A Transaction Model for $\mathcal{H}$ypertext

P.L. van der Spiegel J.Th.W. Driessen P.D. Bruza
Th.P. van der Weide

*

Dept. of Information Systems, University of Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, e-mail: pvds@cs.kun.nl

Published as: P.L. van der Spiegel, J.T.W. Driessen, P.D. Bruza, and Th.P. van der Weide. A Transaction Model for Hypertext. In D. Karagiannis, editor, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 91)*, pages 281–286, Berlin, Germany, 1991. Springer-Verlag.

Abstract

This paper presents a specification of $\mathcal{H}$ypertext systems on several levels of abstraction, and describes translations between these different levels. The $\mathcal{H}$ypertext system will be organised according to a stratified architecture, which is a generalisation of the so-called Two Level Architecture.

We describe a user interface for the $\mathcal{H}$ypertext system, examine the user's elementary operations (transactions) and present the corresponding user view on the system.

Next a conceptual schema is presented, that describes $\mathcal{H}$ypertext from the perspective of the *author* (editor) of the information. We describe the user transactions in terms of operations on the conceptual schema.

An object oriented implementation of this $\mathcal{H}$ypertext system is presented. We show how the object hierarchy is derived from the conceptual schema.

1 Introduction

Current research on $\mathcal{H}$ypertext systems appears to focus around two aspects. Firstly, research is focussed on the design of powerful user interfaces. This topic received the most attention so far. Secondly, research is directed at basic concepts that constitute $\mathcal{H}$ypertext (see for example [2] and [16]).

*This work has been supported by the ESPRIT project APPED (2499).

In this paper we deal with both aspects, and relate them to each other. For this purpose we use a general architecture for $\mathcal{H}$ypertext systems. We follow the approach for Information System Architecture as sketched in [3], and consider a $\mathcal{H}$ypertext system as consisting of the following components (see figure 1):

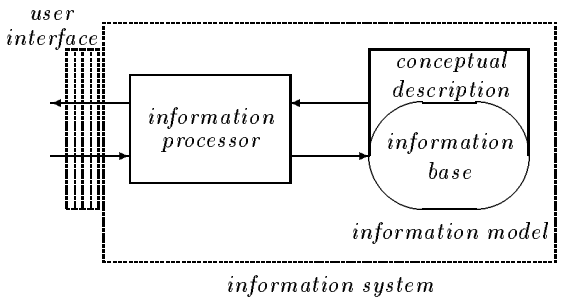

Figure 1: The Information System Paradigm

1. the information model. This is also denoted as application model or as title ([2]). It consists of:
   (a) a conceptual description (specification), describing the structure of the stored information, and the rules that govern modifications of the stored information (such as constraints).
   (b) an information base, containing the stored information according to the conceptual description. This is usually referred to as an instantiation (population) of the conceptual description.
2. an information processor, that processes user requests. The information processor accepts commands from the user via a user interface, interprets them in terms of the conceptual description, and responds in accordance with the information model (both the information base and the conceptual description).

An important difference between $\mathcal{H}$ypertext systems and (conventional) information systems is the concept of *associative link*, that enables the user to jump unconstrained through the information base. In terms of the relational model the associative link may be understood as a cross-reference from any tupel of any table into any tuple of another table.

In traditional $\mathcal{H}$ypertext systems, in contrast with conventional information systems, there is almost no conceptual description of the stored data. The weaknesses of such an approach have been discussed by several authors ([7][15]). There seems to be a growing need to be able to support a conceptual description in relation with $\mathcal{H}$ypertext or documents, especially the combination of structured documents with $\mathcal{H}$ypertext applications looks promising. Our setup combines $\mathcal{H}$ypertext and document description languages, such as ODA ([4]), SGML ([10]) and TEX ([11]).

The outline of the paper is as follows. In section 2 we introduce a multi-level organisation of $\mathcal{H}$ypertext. Three levels are emphasised in this regard.

One level is concerned with the actual information, the second level with the disclosure of the information. The third level is intended to administrate the users' own routes of preference through the $\mathcal{H}$ypertext to their favourite areas of exploration. Such user behaviour, sometimes also referred to as user profile, forms a dynamic aspect, even in an otherwise read-only $\mathcal{H}$ypertext such as CD-Rom applications.

Next we describe the different components of the system, on the basis of the general architecture as presented in figure 1.

As mentioned earlier, navigation is the primary activity of the user of a $\mathcal{H}$ypertext system. For this reason the commands, given to the information processor via the user interface, consist of operations to support navigation. The user interface is a presentation of the system to the user, hiding (as good as possible) internal aspects. It reflects the internal state of the information system, and gives the user the opportunity to change this internal state (for example by clicking a button). These transactions with the system are modelled in section 3, and result in what is called the transaction model.

In section 4 the underlying conceptual schema is discussed in detail. This conceptual schema supports the multi-level organisation of the information as introduced above. We describe how user transactions are translated to operations on the conceptual schema in section 5. Finally we indicate how a transaction model can be implemented in the Object Oriented Paradigm.

2 Multi-level organisation of the information model

In [2] a new mechanism was presented to structure $\mathcal{H}$ypertext into two levels. Similar approaches are discussed in [1] and [12]. The intention of this structure was to provide a means of facilitating user orientation in Hypermedia in response to the well documented "lost in hyperspace" problem [13][14]. The bottom level of this mechanism is called the *hyperbase* and the upper level the *hyperindex*. We denote the hyperbase by $\mathcal{H}$ and the hyperindex by $\hbar$.

The hyperbase corresponds to $\mathcal{H}$ypertext as is typical in current systems (with the extension that both the conceptual description and the actual information are contained).

The hyperindex is an index, organised in the form of a $\mathcal{H}$ypertext, and is used to index the information in the underlying hyperbase. Its underlying principle is that any index term can be *refined* or *enlarged*. When the user arrives at some potentially interesting index information item, the information in the hyperbase, relevant to this particular index item, can be retrieved. In this way the user moves from the hyperindex level to the hyperbase level. This type of browsing, by navigating through the hyperindex, is termed *query by navigation* (QBN). Querying is thus reduced to a form of browsing which is an easily understood form of searching behaviour.

Users of a $\mathcal{H}$ypertext application can be compared with explorers. Slowly but surely they build up knowledge about relevant parts of the hyperbase. They need the possibility to connect relevant information (in their own subjective way). This need is administrated in our architecture by an additional level: the *hypermap*, denoted by $\tilde{m}$. This hypermap can be compared with a made-to-measure road-map (one for each user), which serves as a private guide for the user on how to reach specific information. This feature supports the user, who travels around in the loneliness of the vast information-desert, by recording a kind of historical document of what was achieved.

In this paper we restrict ourselves to the three levels we just described. However, useful extensions are possible, for example a level that indexes the hyperindex, or a hypermap for the hyperindex. Also we do not consider interlayer connections between the hypermap and the hyperindex.

The resulting structure of the information model is shown in figure 2. Details of this figure will be explained in the next section. Each of the $\mathcal{H}$ypertext (sub)structures has its own conceptual description, information base, and set of associative links. Between the levels are connections.

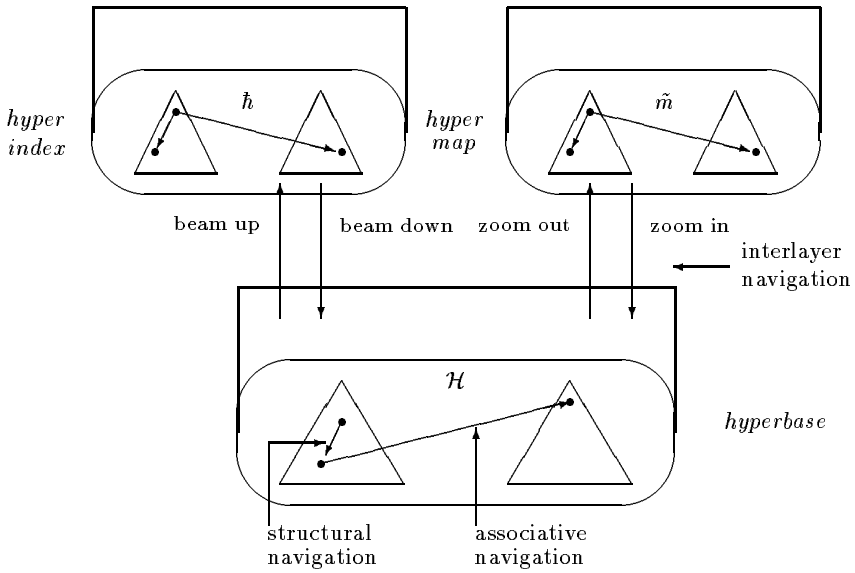

Figure 2: Multi-level organisation of $\mathcal{H}$ypertext

3 The user interface

Now that the multi-level architecture has been introduced, we describe the behaviour of a system based on this architecture. This behaviour will then serve as a base to build the associated transaction model.

To describe the interaction of the system with a user we use the *conceptual dialogue specification technique DST* (see [5] and [6]). First the DST-technique will be introduced briefly. We use the style that was presented in [9].

The intention of a *DST-diagram* is to describe a dialogue, usually between man and machine. A DST-diagram maps out the different stable states of an information system, and the dialogue by which the user can bring about state transitions. A brief description of the intention of the symbols is as follows:

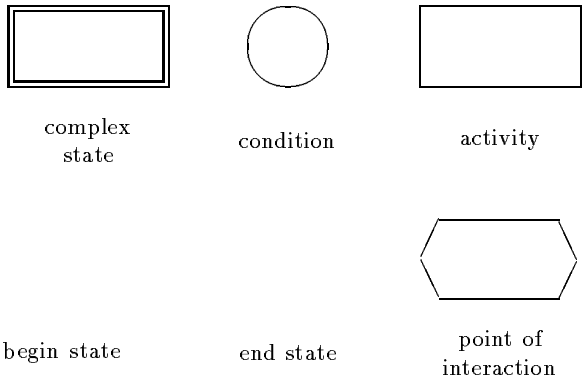

Figure 3: DST-diagram symbols

- The **begin state** and **end state** denote the states in which a dialogue begins or ends respectively.
- A **point of interaction** is a state in which the system waits for user input.
- An **activity** denotes an autarkic action of the system, i.e., an action requiring no (or only trivial) interaction with the user.
- A **complex state** denotes a subdialogue which is refined in an other DST-diagram. By means of complex states the possibility of *hierarchical decomposition* is provided.
- In a **condition** (transition guard) the system evaluates the associated condition. According to its outcome, one of the transitions is selected. Note that conditions are *not* restricted to boolean conditions.

The above symbols can be connected by so-called transition arrows, which may have an associated label.

A well formed DST-diagram is a DST-diagram that conforms to a number of rules. Some important ones of these are:

- A DST-diagram must have a unique begin state and a unique end state.
- The begin state must lead to every other state.
- The end state must be reachable from every other state.

At this point we assume that the DST-diagrams have been sufficiently well introduced to be able to understand the specifications which will follow. For more details about DST-diagrams the reader is referred to [6].

3.1 The Interaction with $\mathcal{H}$ypertext

Imagine we are sitting behind a terminal, navigating our way through the $\mathcal{H}$ypertext system, looking for information. A typical stable state will be that the system has presented some information on the screen, that we are reading and using for deciding what we want next. A collection of information that is offered as a whole to the searcher is denoted as a *presentation*. It is the central point of interaction in figure 4. We make a distinction between the presentation and the underlying object that is manifested by this presentation. We will use the term *context* to denote this underlying object.

From a particular context there are several possibilities to go to an other stable point (piece of information):

1. Typical for $\mathcal{H}$ypertext systems is to offer the opportunity to have follow up on some issue that is presented on the presentation screen. For example, pieces of the text on this screen are highlighted in order to indicate that they can be selected for further reading. These pieces of text then form the *buttons* that can be pushed in order to communicate to the system what next action is required. These connections are termed *associative links*. We use the term **Associative Navigation** for this kind of traversal through the information.

2. Associative links are very flexible as they impose no restrictions on making connections between pieces of information. However, just as `goto` statements in programming languages, they tend to deform rapidly into a muddle. A way to overcome this would be to assign labels to associative links. A better approach, however, is to organise the information according to some structure. An example is the organisation of the information as a book, consisting of chapters, chapters consisting of sections, etcetera.

   Imposing a structure can be compared to the usage of typed variables in conventional programming languages. In this case, the $\mathcal{H}$ypertext system will enforce the compliance with the imposed structure.

   As a result, navigation over the structure is of an other nature than Associative Navigation. It is denoted as **Structural Navigation** in figure 4. Structural Navigation is characterised by the fact that it extends (enlarges) or contracts (refines) the current context. Going from a chapter of a book to one of its sections is an example of context refinement.

3. The next option is navigation between two layers: **Interlayer Navigation**. This kind of navigation is displayed in figure 2 as *beam up* and *beam down* (or *zoom out* and *zoom in*). The underlying idea is traversal from the current context into the best suitable context within the other layer. For example, beam down from the hyperindex into the hyperbase leads to a context in which all relevant contexts are listed in order of relevance.

4. A fourth option is to give a partial description of subject of interest and to go to that subject in the hyperbase. This is often called a **query**. We will not deal with query handling in this paper but refer to [8] for this.

5. Of course there is also the possibility of leaving the $\mathcal{H}$ypertext system.

The various options are depicted in a DST-diagram in figure 4.

3.2 The user's concepts/elementary transactions

In the previous section the behaviour of the system towards the user was introduced. We will now examine the concepts related to this behaviour somewhat further.

The user is always in a certain *context*, represented by a *presentation*. That context stands for a certain *structure* that again is in a *layer*. Every navigation step leads to a change of context. The active context is left, and a new one is entered.

- During associative navigation, little can be said about the new context. We direct ourselves to information, probably in quite another context. It might even be the case that we can reach the requested information in more than one context. For example if we follow a link to the map of a city, then this map may be part of a tourist guide, but may also occur in the context of a detective book, that contains the map for reader convenience. In case of such *context ambiguity* the system has to resolve this confusion, by letting the reader select one context. When the new context is clear, the system will shift to this new

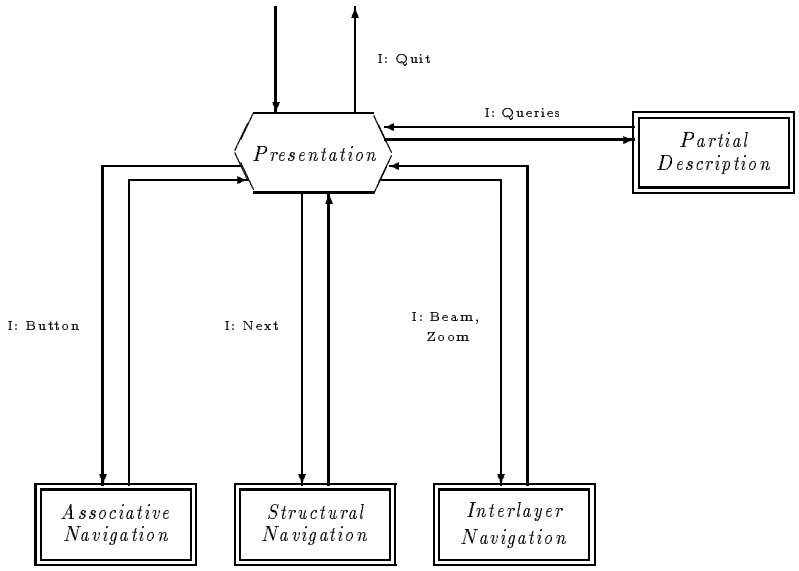

Figure 4: Top Layer DST-diagram Interaction with User

context. A *context shift* takes place. The decomposition of Associative Navigation is presented in figure 5.

- When navigating following the structure only a minor change of context takes place. We call this: *context adjustment*. The effect of structural navigation is enlargement or refinement of the current context. The decomposition of structural navigation is shown in figure 6.
- The navigation between two distinct layers (Interlayer Navigation) can be initiated by selecting in the presentation screen the desired kind of interlayer navigation (*beam up*, *beam down*, *zoom out* or *zoom in*). When the user selects a beam or zoom, the system determines the new context. This new context is in the newly selected layer. Here also a *context shift* takes place.

The various possible context changes and the other concepts of the navigation process are described schematically, using an E.R. like notation, in figure 8.

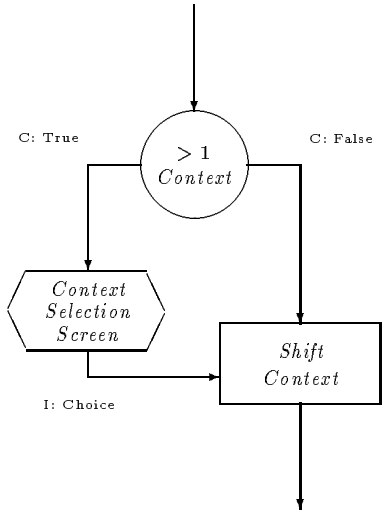

Figure 5: Refinement of Associative Navigation

4 The conceptual description

In the previous section we presented the information structure from the perspective of a person who is only consulting the information. More sophisticated views on the information exist however. In this section we

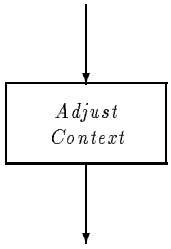

Figure 6: Refinement of Structural Navigation

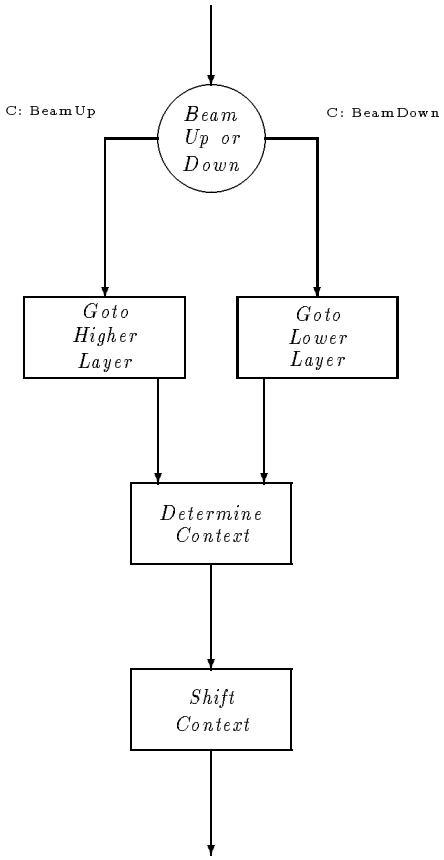

Figure 7: Refinement of Interlayer Navigation

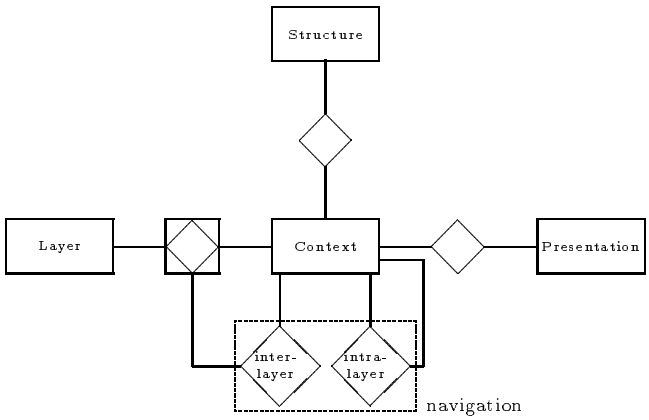

Figure 8: Schematically description of the user's concepts

present the overall underlying information structure, the conceptual description (see figure 9). This will for example be used by the author of the application. We consider $\mathcal{H}$ypertext as consisting of: *structure*, *presentation* and *contents*. We will discuss the information structure from figure 9 in detail in this section.

The basis of the conceptual model is a set of so called *fragments*. Fragments are elementary pieces of information, which are not decomposed structurally into smaller components. The criterion for judging whether a fragment is atomic or not is not necessarily a property of the fragment itself, but rather is dependent on the lowest level of granularity at which the information is to be considered. For example, animation can be considered as a single fragment, or as a sequence of fragments. Each fragment has associated *data* of a particular *medium* (text, video, audio, etc).

Usually information is structured in some well defined form. For example, if the information has the form of a book, it is grouped into chapters, where a chapter consists of sections, etc. There are several approaches for specifying the structure of the information. For example, context free rules are used to describe the structure of a document in SGML [10], which seems to be the emerging defacto standard for this purpose. Other possibilities in this regard are ODA [4] and TEX [11]. We basically adopt the grammar based approach of SGML. As a consequence, an information structure is described by the following components:

- A set of symbols denoting structural elements
- A set of context free rules
- A start symbol

A *rule* comprises a left hand side (relation *lhs*), which consists of a single *structural element* and a right hand side (*rhs*), which is a series of one or more structural elements (see figure 9). Each structural element in the right hand side may be qualified by one of the following occurrence indicators (*occ*):

* the so-called Kleene star, the qualified structural element may occur zero or more times.

\+ the so-called Kleene plus, the qualified structural element should occur at least once.

? the qualified structural element may occur at most once.

The structure definition specifies a framework for the document structure. An *actual structure* is a hierarchy of instances of structural elements conforming to the structure definition. We call such an instance a *molecule*. The hierarchy is modelled by the relation *parent child*.

Links model cross references in the information. In our model, two types of links are distinguished: *context bound* links and *context free* links. A context free link is an association between two fragments that holds independent of any context and is in this sense absolute. A context bound link is also an association between two fragments, but its *source* and *destination* both have associated a certain context. This context is established by the associated molecule-fragment pair.

One of the most powerful aspects of $\mathcal{H}$ypertext is the ability to define different views on the same underlying information. A *view* consists of a structural definition and an associated actual structure. Finally a view contains a set of associated links.

A set of related views together forms a *layer*. Every view has an associated unique layer. Note that in this paper we only consider the layers hyperindex, hypermap and hyperbase.

5 Translation

Up to this point we have described two ways of looking at a $\mathcal{H}$ypertext system. The first is the view from the perspective of a user, the second is the underlying structure of the information. The relation between the two will be described in this section, on the basis of the user interface from section 3 and the internal structure of section 4.

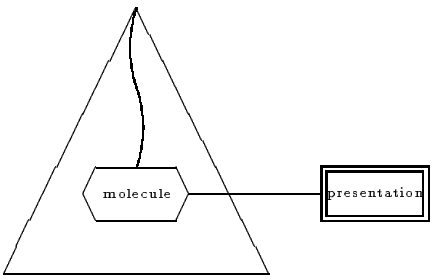

Figure 10: A context

First we consider the stable state in figure 4 which is a *presentation*. As mentioned before this presentation represents a certain abstract object, called context (see figure 8). A context corresponds to a certain molecule (see figure 9), and the path to the root of the corresponding actual structure (figure 10). This structure is related to a view.

The *layer* and the *presentation* entities from figure 8 can be mapped to the same entities from figure 9. The contents of the presentation are modelled

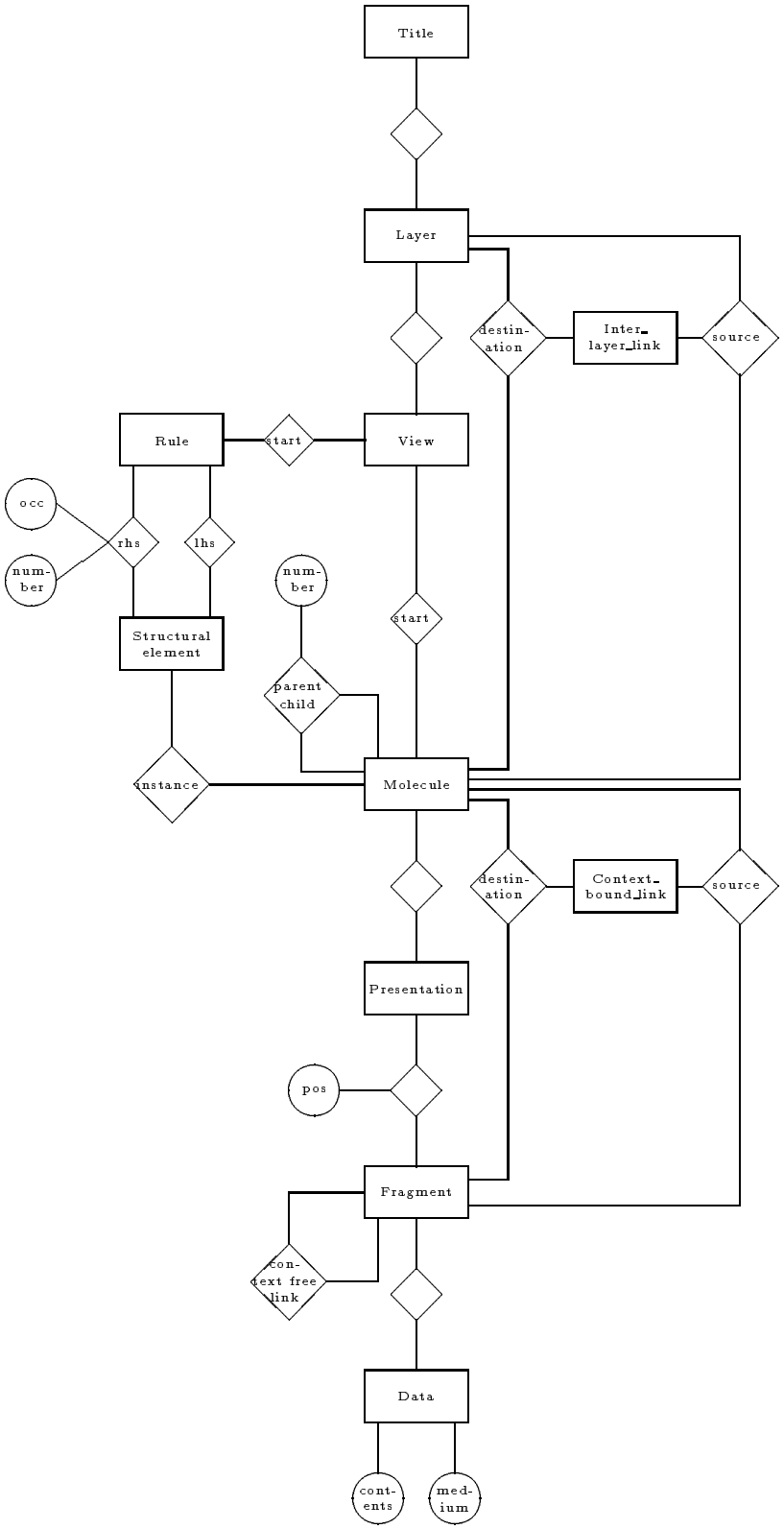

Figure 9: Conceptual Schema of the Information Model

as a partially ordered set of fragments, each on a certain position (relation *pos*) in the presentation.

Operations on the user level can now easily be translated to operations on the conceptual schema. On the elements of the conceptual schema we only need some primitive building and retrieval operations.

- Associative Navigation is achieved by following either context bound or context free links. If the destination of a context bound link consist of more than one context, one of them has to be selected.
- For Structural Navigation the *parent-child* relation over *Molecule* is used. For example, context enlargement implies taking the father of a certain molecule.
- Interlayer Navigation uses the *interlayer link* in an obvious way.

Now the operations on contexts are automatically clear. Context adjustment is a (little) modification of the path which defines the context. A context shift implies building a completely new path in a certain structure.

5.1 An object oriented prototype

In this section we give a short description of a prototype implementation in the Object Oriented Paradigm. Our approach was to implement each entity type of the conceptual schema from figure 9 as a so called class, where a class consists of a data structure and the methods for accessing the data.

The methods within a class and the inheritance structure of the classes are derived from:

1. the relation types of the conceptual schema
2. the usage as described in the transaction model.

This derivation is quite straight forward, and therefore not described in detail. For example, the procedure to present a molecule makes use of the routines that are provided by the class Presentation, that handles the construction in terms of fragments. The resulting class hierarchy is depicted in figure 11.

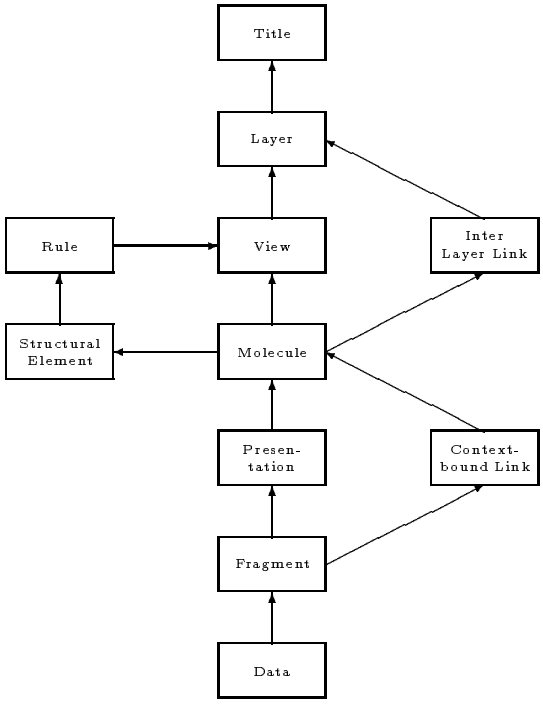

Figure 11: Class hierarchy of the prototype

6 Conclusions

In this paper we have tried to bridge the gap between conventional information systems and document oriented systems (especially $\mathcal{H}$ypertext systems). The resulting system can benefit from the concepts of both worlds. We see that $\mathcal{H}$ypertext manipulation can be interpreted in terms of conventional conceptual schema manipulation. The inverse however does not seem to hold. For example, there is no counterpart for query by navigation in a conventional SQL database. Further research therefore might concentrate on the meaning of such operations in the world of databases.

References

[1] M. Agosti, A. Archi, R. Colotti, R.M. Di Giorgi, G. Gradenigo, B. Inghirami, P. Matiello, R. Nannuci, and M. Ragona. New prospectives in information retrieval techniques: a hypertext prototype in environmental law. In *Online Management 89, Proceedings 13th International Online Information Meeting, London, England*, pages 483–494, 1989.

[2] P.D. Bruza and T.P. van der Weide. Two Level Hypermedia - An Improved Architecture for Hypertext. In A.M.Tjoa and R.Wagner, editors, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 90)*, pages 76–83. Springer Verlag, 1990.

[3] J.A. Bubenko. Information system methodologies - a research view. In T.W. Olle, H.G. Sol, and A.A. Verrijn Stuart, editors, *Information System Design Methodologies: Improving the Practice*, pages 289–318. North-Holland, 1986.

[4] I.R. Campbell-Grant and P.J. Robinson. An Introduction to ISO DIS 8613 - Office Document Architecture - and its Application to Computer Graphics. *Computer and Graphics*, 11(4):325–341, 1987.

[5] E. Denert. Specification and design of dialogue systems with state diagrams. In E. Morlet and D. Ribbens, editors, *Proceeding International Computing Symposium*, pages 417–424. North-Holland, 1977.

[6] E.J.T van Dinter, M.P.W. Martens, A.H.M. ter Hofstede, and S. Brinkkemper. Specification and Implementation of the Conceptual Dialogue Specification Technique DST. Technical Report 90-17, Department of Information Systems, University of Nijmegen, The Netherlands, November 1990.

[7] P. Garg. Abstraction mechanisms in hypertext. *Communications ACM*, 31(7):863–870, July 1988.

[8] F. Halasz. Reflections on notecards: Seven issues for the next generation of hypertext systems. *Communications ACM*, 31(7):836–852, July 1988.

[9] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: Chopping down the methodology jungle. Technical report, Department of Information Systems, University of Nijmegen, The Netherlands, dec 1990. To be published.

[10] ISO8879. Information Processing - Text and Office Systems - Standard General Markup Language (SGML), 1986-10-15.

[11] D.E. Knuth. *The TEXbook*. Addison Wesly, reading, Massachusetts, 1984.

[12] D. Lucarella. A Model for Hypertext-Based Information Retrieval. In *Proceedings of the European Conference on Hypertext - ECHT 90*, pages 81–94. Cambridge University Press, 1990.

[13] J. Nielsen. The art of navigating though Hypertext. *Communications ACM*, 33(3):296–310, March 1990.

[14] B. Schneiderman and G. Kearsley. *Hypertext Hands-On!* Addison-Wesley Publishing Company, 1989.

[15] P. Stotts and R. Furuta. Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. *ACM Transactions on Information Systems*, 7(1):3–29, 1989.

[16] F. Tompa. A Data Model for Flexible Hypertext Database Systems. *ACM Transactions on Information Systems*, 7(1):85–100, January 1989.