Empowering Databases for Context-Dependent Information Delivery

Moira C. Norrie and Alexios Palinginis

Institute for Information Systems, ETH Zurich, CH-8092 Zurich, Switzerland
{norrie,palinginis}@inf.ethz.ch

Abstract. We present a web-publishing platform that was developed by integrating key constructs and operations for web content delivery into the core of an object-oriented database system. In this paper, we focus on the support for context-dependent delivery, inclusive of time-dependent delivery. A general model of context is based on application-specific sets of characteristics. We describe how web documents are dynamically composed from component instances, selected according to a best match of current context state and individual instance characteristics.

1 Introduction

Nowadays, a wide range of commercial tools and technologies are available to support the publishing of both static and dynamic data on the web. While systems such as Coldfusion, Interwoven and Vignette offer sophisticated tools and architectures to meet performance and scalability requirements, they lack a well-defined general model of the basic information concepts underlying web content management [2]. As a result, the continual demand for new requirements in this rapidly evolving domain leads to ad-hoc solutions and increases the complexity of both the system and the development process.

Our goal was to develop a general platform for web publishing based on a simple, but general, model that identifies the key concepts that define a web site. The platform is based on an object-oriented database system which was extended to support web site engineering and operation by integrating the constructs and operations defined by that model into the system. In this way, we empower the database system to not only manage domain-specific data such as *persons* and *publications*, but also to control the delivery of information through the dynamic generation of documents from database objects that define structure and presentation.

In terms of information delivery, it is no longer sufficient to support access through only desktop browsers. A web publishing framework has also to be able to cater for all forms of mobile and novel devices. In addition, it must support user preferences and context-dependent delivery. It is a matter of delivering *the right information, to the right person, at the right time*. It is therefore vital that the framework is flexible enough to adapt to, not only emerging (and even unanticipated technologies), but also the rapidly expanding interaction sphere of hypermedia. For example, in [4], we discuss issues of including digitally augmented paper as a first-class medium in hypermedia systems.

In this paper, we focus on our support for context-dependent information delivery, where the notion of *context* is a general one that can be defined in an application-specific manner in terms of a set of *characteristics*. These characteristics may include the client device, the browser, the language and style of presentation, the user or the access history. Further, temporal context is catered for by associating the components of a web site with temporal properties that control time-dependent information delivery.

We begin in Section 2 with a discussion of the requirements of context-dependent delivery and related work. In Section 3, we then present an overview of our web publishing platform OMSwe and the central notion of web elements. Section 4 describes in detail how the notions of context and state are handled in the system to produce context-dependent delivery. Finally, concluding remarks are given in Section 5.

2 Context-Dependent Delivery

Global and mobile access to information is of increasing importance. This, in turn, has led to requirements for systems that can adapt the delivery of information according to the access device and situation. The content and presentation of documents delivered in response to client requests may vary according to a whole range of factors such as the user preferences, the client device, the communication network, the place and time of the request, the previous information request and so on. Further, the content may vary both in terms of its structure and the format of the basic elements from which it is composed. For example, if we request information about a person via a mobile phone, we may send only essential properties such as phone number and email address, whereas, from a desktop browser, we would deliver, not only full background details, but also associated information such as their publications. Also the format of content elements such as images will depend on the access device and text elements may be adapted in terms of the language and also whether full text or summarised texts are presented.

Some projects adopt a fixed model of context. For example, in the mobile computing domain a number of projects on context-awareness focus mainly on location (see for example [6]). While others in this field have argued for more general models of context e.g. [7], it is still true that the focus in these projects is on the physical infrastructure required to provide context information and less on the effects of context in terms of information delivery.

Since our goal was to develop a general web publishing platform, it was important to have a general model of context enabling specific applications to define exactly what parameters define a context state.

For a given application A, we define its *context domain* C_A as a set of characteristic variables $\{c_1, c_2, \dots c_n\}$ for some $n \geq 0$. Each characteristic variable c_i will have an associated domain of values V_i. Then a characteristic is any given pair (c_i, v_i) for $c_i \in C_A$ and $v_i \in V_i$. A valid *context state* CS_A is any set $\{(c_i, v_i) | c_i \in C_A \text{ and } v_i \in V_i \text{ for } 1 \leq i \leq n\}$.

The notion of context that we define here is the same as that proposed for semi-structured data in [8], where they use the term *dimension* instead of *characteristic* reflecting the fact that they were influenced by the work on multidimensional programming languages. In [8], they present MOEM, an extension of the semi-structured data model OEM to represent multidimensional data, and describe how an MOEM document can be transformed to an OEM document under a specific context state. Similar ideas have been introduced into both HTML [9] and XML [1] documents to support context-dependent variants of documents.

To support such models of context in a web-publishing platform requires a database system that manages information about users, roles, contexts and presentations as well as the application domain. If web-publishing support is to be integrated into the database system, then a notion of state, and specifically context state, will have to be integrated, along with a mechanism for dynamically generating documents from stored data based on the current state. Further, it must be easy for application developers to specify the context domain of an application and context-dependent properties of the content to be delivered.

What we provide to the application developer is a general model of context and how it can be used to manipulate information, its structure and presentation. Our focus is therefore, not on adaptive web sites per se as described in [5], but rather on context models and mechanisms that could be used to support the development of such sites.

We conclude this section by commenting on the wide variety of commercial content management systems that have been developed to support both the development and operation of complex web sites. These systems emerge from the publishing community and, typically, the concepts in which they deal at the information level are primarily document concepts of text, image, URL etc. rather than with semantic concepts of a particular application domain. Thus they deal with an image of a given format, but have no information as to what it is an image of.

This lack of semantic information about content makes it difficult to ensure consistency and to link pieces of content together that represent the same logical entity. For example, if information about a person appears on several pages of a web site, the content editor has to manually keep track of the fact that the associated content actually belongs together as a semantic unit. This means that when information about that person is updated, or even the entire person is to be replaced, it is difficult to track where to make the necessary changes and tedious to make them. If the information is stored together in one place and the different instances that appear on the various web pages dynamically generated from that information, then it is easy to make changes and avoid inconsistencies. Note that this solution applies even if the actual instances differ greatly, for example, a link to a person based on their name, or a full description of a person inclusive of an image. The situation becomes much more critical when one starts to consider multi-format and multi-lingual content.

As a result, existing software solutions for web publishing tend to develop ad-hoc solutions to emerging requirements for context-dependent delivery. At first, entire parallel sites had to be developed for different languages or different access devices. This resulted in major development costs for WAP applications that, in some cases, never proved to be successful financially. With the introduction of XML support and attention to support for specific notions of context such as time and device, the situation is improving, but it still lacks the true flexibility that stems from a general underlying conceptual model.

3 OMSwe and WebElements

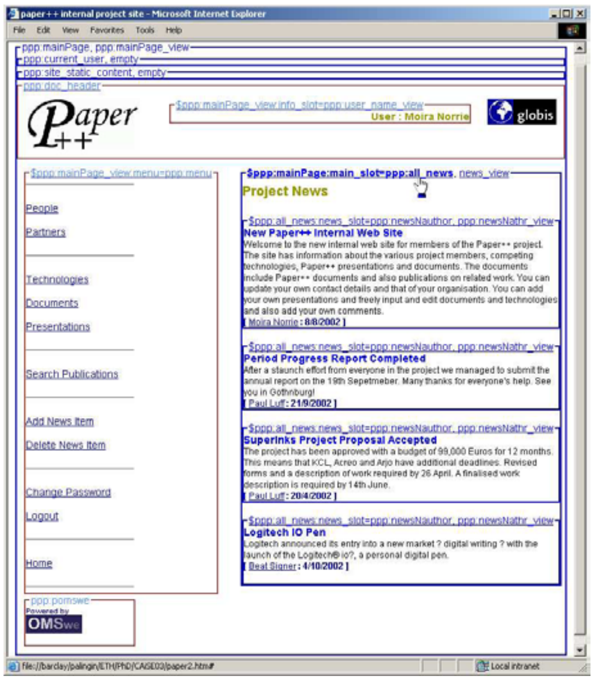
Our overall goal was to investigate what features need to be integrated into a database system to provide full web publishing support for both static and dynamic web sites. In line with our general approach to object data management, we wanted to keep the model both simple and flexible, introducing a minimal number of new concepts into our object data management system OMS Pro [3]. We were seeking an integrated solution that supported both the rapid development and prototyping of applications, supporting changes while ensuring consistency. Also, it was important that we catered for both the development of new systems from scratch as well as publishing existing databases on the web. Last, but not least, we wanted to support universal access from various forms of client devices and browsers.

We begin this section by describing some general features of the OMS Pro system, on which our web publishing platform OMSwe is based. OMS Pro is an object-oriented database system designed to support rapid prototyping and based on the OM model which supports role modelling through multiple instantiation and multiple classification, and also handles links between objects through a first-class association construct. There is a full operational model in terms of methods and triggers bound to types and general application macros bound to databases. The system is implemented in SICStus Prolog and operations may be implemented in either Prolog or our own language OIL (OMS Interaction Language). Not only application data, but also system and application metadata are represented as objects. This uniformity makes it amenable to the sort of extensions that we discuss in this paper where new functionality is associated with new system object types along with appropriate methods, triggers and macros.

As indicated in figure 1, a typical web page can be divided into a number of logical components such as header, footer, menu and the central main information component which displays the variable content. The page shown in figure 1 belongs to the internal web site of a project and the main component lists various news items, each of which is itself a component. Actually, in figure 1, the page is displayed under a debugging option of OMSwe and shows, not only the page content, but also the component structure. By clicking on the names labelling the component frames, the OMSwe objects representing components can be displayed.

A component can be specified in terms of its *content* and its *presentation*. The content defines the dynamic information contained within the component. This can be a combination of information retrieved or generated from database objects and references to other nested components. In this way, the content is defined in terms of a content hierarchy, where each node is bound to a database operation (macro) that generates data. The web publishing framework is based on XML technologies, which means that content generated data is represented in XML.

The presentation specification of a component is defined in terms of *templates* which are defined in XSLT. The templates determine how the content generated data is to be presented, specifying also the appropriate markup and static content (text, images etc) to be included in the target document. For example, it is at the template level that the developer specifies whether XHTML, WML (Wireless Markup Language), Voice XML or some other markup should be included in the document to be sent back to the client. Both the content and presentation of a component may be subject to con-


paper++ internal project site - Microsoft Internet Explorer
File Edit View Favorites Tools Help
ppp:mainPage, ppp:mainPage_view
ppp:current_user, empty
ppp:site_static_content, empty
ppp:doc_header
Paper++
$ppp:mainPage_view:info_slot=ppp:user_name_view
User : Moira Norrie
globis
$ppp:mainPage_view:menu=ppp:menu
People
Partners
Technologies
Documents
Presentations
Search Publications
Add News Item
Delete News Item
Change Password
Logout
Home
ppp:pomswe
Powered by
OMSwe
$ppp:mainPage:main_slot=ppp:all_news, news_view
Project News
$ppp:all_news:news_slot=ppp:newsNauthor, ppp:newsNathr_view

New Paper++ Internal Web Site
Welcome to the new internal web site for members of the Paper++ project. The site has information about the various project members, competing technologies, Paper++ presentations and documents. The documents include Paper++ documents and also publications on related work. You can update your own contact details and that of your organisation. You can add your own presentations and freely input and edit documents and technologies and also add your own comments.
[Moira Norrie : 8/8/2002]

$ppp:all_news:news_slot=ppp:newsNauthor, ppp:newsNathr_view

Period Progress Report Completed
After a staunch effort from everyone in the project we managed to submit the annual report on the 19th Sepetmeber. Many thanks for everyone's help. See you in Gothnburg!
[Paul Luff : 21/9/2002]

$ppp:all_news:news_slot=ppp:newsNauthor, ppp:newsNathr_view

Superinks Project Proposal Accepted
The project has been approved with a budget of 99,000 Euros for 12 months. This means that KCL, Acreo and Arjo have additional deadlines. Revised forms and a description of work required by 26 April. A finalised work description is required by 14th June.
[Paul Luff : 20/4/2002]

$ppp:all_news:news_slot=ppp:newsNauthor, ppp:newsNathr_view

Logitech IO Pen
Logitech announced its entry into a new market ? digital writing ? with the launch of the Logitech® io?, a personal digital pen.
[Beat Signer : 4/10/2002]

file://barclay/palingin/ETH/PhD/CAiSE03/paper2.htm#
Local intranet

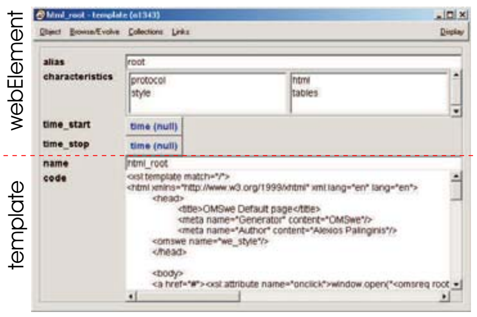
Fig. 1. WebElements of a Project Web Site

text dependencies. We therefore introduced the notion of a *web element* to represent an element of web component specification, whether it is a specification of content or presentation.

Web elements are represented in OMSwe as objects of an introduced system type `webElement`. The subtypes `content` and `template` are used to represent content and presentation specifications, respectively. An example of a `template` object, showing the `webElement` part and the `template` part is given in figure 2.

We achieve context-dependent web pages by selecting web elements according to the current context state. A web element is actually defined by a set of `webElement` objects with the same alias and type (content or template) but different *activation contexts*. An activation context defines the set of context states in which the `webElement` can be used and it is specified in terms of a set of characteristics and a valid start time and stop time as shown in figure 2.

For any given request, the particular `webElement` object used will depend on a best match between the current context state and the set of characteristics and time properties of the set of `webElement` objects with the specified alias. The set of characteristics can be one or more of those defined in the application's context domain as part of a special configuration object. The example of figure 2 has two characteristics `(protocol,html)` and `(style,tables)`. This means that it will match to con-


webElement
template
html_root - template (o1343)
Object Browse/Evolve Collections Links
Display

alias: root

characteristics: protocol html; style tables

time_start: time (null)

time_stop: time (null)

name: html_root

code:

```
<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
        <head>
                <title>OMSwe Default page</title>
                <meta name="Generator" content="OMSwe"/>
                <meta name="Author" content="Alexios Palinginis"/>
        <omswe name="we_style"/>
        </head>

        <body>
        <a href="#"><xsl:attribute name="onclick">window.open("<omsreq root
```

Fig. 2. Template Object

text states where the protocol and style context variables are set to “html” and “tables”, respectively. For all other characteristic variables of the context domain, no values are specified which means that it matches any value. The specification of defaults means that there is always at least one `webElement` object that matches. The matching process is performed by the OMSwe engine and is described in detail in the next section.

The `time_start` and `time_stop` attributes are used to specify the active period for the `webElement`. If they are both set to null, the `webElement` will always be active. The ability to define active periods for a `webElement` object is useful in many cases where information or operations should appear on a web site for limited periods of time, enabling it to be prepared in advance and requiring no manual intervention to alter the appearance of a web page (or even an entire web site) at a certain point in time. For example, after the submission date for conference papers is closed, the submission page will automatically “close”, offering instead a page saying that the deadline has passed. In effect, the active period for a `webElement` is a special form of characteristic that belongs to the context domain of all applications and is specified in terms of two values that define a time interval, rather than a single value. For recurring events, these time intervals can be updated by the appropriate the request handling operations in the database.

Thus, the set of characteristics of a `webElement` together with the attributes for the valid timespan, define a valid context for the `webElement`. We have already pointed out that our notion of context is exactly the same as that described for semi-structured data in [8] and they also consider temporal validity as a special form of context validity.

An XML generator and associated tools were integrated into the database system to allow database objects to be published as XML and then transformed into the appropriate presentations using XSLT templates. We have a general XML schema for OMS objects and provide various options in the generator functions to specify such things as the type view of an object in the case of multiple instantiation and whether or not attributes with null values should be included. As mentioned above, relationships between objects are represented by an association construct and it is frequently the case that we want to include attributes of associated objects in a web component. We therefore introduce a notion of embedded object views in the XML generator to support this.

To ensure a correct matching of contents and templates, we introduced a notion of *content schema*. Templates are associated with content schemas to ensure both type and structure correctness. It is possible that a content schema has more than one template associated with it and, at the same time, a template may be used to visualise more than one schema. Each schema, however, has a default template specified.

4 Context and State

In this section, we describe how the OMSwe engine composes document content through the selection of `webElements` objects based on the current context state.

A system *state* consists of three parts — a set of slot bindings, a set of parameter values and a set of characteristics. The slot bindings are to bind component slots to actual components. In this way the structure of the document is composed from smaller content parts based on the current state. For example, in the case of figure 1, the main component is actually a *component slot* and is dynamically bound to different components to reflect the various contents that appear in the main part of the documents of the web site. The parameters may either be component parameters or parameters for a database operation. As described previously, the context state is the set of characteristics and forms part of the overall system state.

State transition is based on user activation which generally is either link selection or form submission. State information can be encoded in links by means of URL rewriting and in forms by hidden fields. To simplify the process, OMSwe provides special markup as part of OMSML to define links and also forms, together with any state transition information, and to generate these URL encodings. We should notice here that application reactive behaviour should be implemented in the internal server-side session information. This is semantically the appropriate place to handle application specific issues and not on the user

Since the state can often involve many different variables, we further provide options to define an *absolute request* or *relative request*. In the case of a relative request, the new state inherits the bindings, parameters and characteristics of the current state, only overriding the variables specified explicitly in the request. In the case of an absolute request, the new state is actually relative to the default state defined in the configuration object. Effectively, this means that all contextual information will be lost, unless explicitly included in the absolute request.

Figure 3 presents an overview of the state transition process and the resulting document generation. On the left, we indicate the request state and the two options of generating the new state depending on whether it is a relative or absolute request. Once the new state has been generated, the system starts from the root component and, following references to nested components, builds up the document structure and presentation. For each web element referenced, it will select the `webElement` that best matches the current state and this is indicated in the right of the figure by the dark shading of one `webElement` at each node of the content structure. Note that, in general, the selection of a particular `webElement` at one level could have a significant effect on the resulting

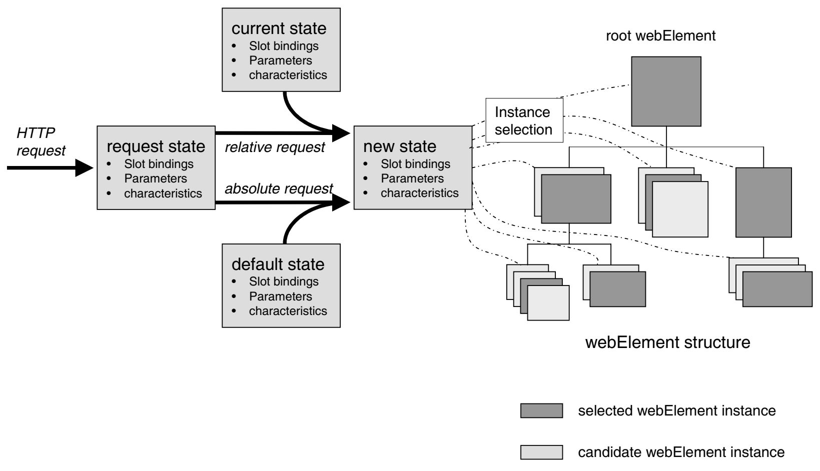


Fig. 3. State Evaluation and Context Matching

document structure, since different `content` objects may have different associations to other `content` objects and hence their content subhierarchies may be quite different. As a simple example, we could generate a second `content` for the root which is associated with a simple HTML template with the message that the site is off-line and associated with a timespan which specifies our next major site update.

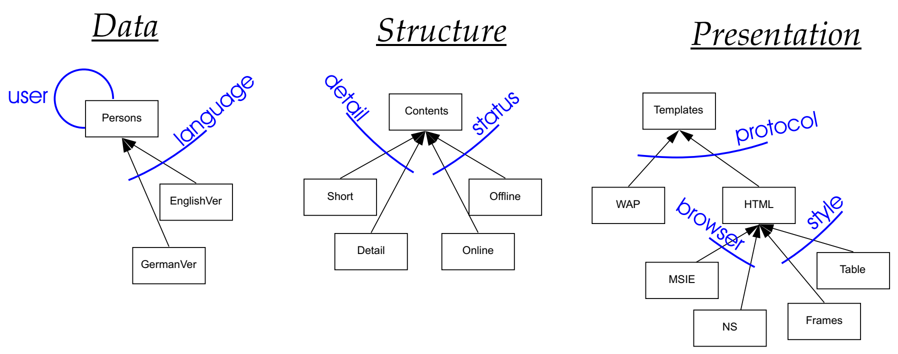


Fig. 4. Context Examples

The context aware mechanism can influence, not only structure, but also data and presentation. As an example, in figure 4, we represent with arcs the different characteristics of an application. With respect to structure, we have contents aimed at large or small devices and also the on/off-line scenario described above. With respect to presentation, the templates used will depend on the protocol and, in the case of html, are further selected based on characteristics browser and style. Finally, the actual data content may depend on the language characteristic. Different versions of person objects will be associated with different language characteristics. Another common characteristic is the user. Accessing data is subject to user access rights for the given user in the current context. The user characteristic is often used to trigger different structural behaviour too. We have a special root `webElement` for the user characteristic with the value `null`. This component will be used when the user is anonymous and for secure sites this is nothing else than the root component of the login page.

We mentioned before that system state transitions are based on user intervention. By browsing the site, the user alters the system state by requesting different components to be placed in different slots. Of course, the user is not aware of this behaviour, but rather it is the developer who ensures the encoding of this information in the links. Many context characteristic values are assigned automatically without direct user intervention. For example, the characteristics of protocol and browser are automatically extracted from the HTTP headers. Even the language and component detail characteristics could be extracted from the headers, but they could also be modified explicitly by the user.

We therefore define different classes of characteristics which influence the matching and inheritance behaviour. The first to discuss are the major/minor characteristics. These are characteristic pairs such as protocol and browser that build a hierarchy. The relationship between major/minor pairs can be specified as *weak* or *strong*.

In the case of a weak relationship, the minor characteristic will only be considered if the major characteristic value matches. We use a '-' to indicate a weak pairing, e.g. `de-ch` could denote German language and Swiss region in a language/region pairing of characteristics. When no content is defined for `de-ch`, then the region information `ch` can be omitted in the matching process and the `webElement` objects defined for only `de` will be considered. Further, when the state defines the language as `de` (without specifying region), all content defined for `de` or `de-ch` are considered as candidates.

In the case of a strong relationship, both characteristic values should match the state for a `webElement` object to be a candidate. For example, consider the browser/browser_version pair. We use a ':' to indicate a strong pairing, e.g. `msie:5.0` denotes Microsoft Internet Explorer version 5.0. If a template is defined for `msie:5.0`, then the current state must specify exactly `msie:5.0` to obtain a match. Since we cannot assume compatibility between all versions of Microsoft Internet Explorer, it would not be appropriate to match this to a context state that simply specifies `msie` as the browser value.

The major/minor characteristic class can be combined with range characteristics and list characteristics. These two constructs can be used in the context definition of a `webElement` to identify a context area. For example, a range definition could be used to specify that one template is compatible with browsers of version `msie:(3.0,5.0)`, while another can be used only for `msie:5.0`.

A web element could further be defined for multiple language/region pairs. For example, one template with `[de-de, de-au]` applies to a site to handle commercial transactions for Germany and Austria, while another with `de-ch` deals with the special case of Switzerland which is not in the European Community. A combination of range definition and lists is also allowed.

A state typically defines many characteristics which opens the issue of defining a best match algorithm. Characteristics are defined as mandatory or optional. For example, the protocol characteristic is mandatory. It would make no sense to send a WAP document to an HTML browser just because the content happens to match all other characteristics such as language, region and style. On the other hand, if no content exists for the Greek language (state defines language=gr), it would make sense to use an English version.

The matching algorithm is further aware of a characteristic prioritisation defined by the developer. It might be the case for instance that we want to give a higher priority on matching the correct language over a less significant characteristic such as the presentation format. We define therefore $0 < p(c_i) \leq 1$ as the priority of a given characteristic.

We are now ready to extend the definitions from section 2 by adding for any webelement k (representing either data, structure or presentation) the webelement profile I_k as a set of $\{(c_i, d_i) | c_i \in C_A \text{ and } d_i \in CD_A \text{for some } i \in 1 \leq i \leq n\}$. The characteristic definition pattern CD_A is any valid definition pattern of a characteristic value such as the above mentioned weak, strong, range and priorities lists.

The matching algorithm will search for the current state the webelements of a given class for the one that best match according to their profiles. For each characteristic pattern definition a characteristic weight $W_{k,i}$ will be calculated according to the following table:

Pattern/Class	State	Webelement k Profile	Weight $W_{k,i}$
mandatory	X	Y (Y $\neq$ X)	0 (not considered)
any	major	major	1
weak	major-X	major-X	1
weak	major-X	major	2
weak	major	major-X	2
weak	major-X	major-Y (Y $\neq$ X)	2
strong	major:X	major:X	1
strong	major:X	major:(A,B)	1 if X in [A,B]
strong	major:X	major	2
any	X	(A,B)	1 if X in [A,B]
priority list	X	[A,..,X,..,B]	$\text{pos}(X)^3$

The matching algorithm will sum the rankings results for all characteristics defined for each `webElement` for the current context weighted by the characteristic priorities, thus the weight of an item k for the current state is:

$$W_k = \prod_{i=1}^{n} p(c_i) * W_{k,i}$$

The one with the highest ranking (lowest number greater than zero) will be chosen. In the case that no `webElement` could be chosen, the default will be used. The current algorithm gives lower ranking when matching values defined deeper in a list construct as indicated in the last row of the table. In this way, we assign the list a priority semantic. Furthermore, the weight of priority is much less than the weight from matching first choice characteristics. In this way, `webElement` objects that match some first choice characteristics will be preferred over `webElement` objects that match even more characteristics but are defined lower in the priority list. The priority weight can be defined differently for each characteristic giving the possibility to even consider the list values as equal to each other.

The number of optional matched characteristics and the matching position in the case of list definitions influence the ranking. In the rare case of equal total weights for two or more candidates, a random or default `webElement` can be used.

Finally, we stress that with this approach it is possible to have a very rich model of context that takes into account many factors, without having to define a unique `webElement` object for each possible combination. This not only reduces the development time, but also unnecessary replication.

5 Conclusions

We have described the general concept of web elements that were integrated into the object data management systems OMS Pro to provide a general web publishing platform. A key feature of the system is its support for global access to information. This means that it can adapt both the content, the structure and the presentation of documents according to the properties of the user, the client device and the application through the use of a single general concept. Specifically, each application can specify a set of characteristics that define contextual factors that may influence the information delivered in response to a user or application request.

The OMSwe system has been used to develop a number of operational web sites and also in teaching. It has also been used to develop a general web interface for OMS databases as well as development and debugging tools for the OMSwe system itself.

References

1. M. Gergatsoulis, Y. Stavrakos, and D. Karteris. Incorportaing Dimensions to XML and DTD. In *In Database and Expert Systems Applications (DEXA'01)*, Munich, Germany, September 2001.
2. M. Grossniklaus and M. C. Norrie. Information Concepts for Content Management. In *Proc. Intl. Workshop on Data Semantics in Web Information Systems (DASWIS 2002)*, Singapore, December 2002.
3. A. Kobler, M. C. Norrie, and A. Würgler. OMS Approach to Database Development through Rapid Prototyping. In *Proc. 8th Workshop on Information Technologies and Systems (WITS'98)*, Helsinki, Finland, December 1998.
4. M. C. Norrie and B. Signer. Issues of Information Granularity and Semantics in Cross-Media Publishing. In *Proc. 15th Conference on Advanced Information Systems Engineering (CAiSE 2003)*, June 2003.
5. Mike Perkowitz and Oren Etzioni. Towards adaptive Web sites: conceptual framework and case study. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1245–1258, 1999.
6. Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.
7. Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999.

8. Y. Stavrakas and M. Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In *Proc. 14th Conference on Advanced Information Systems Engineering (CAiSE 2002)*, Toronto, Canada, June 2002.
9. W.W. Wadge, G.D. Brown, M.C. Schraefel, and T. Yildirim. Intensional HTML. In *Proc. 4th Intl. Workshop on Principles of Digital Document Processing (PODDP'98)*, March 1998.