Erschienen in: Proceedings of the Third International Conference on Web Information Systems Engineering : 12 - 14 December 2002, Singapore ; Workshops / IEEE; Ling, Tok Wang (Hrsg.). - Los Alamitos, Calif. : IEEE, 2002. - S. 150-159. - ISBN 0-7695-1813-3
https://dx.doi.org/10.1109/WISEW.2002.1177858

Information Concepts for Content Management

Michael Grossniklaus
Institute for Information Systems
ETH Zentrum
Zurich, Switzerland
grossniklaus@inf.ethz.ch

Moira C. Norrie
Institute for Information Systems
ETH Zentrum
Zurich, Switzerland
norrie@inf.ethz.ch

Abstract

Content delivery is rapidly emerging as a complex systems domain concerned with multi-channel, multi-format publication of information across user and application domains. A variety of content management solutions have been developed in response to these challenges based on, not only differing technologies, but also heterogeneous approaches. However, none of these present a solution that is both sufficient and consistent. Here we present an analysis of requirements leading to a general model of the information concepts central to content management. This model is the basis for a web content management solution currently under development.

1. Introduction

Together with the internet's evolution towards an important and widely-used information platform, new requirements for creating, publishing and managing content have arisen. Websites have dramatically grown not only in number, but also in complexity. Hence the job of managing a site is no longer the task of one single webmaster but rather of a team of content providers, editors and designers that strive to deliver up-to-date and correct information [14]. To manage and organise the work of such teams, many website owners have established complex workflow and revision processes to ensure quality of content at all times.

To facilitate and support creation of such managed websites, a large number of software solutions have been developed. The concepts and approaches underlying these systems are as numerous as the challenges of today's web publishing. Since many of these solutions were developed incrementally in response to market demand in a rapidly evolving domain, they tend to lack a clear, consistent concept and instead offer a mix of concepts, each tailored to a specific problem. For example, they may handle data from different sources in quite different ways. Further, although they all make claims of separation of content and presentation, one sometimes finds that the solution does not quite match the claim and presentation concepts permeate the content model. On the other hand, many solutions which have strived towards a model-based approach are not sufficient in that they do not address certain requirements such as multi-lingual support.

To address this situation, we wanted to derive a consistent and sufficient model for content management based on an extensive analysis of requirements combined with a study of current solutions. We present these requirements in section 2 and then go on to discuss the extent to which existing solutions meet these requirements in section 3. Following this, we present our own object-oriented model of content management in section 4. At the core of the model there are information concepts essential to content management to separate content, structure, view and presentation. Beside these semantics for content the model also incorporates concepts for user management, business workflows and context-dependent behaviour. In section 5 we then describe the development of a content management server based on this model. Concluding remarks are given in section 6.

2. Requirements

Managing today's websites presents a whole new set of challenges and requirements. As the creation and maintenance of complex sites is now a collaborative effort of a team of professionals with varying technical backgrounds, these requirements involve not only content organisation and storage, but also support for collaborative working and business workflows. In this section, we describe the most important requirements that a content management system should address. As these requirements depend on different kinds of user roles (content editor, web designer, web site manager etc.), we will group them accordingly.

Content Editor. Authoring is the line of action of content providers or editors. Their task is to create and maintain content. Neither do they decide how their content is to be presented nor do they design how it is accessed or navigated. This separation of concerns leads to a set of basic requirements every content management system should meet. The first of these requirements is the well-known separation of content and presentation. By separating these two concepts, a content management system is able to support a number of content authors while ensuring that all content abides to the defined presentation guidelines, such as corporate design or corporate identity.

When looking at the content itself, the requirement of multi-format content is apparent. As the internet is a global institution and can be accessed from virtually anywhere, many modern websites are available in different languages. A great number of these sites are already capable of adapting to the language preferred by the requesting user-agent. Language however is only one dimension where multi-format content is useful. With an ever- increasing set of web browser technologies and support for a wide range of platforms such as mobile phones, media phones and PDAs, content has to adapt itself to many different characteristics. Among these are file format, resolution and size of images or version and level of detail for text. A content management system should not limit itself to fixed dimensions, as is often done for language, but rather support a user- definable and extensible property model that then can be used to deliver the best possible content to the requesting client.

Another important requirement is the notion of users, user roles and permissions. When a potentially large number of authors are working on one website, it is paramount to track the originators of all changes and to control whether they should be allowed to perform these changes. Hence, in addition to versioning, a content management system must offer the possibility of creating and authenticating users. To manage and classify these users in hierarchical groups, the concept of user groups or user roles, such as *content editor* or *content revisor* have to be present in the system. But users and user groups themselves are not sufficient. Customisable and extensible permissions and user rights must also be managed and enforced by the system at all times. A joint model of users, groups and permissions that allows the initiator of a website to implement the required user scheme is therefore cleary needed.

Web Designer. The web's evolution into a publishing platform where global players and large-scale companies present themselves leads to different challanges in terms of the design of webpages. Companies have extended their corporate identity and corporate design guidelines to include the website. Graphical designers and artists are employed to develop the look of the individual pages. Clearly, a content management system has to be aware of this situation and present concepts to solve the problems arising. Again, the previously presented separation of content and presentation emerges as a central requirement to enable such a mode of website development.

As a matter of fact, the presentation and design of a website is dependent on the client requesting the page. Today's web clients vary greatly in terms of capabilities. Desktop browsers capable of displaying HTML are certainly the most commonly used clients. But other browsers such as mobile phones, media phones, PDAs and even voice-based systems have entered the picture. Not only have these devices limited or no support for HTML, but also they are very heterogeneous in terms of display size, colour depth and rendering facilities. Therefore even if such a client supports HTML, it clearly needs a different version than a desktop browser to cater for the different screen dimensions. The possibility to define multiple designs uncoupled from the content is the next requirement that a content management must meet.

To support multiple-presentation channels, these design templates have to be characterisable in much the same way as content in the above section. Allowing such presentation properties at a high granularity level enables the designer to tailor a presentation to a particular device and the content management to select the best design for any request. It is therefore another central requirement for any such system.

Web Officer. Up to now, we have concerned ourselves with content and presentation, but there are other key dimensions in designing a content management system that tend to be forgotten alongside the omnipresent requirement of separating content and presentation. These further requirements that surpass the simple separation of content and presentation are introduced by the line of action of the webofficer. Webofficers have only come into existence since the emergence of large-scale corporate websites. Their job involves determining the overall structure of a website, designing its navigation and deciding upon personalisation options.

To be able to design the navigation and content structure of a webpage independently from the content and presentation, it is necessary not only to distinguish between these two concepts, but to enforce a clear separation of content, presentation and structure. This requirement is particularly needed when creating websites for multiple presentation channels. If structure would be mingled with either content or presentation it would have to be recreated or even duplicated for every presentation channel. Of course, separating structure does not mean that it cannot vary across such channels if necessary.

Designing and creating a personalisation scheme for a website is a further task of a webofficer. Specification and

implementation of such personalisation schemes is therefore another general requirement of a content management system. Such a scheme determines which parts of a site can be personalised and what options are available. For instance, it could be possible for a user of a website to create his own version of certain content. A more restrictive option would be that the common user may only annotate or comment the content of the website. We will call this user-centric personalisation *explicit personalisation* because the user decides how the website should behave. Another form of personalisation is *implicit personalisation.* Here the context of the client influences how the content management system will deliver the pages to the user. Context-awareness is a very important requirement for a content management system as context itself can be used throughout the system to improve personalised and correct content delivery. Context is the counterpart to the characterisation of both content and presentation and is therefore used by the system to select the appropriate data and templates to generate the page. Context can be thought of as a set of *(name, value)* tuples that describe the user, software, hardware and environment of the client browser. To be context-aware, a content management system must allow the specification of valid context dimensions (e.g. browser-version, screen-size, user-location, etc.) and use the gathered values in the process of building a page.

Website Manager. In every major company there exists a set of business processes and predefined workflow that describe how a content object has to evolve until it is published on the corporate website. Using such processes, a manager can guarantee that no incorrect or out-of-date content is ever shown on a page. Since all personnel supervised by the website manager are working on the website via the content management system, these workflows have to be integrated and enforced by the system.

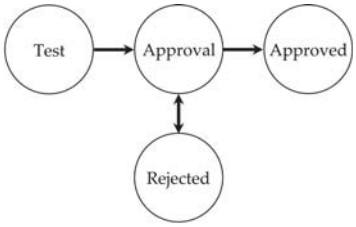

Figure 1. Example Workflow

A simple example for a publishing workflow would be the following. An author initially creates a content object. While he is editing the object, it is in the state of *Test* and cannot be accessed or viewed by anyone else. As soon as he finishes editing and would like to publish the object it changes state to *Approval*. The object is now examined by the supervisor of the object and either transferred to the state of *Approved* or *Rejected*. If an object is approved, it becomes visible for everybody and is live on the website. If it is rejected, the appropriate author is informed and may revise the object and resubmit it for approval. The example workflow described above is shown in figure 1.

This simple workflow consists of four states. However a content management system should be capable to support arbitrary processes as these tend to vary between companies. Of course it must also keep track of changes using versioning or related concepts. The system should further be able to notify the corresponding persons whenever state transitions occur.

Web Engineer. The last group of personnel involved in creating a website are the web engineers. Many webpages today are not only presenting information to a user, but are highly interactive applications such as e-learning systems or e-banking solutions. Web engineers are responsible for these active elements of a website, which may involve programming to implement a certain application logic.

A content management system must provide web engineers with a coherent and open application development platform, that offers basic web functionality such as user tracking and shopping carts as optional software modules. A system that meets this requirement prevents engineers from implementing the same components over and over again for every solution they are building and introduces the principles of good software design into the realm of website programming.

3. Existing Solutions

Most available, existing solutions have been developed to suit a specific need or to address a given problem and are therefore heterogeneous in nature. A common design principle shared by many of these systems is the concept of separation of content and presentation. As previously stated in the requirements section, this concept alone is not sufficient to address all challenges posed by today's websites. In this section, we therefore want to analyse some representatives of existing systems and describe what further requirements they meet and where they still have some shortcomings in comparison to the proposed requirements.

Before discussing concrete content management solutions, it can be observed that all of these systems can roughly be classified as members of one of two approaches that are orthogonal in terms of the role of the database involved. As this classification has proven itself to be useful when characterising such systems, we will describe the two approaches first and use them to analyse specific systems later.

The first approach of designing a content management system has evolved from the database community, we therefore call it the *database-based approach*. With the growing importance of the internet, the need for having web access to databases and publishing the contents of a database on the web has arisen. Many different solutions have been developed that allow databases to be browsed over the web and some of them have gone as far as integrating the concepts of content management into their solutions. The possibility of bringing any existing database to the web clearly is the main advantage of this approach. But there are also other more subtle concepts that can be considered very useful. As the database community has developed a great number of data models that can be used to represent content in a semantically rich way, these models can now be employed to design the data underlying a website, thereby enabling the content editor to work with richer data concepts than just texts and images. However, a data model alone is not sufficient to cover all aspects of modelling a website. Advanced solutions belonging to this category therefore have developed a great number of additional models such as composition, navigation and presentation models. Coupled with such models is often a specific design method assuring quality of the developed product [4]. This is another strong feature of this approach as such models are currently lacking in most web development tools and have not yet been accepted by a broad community. Alongside these benefits and advantages, there are also a number of serious disadvantages and limitations to this approach. The most severe is of course the lack of support for multi-format content. No existing system provides built-in constructs for this concept. Although many database systems have facilities for versioning, they do not allow versions based on context directly. Another drawback that can be observed in most of these systems is their limited or lack of support for dynamic content modification as there are too many models and layers involved between the actual data and its final presentation.

The second approach is orthogonal to the first and has emerged from the publishing community. As this approach focuses around mapping documents to a database rather than bringing a database to the web, we shall call it the *document-based approach*. Instead of using an arbitrary database with a freely definable schema to store the data of a website, systems belonging to this category employ a specialised database with a schema tailored to the data types occuring in documents. Such a schema commonly will include such concepts as texts, images, urls, links, etc. Clearly it is very easy to design a schema like this to separate between the concept of an object and its actual content, thus enabling multi-format objects. Another advantage of such systems is their great flexibility. As there is no complex application model involved, it is very easy to perform small changes to a website without defining new types of objects. On the other hand, with great flexibility always comes loss of control. As there is no possibility to define user-types belonging to the application domain of a website (e.g. book objects in an online store) there is also no semantic information that a set of texts, images and links actually represents such an object. Upon modification of the data, the system is not able to check whether the changes are valid or if they violate an implicitly assumed concept of the application domain. But there are also other more aggravating consequences to the lack of object-orientation that comes with the impossibility of user-defined types. Reuse of data becomes tedious or even infeasible. Imagine the case of a person object consisting of a text (e.g. the person's name), an image and a link (e.g. to the person's homepage). A document-based system can only manage these three pieces of data independently as it offers no notion to group them into an object. If this person appears on multiple pages of a website, the content editor must repeatedly re-establish the concept of a person by hand. Such a procedure is painful at best, but it is also very likely to be error-prone.

Probably the most widely-known solution to the problems of content management in the research world is **WebML** [15, 2, 1]. WebML is a good representative of the first category of approaches. It consists of a set of orthogonal models (structural, compostion, presentation and personalisation) that support a well-defined design process that separates the concerns of all personnel involved. Aside of separate models for different user groups, there is no handling of workflows and user rights built into the model. There are however some other characteristics of WebML that also do not meet the requirements proposed above. In their design of units in the compostion model, there is no clear separation of structure and presentation. Both index units and data units, which are structural concepts, have built-in implicit assumptions about their final presentation. Another drawback is the lack of support for multi-format content for which no concepts are provided in this solution.

A historically important solution addressing the challenges of web site management is **Strudel** [5, 6]. Strudel emerged from the database community and therefore can be seen as another system using the *database-based approach*. By extending and adapting concepts of information integration and management of semi-structured data, Strudel tries to address the problems arising with large-scale website management. At the core of the system is a remarkably clear separation of content, structure and presentation. As is typical for this approach, the content of a website comes from an arbitrary database. StruQL is the query language used in Strudel to access the website data and build page and navigation structures. Finally, templates written in Strudel's own template language transform this structure into the desired markup. Although the publishing aspect of Strudel is very powerful and well-designed, the system lacks other

features required for content management such as the notion of users, user rights and workflows. Most important however is the absence of concepts to support multi-lingual or even multi-format content. Supporting multiple presentation channels is possible but involves a small number of changes to the core of the Strudel system.

From the open-source community comes **Zope** [20], probably the most advanced freely available content management system today. Zope has also to be classified as a system belonging to the *database-based approach* as it is heavily based on its underlying Z Object Database (ZODB) [7]. Zope is implemented in Python and provides separation of content and presenation by means of persistent Python objects and design templates written in their proprietary markup language. As a lot of control of composing pages is given to those templates, websites are hard to port to multiple presentation channels as the logic contained in these templates has to be implemented over and over again. Although Zope is built on a specially developed database, it does not offer multi-format content features and thus has no built-in support for multi-lingual sites. ZODB also imposes another restriction on the Zope system: As ZODB is organised as a hierarchical database, the system's potential in modeling website data is limited.

Obtree C4 [13] is a commercially available content management system which is currently managing the website of many large-scale companies. C4 clearly has to be classified as a *document-based* system as it stores all document objects such as text, images, links, etc. in a specialised database. The system provides full multi-lingual features but has no concept for other content characteristics as presented in the requirements above. Apart from content, the system also manages users, user rights and business workflows and offers a variety of editors to support the requirements of content providers. The page structure is defined by a virtual tree of directories and pages. Inside these pages, templates control the composition and presentation of content. Hence the approach is not meeting the proposed requirement of separation between presentation and structure and publishing websites to multiple presentation channels, can therefore become rather difficult.

Another open-source content management system is **Cofax** [3] which was originally designed as a system to publish online newspapers. As such, it also has to be classified as belonging to the second approach. In contrast to Obtree C4, it not only maps concepts such as text, images, links, etc. to a database, but also has built-in concepts for articles which is clearly a remnant from its original purpose. The system lacks a clear separation of content and layout as it is still heavily targeted to HTML. There is no support for multi-lingual nor multi-format content. Object lifecycles and workflows are limited to the possibility of declaring triggers, but there is no coherent concept to support those notions.

The **Microsoft Content Management Server 2001** (MCMS) [10] is another commercially available product that classifies as a *document-based* system. The notion of pages are central to this system and it is therefore clearly targeted at clients using desktop browsers. Accordingly, there is almost no support for multiple presentation channels. The separation between content and layout is again achieved by using templates that contain references to content objects. Objects can have mutliple languages and revisions. On the other hand, MCMS provides comprehensive support for users, user roles and access rights which can even be integrated with the user authentication mechanisms of Microsoft's operating system. Microsoft has very recently announced the next version of its content management system to be released at the end of 2002. This release addresses many of the problems presented here. It is however to early to judge whether the system will indeed satisfy all of the proposed requirements.

Although this selection of systems is not exhaustive, it can be seen that none of the presented solutions fully meets all requirements. Support for multiple presentation channnels and multi-format content seem to be key problems that are not yet resolved. In the next section, we introduce our model that represents our approach to addressing these problems in a consistent way.

4. Proposed Model

Both the *database-based* and *document-based* approaches have their advantages and disadvantages. The first approach is very strong on conceptual data modelling, supporting both high-level object concepts and semantic information about the application domain. The latter has strong concepts for multi-lingual or even multi-format content and different object versions. As a logical consequence, our research has focused on combining the powerful aspects of these orthogonal approaches into one consistent model that can be used as the basis of a modern content management system. In this section, we will present the important parts of this model and show how these information concepts can be used to satisfy the proposed requirements. But before going into the details of the model, we first introduce some key ideas and concepts that enable database-like application modelling to be integrated with traditional content management.

At the heart of our model is the key concept of separation of content, structure, view and presentation. To adequately meet all proposed requirements, we found that separation of content and presentation is not sufficient. Both structure and view have to be introduced as core constructs into any information model that is to be used for content manage-

ment. Having *stucture* as a concept of its own allows multiple compositions and navigations on the same content as is needed, for example, when targeting heterogeneous client platforms such as PDAs and media phones. One can easily imagine that the navigation structure might differ quite a bit on these two types of devices. On the other hand, separating structure from layout which is, as we have seen, a common problem in existing systems, enables the system to use the same structure with different layouts when appropriate. As an example for this behaviour one can imagine a website where users can personalise how the pages are presented to them, i.e. which colors or font sizes are to be used. The other important concept that we have introduced in our model over existing approaches is the construct of *view*. Views are used to define what parts of an object are shown to the user. Similar to the concept of the same name known from relational databases, a view can be used to select certain attributes of an object to be included in the final rendering of a page. Further, views can be used in our model to decide on the behaviour of associations or references between objects of a certain type. To illustrate the concept of a view one might think of a database model containing the constructs of *author* and *book*. An author object may reference a set of book objects and a book object may reference an author object. For the page of a given author on our website, a view for type author can be used to specify both the attributes of the author to be displayed and whether the books written by this author are to be embedded on the same page. If the books are to be included on the same page, the system will automatically *unroll* the association and follow the semantic link between the author object and the book objects.

Clearly personalisation and context-dependent delivery of content are key issues in designing modern websites. Therefore, in addition to a clear separation of concepts, our model also includes two additional constructs to support personalisation at any level. The concepts of *context* and *characteristics* are very much related in that they are tightly working together in the process of generating the best possible website for any incoming request. The introduction of the notion of context into the model allows specification of what context should be managed and processed by the system. It further allows specification of how this context is represented, i.e. of which context dimensions it consists. A possible example for such a context would be the *software context* which may consist of dimensions such as *os_id* (operating system identification), *browser_id* or *plugin_support*. Upon receiving an incoming request, the system can extract the appropriate values and construct the corresponding context. This context is then passed to the page assembly engine that will use it to select the best matching content, structure, view and presentation. Having talked about matching context against the properties of these data and metadata objects, it becomes evident that there has to be another construct present in the system which allows us to describe or annotate these objects. Currently we have chosen the simple approach of annotating objects that are up for content-dependent selection by means of *(name, value)* tuples. It is however important to note that values can be, not only atomic values, but also may include sets and ranges to gain utmost flexibility in describing the content of an object. Such ranges can for instance be used to characterise the period of time when a given content is valid to be displayed on a live website. Using these two concepts, personalisation and context-dependent, multi-channel presentation are relatively easy to implement. It even becomes apparent that personalisation is just another form of context-dependent presentation as the user and their attributes can be defined as a context as well, for instance a *user context*.

In the remainder of this section, we will present the most important parts of our information model for content management in more detail. These information concepts have been developed in the OM object model [11]. OM is an object-oriented information model featuring powerful concepts such as *object hierarchies*, *collection hierarchies*, *binary associations*, *multiple inheritance* and *multiple instantiation*. Collections are used in OM for role modelling and the classification of objects. Associations provide bidirectional links as a potent stand-alone concept and can be used to semantically connect and navigate large collections of objects.

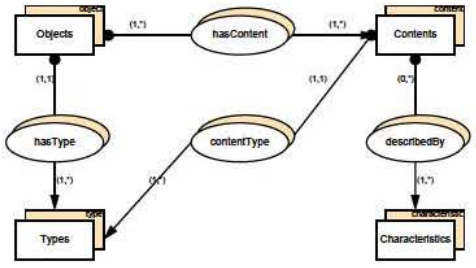

Figure 2. Content Model

The first part of our model shown in figure 2 deals with the management of content. The most important design decision in this model is the separation of an object into the *concept* of an object and the *content* of the object. Only by separating these two notions is it possible to build a system upon this model that meets the requirement of multi-format content. Another concept necessary to meet this requirement is the *characteristics* that are associated with the content of an object to describe it. To ensure type-safety, both the concept of the object and its content have to be of the same type, otherwise the object would be allowed to expose heterogeneous structure in differing contexts. Clearly this

is not a desirable property of a content management system and has to be prevented at this very low level.

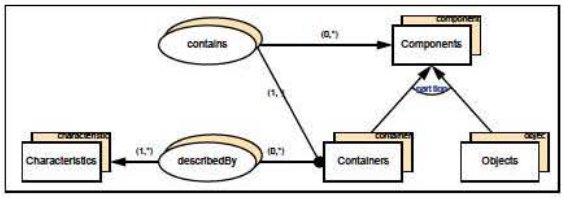

Figure 3. Structure Model

Figure 3 displays another core information concept that is part of our model. After having modeled the content as presented above, it must be structured and composed to be published on a website. We have chosen the simple, yet powerful model of *component* and *container* to build hierarchical structures over content. This design pattern [8] is well known in software engineering to generate complex object trees where components are the leaf nodes and containers are inner nodes. Employing this very basic pattern it becomes to possible to design structural hierarchies with arbitrary levels of composition. In the realm of web content management, one can imagine containers to be folders or pages. Generally, a container is a collection of objects that will be presented together such as a set of person objects on an employee overview page. Component objects on the other hand are the concrete data objects that make up the content of the website. Hence the differentiation between component and container is our expression of the separation between content and structure.

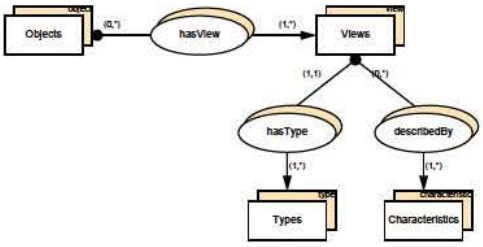

Figure 4. View Model

As motivated before, views are an important concept for personalisation and therefore have been incorporated as a core construct in the information model for content management. Figure 4 shows the part of our model that is concerned with views. A view is always connected to a single type to ensure type safety as a view references attributes and other properties of an object. The other assocation that a view takes part in links it to a set of characteristics that are again used to describe the view and enable the system to dynamically select the best-matching view to a given context at runtime.

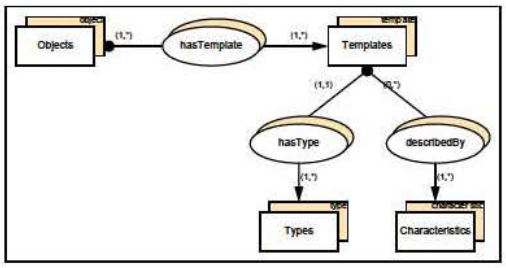

Figure 5. Presentation Model

Figure 5 gives an overview of the presentation part of our model. The first thing to notice is its striking similarity to the view model. On closer examination this makes perfect sense as the two concepts are closely related. When designing the layout of a webpage, one has to decide on the one hand what data should be displayed and on the other hand in what manner the data is to be presented. In our approach, views are used to make the first of those two decisions and presentation is then responsible for the second. The concrete implementation of different presentations is achieved in our system by means of templates. These are XSL templates [19] bound to a certain type of object and described by a set of characteristics. These templates can be used to transform the object content into any desirable markup before it is delivered to the requesting client. In selecting the appropriate templates to perform this transformation, once again the context of the client's software is used to generate the best-matching markup.

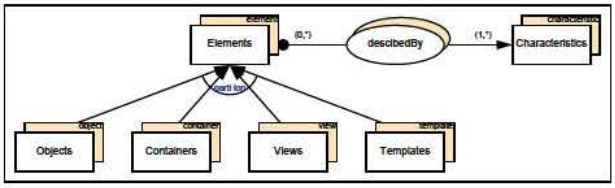

Figure 6. Elements of Content Management

Reviewing the models presented for *Content*, *Structure*, *View* and *Presentation*, it is apparent that these four constructs share common properties. All of them are, for instance, described by *Characteristics* to allow context-dependent behaviour. It is therefore useful to introduce the super-concept *Element* to unify these four types of objects as shown in figure 6. Profiting from this structural abstraction, it will be possible to uniformly address the problems of personalisation, user management and workflows with our model. The actual connection between these concepts however is rather a conceptual relatedness than a feature of the model. They are in fact the basic elements required for content management.

The workflow model we use to meet the requirement of a team of website authors with various user roles is shown in

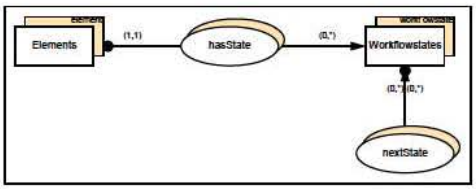

Figure 7. Workflow Model

figure 7. Each content object in our system is linked to a certain *workflow state*. This state controls how and whether the object can be published on the live website and who is permitted to edit or review it. Generally speaking, a workflow is a directed graph with states as its vertices and state transitions as its edges. Our model allows custom workflows to be defined as directed graphs by using the association *nextState* to model the possible edges between two vertices. As it is sometimes desirable to have workflow, not only for content, but also for structure, view and most of all presentation, we have decided to associate the construct *Element* with the workflow states.

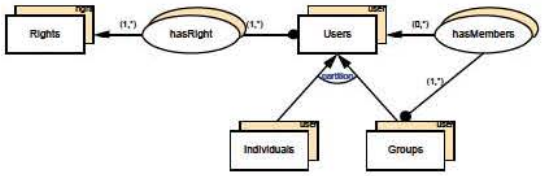

Figure 8. User Model

The last part of our information model that we describe here is the user part of the model shown in figure 8. As is apparent in this graphical representation, there are two kinds of users needed to meet the previously presented requirements. First there are *individual* users, then there are user *groups*. A user group may include a set of individual users, but it can also contain other groups. Thereby it is possible to build fine-grained hierarchies of users and user groups modelled after any situation existing in real-life. Both individual users and user groups may be assigned a set of rights. Profiting from the hierarchical organisation of users, the system may allow inheritable rights by checking the rights, not only at user-level, but also by traversing the user-tree to its root and checking the rights at every node encountered on this path.

Based on the model and concepts in this section, we have developed a prototype system that endeavours to meet all of the proposed requirements using the information concepts described. In the next section, we will give an overview over this system called *eXtended Content Management Server* or XCMServer.

5. XCMServer

Based on the information concepts and the model presented in the last section, we are currently developing a prototype system that will meet the above proposed requirements of content management. To implement the system, we have chosen the Java Platform as it offers a simple way of implementing custom server components. Hence the entry point for any request sent to our content management server is a servlet [9]. This servlet extracts context information from the request and delegates it to the content engine. Based on this context information the content engine assembles the content coming from various data sources into an XML document. The structure of this document is entirely based on the metadata stored in our system using the previously presented information concepts. Together with the XML, an XSLT stylesheet is dynamically generated as well. This stylesheet is also customised using context information about the target platform. In a final step, the XML document is transformed into the appropriate markup using this generated XSLT stylesheet. The result of this transformation is serialised and sent back to the client which renders the document.

Before starting work on the current implementation of our content management system, we have gained experiences from another prototype system that has been designed, implemented and completed in the context of the *eXtensible Infomation Management Architecture* (XIMA) [16]. Within this system, a number of tools have been developed such as a graphical user interface that helps designing websites with the system and enforces a structured development method for this process. This content management system however has not been satisfactory in supporting all of the presented requirements and we have therefore decided to extend it and build a second more complete system based on the information concepts presented in section 4.

As a platform for our metadata and content databases, we have chosen the OMS Pro [18] data management system. OMS Pro is an object-oriented database implemented based on the above presented OM model, providing support for all of its concepts. Describing objects by means of characteristics or properties is central to our approach to content management. We therefore have begun to extend OMS Pro to allow such annotations for objects in the core of the database. Hence it will be possible to manage objects that have many context-dependent incarnations and thus provide the notion of multi-format data at database level. Although this approach seems to be very powerful in the realm of OMS Pro databases it has to be noted that different solutions have to be found for other platforms where the option to extend the database system is not available.

Coupled with the concept of characteristics is the notion of context. When working on a document, the system has to find the best matching content by comparing its characteristics to the context found in the incoming request. As one can imagine, this can become quite complex when the system has to deal with a potentially large number of content axis. Another problem is that when matching context and characteristics, not only equality tests have to be performed, but also checks for interval and set inclusion. We therefore currently are developing algorithms to perform this matching efficiently based on the theoretical foundations presented in [17]. The computational complexity of these matching algorithms is central to our content management system as they have to be applied for every content, structure, view and presentation object involved in the generation of a document. We therefore give special consideration to the performance of this selection mechanism as it will directly influence the overall responsivness of the complete system.

As all pages in our system are generated dynamically from the database, another important factor that has great influence on the performance of the system is the caching strategy used on the server side. In our content management system, caches can be used at various levels of granularity. Not only is it possible to cache objects or queries at the database level, the system can also manage caches for partial or complete XML and XSLT documents and even fully generated pages that are known to change rarely. An overview over our experiences with such caches in web publishing systems is presented in [12].

With the completion of the extension of OMS Pro and the improved matching algorithms the revised content management system will meet the proposed requirements.

6. Conclusions

In this paper we have presented a number of requirements that a modern content management system should meet. These requirements are closely coupled to the set of user roles that exist in a team of people working together to create, design and maintain a website.

Existing solutions are numerous and vary greatly in terms of the underlying approaches. None of them however meet all of the presented requirements. We believe this is in part due to the lack of a clear information concept and model underlying these systems. We have therefore developed information concepts that should be used to address these challenges and introduced a model suited to build a content management system. Central to this model is a clear separation between *Content*, *Structure*, *View* and *Presentation*. These four concepts represent the elements of content management and are used in our model to uniformly realise personalisation, user management and workflows.

Finally, we have given an overview over our prototype implementation which serves as proof-of-concept to the presented model.

References

[1] A. Bongio, S. Ceri, P. Fraternali, and A. Maurino. Modeling Data Entry and Operations in WebML. *Lecture Notes in Computer Science*, 2001.

[2] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): A Modeling Language For Designing Web Sites. *Computer Networks (Amsterdam, Netherlands: 1999)*, 2000.

[3] Smile Les Motoristes Internet. (www.smile fr).

[4] J. Conallen. Modeling Web Applications with UML. White paper, Conallen, Inc., 1999.

[5] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. STRUDEL: A Web Site Management System. In *Proceedings, ACM SIGMOD International Conference on Management of Data: SIGMOD 1997: May 13–15, 1997, Tucson, Arizona, USA*, 1997.

[6] M. Fernández, D. Suciu, and I. Tatarinov. Declarative Specification of Data-Intensive Web Sites. In *Proceedings of the 2nd Conference on Domain-Specific Languages*. USENIX Association, 1999.

[7] J. Fulton. Introduction to the Zope Object Database. In *Proceedings of the 8th International Python Conference*, 2000.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley, 1995.

[9] J. Hunter and W. Crawford. *Java Servlet Programming*. O'Reilly & Associates, Inc., 1998.

[10] Microsoft, Inc. (www microsoft.com/cmserver).

[11] M. C. Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. *Lecture Notes in Computer Science*, 1994.

[12] M. C. Norrie and A. Palinginis. OMSwe: Integrating Full Web Publishing Support into an Object Data Management System. In *Submitted to WISE 2002*, 2002.

[13] Obtree Technologies, Inc. (www.obtree.com).

[14] T. A. Powell. *Web Site Engineering: Beyond Web Page Design*. P T R Prentice-Hall, 1998.

[15] C. S., F. P., and P. S. Data-Driven, One-To-One Web Site Generation for Data-Intensive Applications. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*. Morgan Kaufmann, 1999.

[16] B. Signer, M. Grossniklaus, and M. C. Norrie. Java Framework for Database-Centric Web Site Engineering. In *Proceedings of WebE'2001, 4th Workshop on Web Engineering*, 2001.

[17] Y. Stavrakas and M. Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In *Proceedings of CAiSE'2002, 14th International Conference on Advanced Information Systems Engineering*, 2002.

[18] A. Wuergler. *OMS Development Framework: Rapid Prototyping for Object-Oriented Databases*. PhD Thesis, Department of Computer Science, ETH, 2000.

[19] XSL Transformations (XSLT), 1999. (www.w3.org/TR/xslt).
[20] Zope Corporation. (www.zope.org).