Published in: First Int. Workshop on Digital Libraries (DLip 2001),
Proc. of DEXA-Workshops, München, IEEE-Press, 2001, pp. 933-936

The Object Oriented Document Model of a Meta System for Existing Digital Libraries

Michael G. Bauer
Institut für Informatik, TU München
Orleansstraße 34, D-81667 München, Germany
bauermi@in.tum.de

Günther Specht
Fakultät für Informatik, TU Ilmenau
Postfach 100565, D-98684 Ilmenau, Germany
gspecht@prakinf.tu-ilmenau.de

Abstract

In this paper we describe an object oriented document model of a meta system for existing digital library systems. We show how we solve heterogenity of existing systems in the context of linking and annotating existing digital library documents with a slim and simple approach.

Our approach is already implemented in the system OMNIS/2, which is an advanced meta system for existing digital library systems and enhances existing digital library systems or retrieval systems by additional storing and indexing of user-defined multimedia documents, automatic and personal linking concepts, annotations, filtering and personalisation.

1. Introduction

With the growing number of information that is available to the individual and the different media types, many different digital library systems were developed. These systems are established tools, but users still miss features that would improve their ability to work with documents in digital library systems as it is common with books printed on paper (i.e. adding references, marking pages and annotating text). This led to the development of the OMNIS/2 system[1] which is a meta system for various existing digital libraries [7]. It equips users with a tool that enables them to search several systems at once and especially to benefit from links between documents of different digital library systems (cross linking of documents). Users are able to add links by themselves (with an authoring tool that is part of OMNIS/2) and we will soon offer links that are generated automatically whenever possible (e.g. for bibliographic references or keywords). These features are often asked for by research groups who work with specialized library systems (e.g. VD17, a digital library of all German printings of the 17th century, a former project) [2]. Merging links and documents does not require a modification of the source documents and can even be done without harvesting, which would collect all documents and store the documents in an own huge database. This is achieved by a technique which adds the links at run-time using XSLT [9] right before the documents are presented to the user. We call this technique a posteriori cross linking [1]. In addition it is possible to annotate external documents without a write permission for the library systems which hold the annotated documents. The annotations are not limited to text and can consist of multimedia documents. Users are able to use a personalisation feature to create their own view on the documents and to "work" with digital library systems by themselves. In its current version the system has already access to some 80000 documents of the digital library system of the Faculty for Computer Science at TU München. As the integration of other digital library systems and media libraries is planned (or already underway) and diversity between the digital library is significant we started developing an object oriented document model to overcome this heterogenity. The document model is one of the key features of OMNIS/2. Most of the enhancements the system offers (as a posteriori crosslinking, annotations, personalisation) are based on this document model.

We describe the document model in detail in section 2. In section 3 we explain the architecture of OMNIS/2 and show the environment in which the document model is successfully used. We then give a very short overview on related work (section 4) before we end the paper with a summary (section 5).

[1]OMNIS/2 is funded by DFG (German Research Foundation) within the research initiative "V3D2" ("Distribited Processing and Delivery of Digital Documents") and is part of the Global Inventory Project (an initiative of the G7-countries).

2. The Object Oriented Document Model

2.1. Theoretical Preparations

As OMNIS/2 sits on top of established digital library systems it has to handle several different types of documents which emerge from the underlying digital library systems.

Following this we have at first examined different ways of how to categorize the documents. The most simple one is to categorize the documents according to their source. We call the documents from the underlying digital library systems external documents and refer to the documents uploaded by the users, i.e. userdefined documents and annotations (texts, graphics, etc.) as internal documents. Internal documents are stored in an own relational database system (the so called meta database, see section 3 for details).

The content of internal and external documents strongly differs. Since OMNIS/2 does not use any harvesting the content of external documents is not explicitly stored by OMNIS/2 but remains in the underlying digital library systems and is only retrieved upon request. In our system we only hold a persistent unique identifier for every external document together with some meta data and linking information. For local documents (where we store the content locally of course) we can easily assure this, for external documents it is an requirment for the underlying digital library systems so that they can be utilized for OMNIS/2. This fact is especially important in the context of a posteriori cross linking.

We can also categorize our documents according to structure. Our design of the system as both, a metasystem and a stand-alone multimedia database system, requires that there are not only simple stand-alone documents from various sources. Users must have the ability to compose their own documents from various existing (i.e. external) and user-defined (i.e. internal) documents. The document model of OMNIS/2 therefore distinguishes two types of documents, composites and atoms, thereby following the Dexter Hypertext Model [4]. Composites either consist of one or more atoms, one or more composites or both. A composite can not exist for itself, but always has to contain at least one atom. Consequently composites are internal documents and atoms are either local or external. This results in a hierarchical document structure (see Fig. 1 for the BNF-notion of the document structure and Fig. 2 for an example). This hierarchy is actually a DAG (directed acyclic graph), thus atoms and composites can be shared within the same hierarchy and can occur in several different levels.

Atoms are of a single type or may even be complex if they are accessible as a whole in an external system. In the current system we consider text, images, audio, video and external (i.e. the external documents) as document types. The external documents are further divided into the various document types originating from the integrated external digital library systems. This separation implies a strong encapsulation of document specific features in the document types and motivates the implementation of this document model in an object oriented way.

```
<Document>  ::=  <Composite> | <Atom>
<Composite> ::= {<Composite> | <Atom>}+
<Atom>      ::= internal Atom |
                external Atom
```

Figure 1. Document hierarchy in BNF-notion

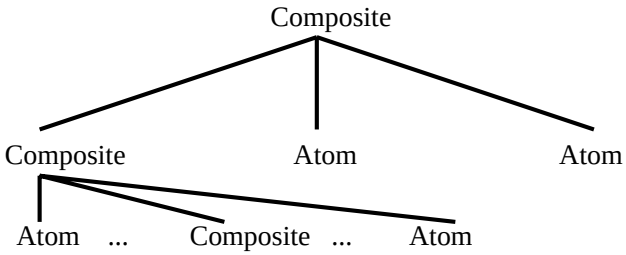

Figure 2. Example of a hierarchical document structure

Another consideration comes from the fact that due to the heterogenity of the underlying systems OMNIS/2 has to find a way to access the different documents in a uniform way.

Taking all these considerations into account we decided that an object oriented design would follow exactly our needs. In our model every document is seen as an object and can exist for its own at runtime. Every object is identified by a persistant unique identifier (UID). The document objects have the ability to create themselves from the database on request. In addition they carry all information about themselves (i.e. anchors, anntotations) and can display themselves for the presentation to the user. A second display method enables a different presentation e.g. for an authoring tool. In the case of composite documents the display method simply calls the display methods of its child documents.

This design enables us to use object-oriented programming techniques although the underlying systems do not offer object oriented features at all.

2.2. Implementation

A very suitable method to implement our document model uses the composite design pattern (a structural design pattern) [3]. This design pattern enables us to represent part-whole hierarchies using objects and also provides a uniform interface to both composites and atoms (see

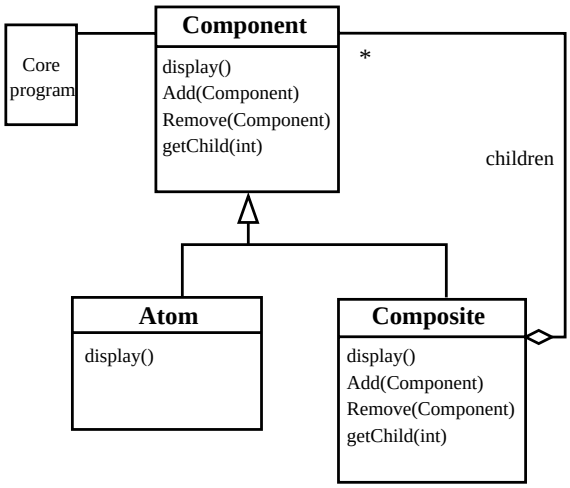

Figure 3. Composite design pattern (UML)

Fig. 3 for the design pattern in UML). It gains its flexibility through the component class which acts as a container. This is especially important for the presentation of the documents with the display methods as it is easy for composite documents to simply call the display methods of their child documents (regardlessly whether they are atoms or composites) which form the document hierarchy. For OMNIS/2 we decided that the documents display themselves in XML, which enables us to use modern techniques like XSLT to process (annotate, link) the documents. The hierarchical structure of XML in addition assists us to preserve the hierarchical structure of our document model. The solution overall combines several important aspects. It is slim but nevertheless very powerful and reduces implementation time drastically (this is also supported by our decision of using Java for the implementation). It hides heterogenity very well in our case by using encapsulation and it is still easily extensible.

3 Architecture of OMNIS/2

The presented object oriented document model is already implemented and used in our meta system OMNIS/2. In the following we briefly describe its architecture.

OMNIS/2 is separated basically into two layers (see Fig. 4). We use established digital library systems as data providers and treat them as large containers with powerful query languages. We assume that the systems use XML to make their data available to the outside world as it is common for modern systems nowadays. It would also be possible to access the digital library systems through a tight coupling by using certain ports the systems provide, but we do not discuss this approach any further in this paper. Our system sits on top of the established digital library systems as a meta system and acts as a service provider to the users, who access OMNIS/2 through a common browser. We decided to implement the meta system as a Java servlet in the Apache webserver. The meta system stores and handles all of the linking information and also the annotations, i.e. multimedia documents that users can upload into the system, in its own relational database. With referential integrity dangling links are avoided within the OMNIS/2 system if referenced user-defined documents are removed. The system itself is not only a meta system for digital library systems, but it can also be used as a stand-alone multimedia digital library system as all features are available as well for user-defined documents.

4. Related Work

Over the years different ideas of object oriented document models emerged. At this point we concentrate only on a few of these proposals.

The Kahn/Wilensky Framework [5] is probably the most similar proposal to our solution. It describes how to name, identify or invoke digital objects (which can be any kind of data-streams) in a system of distributed repositories. In our solution we even go one step further and push functionality into our objects that is very document specific (e.g. the display() methods).

The SODA concept (Smart Objects, Dumb Archives) [6] uses a very interesting encapsulation technique where traditional archive functionality is transferred to document objects. These objects are modeled as buckets and the buckets provide all functionality for the actual documents. Buckets are most similar to digital objects mentioned in [5]. For the SODA concept the archives themselves offer only very simplistic methods (put, delete, list, info, get). In OMNIS/2 we also put functionality into our document objects. We have to work however with smart archives which know about the content of the documents. This motivated our encapsulation strategy to hide this.

The Stanford Infobus [8] is a very general approach towards interoperability for digital libraries. The Infobus itself enables participating systems to see documents as objects and to move them on the Infobus between different digital library systems and clients. The design is very powerful and goes beyond the scope of OMNIS/2 but requires participating systems (in the case of OMNIS/2 the underlying digital library systems or third parties) to implement the specification, which we do not.

5. Summary

Heterogenity and interoperability are fundamental issues in the field of digital library systems. We presented an object oriented approach for a meta system for existing digital

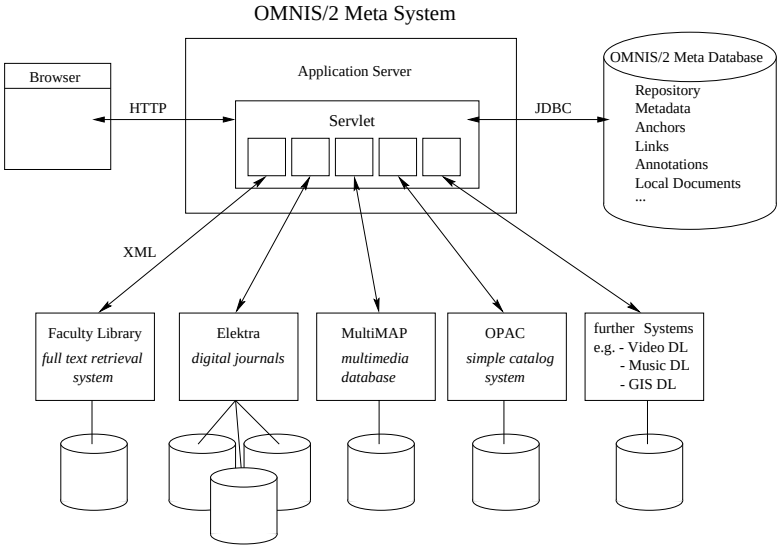

Figure 4. Architecture of OMNIS/2

library systems, where we have to handle different formats of documents and in addition want to enhance the existing documents by links and annotations without changing the source documents. We have also shown how to implement our approach by using a design pattern which results in a slim but still powerful solution. Additionally we presented the meta system OMNIS/2 which provides the environment in which the document model is successfully used.

References

[1] Bauer M.G., Specht G., *Enhancing Digital Library Documents by A Posteriori Cross Linking Using XSLT*, Proc. of the 5th European Conf. on Research and Advanced Technology for Digital Libraries (ECDL 2001), Springer, LNCS, 2001.

[2] Dörr M., Haddouti H., Wiesener S., *The German National Bibliography 1601-1700: Digital Images in a Cooperative Cataloging Project*, Proc. of ADL'97, Washington DC, IEEE Computer Society, 1997, pp. 50-55.

[3] Gamma E., Helm R., Johnson R., Vlissides J., *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[4] Halasz F., Schwartz M., *The Dexter Hypertext Reference Model*, Comm. of the ACM, 37(2), Feb. 1994, pp. 30-39.

[5] Kahn R., Wilensky R., *A Framework for Distributed Digital Object Services*, cnri.dlib/tn95-01, May 1995, http://www.cnri.reston.va.us/cstr/arch/k-w.html

[6] Nelson M.L., Maly K., Zubair M., Stewart S.N.T., *SODA: Smart Objects, Dumb Archives*, Proc. of the 3rd Quropean Conf. on Research and Advanced Technology for Digital Libraries (ECDL 1999), Springer, LNCS 1696, 1999, pp. 453-464.

[7] Specht G., Bauer M.G., *OMNIS/2: A Multimedia Meta System for existing Digital Libraries*, Proc. of the 4th European Conf. on Research and Advanced Technology for Digital Libraries (ECDL 2000), Springer, LNCS 1923, 2000, pp. 180-189.

[8] Stanford Digital Library Project: http://diglib.stanford.edu/

[9] *XSL Transformations (XSLT), Version 1.0*, http://www.w3.org/TR/xslt