Journal of Universal Computer Science, vol. 4, no. 4 (1998), 501-521
submitted: 10/1/98, accepted: 23/1/98, appeared: 28/5/98 © Springer Pub. Co.

Building Hypermedia with Objects and Sets

Erik Duval, Koen Hendrikx, Henk Olivié
Departement Computerwetenschappen, K.U.Leuven
Celestijnenlaan 200 A, B-3001 Heverlee, Belgium
{Erik.Duval,Koenh,Olivie}@cs.kuleuven.ac.be

Abstract

In this paper, we present a data model for hypermedia structuring and navigation. The three fundamental building blocks are values, objects and sets, for modeling atomic and aggregate content, as well as navigational structure. In order to formally define operators that enable an end user to access elements of a set, zoom in on or out of a set, we introduce the concepts of user state, topology of a set and anchors.

1 Introduction

This paper deals with a new hypermedia data model. Generally speaking, a *data model* defines the constructs available to structure data. These constructs should make it possible to capture as much as possible the meaning or semantics of the real world - or at least of the part being modeled [**?**].
A *hypermedia data model* then, is a set of constructs that can be used to structure data in a hypermedia fashion. Such a model defines static structural aspects, as well as dynamic operators for creation, access to and deletion of the objects involved.
The most simple hypermedia data model is the *basic node-link paradigm*: information is organized in chunks, called 'nodes', and interrelated by 'links' [**?**]. The simplicity of the basic node-link paradigm can result in poor navigation structures that cause disorientation to exploring users.
The basic challenge in any attempt to enrich the basic node-link paradigm is to propose additional or alternative data structuring facilities, while retaining as much as possible the simplicity and flexibility of the original approach.
In earlier publications, we presented previous versions of our data model [**?**, **?**], as well as a distributed hypermedia data management system based on our data model [**?**].
This text is structured as follows: sections 2, 3 and 4 define the fundamental building blocks of our model (values, objects and sets). Sections 5, 6 and 7 explain how the concepts of user state, topology operator and anchor provide the

framework for defining navigational facilities. Section 8 covers the navigational operators themselves. The current status of our implementation efforts, as well as our plans for the future are presented in section 9. We conclude by comparing our work with related research in section 10.

2 Values

Values are atomic in our model, i.e. they have no (accessible) internal structure. A class D groups all values:

$D = \{x \mid x\ is\ a\ value\}$

D is further divided in n_d subclasses D_i. These classes define subsets of D with similar values over which the same operations are defined:

$\forall i \in [1..n_d] : D_i \subset D$

These subclasses can be divided in subclasses in turn, and so on. The end result is an object-oriented class hierarchy. (We don't elaborate on issues such as shared subclasses with multiple inheritance here. These are treated in any book on object-oriented data modeling.) Figure 1 illustrates such a hierarchy for common domains, based on [?]. In this class hierarchy, a distinction is made

value:	discrete value:	identifier	
		date	
		year	
		month	
		text	
		still image:	GIF
			JPEG
		graphics	
	continuous value:	audio:	WAVE
			AU
		video:	MPEG
			QuickTime
		animation	

Figure 1: Value Class Hierarchy

between discrete and continuous values, i.e. between values whose visualization takes place immediately (at least conceptually, although in practice it may take some time to render for instance a JPEG image) and values whose visualization is a process that takes place in a time span (like for instance a six seconds video clip).

The class of identifiers in figure 1 is somewhat peculiar as these values are intended for reference to particular objects only, and have no further meaning for end users. They will be used internally by a hypermedia system.

3 Objects

Objects aggregate values: we model their internal structure with the *attribute* concept. Each object is modeled as a tuple of attribute values. An object class O_i is defined as a tuple of n_i attribute definitions, each defining the name a_i and domain D_i of the attributes. A domain D_i is a value class from which the appropriate values for an attribute a_i must be drawn.
Thus, the definition of an object class O_i can be formalized as follows:

$$O_i =<< a_1, D_1 >, < a_2, D_2 >, \ldots, < a_{n_i}, D_{n_i} >>$$

An object o from class O_i can then be modeled as a tuple of attribute values:

$$\forall o \in O_i : o =< v_1, v_2, \ldots, v_{n_i} >$$

These attribute values must belong to the appropriate domain:

$$\forall i \in [1, n_i] : v_i \in D_i$$

Further on, we will denote the value of attribute a_i for an object o as follows:

$$a_i(o) = v_i \in D_i$$

(In many object-oriented languages, this is also denoted as $o.a_i$.) We also define a set O that groups all objects:

$$\begin{aligned} D &= \{x \mid x \ is \ an \ object\} \\ &= O_1 \cup O_2 \cup \ldots \cup O_{n_o} \end{aligned}$$

where n_o is the number of object classes.
Just as for values, an object-oriented class hierarchy can be defined for objects, as illustrated by figure 2, which represents a (very) limited class hierarchy for objects on people, publications, courses, organisations and projects. Generally speaking, the class hierarchy for values is more general and will often be fixed for a particular hypermedia system. The object class hierarchy is more application specific and will typically be defined by the developers of a hypermedia application.
As an example, the class definition for a class *Article* is illustrated in figure 3. Like all objects, articles are identified by a unique identifier and have a name, which doesn't need to be unique but is more meaningful to end users. (For articles, the title can typically act as name.) The other attributes are specific to the class of article objects. All attributes are defined over a domain of figure 1.

object:	publication:	article in a journal
		book
		paper in proceedings
		technical report
	person	
	course	
	organisation	
	project	
	journal	

Figure 2: An object class hierarchy

Attribute	Domain	Refers to	Attribute	Domain	Refers to
identifier	identifier		nr	text	
name	text		beginpage	text	
author1	identifier	Person	endpage	text	
author2	identifier	Person	content	text	
...	...	...	note	text	
journal	identifier	Journal	creator	identifier	Person
year	year		created	date	
month	month		last_modifier	identifier	Person
volume	text		last_modified	date	

Figure 3: Definition of the class article

Note that a referential constraint must hold for all attributes (defined over the domain identifier) that refer to other objects, as defined by the 'Refers to' column in figure 3. As an example, the identifier of the first author must refer to an object from class $Person$:

$$\forall a \in Article : (\exists p \in Person : author1(a) = identifier(p))$$

Operations must be supported by all object classes to create, delete, edit, etc. objects. We will not consider authoring aspects in this paper, so that only the operation to *visualize* objects is relevant in the present context. For some objects, visualization may be a process that proceeds over time. This will typically be the case if a continuous value (see figure 1) is involved.
The attentive reader will have noticed that we make abstraction of the possibly distributed nature of a hypermedia system, where the visualize operation would typically involve a client request to obtain the relevant values from a server, the transmission of these values from server to client and their rendering by a client.

4 Sets

Sets model structure superimposed over objects. In this way, sets can be used to define context. This will become clearer in the following sections, when we explain how sets define the navigational functionality.
The elements of a set are either objects or other sets, that in turn contain either objects or other sets, etc. More formally, we define a set S that includes all sets:

$$S = \{x \mid x \ is \ a \ set\}$$

All elements in a set are either sets or objects:

$$\forall x \in S : x \subset (S \cup O)$$

This set concept should not be confused with the domain concept introduced in section 3. A domain is a set (or rather a class) of values that are appropriate for an attribute of an object. The sets we discuss in this section contain objects and other sets; they are used to define navigational contexts.
There are two ways to define which elements belong to a particular set:

- An *intensional* definition is based on search criteria that identify the relevant elements. As an example, a set i can include all publications whose first author is 'T. H. Nelson'.

$$\begin{aligned} i = \{a \mid\ & a \in Article \\ & \wedge(\exists p \in Person : Identifier(p) = author1(a) \\ & \qquad \wedge Name(p) =' T.\ H.\ Nelson')\} \end{aligned}$$

- In an *extensionally* defined set e, objects are 'manually' added to a set by an explicit request. As an example, one can define a set that models a research unit and then manually add:
  - a text t that describes the work of this group,
  - an intensional set i with all publications by members of the research unit,
  - an MPEG video clip v introducing the projects the group participates in.

  The result is a set

  $e = \{t, i, v\}$

Generally speaking, the intensional mechanism is more suited for systematic modeling of large-scale hypermedia applications, because its definition will automatically identify all relevant elements. Moreover, the set will always be up to date: when new relevant objects are defined, they automatically belong to the set, without any 'manual' user intervention.
In fact, intensional and extensional definitions can be combined: search criteria can be defined for a particular set so that all objects that satisfy these criteria are included in the set, as well as some manually added objects that do not satisfy the prescript. As an example, publications about 'digital libraries' can be grouped in a set by defining a prescript that identifies all publications with this string in their title. Relevant publications whose title doesn't include this string can be added manually.
Operations over S enable users to create, delete and edit sets. For (partially) extensional sets, operations to manually insert and remove elements are also supported. As we already mentioned, authoring aspects are not covered by this paper, so that we don't detail these operations here. The interested reader is referred to [**?**]
Set membership does not need to be hierarchical: elements can be shared when they belong to members of different containers. In fact, this is a very important point, as it allows a re-use approach to hypermedia application development [**?**]. Loops, representing recursive memberships, are also allowed, although these should be used with caution, as they can lead to user disorientation.

5 User State

In this section, we introduce the *user state* concept u. The user state consists of two components:

$u =< h, c >$

- The navigational *history* h of the user is defined as follows:

  $h =< s_0, s_1, \ldots, s_n >$ with $s_0, s_1, ..., s_n \in S$

  The intended meaning is that this component is a sequence of sets s_i, that have been current sets in a particular session (see below). This will be clarified below when we explain that s_i can be made current by zooming in on one of the elements of s_{i-1}. At the beginning of a session, h is empty.

- The *current situation* is modeled by a component c in u, defined as follows:

  $c =< s, e, t >$ with $s \in S, e \in (S \cup O), t \in R^+$

  where s models the *current set*, e models the *current element* and t models the *current time*. The current time is the time that has elapsed since the visalization of e started (see also section 8). This is only relevant if e involves a continuous value and can be used to adapt the list of accessible elements while the current element is displayed, as is elaborated in section 6.

  We impose the constraint that only elements of the current set can be current. This means that:

  $e \in s$

The main point is that all relevant contextual information is modeled by the user state u, and that the navigational operators are defined in terms of the effect they have on the user state (see section 8).
Note that the above definition of the user state u can be modified for specific needs. In the case of educational hypermedia applications for instance, the history component h can also include the results of the student on self assessment tests. Or h can also include the sequence of elements that have been accessed (see section 8) for each of the sets s_i that belong to h. The current situation c can also include the user identity, for instance in order to log the actions of a particular user.
In the remainder of this paper, we will stick to the definition of user state given above, which contains all information needed to formally define navigational operations.

6 Topology

6.1 Definition

The *topology operator* θ defines which elements are accessible in a particular situation, i.e. to which elements the access operator can be applied (see below). More formally, θ can be defined as follows:

$\theta(< s, e, t >) =< e_1, e_2, \ldots, e_m >$ where $e, e_1, e_2, \ldots, e_m \in s$

Note that the result of θ for a particular situation $< s, e, t >$ is a sequence of elements, all belonging to the set s. We impose this constraint because the set s models a navigational context. As will be explained in section 8, the access operator, applicable to the elements of $\theta(< s, e, t >)$, should not leave this context. (Switching contexts is achieved with the zoom in, out and up operators.)
Intuitively, the *meaning* of θ is as follows: in a particular situation $c =< s, e, t >$, the user can navigate from the current element e (in the current set s) to the elements $e_1, e_2, \ldots, e_m$ (that also belong to s).
In essence, the topology operator replaces the (much) more conventional link concept that is often seen as the essence of hypermedia data modeling [**?**]. In conventional, graph based hypermedia data models, a link connects source nodes with target nodes. (In most cases, only one source and target node is allowed per link.) This can easily be modeled by our topology operator as well: the source node corresponds to the element e, and the target nodes correspond to the elements $e_1, e_2, \ldots, e_m$.

6.2 Example

As an example, consider figure 4, which illustrates a simple conventional graph based hypermedia structure with three unidirectional links (from e_1 to e_3, from e_1 to e_4 and from e_4 to e_5), two bidirectional links (between e_1 and e_2 and between e_3 and e_4) and a multi-target link (from e_5 to e_2 and e_4).
This navigational structure can be modeled as a set

$s = \{e_1, e_2, e_3, e_4, e_5\}$

with the topology operator defined as follows:

$$\begin{aligned} \forall t \in [0, \infty[: \; & \theta(< s, e_1, t >) =< e_2, e_3, e_4 > \\ & \theta(< s, e_2, t >) =< e_1 > \\ & \theta(< s, e_3, t >) =< e_4 > \\ & \theta(< s, e_4, t >) =< e_3, e_5 > \\ & \theta(< s, e_5, t >) =< e_2, e_4 > \end{aligned}$$

In this example, the topology operator is defined in an extensional way, i.e. by explicitly indicating the identities of the objects that constitute its result. Used in this way, the topology operator adds to the conventional link concept:

- the idea of *context*, modeled by the current set s: This is equivalent with the web concept in Intermedia [**?**], which allows the definition of different link structures over a set of nodes. In other words: the context (i.e. the current web in Intermedia, or the current set in our model) defines the

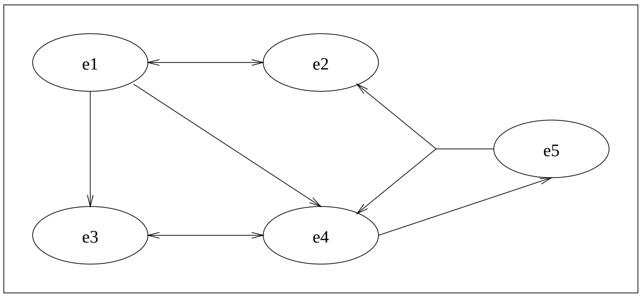

Figure 4: A Conventional Hypermedia Example

navigational structure (i.e. the links in Intermedia or the result of the toplogy operator in our model).

In hypermedia systems based on more limited data models (with the World-Wide Web as the most noteworthy representant [?]), there is only one context, which basically means that the result of the topology operator depends on the current element e only.

- the notion of *time* that has elapsed since the visualization of the current element e started: This concept can be used to make elements accessible only for a particular time span during the visualization of the element e. As an example, if the topology operator is defined as follows:

  $\forall t \in [2,6] : \theta(< s, e, t >) =< e_1 >$

  then, during play-out of e (this can be a video clip), the element e_1 (for instance a biography of an actor) is only accessible from two until six seconds after the clip started (maybe because that is when the actor appears in the clip).

6.3 Intensional Definition

The topology operator is an especially powerful concept when it is defined in an intensional way.
As a simple example, we can define a set $Pubs$ that holds all articles, persons and journals. For this set, we can define the topology operator intensionally in the following way:

$$
\begin{array}{l}
\forall t \in [0,\infty[, \forall a \in Articles : \theta(< Pubs, a, t >) =< p_1, p_2, \ldots, p_k, j > \\
\quad \text{where} \quad p_1, p_2, \ldots, p_k \in Persons \\
\quad \text{and} \quad j \in Journals \\
\quad \text{and} \quad author_1(a) = p_1 \\
\quad \text{and} \quad author_2(a) = p_2
\end{array}
$$

and $\ldots$
and $author_k(a) = p_k$
and $journal(a) = j$

This means that, if $Pubs$ is the current set, and the current element is an article, then elements that correspond with all persons that contributed to the article as an author, as well as the journal it was published in are accessible. In a similar way, all articles a person contributed to can be made accessible automatically when the current element corresponds to this person.
It is important to note that the topology operator thus completely removes the need for manual linking of elements. This is very important because it greatly alleviates the *maintenance* problem:

- When an element is *deleted*, it simply no longer satisfies the search criteria that the intensional definition of the topology operator is based on. If the topology operator is defined in an extensional way, then the element will automatically be removed from its result. This is not further detailed here, because authoring aspects are outside the scope of this paper. The interested reader is referred to [?]. The net result is that no dangling links can arise.
- Moreover, when a new element is *inserted*, it will automatically be identified by the topology operator if it is relevant. Referring to the example mentioned above, when a new article is created, it is automatically linked to available information about the authors and journal, and it will itself automatically be accessible from elements that correspond to the authors and journal.

7 Anchors

7.1 Definition

The topology operator defined in the previous section corresponds to the notion of links in conventional hypermedia data models. Also needed is a concept that defines 'active spots', i.e. a construct that enables an end user to access one of the accessible elements. For this purpose, we generalize the conventional anchor concept.

- If the current element e is an *object* from class O_i, then the anchor operator α defines the anchors that can be activated to access the accessible elements $< e_1, e_2, \ldots, e_m >$, as defined by the topology operator θ for a particular situation (see section 6).

  Formally, α can be defined as follows:

if $e \in O_i =<< a_1, D_1 >, < a_2, D_2 >, \ldots, < a_{n_i}, D_{n_i} >>$
and $\theta(< s, e, t >) =< e_1, e_2, \ldots, e_m >$
then $\alpha(< s, e, t >) =< y_1, y_2, \ldots, y_m >$
where $y_1, y_2, \ldots, y_n \in \{a_1, a_2, \ldots, a_{n_i}\}$

This means that, for every accessible element e_i (as defined by the topology operator θ), the anchor operator α defines an attribute y_i from the class O_i that the current element e belongs to.

- If the current element e is a *set*, then there are no attributes that can be used to define the anchors. In that case:

if $e \in S$
and $\theta(< s, e, t >) =< e_1, e_2, \ldots, e_m >$
then $\alpha(< s, e, t >) =< name(e_1), name(e_2), \ldots, name(e_m) >$

This means that, in this case, the names of the accessible elements act as anchors.

Intuitively, the meaning of α is as follows: if the the current element e is an object from class O_i, then the user can navigate to the elements $e_1, e_2, \ldots, e_m$. This is defined by the topology operator $\theta(s, e, t)$. The values of e for the attributes $y_1, y_2, \ldots, y_m$ act as anchors: in other words, these attribute values are displayed in such a way that they can be activated by the end user to access the corresponding elements $e_1, e_2, \ldots, e_m$. If the relevant attribute values are object identifiers, then the names of the corresponding object will be displayed, rather than the value of the identifier itself.
If the current element e is a set, i.e. if $e \in S$, then the names of the accessible objects are displayed in such a way that they can be activated by the end user in order to access them.

7.2 Example

As an example, an article (see figure 3) can be defined as in figure 5 (see also [?]).
Suppose that this article is the current element e, and that the topology operator θ and the anchor operator α are defined as follows for a set s that e belongs to:

$\theta(s, e, t) =< 2345, 3456 >$
$\alpha(s, e, t) =< author1, journal >$

In this case, when the article e is visualized, the name of object 2345 will be shown as the value of the attribute $author1$, and this name will be active, i.e. when a user activates it (for instance by clicking on it), then object 2345 will be accessed (see below). A similar situation exists with respect to object 3456 and the attribute $journal$.

Attribute	Value
identifier	1234
name	'Hypertext: An introduction and survey'
author1	2345
journal	3456
year	1987
month	September
volume	'2'
nr	'9'
beginpage	'17'
endpage	'41'
...	...

Figure 5: Definition of an article instance

Note that the value of one attribute a_i can act as anchor for several objects. In that case, a_i appears several times in the y_i of $\alpha(s) =< y_1, y_2, \ldots, y_n >$. This corresponds to a multi-target link in conventional hypermedia models.
Note also that not all attributes that refer to other objects must be made active. For the example just mentioned, the topology and anchor operators can also be defined as follows:

$\theta(s, e, t) =< 2345 >$
$\alpha(s, e, t) =< author1 >$

In this case, when e is visualized, the name of object 3456 is still shown as value for the attribute journal, but it is no longer active, and the corresponding object is no longer accessible.

8 Navigation

8.1 Introduction

In this section, we define operators that enable an end user to navigate the structure defined over objects and sets using the concepts of the previous sections. The access operator (section 8.2) can be used to change the current element e within the current situation $c =< s, e, t >$, leaving the history component $h =< s_0, s_1, \ldots, s_n >$ of the user state $s =< h, c >$ unaffected. The zoom in, out and up operators (sections 8.3, 8.4 and 8.5) enable an end user to change the current context s as well as the history component h.

8.2 Access

The access operator $\mathcal{A}$ can only be applied to an accessible element of the current set. The result is that that element becomes the current one, and that it is visualized. This can be formalized as follows:

$$\mathcal{A} : e', u \rightarrow u' \text{ where } u =< h, c > \\ u' =< h, c' > \\ c =< s, e, t > \\ c' =< s, e', 0 > \\ e' \in \theta(c)$$

Note that the history component h of the user state u remains unaffected. In other words: we do not record the navigation of a user within a given context (modeled by the set s of the situation c) in the history component h. As mentioned in section 5, the definition of the history component can be modified in order to record this information, leading to more complex as well as more generalized definitions of the navigational operations.
Also note that e' not only becomes the new current element, but that, as a side effect, the visualization of element e' is started. This implies that the current time is initialized as zero and will start 'running' from the moment that $\mathcal{A}$ is applied.
The role of the access operator $\mathcal{A}$ is similar to that of following a link in the basic node link paradigm: it enables an end user to navigate between information items (elements rather than nodes).
It is important to note that very few assumptions are made about the content of e' in the definition of $\mathcal{A}$. The sole requirement is that, in response to the access operator $\mathcal{A}$, it must be possible to visualize e'. What such a visualization should entail (rendering a graphic, displaying a video clip, etc.) is completely open. This can also be a more complicated series of events (like an interactive question and answer application). In this way, our model adheres to an open hypermedia approach, as content created with external applications can be incorporated in hypermedia structures [**?**].

8.3 Zoom In

Applying the zoom in operator $\mathcal{I}$ is only possible if the current element e is a non-empty set. The effect is that e becomes the current set, and that the access operator $\mathcal{A}$ is applied to the head $\gamma(e)$ of e. For all non-empty sets s, the head is a particular element of s:

$$\forall s \in S : s \neq \phi \Rightarrow \gamma(s) \in s$$

This concept is introduced to define what should be visualized when s is zoomed in upon. The head thus functions as a sort of 'entry point' for sets.
More formally, the zoom in operator can be defined as follows:

$$\begin{aligned}\mathcal{I} : u \rightarrow u' \text{ where } & u =< h, c > \\ & u' =< h, c' > \\ & h =< s_0, s_1, \ldots, s_n > \\ & c =< s, e, t > \\ & e \in S \\ & e \neq \phi \\ & h' =< s_0, s_1, \ldots, s_n, s > \\ & c' =< e, \gamma(e), 0 >\end{aligned}$$

This means that the history component is extended by appending the previously current set s to it. The previously current element e must be a non-empty set and now becomes the new current set. Its head is accessed and thus becomes the current element, while, as a side effect, its visualization is started and the current time is initialized.

8.4 Zoom Out

The opposite effect can be achieved by applying the zoom out operator $\mathcal{O}$: basically, the situation before the last zoom in becomes the current situation again. In a formal way, the zoom out operator $\mathcal{O}$ can be defined as follows:

$$\begin{aligned}\mathcal{O} : u \rightarrow u' \text{ where } & u =< h, c > \\ & u' =< h', c' > \\ & h =< s_0, s_1, \ldots, s_n > \\ & c =< s, e, t > \\ & h' =< s_0, s_1, \ldots, s_{n-1} > \\ & c' =< s_n, s, 0 >\end{aligned}$$

The new current element s is the set that was current before the zoom out operator $\mathcal{O}$ was applied. This element is accessed, and, as a side effect, it is visualized and the current time is re-initialized. The new current set s_n is the set that was previously added to the history h by zooming in on s_{n-1}. This set is now removed from the history h'. (Note that, in this respect, the history doesn't model a complete navigational path, but only the part of it that lead to the current situation. If the user backtracks by zooming out, then information about the retraced steps is discarded.)
If h is empty ($h =<>$), then the zoom out operator $\mathcal{O}$ is inapplicable.

8.5 Zoom Up

In order to enable an end user to discover to which sets an element belongs, we define an additional operator. The zoom up operator $\mathcal{U}$ enables a user to make current a set s' that contains the current element e. More formally:

$$\mathcal{U} : u \rightarrow u' \text{ where } u =< h, c >$$

$$u' =< h', c' >$$
$$h =< s_0, s_1, \ldots, s_n >$$
$$c =< s, e, t >$$
$$h' =<>$$
$$c' =< s', e, t >$$
$$e \in s'$$

This means that the current element remains the same (i.e. e) in the new situation c'. The current time t also remains the same. This menas that, at least conceptually, the zoom up operator takes effect instantaneously. The current set changes from s to a set s' that the current element e belongs to. Finally, the history component h' is emptied. (More on that in a note below.)
The zoom up operator $\mathcal{U}$ thus enables a user to explore all contexts that a particular element belongs to. The rationale for this operator is that, if a particular element is relevant, then it often makes sense for the user to explore other contexts it appears in as well. Note that the operator is indeterministic if the current element e belongs to more than one set. (If it belongs to only one set, then the effect of the zoom up operator is identical with that of the zoom out operator, except with respect to the history component h.) In a practical implementation, the user can be presented a list of candidates for s', i.e. a list of all sets that contain the current element e. By choosing one of these elements, the result of $\mathcal{U}$ can be determined.
Note: One could question whether it is appropriate to start a new sequence $h' =<>$, when $\mathcal{U}$ is applied. An alternative would be to add the previously current set s to the sequence, thus obtaining $< s_0, s_1, \ldots, s_n, s >$. This option is not adopted in our model, because the semantics of the sequence $< s_0, s_1, \ldots, s_n >$ is meant to imply that one can navigate from s_n to the current set s by applying the zoom in operator $\mathcal{I}$. If s is added to the sequence, this property no longer holds: one cannot zoom in on s to make s' current. Another alternative would be to create a new sequence $(s'_1, ..., s'_m)$ that would satisfy this semantics. In other words: one could define a new sequence from the root set to the current set. There are two reasons why we don't follow this approach:

- such a sequence is not necessarily unique,
- a user would be able to zoom out and make current a set s'_m he hasn't visited yet. This would be rather confusing.

Hence the choice in our model to start a new sequence altogether when the zoom up operator is applied.

8.6 Example

In order to clarify the meaning of these operators, consider the following example:

$group = members, projects, publications, courses, intro$
$\gamma(group) = intro$

A set $group$ is defined that contains information about a research unit. Four elements of this set are sets themselves with information about the different members, projects the research unit participates in, publications by members of the unit and courses that are taught by these members. A fifth element is an object with a general introduction about the group and its activities. (This can include a video or text value.) The last element is the head of the group.
Suppose that a user starts his session by zooming in on this group. In that case, the user state u contains an empty history component $h = <>$ and a current situation $c =< group, intro, 0 >$. If the topology operator is defined as follows:

$$\theta(< group, intro, t >) =< members, projects, publications, courses >$$

then the user can access an arbitrary other member. Suppose he accesses members, then the current situation changes to $< group, members, 0 >$. If the user now zooms in, then the user state u contains a modified history component $h =< group >$ and the current situation becomes $c =< members, HO, 0 >$ if the set $members$ and its head are defined as follows:

$$\begin{aligned} members &= HO, KC, ED, KH, RVD, BV \\ \gamma(members) &= HO \end{aligned}$$

where the elements of $members$ are indicated by their initials. If among the elements of HO is a set on his publications, then the user can zoom in on this set and access one of his articles. This article can belong to sets with publications of all the members that contributed to it. The user will find this out if he applies the zoom up operator when this article is the current element.

8.7 Important Features

There are two important features to navigation in our model:

- First of all, two orthogonal dimensions of navigation are supported over sets that group elements:
  - users can zoom in on and out of sets, in order to navigate over the 'belongs to' relationship between sets;
  - by accessing different elements of the same set, users navigate over the 'belongs to the same set' relationship between elements.

  Generally speaking, the first dimension is more concerned with macrostructure: each time an end user zooms in on or out of a set, the navigational micro-context (i.e. the elements of the current set) changes.

  This two-level structure can help to prevent the users from becoming disoriented, as it enables an author to make a distinction between locally related units of information and more global relationships between one context and other, related contexts.

- All *operators are addressed to one particular element* and do not affect the internal structure of other elements. (This also holds for authoring operators [?].) This leads to *self-containment* of sets: a set is a self-containing module that can be edited and navigated independently of others. When a set s is deleted, all interrelationships between members of s cease to exist as well. However, the members of s survive as independent entities; only their interrelationships in the context of s are destroyed. This self-containment is particularly important with regards to re-use, i.e. when an element is shared by several sets [?].

9 Current Status and Plans for the Future

9.1 Current Status

We have developed an environment that supports the data model just presented [?]. This environment is called HOME (Hypermedia Object Management Environment). Technologically, it is based on a commercially available DataBase Management System (in casu Oracle). All information stored in HOME is accessible through the World-Wide Web, using a gateway mechanism.
Storage of objects and values is strictly separated from the storage of data to represent the topology operator θ and the anchor operator α. In this way, different data models can be supported over the same set of basic values and objects.
As an illustration, figure 6 demonstrates how the data needed for θ and α can be modeled in a relational database. A tuple $< s, i, c, a, i', c', t_b, t_e >$ in this relation represents the fact that if the current element is the element with identifier i from object class c, and the current set is identified by s, then the element identified by i', from class c' is accessible when the current time belongs to the interval $[t_b, t_e]$. The anchor is the value of the current element for the attribute a of class c.

Set_Identifier
Current_Element_Identifier
Current_Element_Class
Current_Element_Attribute
Accessible_Element_Identifier
Accessible_Element_Class
Begin_Time
End_Time

Figure 6: An implementation of the topology operator

9.2 Plans for the Future

We are currently working on a number of further developments and applications of the data model presented above:

- In order to further validate the model, we are developing a number of applications, especially in the field of educational hypermedia, including support material for courses on multimedia modeling and programming, as well as an introductory course on computer science concepts.
- We are investigating further extensions to the anchor concept presented above, so that we can for instance apply search operations on the value of an attribute to define the anchor. As an example, this approach would make it possible to identify all occurrences of a search pattern in a text value as anchors.
- Especially with respect to consistent authoring of large-scale hypermedia applications, the development of a suitable design method becomes very important. The model presented above provides a good fundament for such a method, because the operators can be defined intensionally. Our first results in this area can be found in [?],

10 Related Research

The model just presented is mainly influenced by early work in this area [?] and the HM data model [?] [?] [?].

10.1 Node-Link Paradigm

An object in our model corresponds to a conventional hypermedia node. Just like some systems support an elaborate taxonomy of node types, our model includes an object-oriented class hierarchy for objects and values. The function of a set in our model is somewhat similar to that of a composite node in more advanced node-link models [?].
In the Dexter model, an attempt to formalize the common features of node-link hypermedia models, data structuring is modeled by the storage layer [?]. (The run-time layer is more concerned with end user interaction, and the within-component layer deals with the internal structure of objects.) Atomic components in Dexter correspond to objects in our model. Composite components loosely correspond to our notion of sets. However, in Dexter, the 'is part of' hierarchy must be a directed acyclic graph, whereas our model allows loops. The Dexter model can be extended with set-based data structuring [?]. A slot in this extended model corresponds with an object in our model. Nodes and sets both correspond with our notion of sets.

10.2 Gopher

In Gopher, information is structured in collections (visualized by most Gopher clients as menu's) [?]. Gopher collections contain either subcollections or documents; this is equivalent to sets that contain other sets or objects. In Gopher, searches can be launched when a collection item is activated, in order to identify the collection elements. This facility corresponds to intensional sets in our model.

All collection items in the current Gopher collection are always accessible. In other words, the topology operator always delivers all elements (but the current one) from the current set. Selecting an item in a Gopher collection corresponds to accessing an element in our model if the item is an object. If it is a subset, then activating the item corresponds with the zoom in operator of our model. Zooming out corresponds with going up one level in the collection hierarchy. There is no Gopher equivalent for accessing a set or the up operator.

10.3 Hyper-G

In *Hyper-G* (now commercialized as Hyper-Wave), documents can be grouped in collections, that can in turn belong to other collections [?]. Hyper-G documents can be described by attribute-value pairs, just as attributes can be defined for specific object classes in figure 3. A Hyper-G collection is similar to our set concept, but hyperlinks can refer to documents across collection boundaries, which is not allowed for the topology operator in our model. Only extensional sets are supported in Hyper-G: although it is possible to carry out a search (which, as mentioned above, correspond to prescripts for intensional sets), searches cannot have persistent status.

In Hyper-G, all elements of a collection are always accessible. Hence, the result of the topology operator always includes all elements from the current set but the current element. An exception is a specific kind of collection, called a sequence, where only the next and previous element of an ordered set are accessible. The access, zoom in and zoom out operators, as well as the head concept, have straightforward equivalents in Hyper-G. However, when an object is the head of one collection in Hyper-G, then it is head in all collections it belongs to. Our model is more flexible, as head status is specific to an object's membership of a set. The up operator is not supported as such in Hyper-G, but it can be simulated, using the facility to generate 'overview maps' of the collection structure.

Time-dependent behaviour of objects is defined in Hyper-G using SyncScript [?]. This is a more procedural approach than our topology operator, which is more declarative in nature. SyncScript is also concerned with other issues besides time-dependent link structures, like for instance display parameters.

10.4 HM Model

Sets in our model correspond to so-called S-collections in the HM model [?]. Objects correspond to void S-collections.
In the HM model, topology operators are predefined. In an *envelope*, all elements are always accessible, whatever the current element is. In a *folder*, elements are ordered, for instance as $e_1, e_2, \ldots, e_n$. At each moment, the elements immediately 'previous' and 'next' to the current element are accessible. In a *menu*, if the head is current, then all other elements are accessible. If not, then only the head is accessible. This list of predefined, generic topology operators is summarized below:

$$
\begin{array}{lll}
envelope(s) & \Rightarrow \Theta(e,s) = s \setminus \{e\} & \\
folder(s) & \Rightarrow \Theta(e_i,s) = \{e_{i-1}, e_{i+1}\} & (1 < i < n) \\
 & \quad \Theta(e_1,s) = \{e_2\} & \\
 & \quad \Theta(e_n,s) = \{e_{n-1}\} & \\
menu(s) & \Rightarrow \Theta(\gamma(s),s) = s \setminus \{\gamma(s)\} & \\
 & \quad \Theta(x,s) = \gamma(s) \text{ where } x \neq \gamma(s) \wedge x \in s &
\end{array}
$$

The head concept and the navigational operators in the HM model are similar to the equivalent notions in our model.

Conclusion

In this paper, we have proposed a hypermedia model that relies on values, objects and sets as basic building blocks. Structure can be defined over these elements with a topology operator. An anchor operator defines which components of the current element can be activated to access the currently accessible elements. Navigation proceeds by accessing elements, zooming in on sets, or, conversely, zooming up or out from them. These operators are all defined in a formal way, using the concept of user state. An important point is that both set membership, as well as the topology and anchor operators can be defined intensionally, which greatly simplifies maintenance of large-scale hypermedia applications.

Biographies

Erik Duval graduated as an engineer in computer science at the Katholieke Universiteit Leuven in 1989. He obtained his PhD at the same university in 1995. Currently, he is a post-doctoral scientific cooperator in the research unit on hypermedia and databases at the computer science department of the K.U.Leuven.
Professor *Henk Olivié* holds 'licentiaat' degrees in mathematics from the RijksUniversiteit Gent and in computer science from the K.U.Leuven. He obtained

his PhD from the University of Antwerpen. Since 1989, he heads the research unit on hypermedia and databases at the same university.
Koen Hendrikx obtained his degree as engineer in computer science at the K.U.-Leuven in 1993. He then joined the research unit where he is preparing a PhD. on the subject of design methods for hypermedia applications.
The research interest of the unit on hypermedia and databases focuses on distributed hypermedia systems, hypermedia data models and development methods, and the application of hypermedia technology for open and flexible learning.