2.5 Relationship types

In this section there are two aims. First, we want to find a set **R** of directed relations **r** of various types between the concepts in order to extend the knowledge of the domain model, so $\mathbf{R}_{domain} = \sum_i \cup \mathbf{r}_i$ where $\mathbf{r}_i = (\mathbf{k}_o, \mathbf{k}_p, \textit{relationship\_type})$. This expression should be read "concept $\mathbf{k}_o$ is 'relationship_type' to/by concept $\mathbf{k}_p$". Secondly, we want to discuss the possibilities for automatically extracting these relations, ensuring quality to the process.

Along with all selected concepts, there is a set of candidates **L** (ordered by value) that were not selected, i.e. $\forall\ \mathbf{k}_i\ \exists\ \mathbf{L}_{\mathbf{k}i} = \{\mathbf{c}_a, \mathbf{c}_b, \mathbf{c}_c...\}$, where the value of $\mathbf{c}_a \geq \mathbf{c}_b \geq \mathbf{c}_c$ etc. Repeating one of the examples used in the previous section, say the system produced the candidates "sun", "rain", "day" and "wind" for an element, where the candidates had values of 14, 12, 9 and 3, respectively. Then the head of the ordered list is the concept selected, in other words "sun" becomes the concept of the knowledge entity. The tail of the original list hosts all the candidates that were not selected.

For stand-alone individual documents a representation of the knowledge in terms of conceptual names may seem sufficient. However, for a collection of q documents we reveal that in $\mathbf{K}_{domain} = \mathbf{K}_{doc\_1} \cup \mathbf{K}_{doc\_2} \cup \mathbf{...} \cup \mathbf{K}_{doc\_q}$, there obviously are identical items and relations among the elements, reflected through similar concepts and implicit conceptual relations. If we make a peek into the adaptive phase of the interaction, the gap of knowledge in the user model can be bridged by means of selecting appropriate concepts from the domain model and present corresponding knowledge sources to the user. Without representing some sort of connections or relations between the concepts (i.e. the nodes) in the domain model, it is difficult to embody the process of selection in a thoroughly considered plan for a sequence of adaptations.

Many types of relations can be identified in order to extend the domain model and the adaptive performance of the system. In their AHAM system, Wu et. al use the

1. As noted in chapter 4 "Adaptivity", page 19, a system that takes initiative to, proposes, selects and executes the adaptation, is called self-adaptive [Malinowski+92].

prerequisite, inhibitor and part-of relations [Wu+01], which illustrate conceptual dependencies in terms of adaptation. Before exploring the characteristics of conceptual hierarchies and prerequisites, we introduce the discussion searching for concepts that act as deeper explanations for others. Again we rely on the use of Heuristics, this time for the *identification* of the different relationship types.

The system should be able to provide the user with many knowledge sources in order to broaden the understanding of a subject. For the user, the most conspicuous relationship type is clickable hyper references (links) provided by `<A HREF>` in HTML, as they relate either documents, elements, or a combination. What does such explicitly stated relations tell the user? Why does the user click on links after all? Is there a reason for the author to embed hyper references in a document?

Heuristic HR-1 states that identifying links provides a means for the system to find deeper explanations of a concept. First, links invite the user to move around in hyperspace. Second, and of importance for deducing this Heuristic, they are designed by the author and likely to point to information of relevance for the present context. That is, the author invites the user to find more information by following the links. Therefore, linked knowledge sources should be represented as relations in the domain model through connecting the corresponding concepts with the *has_deeper_explanation* relationship type, directed from the source to the destination, so the set $\mathbf{R}_{\text{has\_deeper\_explanation}} = \sum_i \cup \mathbf{r}_i$, $\mathbf{r}_i = (\mathbf{k}_o, \mathbf{k}_p,$ *has_deeper_explanation*$)$. Most likely, such relations would occur between element concepts and document concepts, provided that most links exist within important sections and point to documents. In particular, a link referring to somewhere specific in another document (c.f. `<A HREF="url#location">` in Table 5–2), indicates a conceptual relation between the element concept hosting the link and the element it refers to.

Table 5–9: Heuristics for finding deeper explanations

HR-1 Hyper references between two knowledge sources somewhere within the domain are also relations of type *has_deeper_explanation* between the corresponding concepts.
HR-2 External links with destinations outside of the domain are slightly different from internal ones, and the corresponding relations between such knowledge sources should be labelled as *external*.

Let us exemplify HR-1. The relation should be directed from the source of the link to the destination. Assume that the link `<A HREF="agentsmore.html">learn more about agents</A>` has a source element whose concept is found to be the term `agent`, and that the document "agentsmore.html" is conceptualised as `reactiv`. Then the relation to be set up is:

- **r** = (**agent**, **reactiv**, *has_deeper_explanation*).

Instead of duplicating information that already exists elsewhere (outside the domain), the author may choose to set up external links. Furthermore, as pointed out by Heuristic HR-2, the external links are of less importance than the internal ones discussed above, as their destinations are more likely to go beyond the scope of the domain. Still, such links should broaden the user's understanding, hence the corresponding relations should be labelled *external*, so that $\mathbf{R}_{\text{external}} = \sum_i \cup \mathbf{r}_i$ where $\mathbf{r}_i = (\mathbf{k}_o, \mathbf{k}_p, \textit{external})$. Notice that due to Heuristic HC-6 (page 47), the adaptive hypertext system has a means to identify both relations and concepts in one turn, thus strengthening the participatory role of hyper reference elements in the process of building the domain model. From this we can advantage when searching for a label for the external concept, namely by conceptualising the <A HREF> element and let the winner candidate describe the external concept that the link points to.

So much for explicitly stated references. A document consists of subordinate elements subject to conceptualization, and their contents are likely to be somehow related since they appear in the same document. This kind of implicit hierarchy is not constrained within the limits of single documents, but apply between documents and possibly even between domains, as illustrated in Figure 5–6.

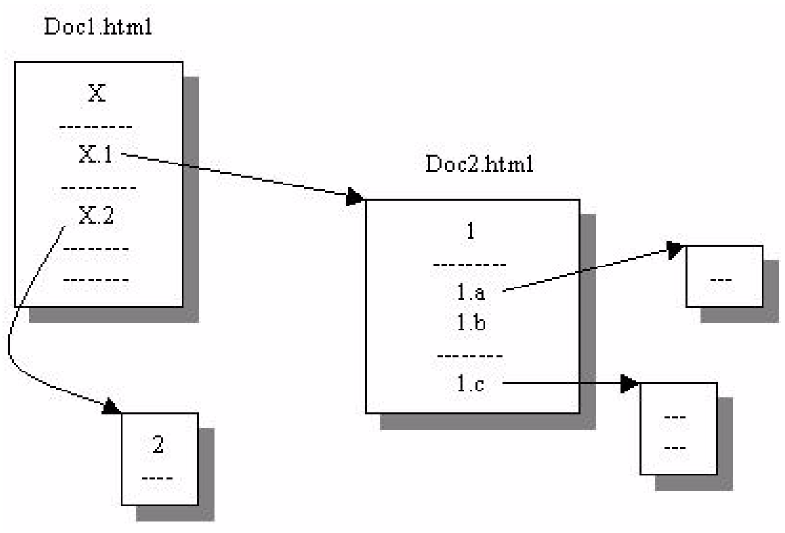


Figure 5–6: The two sections (X.1 and X.2) of the first document are related since they are both in the context of the more general document subject (X). The document hierarchy shows how section X.1 is superior to the entire second document, as illustrated by the directed arrows.

Heuristics for the conceptual hierarchy guide the search for the sets of *parent* and *synonym* relationship types, denoted $\mathbf{R}_{\text{parent}}$ and $\mathbf{R}_{\text{synonym}}$ respectively.

Table 5–10: Heuristics for the conceptual hierarchy

HR-3 All element concepts found within a document are children of the document concept.
HR-4 There is a *parent* relation if a member from a set of candidate concepts equals an already found concept.
HR-5 If two concepts have some joint members from their set of candidates, the concepts are synonymous.

In agreement with the assumption that the concept abstracted from the document has a context superior to the concepts at the element level, Heuristic HR-3 suggests relating the document concept and the corresponding element concepts through the *parent* relationship type, instead of relating all the element concepts found in a document with each other. In other words, such elements are implicitly related through their cohabitation in the same original document, a structure which should be kept intact in the final domain model. Note that due to concept similarity, implicit parental relations between different documents or elements can be caught simply by comparing the conceptual descriptions. For instance, a document conceptualised to `agents` with four elements conceptualised to `reactiv`, `interfac`, `autonom` and `dictionary`, respectively, leads to the relations

- **r** = (**agent**, **reactiv**, *parent*)
- **r** = (**agent**, **interfac**, *parent*)
- **r** = (**agent**, **autonom**, *parent*)
- **r** = (**agent**, **dictionary**, *parent*)

Figure 5–7 illustrates the merging of the conceptual representations of the two documents from Figure 5–6. Note that one element concept of the first document equals the document concept of the second, namely $\mathbf{k}_3$. The *parent* relations correctly capture the hierarchy "$\mathbf{k}_6$ is secondary to $\mathbf{k}_1$" through the relational sequence $\mathbf{r}_2$ to $\mathbf{r}_5$. The system must somehow record that $\mathbf{k}_3$ now both represents a document ("Doc2.html") and an element ("X.2"). Since the document is an element of type `<HTML>`, the same method apply for the situation illustrated in Figure 5–5 (page 51). The implementation specific details are discussed in the next chapter.

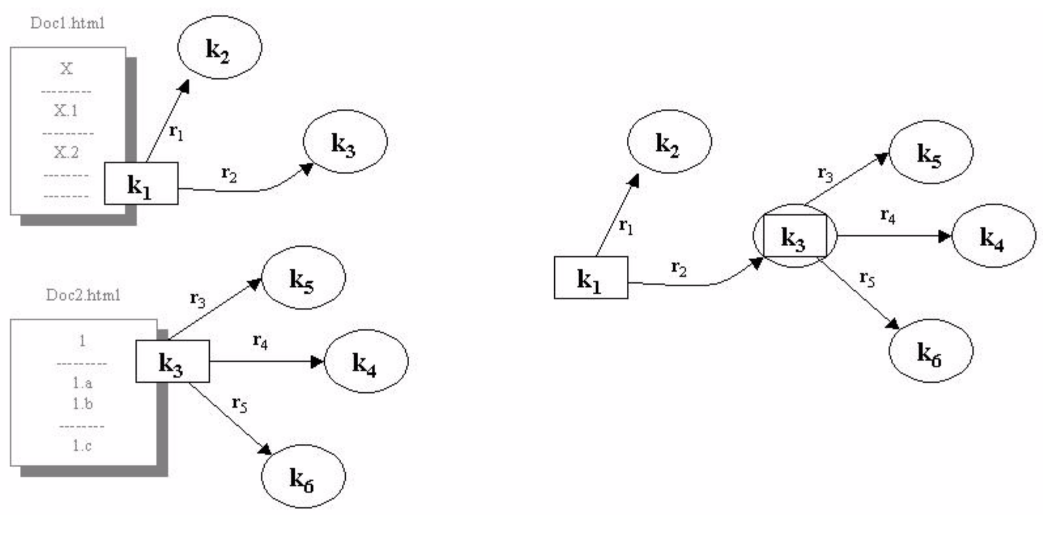


Figure 5–7: The relations are of type parent, correctly capturing the concept hierarchy

Now let's turn to the last two Heuristics from the set of hierarchy. As far as the loser candidate concepts concern, their lives should not be ended despite their loss in the fight for presidency in the selection process. Given new importance by Heuristic HR-4, they can finally rejoice and contribute in the process of identifying parent relations. If there, for two concepts $\mathbf{k}_1$ and $\mathbf{k}_2$, exists a candidate $\mathbf{c}'$ in the loser set $\mathbf{L}_{\mathbf{k}1}$ with the same conceptual description as $\mathbf{k}_2$, then the context of $\mathbf{k}_1$ is likely to be superior to that of $\mathbf{k}_2$. In other words, if a knowledge source has the concept `softwar` and that one of its loser candidates `user` is the concept of another knowledge source, then the `softwar` concept is parent to `user`. More formally, if $\exists\ \mathbf{c}' \in \mathbf{L}_{\mathbf{k}1}$, $\mathbf{c}' = \mathbf{k}_2$, then $\mathbf{r} = (\mathbf{k}_1, \mathbf{k}_2, \textit{parent})$. Additionally, if the reverse also hold at the same time (i.e. $\exists\ \mathbf{c}' \in \mathbf{L}_{\mathbf{k}2}$, $\mathbf{c}' = \mathbf{k}_1$) then the relationship is not of parenthood, but rather of a more synonymous nature. Heuristic HR-5 therefore claims that two concepts are synonymous if their **L** sets have some common denominators. More generally, note that whenever the **L** sets of two concepts share candidates so that $\mathbf{L}_{\mathbf{k}i} \cap \mathbf{L}_{\mathbf{k}j} \neq \{\varnothing\}$, the synonym relation is present between concepts $\mathbf{k}_i$ and $\mathbf{k}_j$. Finally, due to the values associated with each candidate, their joint sum provides a means to determine the strength of the relation. E.g, a threshold might be used so that relations weaker than the threshold can be turned down, hence controlling the size of the sets $\mathbf{R}_{\text{parent}}$ and $\mathbf{R}_{\text{synonym}}$. Figure 5–8 illustrates the role of the candidate sets.

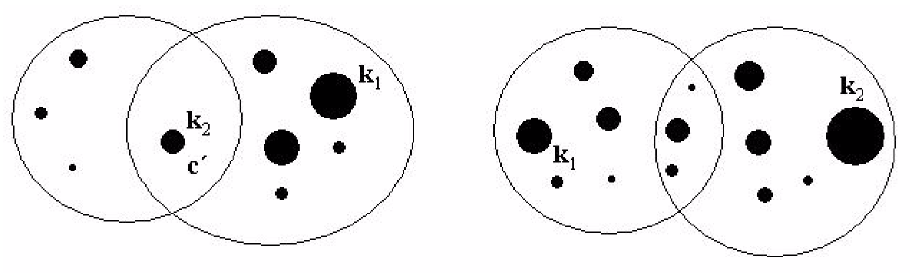


Figure 5–8: Heuristic HR-4 is illustrated to the left and HR-5 to the right. Different values of the candidate concepts are reflected by the various sizes of the black dots.

An interesting question is whether other relationship types can be found or should be embedded in the domain model. Obviously, some concepts are more difficult to understand than others. Does the presentational sequence matter during user interaction? Are some concepts redundant for some users, but highly necessary for others? How can we improve and facilitate intelligent reasoning in the adaptation process? The *prerequisite* relationship type adds knowledge to the conceptual hierarchy as it rely on a deeper understanding of the semantics of the concepts. If concept $\mathbf{k}_1$ is a prerequisite for concept $\mathbf{k}_2$ it means that $\mathbf{k}_1$ should be presented to the user before $\mathbf{k}_2$, whereas if $\mathbf{k}_1$ inhibits $\mathbf{k}_2$ the latter is no longer desirable in a presentation once the first is known. Note that it can be possible to infer the inhibitor relation from a sequence of prerequisites, though not necessarily: if $\mathbf{k}_1$ is a prerequisite for $\mathbf{k}_2$ which in turn is a prerequisite for $\mathbf{k}_3$, $\mathbf{k}_3$ most likely inhibits $\mathbf{k}_1$. The next set of Heuristics therefore focuses on ways to find prerequisites only.

Table 5–11: Heuristics for prerequisite relations

HR-6 A term in a knowledge source $\mathbf{KS}_1$ which is neither selected as concept nor in the upper list of candidates, yet a member of the DSL and chosen as concept for another knowledge source $\mathbf{KS}_2$, is prerequisite to the concept of the first knowledge source $\mathbf{KS}_1$.
HR-7 In the presence of a relation between two concepts, together with an unique, explicitly stated path of documents connecting the two corresponding knowledge sources, there is a set of prerequisite relations between the concepts of each document (but the first one), and the destination of the path.

As seen before, only the most valuable terms of a knowledge source were subject to abstraction. Remember that all candidates are terms but not all terms are candidates. There are two steps used by Heuristic HR-6 in order to infer prerequisites, visualised by Figure 5–9 and explained in the following.

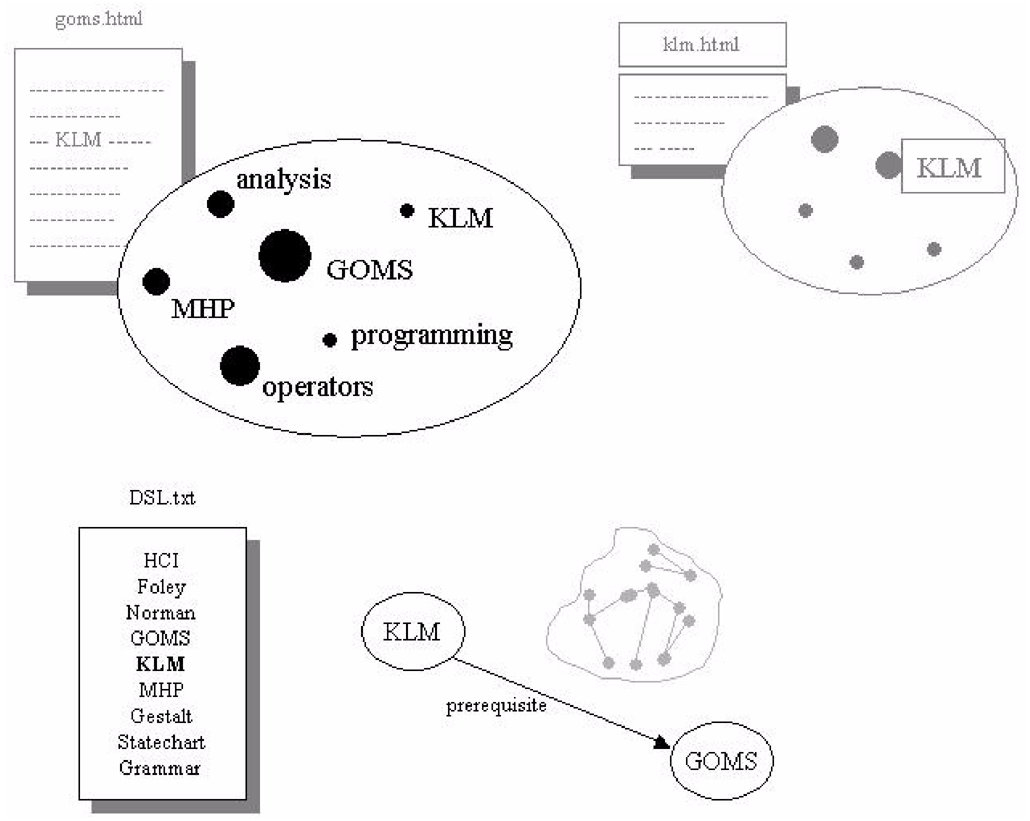


Figure 5–9: In the knowledge source “goms.html”, “GOMS” is selected as concept. One of the low-score terms “KLM” also occurs in the domain specific list. Since the “KLM” concept is already identified as a concept of another knowledge source (namely “klm.html”), the system concludes it to be important for the user’s understanding of the “GOMS” concept. This is indicated in the domain model through the prerequisite link.

For a randomly picked knowledge source **KS**, assume $\mathbf{k}_i$ was selected as the concept. The first step towards the prerequisites is to check if a term **t** from **KS** is assigned a low score during the conceptualization process. Then the system believes it to fall short as a key issue. Therefore, due to the Heuristics of the previous section (page 46 and on) it is unlikely that such a **t** is explained thoroughly in the text of **KS**. Second, if **t**, despite its low value, turns out to be a DSL member, it is for certain that it represents important knowledge of the domain as a whole. Since this knowledge is not debated in **KS** during user exploration, we conclude that if a concept **k´** from another knowledge source **KS´** that equals **t** already exists in the domain model, then it is a prerequisite to $\mathbf{k}_i$. In other words, the system can identify prerequisite relations $\mathbf{r} = (\mathbf{k}', \mathbf{k}_i, \textit{prerequisite})$ by comparing the DSL with the “less important” candidates from $\mathbf{Lk}_i$.

The last Heuristic depends on the results from the previous ones, as it makes use of one of the discovered implicit relations of any type between two concepts $\mathbf{k}_i$ and $\mathbf{k}_j$, i.e. $\mathbf{r} = (\mathbf{k}_i, \mathbf{k}_j, \textit{relationship\_type})$. A sequence of linked knowledge sources (stated explicitly by hyper references) make up a path, and therefore the corresponding concepts constitute an explicitly stated path $\mathbf{P}_{\mathbf{k}i,\ \mathbf{k}j} = [\mathbf{k}_i, ..., \mathbf{k}_j]$, which is at least true at the document level. If both an **r** and some **P** exist for two concepts $\mathbf{k}_i$ and $\mathbf{k}_j$ as exemplified in Figure 5–10 below ($\mathbf{k}_1$ to $\mathbf{k}_9$), Heuristic HR-7 gives birth to another part of the set of prerequisites $\mathbf{R}_{\text{prerequisite}}$, namely $\sum_s \cup\ \mathbf{r}_s$ where $\mathbf{r}_s = (\mathbf{k}_n, \mathbf{k}_j, \textit{prerequisite})$ and $i < n < j$. The existing relation that led to the prerequisites, can be deleted since it duplicates the information.

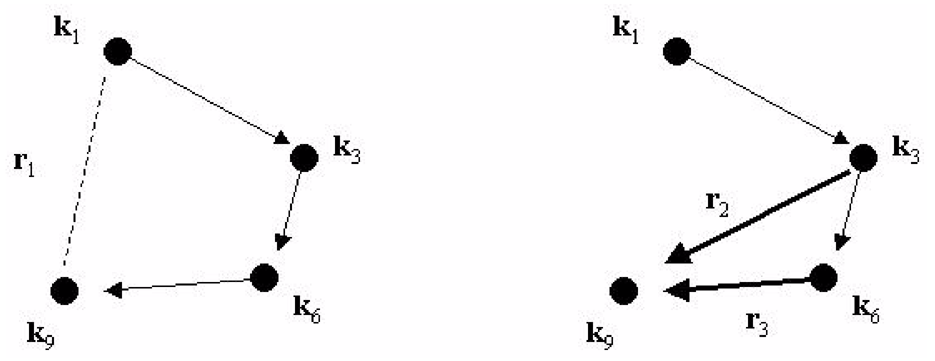


Figure 5–10: The explicitly stated path $\mathbf{P}_{\mathbf{k}1,\ \mathbf{k}9} = [\mathbf{k}_1, \mathbf{k}_3, \mathbf{k}_6, \mathbf{k}_9]$ together with the implicitly discovered relation $\mathbf{r}_1 = (\mathbf{k}_1, \mathbf{k}_9,$ *relationship_type*) leads Heuristic HR-7 to find two prerequisite relations, namely $\mathbf{r}_2 = (\mathbf{k}_3, \mathbf{k}_9,$ *prerequisite*) and $\mathbf{r}_3 = (\mathbf{k}_6, \mathbf{k}_9,$ *prerequisite*).

In short, say a parent relation exists between the two concepts `debat` and `agent`, whose knowledge sources are “debating.html” and “agents.html” respectively. Moreover, when there also is a sequence of links from the two sources, e.g. from “debating.html” through “software.html” and “iui.html”, to “agents.html”, then the following prerequisites should result [1]:

1. When the sources “software.html” and “iui.html” are conceptualised as *softwar* and *iui*, respectively.

- **r** = (**softwar**, **agent**, *prerequisite*)
- **r** = (**iui**, **agent**, *prerequisite*)

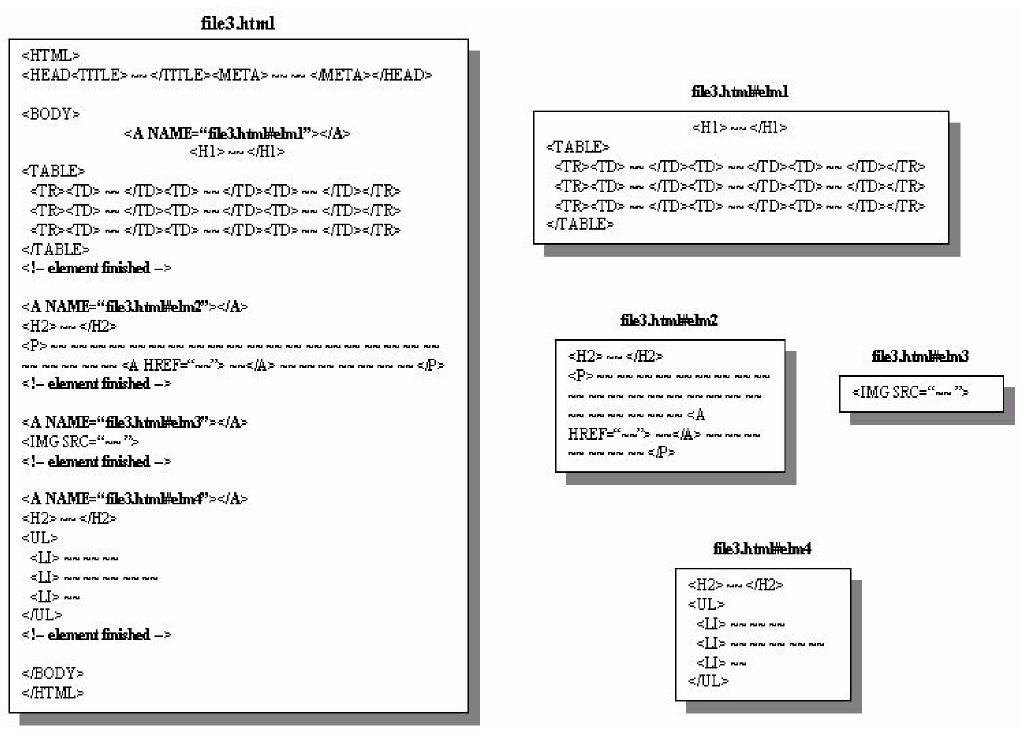
5.2.6 Completing the domain model

By now the system has identified a set of concepts $\mathbf{K}_{domain} = \Sigma_i \cup \mathbf{K}_{doc\ i}$ where and a set of relations $\mathbf{R}_{domain} = \{\mathbf{R}_{has\_deeper\_explanation}, \mathbf{R}_{external}, \mathbf{R}_{parent}, \mathbf{R}_{synonym}, \mathbf{R}_{prerequisite}\}$. Together, the sets constitute the building blocks of the domain model, i.e. representing the knowledge of the domain, which is an important basis for the adaptation engine to perform its tasks producing adaptive presentations for the user. As previsioned in section 4.5.1 “Building a domain model”, page 30, the system can not be expected to construct a perfect domain model. Furthermore, as pointed out by Davis et. al, an imperfect model will lead to incorrect conclusions [Davis+93] which even more necessitates the need for revision and manual adjustment from a domain expert (i.e. the author).

When generating hypertextual presentations in the adaptive phase, the appropriate knowledge sources must be called forth. For efficiency reasons, the elements extracted from the documents should be stored in a database or in many small files (one file for each element). First, explicitly storing this information would ensure quicker customization of documents. Second, when in the adaptive phase the system must choose ingredients of the document to be generated. Since each knowledge source is formatted in HTML, scripting languages like `PHP`, `JSP` or `ASP` can be used to generate documents easily.

The notion of “document concepts” helped in the process of relating the different knowledge sources. Tempting as it may seem, merely storing the elements found in individual small files in the final knowledge base yields a potential pitfall. First, it would be difficult to reconstruct the original documents from the extracted, new knowledge sources, despite the parent relation, since only the elements classified as important were subject to conceptualization. In the worst case the original documents might be poorly tagged, hence leaving the system to omit a lot of important information in its domain model. Second, it would be difficult for the system to allow a user to switch between adaptive and original mode.

Retagging the source documents, i.e marking up each element with an unique ID using the <A NAME> tag, would elegantly solve these problems. Even better, retagged documents allow for other variants of adaptations including dynamic link generation to specific areas of interest, adding or removing sections of the original documents and so forth. Retagging the documents therefore yields an essential supplementary source of information for the system, second to the file/database representation. The storage issue is shown in Figure 5–11.



file3.html

```
<HTML>
<HEAD<TITLE> ~~ </TITLE><META> ~~ ~~ </META></HEAD>

<BODY>
                <A NAME="file3.html#elm1"></A>
                        <H1> ~~ </H1>
<TABLE>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~ </TD></TR>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~ </TD></TR>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~ </TD></TR>
</TABLE>
<!-- element finished -->

<A NAME="file3.html#elm2"></A>
<H2> ~~ </H2>
<P> ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~
~~ ~~ ~~ ~~ ~~ ~~ <A HREF="~~"> ~~</A> ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ </P>
<!-- element finished -->

<A NAME="file3.html#elm3"></A>
<IMG SRC="~~ ">
<!-- element finished -->

<A NAME="file3.html#elm4"></A>
<H2> ~~ </H2>
<UL>
  <LI> ~~ ~~ ~~
  <LI> ~~ ~~ ~~ ~~ ~~
  <LI> ~~
</UL>
<!-- element finished -->

</BODY>
</HTML>
```

file3.html#elm1

```
                    <H1> ~~ </H1>
<TABLE>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~ </TD></TR>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~ </TD></TR>
  <TR><TD> ~~ </TD><TD> ~~ </TD><TD> ~~ </TD></TR>
</TABLE>
```

file3.html#elm2

```
<H2> ~~ </H2>
<P> ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~
~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~
~~ ~~ ~~ ~~ ~~ ~~ <A
HREF="~~"> ~~</A> ~~ ~~ ~~
~~ ~~ ~~ ~~ </P>
```

file3.html#elm3

```
<IMG SRC="~~ ">
```

file3.html#elm4

```
<H2> ~~ </H2>
<UL>
  <LI> ~~ ~~ ~~
  <LI> ~~ ~~ ~~ ~~ ~~
  <LI> ~~
</UL>
```

Figure 5–11: To the left, an example of a retagged document holding information about the elements. For efficiency reasons, the elements are also stored in individual smaller files (to the right). The representation in total yields flexibility for the adaptation in many senses

5.3 Generating adaptive presentations

With knowledge of the domain model a step is taken towards tailoring documents to each user. The next concerns user modelling. This section *briefly* suggests some overall pragmatic issues in order to place the work from the previous sections in context of an AHS. In particular, we show how some of the previous made commitments ensure intelligence a the system's behaviour.

5.3.1 Structure of the user model

According to Heuristic HC-7, one concept represents many knowledge sources. This implies that a more complex representation of which concepts are actually learned should be embedded in the domain model. An appealing solution is to associate the knowledge state with each concept, so that the degree of user knowledge on a concept may vary. E.g. the user might have *no* knowledge at all, *incomplete, complete* or *deeper* knowledge about a concept.

A user model (UM) is a knowledge representation (KR) that must represent the knowledge state of each user. In general, a KR play five main roles [Davis+93], each leading to important properties. First, since the KR is a surrogate for some real entities, it is a source of error. Second, we need to make some decision on "what to see" in order to hide some of the potential complexity, and these ontological commitments constrain the view of the task at hand. Third, the KR is only part of intelligent reasoning. Different definitions of intelligence contribute both to the selection of what representation to use and to the form and content of the inferences that can be legally made. Finally, the last two roles deal with efficiency and expression issues, which is of less importance for the following discussion.

At first glance, it seems natural to copy the building blocks from the domain model (DM) into UM. However, several questions concerning the interaction are at hand. What should be presented to the user at the first time of interaction? How can adaptations be made based on empty user models? Do all users have equal preferences? When is the content learned? In order to answer these questions, we outline a structure for the user model, keeping the nature of a generic KR in mind.

The first two questions have straightforward solutions. First, the user could browse the original documents leaving for the system to record the concepts encountered (note that this is possible due to the retagging introduced in the previous section). After some initial interaction steps, UM would hold some few concepts (i.e. nodes), and adaptive behaviour can take place. The second option is for the system to present the information associated with a start-node, explicitly specified by the author, or randomly picked. Third, if the user has used the AHS earlier in other, related domains, as might well happen in educational contexts, there is a chance for similar concepts from the user model **UM´** of the already learned domain and the concepts to be learned from the present domain **DM**, thus allowing the system to initialise the present user model **UM**. Figure 5–12 illustrates a situation with a

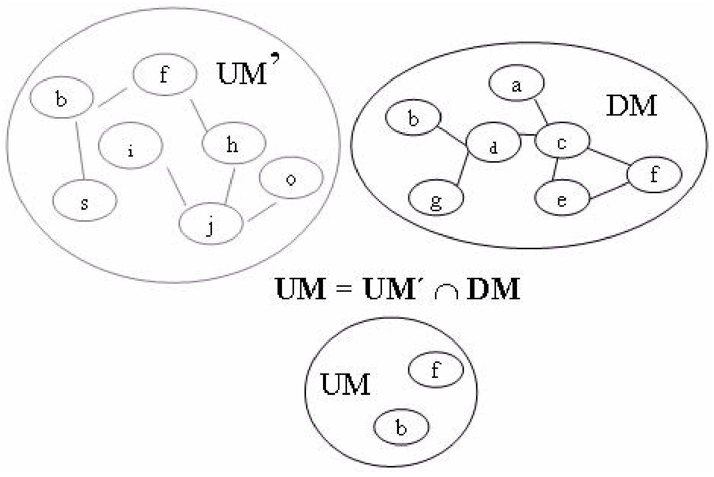


Figure 5–12: The user has already learned a domain UM', which has some common concepts with the DM at hand. Note that since the concepts "b" and "f" are not directly related in DM, they remain so in the user model.

previously learned user model **UM´** = {b, f, h, i, j, o, s} , where the letters represent concepts, and the domain model to be learned is **DM** = {a, b, c, d, e, f, g}. The system can easily initialise a new user model **UM** through the operation **UM´** $\cap$ **DM,** i.e. **UM** = {b, f}.

The third question points out an additional benefit of embedding user models in adaptive systems. Not only the knowledge level will vary among users. Presentational and learning style preferences are also likely to differ, especially for hypertextual environments. Therefore, user models for an AHS should include preferred learning style or media types, background, skills of the user etc. Additionally, drawing assumptions on the level of the user skills, can be useful. More skilled users should be presented advanced features and immediately understand the underlying concepts, whereas novices would need a mix of many different presentations of the same concept. A stereotype approach as discussed in section 4.3 “Models add power to adaptive systems”, page 24, can be used for this part.

There are many factors to be considered when analysing user behaviour, e.g. how a node (concept) is accessed, which information is contained in the node and the time spent [Brusilovsky96]. Griffin suggests that a time interval is more appropriate, since people also can get distracted from their task [Griffin97].

To summarize, the user model should at least contain information on concepts, user characteristics (expert, intermediate, novice), user preferences (illustrations, text, summaries), state of the concepts (complete, ready-to-learn, incomplete) and the relations.

5.3.2 Adaptation Model

The system should tailor documents for the user in order to fill the gap of knowledge. In its simplest form the selection task constitutes the relative compliment $\mathbf{K}_{\mathrm{DM}}$ - $\mathbf{K}_{\mathrm{UM}}$, that is picking those concepts from the domain model that has not yet been offered to the user. However, such a solution is not sufficient for an intelligent behaviour. In which order should the concepts be presented? When is a concept actually learned? How to best bridge the gap of knowledge for users with varying skills? Obviously, the system lacks an important component.

According to Wu et. al, the adaptation model states how the system can perform its adaptation based on a set of adaptation rules. In order to separate the implementation dependent aspects from the explicit models AM, DM and UM, some sort of adaptation engine (AE) responsible for performing the adaptation [Wu+01] and updating the user model [Kules00] is necessary. For our AHS, an AE provides the composition of documents on the fly by selecting knowledge sources as pointed to from the concepts in DM, through reasoning on UM, based on rules from AM. Henceforth, the process of tailoring documents is referred to as how to select appropriate concepts.

Due to the presence of the conceptual states in UM and the relationship types in DM, the AE can add intelligence to the process of selecting concepts. Concepts in

UM have values indicating at which level they are understood. The relationship types of DM indicate how the concepts are related. Keep in mind that the goal of serving the user with information would be twofold: both to plan for a sequence of concepts and to complete their respective knowledge states. In the following both parts are intertwined, since the adaptive documents should result based on both considerations.

In order to fill in more knowledge about a concept and complete its state, corresponding sections not yet learned could be offered to the user. Following *has_deeper_explanation* and *parent* relations should extend the user's understanding of a concept, e.g. when presenting a concept that is a deeper explanation of another, the status of the latter should be updated from "incomplete" to "deeper". For concepts that don't have any attached deeper explanations, the deeper level can be obtained when all knowledge sources associated with the concept is learned. Furthermore, traversing nearby concepts, not only increases the respective conceptual states. In general, for the task of bridging the gap between concepts far apart in UM, relations from DM would usefully guide the AE to construct paths that in sum would network the gaps. The rules from AM would specify what to do when encountering different relationship types, hence contributing to the resulting curriculum. In particular, rules in the AE might use the prerequisite relations to verify whether the prerequisite concepts are satisfied every time a concept is asked for, and if not, simply present the prerequisites first. Finally, for expert users external relations would be useful as they most likely extend the context of the domain (c.f. HR-2, page 53). Likewise, external resources should not be presented to novices until all concepts are known.

5.3.3 Various forms of adaptations

Hitherto, the task of our AHS was to dynamically generate documents based on knowledge extracted from the domain. However, the structure of DM and the preservation of the original (retagged) documents presented in this thesis should provide for many sorts of adaptations. It is interesting to briefly explore some of the possible variants. We close this chapter by proposing three options acting as either stand alone solutions or extensions to our AHS.

One option that makes its actions quite transparent, is for the AHS to follow the predefined curriculum and content of the original documents, *greying out* undesired fragments, *insert* new sections, and dynamically adding constructed links (which is referred to as *link construction* in the literature). Greying out an undesired fragment would be based on whether the concept representing that section is learned or not. Sometimes, prerequisite relations would trigger the system to add more information on a page, e.g. based on incomplete knowledge of prerequisites[1]. Link generation can easily be accomplished due to the form of the marked-up sections from the retagging phase (c.f. Figure 5–11), allowing dynamic links to refer to specific areas of the original documents as well as the documents as a whole.

1. The system might simply embed prerequisite concepts whenever their knowledge states are incomplete.

As a second option, based on the element types, we propose that short summaries can be customised for users asking for a synopsis of the domain or some subject. In particular, a *synopsis* for one concept can be made by generating lists out of knowledge sources that originally were e.g. heading-elements. Likewise, if a *resumé* of the domain is to be made, the system can use the same strategies for concepts it infers to be more important than others, that is for “key concepts”. As an example, an Heuristic which claims the number of relations to and from a concept indicate its importance, would be provide a simple way to identify key concepts. More sophisticated methods are obviously possible, like a more thorough analysis of DM, searching for the most centralised concepts. Remember from the discussion in the first section of this chapter where we debated what form the concepts should take on. Since we stuck to descriptive terms stolen directly from the knowledge sources instead of building a vector representation, *even shorter summaries* can also be provided by listing a range of concepts and present them as a keywords-list. This additional benefit would require no extra work but a few rules to select and display the most suitable concepts.

Lastly, if the user has insufficient or incorrect knowledge about a concept, e.g. revealed through predefined, provisional tests, the system should decrease its state in UM to the appropriate level, so that previously presented concepts can be redisplayed. This process possibly includes a reorganisation of the first curriculum or a more extensive use of illustrations. Furthermore, assume that the system can *visualise* DM, i.e. drawing the domain model network on the screen. When combined with the user model, the knowledge gap could be shown to the user, who would thus be able to place what was already learned into a larger context understanding what is around the next corner. Note that the ability for the user to puzzle the context of the part of DM which is not already walked-through, is again due to the conceptual descriptions.