Object-Oriented Hypertext Design : Authoring for Reuse

K. Hendrikx, H. Olivie, E. Duval
Department of Computer Science,
Katholieke Universiteit Leuven, Belgium
{koenh,olivie,Erik.Duval}@cs.kuleuven.ac.be

Abstract

In this paper we present a new approach for the development of hypertext information systems that avoids a number of traditional hypermedia authoring problems, and facilitates the maintenance of links and the reuse of information resources.

The model is set-based, and integrates features from object-oriented software design models. This helps authors to make a comprehensive (and comprehensible) design before starting an implementation.

We also briefly describe an authoring system that implements this model, and which uses an existing relational database system to store documents, links and meta-information.

Because of the current momentum of the WWW, it is unlikely that it will soon be replaced by any of the other systems that try to alleviate the stated problems, like Hyper-G [3], Xanadu [4] or Microcosm [5]. A realistic approach would be, in our mind, to develop design and authoring tools that allow users to design and create information webs or information systems that use the WWW as an implementation platform, but built with higher level primitives.

We have developed an object-oriented hypermedia model (OOHM) that defines such primitives, and an object-oriented hypermedia system (OOHS) that supports the model and translates these primitives to e.g. an implementation on the WWW.

1. Introduction

As the world-Wide Web (WWW) becomes increasingly important as a system to implement all kinds of information systems, like Campus-Wide Information Systems, problems with regard to the efficient design and maintenance of large volumes of hypertext become more urgent. It is widely agreed that the WWW has severe problems when it comes to maintenance of hypertext [1]. Another problem is the implementation of search tools which allow more complex queries than just scanning for keywords. The simplicity and openness of the WWW, its most prominent advantage, is also a major cause of its problems. Static, embedded and untyped links make hypertext easy to author, but hard to maintain. Apart from that, the used model makes hypertext – or rather it allows hypertext to be – very difficult to understand [2]. Also, the open-ended CGI (Common Gateway Interface) mechanism used to implement search mechanisms encourages everyone to write his own search engines and interfaces, making it harder for non-experienced users to find their way.

2. A hypermedia modeling principle

When authoring large amounts of hypermedia, one needs both a modeling principle and a design method. A modeling principle is a set of constructs that in some way specify the behavior of (a part of) the real world. An instance of such a model is fairly stable and is independent of any specific application. The object-oriented modeling principle is a very good model since it produces models that are very application-independent.

Once a model of reality is available, the model can be translated into a number of applications, preferably with the aid of automated tools. For general software systems, object-oriented programming environments are available. Entity-relationship models can be translated automatically to a relational schema [6] to produce a database implementation. As the application depends on the model, the production of the model is the most critical phase.

There exist modeling principles specifically directed to the development of hypermedia systems, like RMM [7] and HDM [8] which are based on the ER model, and OOHDH [9] which is based on an object oriented-model. When using these models, a third development

phase is required : the navigational design. This navigational design precedes the actual application development, but still is more or less independent of the application. It defines consistent ways in which objects (or entities) are visualized and can be navigated. Common to these navigational design phases is also that they provide for different 'viewpoints' : the visualized and navigational aspect of an object may differ depending on the context from which it is viewed.

We aim to define a hypermedia model that is powerful enough to allow the (almost) automatic implementation of a hypermedia information system from an instance of the model. Our model differs from the existing models in that it is specifically aimed at supporting the authoring of comprehensible hypertext, instead of making existing databases accessible through hypertext.

In the remainder of this paper we propose OOHM, an object-oriented modeling principle for hypermedia databases, and an implementation called OOHS, a system that converts instances of OOHM to a hypermedia implementation.

2.1. The hypermedia model : OOHM

The basic features of OOHM are : (1) it is set-based, which mainly serves to overcome what is sometimes called 'the static link deficiency', and at the same time helps to avoid disorientation, and (2) it is object-oriented. Objects are classified in an object-oriented hierarchy and with each class a visualization script is associated. This provides a fairly general way to define how documents are to be visualized – within a given context – and how available links are visualized.

We will explain these features in more detail in the remainder of this section.

2.1.1. The set-based nature. Embedded links in classic node-link hypertext are very similar to goto statements in programming. As goto statements, they are easy to understand and implement, but make maintenance a lot more difficult. Structured programming [10] largely eliminated the need for gotos in conventional programming. Moreover, structured programming was better suited for the implementation of structured top-down designs. In a similar way, the OOHS eliminates the need for embedded links. Instead, links are generated automatically from a higher level specification, like machine code is generated from a high level programming language. This facilitates the

reuse of existing components, since links to and from those components are of a more abstract nature and can be updated automatically [11] [12].

The model used is the following. The hypermedia database consists of sets and documents, both referred to as *objects*. All objects belong to one *class*, as will be explained below. A set can contain any number of documents and/or other sets, much like a directory can contain other directories and files. An object may be contained in more than one set. Navigation is limited to (a) *zooming in* to members of the current set or members thereof, (b) *zooming* (any number of levels) *out*, or (c) *panning* to siblings in the current set. This limitation has the advantage that it avoids disorientation of the user, while still being relatively flexible.

The model we have described up to now is very similar to those of Gopher [13] and Hyper-G [14]. As the OOHM, these claim to be easier to navigate as they avoid disorientation. Because links are not stored inside documents but in a separate database they are also easier to maintain. This also has the following advantages [15] :

- links can be attached to read-only documents (e.g. stored on CD-ROM);
- links can be followed backwards;
- local maps of the link structure can easily be computed;
- it is relatively easy to impose consistency constraints, e.g. when deleting a document, and these constraints can readily be implemented by the underlying database system.

Set based hypermedia systems need some rule to define what is to be visualized when a user accesses a set. Gopher simply visualizes links to all member objects in the form of a menu. Hyper-G has the concept of a *head-element*. When a Hyper-G collection is accessed, its head is visualized along with links to its other members. The head may itself be a collection, in which case the visualization task is passed on to its head in turn. OOHM uses the same mechanism, with the slight simplification that the first element of a set is always the head. This means that when a set is accessed, its first element will be visualized, along with links to the other elements.

The most important difference between OOHM and Gopher or Hyper-G is that OOHM allows an author to be much more specific in specifying how objects are to be visualised, by using the mechanism explained in the next subsection.

2.1.2. Typing of objects and template scripts. An important additional feature of OOHM is the classification of objects. Each class of objects has a number of attributes and methods associated with it. These methods are :

- **Access** : generate a hypertext page (e.g. HTML) visualizing the object and its navigational structure
- **Map** : generate a local map of the structure of the object and its surroundings
- **Print** : generate a printable form of the object and the objects contained within it.
- **GenForm** : produce a form with a field for each of the object's attributes and present it to the user;

These four procedures are very similar. They produce some output based on the object's data fields and a visualization script. Additional methods are :

- **RecForm** : update the database according to the user's input in a previously generated form;
- **Insert** : insert a member into a set;
- **Delete** : delete a member from a set;
- **Query** : if the set is of type 'query' (or any of its descendants) then assume its content is an SQL (Structured Query Language) query, execute this query and use the result to define membership of the set.

The first four procedures are specified by each class through an SGML [16] visualization script. We will illustrate this with an example.

```

1 <head><title><include unit.title>:references</title>
2 </head><body>
3 <center><table border><tr>
4 <th><h2><include unit.title>:references</h2><tr>
5 <th><link sibling=what icon=w.gif alticon=wt2.gif>
6 <th><link sibling=where icon=wr.gif alticon=wr2.gif>
7 <th><link sibling=example icon=ex.gif alticon=ex2.gif>
8 <th><link sibling=question icon=q.gif alticon=q2.gif>
9 <th><img src=refs.gif>
10 <th><link dir=up icon=up.gif></table>
11 </center>
12 <include body>
13 <dl><foreach member=reference><dt>
14 </foreach member=person>
15 <include attrib=name sep=","></foreach>
16 <dd><i><link this><include attrib=title></link></i>
17 <include attrib=year prefix=",">
18 </foreach></dl><hr>
19 <foreach sibling=subunit.item prefix="<ul>"
20 postfix="</ul>">
21 <li><link member=references><include title>
22 </foreach></body>

```

Figure 1. Visualization script for the 'references' class

Figure 1 shows an access visualization script for the class 'references'. The script produces a list of bibliographical references in HTML. The script does the following : for each member (line 13-18) of type *reference* do the following : for the members of type *person* (line 14-15) (the authors) list the names, separated with a comma. Next the title (line 16) is printed in italic and the year of publication (line 17).

The title serves as a link to the actual reference. When that link is activated the access script for 'reference' or one of its derived classes (like article or journal) is used to display the reference in more detail, with links to the authors etc.

Lines 5 to 10 produce a number of links to sibling objects in the form of icons. The *icon*-parameter indicates the icon to be used, while the *alticon*-parameter is used to visualize an unavailable link. (See also Figure 5).

This system is relatively general, since it is possible to generate other SGML document types, like for instance HyTime [17] or MHEG [18].

2.2. Advantages of OOHM

Apart from the advantages regarding navigation and reuse a few other advantages can be attributed to this model.

2.2.1. Different viewpoints. OOHM originated from research into educational hypertext [19]. In our view, the most important added value of hypertext is its ability to offer different viewpoints, different ways of looking at the same information. As an object can be a member of more than one set, each of those sets can provide an alternate view to the information. This means that some objects represent the base information, while others represent views of that information.

The zoom-in feature may be used to navigate from a less detailed viewpoint to a more detailed one, as was shown in the preceding example of bibliographic references.

2.2.2. Meta-information and querying. Apart from plain text and other media objects the hypermedia database includes extra information. This meta-information provides additional information about the function and content of each information element and can be used to formulate and execute more sophisticated queries than just keyword scans.

This *meta*-information can be of three forms :

- *link information*, specifying which document or set is part of which set, and the order in which objects occur within a given set;
- *type information*, specifying that a certain document or set is of a certain class;
- *attribute-value pairs*, specifying additional alphanumeric information about a certain document or set.

An example of how we are using alphanumeric attributes to achieve a close to natural language querying capability for a large information system is the following. For each object of class 'item' put in the database the author asked herself which concepts are involved [20] [21]. The concepts are later placed in a hierarchy (see also Figure 4).

Figure 2. An example of a generated query form.

It is relatively straightforward to automatically generate a search form in which the user can construct a question by selecting from two selection boxes. It is equally straightforward to find the possible answers to that question by executing a simple SQL query on the relational database. The result of the query is a set containing all sets and documents that satisfy the given criteria. An example query form is shown in Figure 2.

2.2.3. Separation of structure and content. The ability to separate structure and content is an important feature of authoring tools that can simplify modifications. Separation here means mutual independence, meaning that structure is independent of content and content is independent of structure. The first is true for most systems, so that the content can be modified without having to modify the structure. The rationale for this is that content is more likely to change than structure, so one first decides on a structure and then implements the content. This implies that the content becomes dependent on the structure. However, on the rare occasions that the structure does have to be changed, it is likely that everything has to be changed. It is therefore desirable to achieve mutual independence.

In OOHM the structure is independent of the content, and content is independent of structure – at most there is dependence on very local structure.

2.2.4. Support for structural templates. A structural template is a template that, when instantiated, creates a new object, and within it a number of placeholders for other objects. For instance when a bibliographical reference is created, it has placeholders for its authors so that the user is reminded that he or she should fill these placeholders with new or existing author objects. Other examples of structural templates are given in the next section.

We feel that this type of template features are important tools within authoring systems: they actually aid authoring. More about templates in hypertext authoring can be found in [22].

3. A hypermedia design method

As was mentioned earlier, having a hypermedia model is not enough. We also need a method to build a model of a part of the real world and a method to translate the abstract model into an application. Our answer to this problem is to use existing methods and tools of object-oriented software design to create an OOHM model, and use OOHS to implement and populate it.

In OOHM each class either represents a separate type of information or a separate view of existing information. In the design stage of a hypermedia database, these different classes are identified, along with their relationships. Next the layout and navigational structure for each class is defined by writing the scripts for each class. This step may be integrated into the analysis by extending the analysis primitives as was done e.g. in RMM (Relationship Management Methodology) [7], where the entity-relationship model is extended with navigational primitives.

3.1. Examples

3.1.1. Example a : bibliographical references. Figure 3 shows a part of an object-oriented design diagram detailing the structure of bibliographical reference information. The specific model used here is the EROOS model as developed by Steegmans in [23], which is closely related to the ER-model, but does not distinguish classes and relations.

Figure 3. Example on an object design diagram

As can be seen from the diagram in Figure 3, a reference is either a *publication*, or an *article*. A publication can either be a *journal*, a *book* or a *proceedings*. An article may have *appearances* in publications. Each reference has a number of *authors*. An author is a *person*, having a name and *address*, and may work at an *organisation*. As persons and organisations can both have an address and a homepage, they are generalised into *generalised person*. Each reference itself may be *available from* a number of internet addresses.

The object design diagram includes information about the relationships between objects. The example for instance shows that a 'person' object may contain a reference to an Internet address as this person's homepage, and a number of references to locations. From this structural template can be derived for *person* that has placeholders for a number of organisations (works at) and a homepage. When a new person or author object is created, empty placeholders are generated for the homepage and a set of organisations.

Figure 4. Example diagram of learning material

3.1.2. Example b : learning material. Figure 4 shows a simple diagram modeling e.g. learning material, which we use in a specific application. The material consists of a number of *units* which are organised in a hierarchy and that contain *items* which are subclassed into five types.

The first '*what*' represents a textual explanation of the current unit. The second '*where*' item explains the context in which the unit's content is important. A '*how*'-item contains some procedure or procedural knowledge. An '*example*'-item contains an example, a '*references*'-item contains a number of reference-objects (not detailed in this diagram). Units are constructed so that they contain at most one of each type of item. This constraint can be (formally or informally) built into the model, and the authoring tool will warn when a new item was inserted where it should not. Such constraints can be implemented by a trigger mechanism provided by the relational database management system used for storage.

As each item refers to some *concepts*, which in themselves are structured by the *prerequisite* and *subconcept* relationships, a secondary entrance to the information is provided. Students can go through the material according to the units, as it was conceived by the author, or via the concepts. The concepts-view provides quick access to specific information, while the units-view provides access in a more systematic order.

4. Implementation aspects

We have implemented a software system that supports the proposed design model and can be used to generate hypertext webs for the World-Wide Web, or any other hypertext system. It consists of three programs that are in fact user interfaces to an underlying database. These programs are :

- the set and link editor : a user-friendly interface that allows the creation and maintenance of sets and links and immediately visualizes the result;
- a class editor : to create and edit classes and inheritance diagrams;
- the content editor : an integrated text-editor with built-in SGML checking capability.

We are currently thinking about integrating a sophisticated object-oriented design tool into the class editor.

4.1. The use of a relational database

A relational database is used to store link information as well as meta-information, classes, templates, inheritance diagram, access rights etc. This has a number of important advantages. One can depend on existing and state-of-the-art software that has its own support for multi-user management, network access and concurrency. Also a standardized query language, SQL, is used to query and modify the database and provides a secondary interface to the data. Through SQL the author can retrieve information that would otherwise be difficult or impossible to collect, and through update queries large structural changes can be made automatically.

4.2. Collaborative authoring

Figure 5. Screenshot from the authoring environment

The database is accessed through the ODBC (Open Database Connectivity) standard. This allows any relational database to be used for storage and querying. By using a distributed database system, like Oracle, the database can be accessed and modified from anywhere on the Internet.

Figure 5 shows a screenshot from the authoring environment where the set editor is used in conjunction with netscape.

5. Relation with object-oriented software design methodology

For a system to be truly object-oriented, it must at least expose the following three features : inheritance, polymorphism and encapsulation. In OOHS these features appear in the following form :

1. *Inheritance* : classes of hypertext or hypertext-sets can inherit a number of features from (one or more) parent classes. These features are :
 - visualization of content and hyperlinks for a certain hypertext system;
 - production of hardcopy of content and hyperlinks;
 - generation of a local map of the hypertext structure;
 - a set of attributes and their domains for meta-information to be used for searching the database;
 - a form or program that helps the author fill in these attributes.

In the case of multiple inheritance, the visualization are that of the first parent class. The attributes and forms are appended. Each feature can either be simply inherited, redefined, or extended by a child class.

2. *Polymorphism* : links referring to parent classes can later be instantiated with an object of any of the subclasses. I.e. a link to a parent class can be resolved by any object that inherits from that class. For instance, a link to a reference with either point to an article, a journal, a book or a proceedings.
3. *Encapsulation* : each class has a carefully defined interface and all interaction with it must be through that interface. Hypertext classes on OOHS have a predefined interface in the sense that the class defines how it is to be visualized and with what other classes it is linked.

6. Future work

The main objective of our research was to improve hypertext design by introducing more sophisticated paradigms of software development into it. It is reasonable to expect that the application of object-oriented design methods will improve the productivity of authors by enabling reuse of existing resources. On the other hand we think it will improve the comprehensibility of the product because the analysis and design will be more closely related to real-world objects and authors will be forced to adhere to a relatively high-level design.

The second step was to implement a system that supports the implementation of hypertext following the

proposed model. This could be compared to the development of an object-oriented programming language.

We are now considering the use of object-oriented design tools to aid the design of complex hypertext systems. Many of the existing tools of this kind are general enough to be used for this purpose. The tools could generate SQL update queries to define new classes and SGML template scripts instead of program code. In this way existing and sophisticated design tools can be integrated into the authoring system and into hypermedia design in general.

7. Conclusion

We have defined a new hypermedia design model with the following features :

- the model is set-based, allowing the reuse of sets and documents;
- it allows the same information to be accessible from different viewpoints;
- it allows for templates of sets to be prepared and instantiated;
- it allows for meta-information that can be used by the search engine;
- it separates content and structure.

Important features of our implementation of the authoring tools based on this model are :

- link information and meta-information can be stored in any relational database (standalone or via network) allowing it to be queried and modified with SQL and other existing database maintenance tools;
- in a networked environment the information can be stored centrally and edited by multiple authors concurrently;
- the templates have object-oriented features such as inheritance, polymorphism and encapsulation, which facilitate modifications.

We have developed a design method and implemented an authoring system that supports this model and method.

We are currently investigating formal object-oriented design methods in order to integrate them into our system. The goal is to use sophisticated object-oriented computer aided software engineering tools to support the design and implementation of hypermedia systems.

8. References

- [1] R. Cailliau, "About WWW", *Journal of Universal Computer Science*, 1(4):221-230, April 1995. (http://www.iicm.tu-graz.ac.at/Cjucs_root).
- [2] M. Thüring, J. Hannemann, J.M. Haake. "Hypermedia and Cognition: Designing for Comprehension", *Communications of the ACM*, 38(8):57-66, August 1995.
- [3] F. Kappe, H. Maurer, N. Sherbakov, "Hyper-G, a universal hypermedia system", *Journal of educational Multimedia and Hypermedia*, 2(1):39-66, 1993.
- [4] T. H. Nelson. "The Xanadu ideal", *Technical report*, Xanadu On-Line Publishing, 3020 Bridgeway 295, Sausalito CA 94965, USA, 1994.
- [5] W. Hall, G. Hutchings, S. White. "Breaking down the barriers: an architecture for developing and delivering resource based learning materials", *World Conference on Computers in Education VI, WCCE'95, Liberating the Learner*, Birmingham, U.K., pp. 623-634, Chapman and Hall, July 1995.
- [6] R. Elmasri, S. B. Navathe, *Fundamentals of Database Systems*, Benjamin/Cummings, second edition, 1994.
- [7] T. Isakowitz, E. A. Stohr, P. Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design", *Communications of the ACM*, 38(8):34-44, 1995.
- [8] F. Garzotto, P. Paolini, D. Schwabe, "HDM2 – Extending the E-R approach to Hypermedia Application Design", *12th International Conference on Entity-Relationship Approach*, pp. 175-186, 1993.
- [9] D. Schwabe, G. Rossi, S. D. J. Barbosa, "Systematic Hypermedia Application Design with OOHDM", *ACM Conference on Hypertext 96*, pp. 116-128, 1996.
- [10] D. Gries, *The science of programming*, Springer New York, 1983.
- [11] H. Van Dyke Parunak, "Don't link me in: Set based hypermedia for taxonomic reasoning", *Third ACM Conference on Hypertext*, San Antonio, Texas, pp. 233-242, ACM, December 1991.
- [12] E. Duval, H. Oliivié, N. Sherbakov, "Contained Hypermedia", *Journal of Universal Computer Science*, 1(10): 687-698, October 1995.
- [13] M. P. McCahill, F. X. Anklesaria, "Evolution of Internet Gopher", *Journal of Universal Computer Science*, 1(4): 235-246, April 1995.
- [14] K. Andrews, F. Kappe, H. Maurer, "The Hyper-G Network Information System", *Journal of Universal Computer Science*, 1(4): pp. 206-220, April 1995. (http://www.iicm.tu-graz.ac.at/Cjucs_root).
- [15] B. J. Haan, P. Kahn, V. A. Riley, J. H. Coombs, N. K. Meyrowitz, "IRIS Hypermedia Services", *Communications of the ACM*, 35(1):36-51, 1992.
- [16] C. F. Goldfarb, *The SGML Handbook*, Oxford University Press, 1990.

- [17] J. F. Koegel Buford, *Multimedia Systems*, SIGGRAPH Series, ACM Press, 1994.
- [18] F. Colaitis, "Opening up multimedia exchange with MHEG", *IEEE Multimedia*, 1(2), 80-84, 1994.
- [19] K. Hendriks, E. Duval, H. Olivié, "Hypermedia for Open and Flexible Learning", *World Conference on Computers in Education VI, WCCE'95, Liberating the Learner*, Birmingham, U.K., pp. 349-361, Chapman and Hall, July 1995.
- [20] J. Elen, K. Hendriks, K. Proost, "Towards an Electronic Performance Support System for University Teachers: Digit", *Twenty-first Conference on Improving University Teaching*, 1995.
- [21] K. Hendriks, J. Elen, H. Olivié, "Structured Hypermedia Authoring: a Simple Tool for the Design and Implementation of Structured Hypermedia Databases", *Third International Conference Computer Aided Learning in Science and Engineering (CALISCE '96)*, San Sebastian, Spain, pp. 279-287, Springer, 1996.
- [22] J. Nanard, M. Nanard, "Hypertext design environments and the hypertext design process", *Communications of the ACM*, 38(8):49-56, August 1995.
- [23] E. Steegmans, J. Lewi, M. D'Haese, J. Dockx, D. Jehoul, B. Swennen, S. Van Baelen, P. Van Hirtum, "EROOS Reference Manual. Version 1.0", Katholieke Universiteit Leuven, Departement Computerwetenschappen, CW rapport nr. 208, januari 1995.