A Virtual Hyperbooks Model to Support Collaborative Learning

Gilles FALQUET and Jean-Claude ZISWILER
Centre Universitaire d'Informatique, Université de Genève
24, rue Général-Dufour – CH 1211 Genève 4 – Suisse
{gilles.falquet,jean-claude.ziswiler}@cui.unige.ch

Abstract. Learning by collaboratively writing scientific hyperbooks requires specific software tools. We present a model for creating, managing, and viewing hyperbooks. This model is comprised of a re-usable document repository (fragments repository), connected to a domain ontology. The model takes into account the notion of point of view, allowing a user to read the hyperbook according to a specific reading objective or to his or her profile. The model also includes an interface specification language for the creating different hypertext views of the hyperbook contents. The hyperbook model we propose is an example of virtual document model because the hyperdocuments the reader/writer actually sees are not stored but generated by assembling stored fragments according to an interface specification A purely declarative language allows the definition of the views that make up the interface of the hyperbook. We also present the architecture of a hyperbook management system which is based on a database management system and a hypertext view generation system for databases.

Keywords. Scientific hyperbooks, electronic books, reusable documents, virtual documents, ontologies, views, point of view.

1. Introduction

During the last few years we conducted several pedagogical projects that consisted in the collaborative construction of a scientific hyperbook. The principle was that the core of the hyperbook was made of lecture notes written by the teachers, and students were asked to produce new documents for the hyperbook. The teaching objectives of these projects were

- to help the students see the relationships that exists between the different concepts presented during the course (hence the hypertextual nature of the book)
- to give student the opportunity to participate in the collaborative writing of a large electronic document
- to show that the same subject matter can be seen from different point of views

Our first experiment with basic Web tools (an HTML editor, a drawing tool, and an HTTP server) clearly showed the need for a more sophisticated collaborative writing and publishing environment. Thus we decided to develop a Web based, database-backed, hyperbook management system. This system, and particularly its underlying models, evolved according to the needs and problems we observed.

The issues that appeared particularly important in this pedagogical context were the following:

Documents. It is necessary to make sure that the identification of a document (or document fragment) is stable over time and independent of its location (in this respect URLs

are not sufficient). It is also important to have a means to categorize documents, so as to facilitate their retrieval and re-use. The information content of a document should not be cluttered with presentation or linking markers, contrary to HTML documents where linking tags must be explicitly inserted in the text.

Links. Our first experiment showed that students had difficulties creating hypertext links between pages. These difficulties were mainly caused by technical reasons (volatile URLs, access rights to HTML files, etc.). When students were provided with efficient tools to link their pages we observed a proliferation of links that were only marginally relevant. Thus it is necessary to help the writers create meaningful and informative links.

Reading and writing interfaces. Reading a hypertext can cause cognitive overload because the reader has to manage his or her own reading path, as opposed to the linear reading of a simple text. Thus readers should be provided with hyperdocuments that can be read sequentially without navigation effort. On the contrary, writing and linking small text chunks is generally easier than constructing large sequential texts. Thus the reading interface must present linear texts that result from the assembly of small pieces of information. In addition, reading difficult and/or new material (sometimes called active reading) involves several auxiliary activities such as annotating, highlighting, summarizing, etc. Thus an effective reading interface should provide tools to support these "writing" activities.

Terminology and concepts. In scientific writing, terminology (the definition of concepts and their relationships) plays an important role. Scientific writings either refer to well-known concepts of the studied field or they contain new concept definitions. Thus, writing and reading a scientific hyperbook entails referring to and updating a domain ontology.

Point of views. It must be possible to read (or browse) the hyperbook according to different point of views. A point of view is a specific perspective on the book's domain, it can also be a reading objective (in-depth reading, overview, etc.).

The purpose of this paper is to present the hyperbook model that emerged from the combination of this pedagogical effort and our research work on hypertextual interfaces for databases. In fact, this model is a virtual hyperbook model because the documents and links that the user sees are generated from the information and knowledge fragments that are stored in the hyperbook. In other words, these documents are only virtual (or potential) in the hyperbook database.

The rest of the paper is organized as follows: the next section gives some background on the research area we are working in; section 3 presents the structural part of the model, section 4 presents in more detail the "hyperbook ontology" which is the core of the model; section 5 presents the interface and interaction part of the model, section 6 shows how this model can be implemented with a hypertext view system. Finally the conclusion pro-poses future research directions.

2. Background and Related Works

This hyperbook model we propose is build on ideas and concepts developed in the fields of hypertexts, document management, databases, Web interfaces, knowledge bases, and collaborative working. It can be seen as an implementation of the idea of personalized virtual document.

The concept of collaborative work has prompted the development of several Web-based tools, such as BSCW [1] or Learning Space (based on Lotus Notes). Most of these tools are essentially centralized document repository systems or coordination systems. Although they propose a Web interface, these tools are not aimed at collaborative hypertext writing.

In the hypertext field, it is striking to note that "pre-Web" systems were structurally and functionally richer than the Web hypertext model, which aimed at simplicity and de-

centralization. Hypertext research has been headed toward different domains and objectives that are of interest to us. Systems like Intermedia [2] or Storyspace [3] were essentially developed to study and produce hypertext literature; other systems, for instance KMS [4] or MacWeb [5], aimed at knowledge sharing and management; finally, some systems (HyperCard, NoteCard) were closer to highly interactive application development tools. Theoretical works have studied the fundamental notions of link, anchor, node composition, navigation, etc. and lead to the Dexter reference model [6].

Several models and systems have also been proposed to integrate the notions of book and electronic publishing, in order to create hyperbooks. The aim can be either to create hypertext versions of existing books [7], or to generate electronic books from paper books [8], or to directly write (hypertextual) electronic books [9], or to integrate existing electronic documents [10].

The hypertext personalization problem has attracted many research works that lead to the definition of models and techniques for adaptable and adaptive hypertexts [11], [12]. Adaptable hypertexts can present different contents, or differently organized contents, depending on the user's profile. Adaptive systems can automatically update the user's profile by observing his or her behaviour, in this case the adaptation is dynamic. A well known example of adaptiveness occurs in Web browsers: once the user has visited a page, all the links to this page are shown in a specific color to indicate this fact. In [13], the authors propose an adaptive hypertext model that includes a domain model, a user's model, and adaptation rules. The domain model is a semantic network comprised of concepts and relations between the concepts. This model is essentially used to define adaptation rules that depend on what concept the user knows or masters.

Recent research works concentrate on the notion of personalizable virtual documents [14], [15], [16]. These documents are sets of elements (generally called fragments) associated to filtering, organization, and assembly mechanisms. Given a user profile or reading objectives, theses mechanisms will produce different (real) documents that should meet the user needs. For instance, [17] proposes a virtual document model that is based on four ontologies, namely, a domain ontology, a metadata ontology, a document ontology, and an application ontology. The model we present here belongs to this approach.

3. Structural Aspect of the Virtual Hyperbook Model

It is generally accepted that virtual documents [15] are made of fragments (or “pieces of information”) that can be assembled to constitute directly readable real documents or hyperdocuments. We will thus consider that the basic informational contents of a virtual hyperbook consist of a collection of re-usable document fragments. A virtual hyperbook also contains an ontology that formally represents the concepts of the domain the hyperbook is about. Every fragment can be linked to one or more concepts through typed links that indicate the specific role played by this fragment with respect to this concept. The third component of a virtual hyperbook is the hyperbook ontology. Its purpose is to represent the different linking structures of the hyperbook (links between concepts and fragments and links among fragments), the fragment's organization, and the different point of views on fragments and concepts. The interface specification will be based on this ontology to generate the readable hyperdocuments.

3.1 Fragments

The basic informational contents of the hyperbook are made of reusable fragments. A fragment has a content and belongs to one or more categories. The content of a fragment is a tree of XML or XHTML elements. The category of a fragment indicates its intrinsic nature. Typical categories are: statement, question, theorem. Categories must not be confused with roles played by fragments with respect to the domain concepts. For instance, if a fragment is an example of the concept *cyclic graph*, it is at the same time counter-example of the concept *tree*.

Fragments can be connected by structural links to form compound fragments. These typed links indicate the roles played be the different fragments in the compound fragment. For instance, an exercise could be made up of a question fragment, one or more answer fragments, and a discussion. Compound fragments can have different purposes, they can represent pedagogical units (an exercise), or argumentative units (an issue related to positions and arguments), or even hyperbook management units (group discussions or weblogs). For instance, a discussion structure could be made up of *topic* and *message* fragments connected through *about* and *reply-to* links.

The important point is that direct links between fragments are purely structural while semantic links will be inferred by referring to the domain ontology.

Since the set of fragment categories and link types depends on the subject of the hyperbook, there are no fixed, predefined, categories and types. In fact, the fragment categories and fragment link types are defined in the hyperbook ontology (see section 4).

3.2 Domain Ontology

It is common in virtual document architectures to distinguish between the document fragments and the semantic structure. The latter, for example an ontology or a conceptual graph, describes the domain and is used for indexing or qualifying the fragments. The domain ontology is intended to hold a formal representation of the domain's concepts.

3.2.1 Concept Definitions

The concept definition language is a graph-based version of some description logic language. In this formalism a concept is either

- a primary concept
- a conjunction or a disjunction of a concepts
- the complement of a concept definition
- a role restriction made of a quantifier, a role name, a minimal and a maximal cardinality, and a range concept

A role restriction is represented by an arc pointing to the range concept and labelled with the quantifier, role name, and cardinality constraints. For instance, in the graph shown in Figure 1, the arc labelled "all component(4,4)" from quaternion to real number means that a quaternion has at least and at most four components that are real numbers and that all the components of a quaternion are real numbers. This same graph shows that a quaternion is a number which has exactly four real components, a multiplication operation, and an addition operation. It also shows that the addition of quaternions is commutative while the multiplication is not.

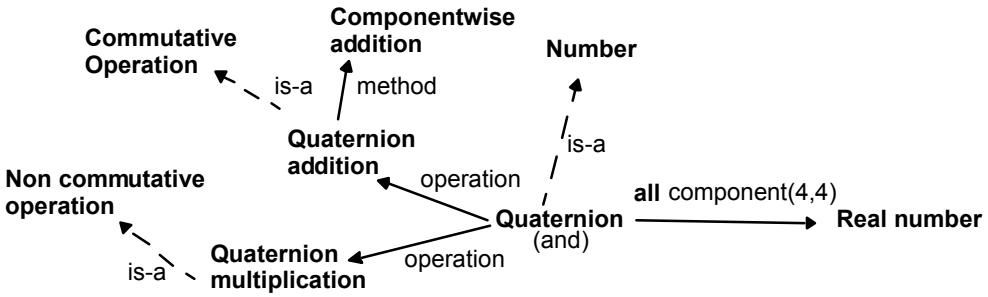

Fig. 1. Concept definitions

Although the language is expressive, it is not always necessary to use all of its features, in particular when the ontology is small. However, when the ontology becomes larger it may contain concepts that are only subtly different. In this case a more precise description of each concept is crucial to show their differences and similarities. This happens when one seeks to exhaustively describe a domain or a category of objects.

3.2.2 Concepts and Point of Views

It is a well-known fact that different experts would give different definitions of the same concept (or what they think is the same concept). For instance, the electron concept would be defined as massive particle with a negative unit charge that is insensitive to strong interaction. A definition provided by a chemist would probably be different, for example: "electric corpuscle that can be dragged away, caught or shared between atoms and molecules" whereas for the electronics engineer it is "the smallest charge carrier able to move in electric circuits".

Since one of our design objectives is to present the subject matter according to different point of views, the model allows for point of view dependent definitions of concepts. To implement multiple point of views in the ontology each arc and node can be associated to a point of view (see Figure 2). Hence the definition of a concept according to a point of view is obtained by selecting those arcs that belong to the desired point of view or to a more general point of view (as we will see in the next section, point of views are hierarchically organized).

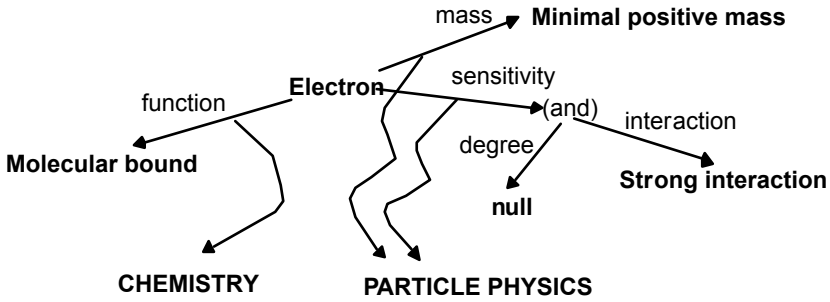

Fig. 2. Different point of views on the same concept

3.2.3 Roles of the Domain Ontology

Apart from precisely defining the domain's concepts, a domain ontology can play several important roles in a scientific hyperbook. In [18] Landow explains that an important hypertext design problem is how to enter the hypertext. A domain ontology provides a good entry point in the hyperbook because the number of concepts in the ontology is generally much smaller

than the number of information fragments. Thus the user can browse the ontology and then go down to the fragments that are connected to the concepts he or she is interested in.

De Bra [19] mentions that the ontology is also a useful tool to personalize the reader's navigation in a hyperbook. If the system memorizes what concepts are known and not known to the user, then it can propose fragments that the user should read.

In our case, the domain ontology will play an essential role in inferring links between information fragments, as we shall see in the next section.

4. The Hyperbook Ontology

The hyperbook ontology is the application ontology of the hyperbook. Its role is to describe the relationship between the fragment repository and the domain ontology; to describe the structures that exist in the fragment repository; and to associate concepts and links to the relevant point of views. The main classes (concepts) of the hyperbook ontology are shown on the diagram of Figure 3.

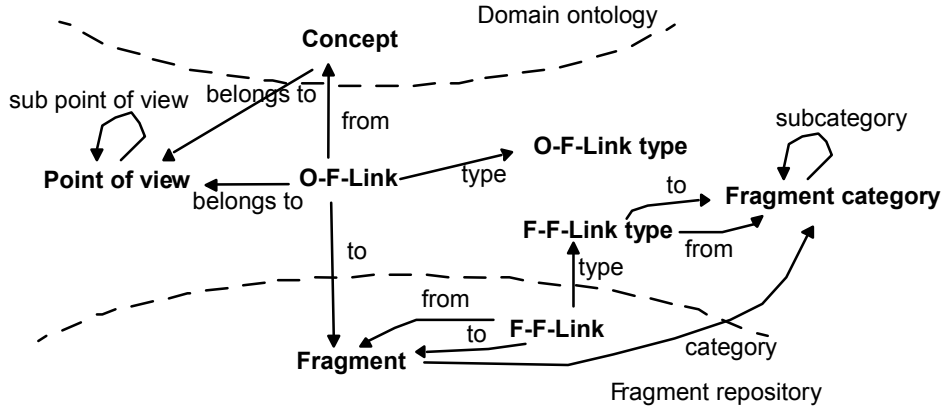

Fig. 3. Classes of the hyperbook ontology

This diagram also shows how the hyperbook ontology is connected to the domain ontology and the fragment repository.

It is important to observe that if the classes are fixed, their instances can be defined specifically for each hyperbook. Hence every hyperbook will have its own fragment categories, link types, point of views, etc.

4.1 Fragment Structures

As mentioned in section 3, the fragment categories and fragment-to-fragment link types (instances of *F-F-Link type* and *Fragment category*) determine the document model. They can reflect structural relationships between fragments (composition links) as well as rhetoric, argumentative, or narrative relationships.

4.2 Links between Fragments and Concepts

The domain ontology plays two roles. On one side it describes the concepts of the domain. On the other side, it serves as a reference to describe the information content of the fragments. By

establishing typed links from fragments to concepts, one can qualify not only what the fragment is about but also what relationship it has with the domain concepts. Typical link types are:

- *instance*, *example*, *illustration*: the fragment describes a particular instance of the referred concept
- *definition*: the fragment contains a textual (or audio, or graphical) definition of the concept
- *property*: the fragment describes a property of the concept
- *reference*, *use*: the fragment refers to the concept (it is necessary to know the concept to understand the fragment)

These links play a crucial role to establish relevant links between fragments and to generate interface documents. The idea is to replace direct linking between fragments (often called horizontal linking) by inferred links that correspond to paths starting from a fragment, going through one or more ontology concepts, and ending on another fragment. Inferred links are preferred to direct links because users (authors) are generally able to establish correctly typed links from the fragments they write to the relevant concepts. When they are asked to link their fragments directly to other fragments they have difficulties finding relevant fragments to link to and deciding on what type of links to establish.

The following figure shows two derived links (1) and (2) obtained by going up to the domain ontology and then down to another fragment.

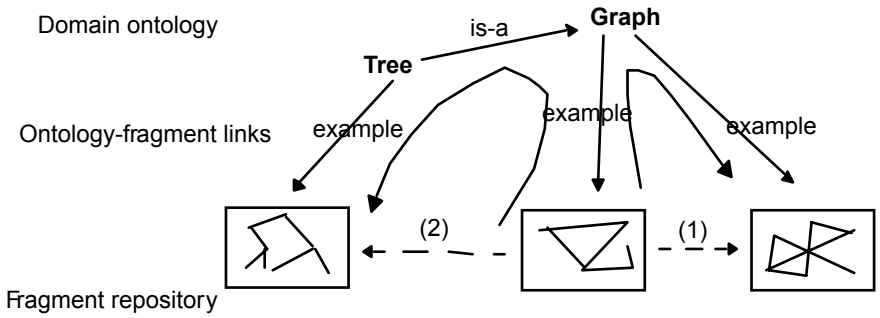

Fig. 4. Link inference through the domain ontology

Since the ontology has a graph structure, it is possible to express link inference with path expressions.

4.2.1 Expressing Inferences by Graph Expressions

An interesting property of the hyperbook model is that semantically meaningful links can be obtained by simple inference rules that consist in path expressions. If we consider the global labelled graph formed by the domain ontology, the fragment collection, and the concept to fragment links, a path expression is an alternated sequence of nodes and arc specification. A node specification is composed of a node type (concept or fragment), a category name (for fragments) or a term (for concepts). An arc specification is composed of a link type, a traversal direction, and a point of view. In addition, each node and arc can be associated to a variable. An instance of a path expression is a path in the hyperbook graph that satisfies all the specifications of the path expression. For instance, link (1) of Figure 4 is an instance of the path expression

fragment ←example – concept – example→ fragment

(start with a fragment, traverse an example link backwards to reach a concept, then traverse an example link to a fragment.). Depending on the link types and fragment categories of the hyperbook, it will be possible to define link inference paths that have a precise and useful meaning for the reader. The following expressions show examples of link inferences that typically occur in the hyperbooks we consider.

fragment F_1 ←example – concept C_1 ←is-a* – concept C_2 – example→ fragment F_2

F_1 is linked to F_2 if F_1 is an example of a concept C_1, and C_1 has a sub concept C_2, which has an example F_2. The ←is-a* – notation represents the traversal of zero, one or more *is-a* taxonomic links in the generic to specific direction. Link (2) on Figure 4 corresponds to this expression.

fragment F_1 ←example/physics – concept C_1 –*→ concept C_2

If F_1 is an example of concept C_1 for the physics point of view, link it to every concept C_2 directly connected to C_1 through any kind of link.

fragment F_1 –uses→ concept C – is-a*→ concept D – property→ fragment F_2

If fragment F_1 refers to concept C, create a link to every fragment that describe properties of any concept D that is more generic than C. If F_1 is an exercise, this will link it to all the properties of the concepts required by the exercise.

An additional property of this link inference method is its robustness with respect to the hyperbook evolution. Since the domain ontology is usually more stable than the hyperbook's fragments, a link to a concept will probably have a longer lifetime than a link to a fragment. Moreover, inferred links are, by definition, always up to date.

4.2.2 A Remark about Instances in the Domain Ontology

There are two ways to represent concept instances. If a fragment describes a concept instance, e.g. the fragment shows a graph, it can be linked to the concept through an instance or example link. However, if an instance plays an important role in the domain it should be represented in the domain ontology, so that it can be referred to by fragments or other concepts. In this case we would have an atomic concept representing the instance, it would be connected to the concept through an instance link and to the fragment through a definition link. For instance, if we consider that the "complete graph with 3 vertices" is a remarkable instance of graph, it must be present in the domain ontology.

4.3 Point of Views

A point of view corresponds either to a category of user (*student*, *researcher*, *journalist*, …) or to the point of view adopted by a user at a given time (corresponding to its reading/writing objectives). For instance, a student could read a hyperbook about *algorithms and data structures* with a software engineering mind set when he or she is developing software. The same person could also read the hyperbook with a theoretical mind set when he or she is studying complexity theory.

The notion of point of view applies both to the concepts of the domain ontology and the ontology-fragment relationships. Thus any concept (in fact any node or link of the ontology graph) an any ontology-fragment link may belong to zero, one, or more point of views.

Point of views must be "reasonably" non-contradictory. If a concept *C* belongs to point of views *P* and *Q* and if an object *o* satisfies the definition of *C* according to *P* (i.e. considering only the parts of *C*'s definition that belong to *P*), it should "in general" satisfy the definition of *C* according to *Q*. In other words, the extensions of C according to *P* and *Q* should be almost equal.

Some point of views can have sub-point of views that are more specialized. For instance, the *computing* point of view could be specialized into *computing theory*, *software engineering*, and *artificial intelligence*. This means, for instance, that an object (concept or link) may belong to the *software engineering* point of view only if it belongs to the *computing* point of view.

5. Interface Model

According to the virtual document approach, the user interface of an hyperbook is made of derived hyperdocuments obtained by assembling selected fragments. The specification of the views must also take into account the point of view adopted by the reader since it may influence the selection and the assemblage of the fragments. Given the richness of the static hyperbook model it is impossible to design a single "optimal" reading and writing interface. This is why the interface model is intended to specify various views on the hyperbook content, allowing the interface designer to adapt the interface to each particular hyperbook. In addition, the interface specification language enables the designer to create simple interfaces that hide the details of the underlying hyperbook model.

5.1 Interface Documents

The hyperbook interface is a hypertext whose nodes are derived documents. The interface specification defines the building rules to apply when creating these documents. The interface specification language is an extension of the Lazy language, which was designed to specify and implement hypertext views on top of relational databases [20]. The most important characteristic of Lazy is its declarative approach. Instead of writing procedural code to program the construction of the hypertext view, one can declare what the selection and assembly criteria are. Another important point is the hypertext model supported by Lazy. In a Lazy specification, hypertext links can be reference links (like HTML links), or inclusion links (the target node is included in the source node at the link location), or expand-in-place links (the target node appears within the source node when the link is clicked). This rich linking model is well adapted to the construction of complex heterogeneous interface documents (in [21] we showed how to create sophisticated hypertext documents to "read" databases with Lazy).

An interface specification consists of a set of node schemas that will be instantiated on demand to produce the actual interface documents. Hence, the interface nodes (the documents the user sees) are instances of nodes schemas. A node schema is comprised of

- a selection part (what fragments and concepts to select)
- a content description (how to arrange the selected objects, which attributes of the selected objects to display)
- a content structure (XML markup tags within the content description)
- reference, inclusion, and expand-in-place links to other node schemas

Example 1. The following node schema selects all the fragments connected to a given concept, its content is made of all the fragment title and contents together with the link type.

```
node examples_of[C]
        <title> "Fragments related to ", C.term </title> ,
        {   <subtitle> L.type, ": ", F.title </subtitle>
            <text> F.content </text>
        }
from Concept C –L→ Fragment F
```

The selection expression is in fact a path expression (shown in 4.2). An instance *example_of*[*x*] of this schema is obtained as follows:

1. select all the fragments *F* connected through a link *L* to concept *x*.
2. generate a <title> element containing the term that denotes concept *x*
3. for selected *L* and *F* generate a <subtitle> element containing *L*.type, the constant ": ", and *F*.title; a <text> element containing *F*.content (the content of *F*)

Example 2. This example illustrates the virtual document idea. It consists in generating a semantically consistent and sequentially readable document by assembling separate fragments. The node schemas shown in Figure 5 (markup tags have been omitted) specify a document that contains

- the textual definition of a concept (found in a fragment linked through a "definition" link)
- the content of all the fragments directly linked to this concept in the "theory" point of view
- links to directly related concepts

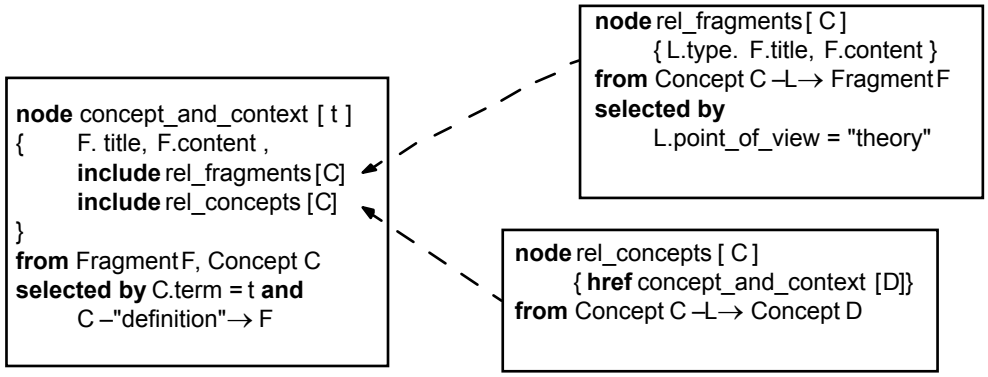

Fig. 5. Definition of a composite node schema through inclusion links

These node schemas show that the interface definition language refers to the fragment repository, the domain ontology and the hyperbook ontology in a uniform way. Hence it facilitates the creation of interface documents to access the hyperbook at any level.

The following figure shows an instance of the *concept_and_content* schema generated on an actual hyperbook (in French).

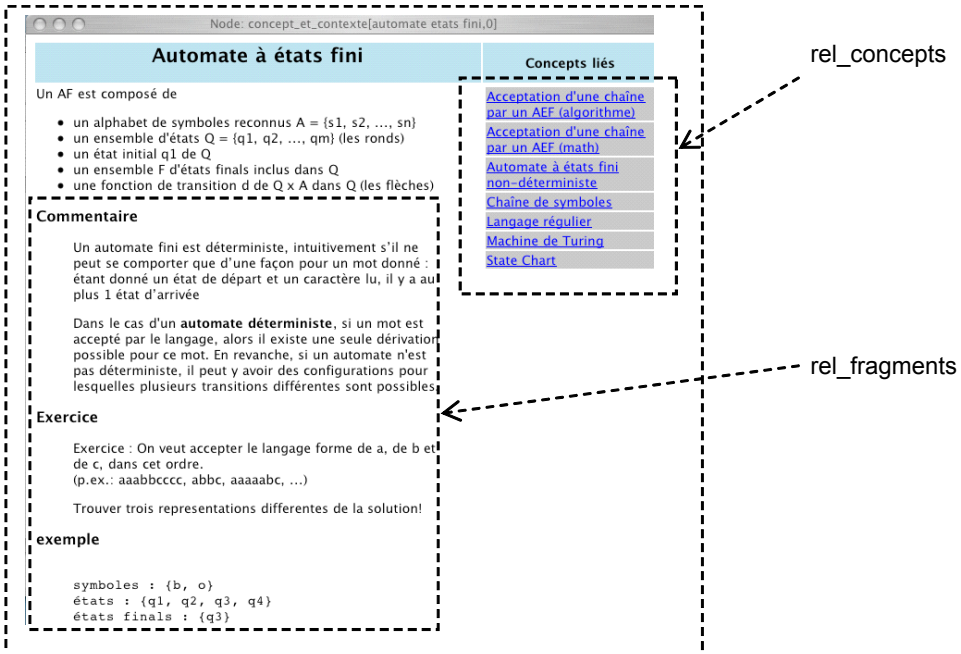

Fig. 6. Instance of a composite node

5.2 Interaction (the Writing Environment)

As mentioned in the introduction, one of our objectives was to support the hyperbook writing activity as well as the active reading of the hyperbook. This implies that the system must enable the users to create new fragments and concepts and to establish links among them.

The interface model supports this active part of the interface through input documents (e.g. forms) and active hypertext links. An active hypertext link is a hypertext link that triggers an action when it is followed. These actions can create, update or delete objects in the virtual hyperbook (an example is given in the next section). The idea is to use the navigational approach as much as possible for updating the information content.

For instance, the following node schema is intended to show the content of a fragment *F*. If the user clicks the "Add Note" active link, this will insert a new fragment *G* into the fragment repository and a new link of type "note" from F. Then it will jump to the node *write_note*[*G*, *F*] that will display an input area to write the note content, update the note fragment and return to fragment *F*.

```
node show_fragment [ F]
{       F.content
        active href write_note[G ] (
                on "Add Note" do new Fragment G; new Link [from: F, to: G, type: "note"]
        )
}
from Fragment F

node write_note[G, F]
        active href show_fragment[F] (
                input t = textarea( ),
                on "Save Note" do update Fragment G [content: t]
        )
```

6. Implementation

During the last two years we have implemented several virtual hyperbook systems based on different versions of this model. We took advantage of the Lazy hypertext view generation system to get straightforward implementations on top of relational databases. The hyperbook model that we have presented here is easy to translate to the relational database model. This results in a relational database schema (a set relational tables) that represents the domain ontology, the fragment repository, and the hyperbook ontology. For instance, the concepts, concepts-to-fragment links, and fragments are represented in the following three table schemes:

```
Concept(id, term, operator, …)
C_F_Link(from, to, type)
Fragment(id, category, content, …)
```

Thus we can use the well-established relational database technology (which handles all the concurrency, security, or query optimization issues) to store the contents of the virtual hyperbook.

Once the hyperbook model has been put in relational form one can define the interface documents with the relational version of the Lazy language. An interface specification is thus a set of Lazy node schemas that refer to the relational tables of the hyperbook. For instance, the following node schema is equivalent of the *concept_and_contect* node schema of Example 2

```
node concept_and_context[ t ]
{       F. title, F.content ,
        include rel_fragments[C.id]
        include rel_concepts[C.id]
}
from Fragment F, Concept C, C_F_Link L
selected by C.term = t and C.type = "definition" and L.from = C.id and L.to = F.id
```

In fact, the node schemas on the hyperbook model can be automatically translated to node schemas on the relational hyperbook schema.

The current system uses the Lazy node server that dynamically generates HTML pages by querying the fragment and link database according to the node schemas. The Lazy system compiles the node schemas and stores their compiled form (a set of SQL statements) into a dictionary. The Lazy node server is a servlet that runs in an HTTP server. When the hyperbook user requests a node instance, the node server executes the compiled form of the corresponding schema (with the appropriate parameters) and send the resulting HTML or XML document to the user's browser. The node server also manages the inclusion and expansion links by recursively executing the appropriate nodes.

7. Conclusion

We have discussed a generic model for representing multipoint of view scientific hyperbooks in the form of virtual hyperdocuments. In this model, the hyperbook ontology plays a crucial role to interconnect the domain ontology and the information fragments, to support the multipoint of view aspect, and to generate views for accessing the hyperbook. By defining fragment categories and link types, a hyperbook designer can adapt the model to a particular domain or task. Then he or she can design a suitable interface by writing document specifications in the form of node schemas.

We have implemented several virtual hyperbook systems that are based on this model. During the last year students of the "formal tools for information systems" course have collaboratively written course notes with such a hyperbook system. The course instructor was in charge of creating the domain ontology and fragments with textual definitions of the concepts. The students' task was to create examples, remarks, exercises, historical notes, etc. to store them into fragments and to correctly link these fragments to the corresponding concepts. The reading/writing interface was comprised of about 30 node schemas. It enabled the user to read, write, and link fragments but also to navigate within the domain ontology, to compare concepts (viewing them side by side), to selectively display fragments related to a concept, etc. The students who used the system were able to produce good quality fragments, probably because they could concentrate on specific and limited tasks and because they did not had to take care of the hyperbook structure. The fragments were correctly connected to relevant concepts, however, the links to other fragments (horizontal links) were of much lower quality. This is why our next experiment will focus on testing the effectiveness of automatic link inference.

We used the same system to create a research oriented hyperbook. In this case we added a group discussion environment by defining suitable fragment categories (topic, message, etc.) and link types (reply, argument, etc.). The ability to manage multiple point of views is particularly useful in a research hyperbook because some concepts not yet well established and several concurrent definitions may co-exist.

In the near future we intend to work on the interface model, with the aim to define new ways of presenting and interacting with virtual documents. We also intend to study the management of digital libraries made of hyperbooks: How to organize and search such a library? How to integrate several hyperbooks?

References

[1] Appelt W., Mambrey P., "Experiences with the BSCW Shared Workspace System as the Backbone of a Virtual Learning Environment for Students". Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications ED-MEDIA 99, Seattle, June 1999.

[2] Garret L. N., Smith K. E., Meyrowitz N., "Intermedia: Issues Strategies and Tactics in the Design of a Hypermedia Document System". CSCW '86 Proceedings, Austin, Texas, Dec 1986.

[3] Bernstein M., "Storyspace". Proceedings of the thirteenth Conference on Hypertext and Hypermedia, 2002, College Park, Maryland, 2002, p. 172-181.

[4] Ackscyn R., McCracken D., Yoder E., "KMS: a distributed hypermedia system for managing knowledge in organizations". Communications of the ACM, vol. 31, July 1988, p. 820-835.

[5] Nanard J., Nanard M., "Should Anchors be Typed Too? An Experminent with MacWeb". In Proc. ACM Hypertext '93 Conference, 1993, p. 51-62.

[6] Halasz F., Schwartz M., "The Dexter hypertext reference model". Communications of the ACM, vol. 37, no 2, 1994, p. 30-39.

[7] Rada R., "Hypertext writing and document reuse: the role of a semantic net", Electronic Publishing, vol. 3, no. 3, 1990, p. 125-140.

[8] Landoni M., Crestani F., Melucci M., "The Visual Book and the Hyper-Textbook: Two Electronic Books One Lesson?" RIAO Conference Proceedings, 2000, p. 247-265.

[9] Fröhlich P., Nejdl W., "A Database-Oriented Approach to the Design of Educational Hyperbooks". Proceedings of the Workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan, 1997, p. 18-22.

[10] Brusilovsky P., Rizzo R., "Map-Based Horizontal Navigation in Educational Hypertext". ACM Hypertext 2002 Conference, College Park, Maryland, 2002.

[11] De Bra P., Calvi L., "Creating Adaptive Hyperdocuments for and on the Web". Proceedings of the AACE WebNet Conference, Toronto, 1997, 149-155.

[12] Brusilovsky, P., "Methods and techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction". In Adaptive Hypertext and Hypermedia, Kluwer, 1998, pp. 1-43.

[13] Wu H., de Kort E., De Bra P., “Design Issues for General-Puropose Adaptive Hypermedia Systems”, ACM Hypertext 2001 Conference, Aarhus, 2001, p. 141-150.
[14] Crampes M., Vercoustre A. M. , Nanard M. , Ranwez S. (Eds), “Acte de l’atelier Documents Virtuels Personnalisables : De la Définition à l'Utilisation”, de la 11ème Conférence Francophone sur l'Interaction Homme-Machine, Montpellier, novembre 1999. URI: http://www.lgi2p.ema.fr/~multimedia/ihm99/ (juillet 2002).
[15] Ranwez S., Crampes M., “Conceptual Documents and Hypertext Documents are two Different Forms of Virtual Document”. Workshop on “Virtual Documents Hypertext Functionality and the Web” of the 8th Intl World-Wide Web Conference, Toronto, 1999.
[16] Garlati S., Crampes M., “Actes du congrès Documents virtuels personnalisables (DVP 2002)”. Brest, 2002.
[17] Iksal, S., “Spécification déclarative et composition sémantique pour des documents virtuels personnalisables”. PhD thesis, ENST Bretagne, Brest, 2002.
[18] Landow G. P., “Hypertext 2.0. The Convergence of Contemporary Critical Theory and Technology”. Johns Hopkins University Press, 1998.
[19] De Bra P., “Adaptive educational hypermedia on the web”. Communications of the ACM, vol. 45, no. 5, May 2002.
[20] Falquet G., Nerima L., Guyot J., “Languages and Tools to Specifiy Hypertext Views on Databases”. In P. Atzeni, A. Mendelzon, G. Mecca (Eds.), “The World Wide Web and Databases”. LNCS vol. 1590, Springer, 1999.
[21] Falquet G., Nerima L., Guyot J., Vanoirbeek C., Rekik Y. A., “Des documents virtuels pour lire les bases de données”. Atelier “Documents Virtuels Personnalisables” de la 11ème Conférence Francophone sur l'Interaction Homme-Machine, Montpellier, 1999.