Abstract

As originally conceived, the World Wide Web was intended for the purpose of sharing information. Many websites realise this aim by publishing pages from a data repository which supports browsing and searching. Building such a site is challenging in the first place, but since the information to be published will be spread among the community who wish to share the data, maintenance is an even greater challenge. In this case, the developers must continuously solicit, receive, moderate and integrate new information into the web site. The additional task of moderation is considerably more time consuming even than the initial web site construction and requires appropriate tools to expedite the process.

This paper describes the prototype of an architecture for constructing web sites which take part in a cycle of publishing and receiving data, in which the pages also solicit fresh information by the use of forms and e-mail. The architecture facilitates both site construction and maintenance, by allowing the date to be declaratively specified and then generating the data repository, web pages and moderating program automatically.

1. Introduction

As originally conceived, the World Wide Web was intended for the purpose of sharing information. Physicists at CERN wanted to share their research data and needed an infrastructure with a usable interface to do so. Today there are many web sites whose purpose is to make available a coherent body of information on a particular topic. The sophistication of such web sites can range from the massively hyperlinked and indexed Internet Movie Database to those many useful small scale sites which list the research activity in some area based on the work of a research group. They range in subject matter from serious academic activities (e.g. Grid projects), through intermediate sites which combine academic and leisure interest (such as the Jane Austen information page) to the completely frivolous (pezcentral).

Indeed, we can look forward to a time when there is a web site corresponding to every area of human interest, provided that someone has access to the information required and is prepared to take the time and trouble to build and maintain it. Such web sites start from one or more people turning a collection of information they have into a static web site, but quickly evolve into a set of pages largely generated from an organised repository. Sometimes, the information to be shared will be easily available to the authors, which means that the web site can be built by one group acting alone. More typically, however, the information will be spread among the interested community so that the web site maintainers will generally only have incomplete information. In this case, the authors must solicit, receive, moderate and integrate new information into the web site. Examples of sites adopting this approach are audiorocket.com which allows new musicians to publicise their work and freshfolio.com which performs the same role for emerging artists.

This process usually involves a single point of invitation on the site for visitors to submit fresh information by mail. The maintainer must then sift through submissions and engage in a moderation process to remove illegal or offensive material; to correct errors; to merge different information; and then to integrate this with the existing data repository. Clearly this is a time-consuming repetitious process. The work reported here attempts to improve the situation by providing a more sophisticated structure for publishing and receiving data together with tools to assist the maintainer. A live repository is used to generate the web site, the pages of which solicit fresh information by use of context-dependent forms or e-mail. Submissions end up in a second (transit) repository and the moderating program controls the transfer of data to the live repository.

The second important contribution of the work presented here is that the structure described in the previous paragraph is regular and completely independent of the area of interest. This means that by using a sufficiently rich data model, all of the components can be automatically generated from a schema describing the web site information. The repository, the site structure and the middleware which generates the web pages can also be generated since precisely the same programming structures are required for any web site of this kind.

As such the prototype falls into the category of declarative web site management, typified by Araneus [1], Strudel [2] and Tiramisu [3]. The work discussed here extends these products in considering the web site to constitute not just the web page structure, formatting, software and data repository, but also the off-line maintenance software. In a site of this kind, the moderating program is as important component as any of the stylesheets or page generation software. If moderation is time consuming, the site will eventually be allowed to fall into disuse. Thus, it is important that the
moderating program also be generated from the
declarative model and this is achievable since it will vary
only in the types of the data fragments it is manipulating.

The paper is structured as follows. The next section
describes an architecture for building information bearing
web sites and is followed by a section discussing how this
can be extended to allow collaborative input. The main
section describes how instances of such an architecture
can be automatically generated and this is followed by a
discussion of related work and work yet to be carried out.

2. Information Bearing Web Sites

Web sites of this type consist of a number of different
kinds of page some of which, e.g. home or help pages,
may be statically written, but most will be dynamically
generated from a data repository. Two kinds of page are
numerically dominant, therefore being the most time
consuming to maintain. The term collection page will be
used to discuss pages which list all of the entities of a
particular type with each entry containing a hyperlink to
further information. The term entity page will be used to
refer to the kind of page which displays information about
one entity. This term has been chosen since such a page
will correspond to an entity held in the repository from
which the web site is generated.

The repository may be any data holding structure, but
our work has concentrated on the content being held as a
mixture of relational and XML data. The nature of this
repository is discussed in detail elsewhere [4], but,
briefly, it is a typed hybrid integrated structure in which
XML and relational data connect by foreign key and
XPATH references. Abstractly, the repository will be a
collection of entities each of which has two key values to
help locate it - a unique database key and a human
intelligible key not necessarily guaranteed to be unique
(of course these may be the same in some cases). When
accessed, the page will be generated by middleware as
shown in Figure 1. The hyperlink sends a database key to
the server. The server uses this to extract the data about
the entity from the repository and then merges this data
with an HTML template specific to this kind of entity.

![Figure 1. The Generation of an Entity Page](image)

The repository structure to support this activity
consists of a number of entity types, the schema for which
is designed using a data modelling technique such as
Entity Relationship or UML models. The middleware
program will be constructed to expect exactly the set of
types in the database. It will have a set of templates for
exactly those types. We could therefore reasonably
expect that the data modelling design could be used not
only to generate the repository schema, but also the
middleware program and the set of templates. In fact
each template consists of two parts: a mapping from the
data structure to the page layout and a formatting style.
The former can be generated in such a way as to link with
the latter as one of a number of selected stylesheets,
whose structure can also be generated for later
completion. The issue of automatically generating the
software from the schema is the topic of the Section 4.

3. Collaboratively Built Web Sites

The next issue is the requirement to permit the
community to collaborate in populating the repository
from which the web site is generated. Typically this is
achieved informally with the web site gathering this
information and adding it to the web site in an ad hoc
manner. The architecture presented here has a more
systematic method of eliciting information which allows
the streamlining of the information flow from the person
with the knowledge to the repository (and hence to the
web site). Each page is extended with invitations to
inform the web site maintainers with more knowledge.
Two mechanisms are provided: e-mail links and form
links, both of which ask the visitor "Can you tell me
anything more about this entity?" or "Do you know about
any more of these?". The e-mail message enables the
visitor to send information as plain text from which data
can be copied, while the form will provide more structure
and hence support the automatic process of assimilation
more easily. Both are accompanied by the date and the
identities of the page from which they have been sent,
and the sender. These are jointly called the incoming
message metadata which provides context for the inter-
pretation of the message content. The material provided
by the visitor ends up in a secondary repository, termed the
transit repository. In the system presented here, the
form data is stored relationally and the e-messages are
stored in a mail file. The e-mail messages may be
accompanied by multimedia attachments and the
incoming repository since it uses a mail file is structured
so that attachments are easily stored.
The two repositories are tied together by a moderating program which permits the site maintenance staff to view the incoming material, edit it and move it to the live repository. Figure 2 shows the cyclic nature of the overall structure.

The moderating program is loaded with the data from the incoming repository and has separate parts for accessing the two kinds of transit data, one for e-mail messages and one for form data. It also has two separate parts for placing the accepted data in the live repository, one for putting the data in a table and one for putting data in an XML document. The combination leads to four different uses of the interface:

- e-mail-XML: An e-mail reader permits the maintainer to select text from the e-mail message and copy it into an XML skeleton.
- e-mail-relations: The text can be copied into a form which causes a database update when complete.
- forms-XML: This transfer is semi-automated as the elements of the form correspond to elements of the XML document. The data can subsequently be edited and the maintainer can eventually add the document to the live repository.
- forms-relations: The incoming data is automatically placed into one or more forms which, if accepted, cause a database update.

Figure 3 shows how this works in practice. The frame for viewing transit data is shown at the top left. Using this, the moderator can pick an incoming form and view the data submitted together with the message metadata. On each form is an accept button which sends the form data to editing windows which precursor submission to the live repository. In accepting the form, the data will be normalised and placed in one or more editable forms or an editable XML document appropriately tagged. The frame for viewing incoming e-mails shown bottom left shows the message and allows text to be selected for copying either into the XML document or into fields in a form.

The right hand side shows the second half of the moderating process. At the top is the window which controls data entry into the relational part of the live repository. Forms are created either automatically from incoming form data or manually by the moderator. Data entered there can be edited and eventually a submit button is pressed to send the data to the live repository. Partially edited data is held persistently between runs of the moderating program, so that the moderator can pause the site update process at any time. XML documents are edited as shown in the lower right corner. The frame gives the maintainer facilities to add new tags, edit the data and ultimately send the document to the live repository.

The moderating program therefore has a structure which is based on the nature of the data being managed – the repository schema. It is to the nature of the generation process that the paper now turns.

4. Building Such Web Sites

The important characteristic of the architecture is that it is entirely regular and will not vary from web site to web site. It therefore becomes possible to generate sites automatically. This section describes how this is done.

The creation of such a web site starts from the specification of an enriched data schema. It is no longer appropriate to describe the schema as representing the repository because it must do more than that – it must also describe the structure of each kind of web page (which data fragments appear on which pages), the forms that appear on the web pages and the nature of the transit repository. Having achieved this, the structural parts of the web templates, the moderating program and both of the repositories can be generated from the schema.

The schema is entered as a collection of entities, their attributes and the relationships between them. The schema structure is illustrated in Figure 4 for the example of a library wishing to display its stock. The principle entity types are ones representing the collections of books, authors and publishers.

In this simple example, several aspects of the data are indicated by the various columns. The first three columns describe the entity type and the names and domains of its components - domains can be atomic (e.g. number, date or string), a MIME type or one of the other entity types. The next column determines whether the data is to be stored relationally, as XML or (in the case of multimedia data) as a file. Then follows two columns identifying which fields function as the two keys – a unique database key used in all of the querying, and a human intelligible key which is not necessarily unique. The cardinalities of the relationships are in the next column. The final columns determine how the data is deployed on the web site. The ColPage column determines which fields will appear on the collection page of this type – the human detectable key field can be expected to appear, but other fields can appear as well. The EntPage column determines which fields will appear on an entity page of this type. The FormIn column determines which fields will appear on a form for submitting fresh information.
Looking at the particular data, we see that Book has a ready made database key in the ISBN, but the title has been chosen to be the field that web site visitors are expected to use - the collection page will list both. The entity page for books shows all the attributes, while the form asks visitors to fill everything except page length. Authors have a system generated ID which is always hidden – their name is what people use and this appears on the collection page. As well as simple data, the type can handle an image field, although the image subtype is left undefined since the same field might reasonably hold files in a variety of image formats. The meaningful attributes appear on the form and the entity page. Publisher data is held in XML and again has an automatically generated ID which is never visible. All the other data appears on the entity page the form. Relationships are entered in both directions and are indicated as inverses (not shown here). Given this information we are in a position to generate the web site.

The live repository is generated using standard database techniques. Stripped of its web characteristics, this is an ER diagram and can be transformed into a hybrid relational and XML schema fairly simply [4]. There are also extra tables added to hold details of the source of information. The principle ones are the Contributor and Contributions tables which record the contributor and date of each contribution.

Since some of the components of an entity consist of multimedia files, the repository also has a collection of file directories, one for each entity type. The individual files are numbered using the entity identifier and a secondary number to distinguish multiple multimedia files for a single entity.

The collection page template is parameterised by the entity type, the database and user keys and any variables that have been specified to appear in the collection page. A similar template is constructed for the entity pages, but this time the heading refers to the human intelligible key and the table entries are all of those fields which are listed as being in the entity page. Where those fields represent relationships, the same fields shown on the collection page are used and these are put in as hyperlinks in the same way as on the collection page. If the relationship is one:one or many:one then a single table row suffices, otherwise a sub-table is created. Multimedia data is added using <img> and <embed> tags.

The form page displays a form with all of the fields indicated in the schema being available for entry. The precise nature of the form controls depends upon the domain of the field being requested. Current values of any fields are sent in the form so that the visitor can see what information the site already has.

The transit repository consists of two parts: a POP3 mail file which has no site-specific structure and a database of tables – one for each form type. The tables are un-normalised as normalisation happens as part of the moderation process. Their structure is completely described from the same metadata as the forms themselves and so is easily constructed from the schema.

The moderation program has four components as indicated in Figure 3 and the components are generated as follows:

- The e-mail reading component consists of a panel with the message subjects, dates and so on. Selecting one of these opens another panel with the message in it. All of the message metadata are extracted and held in ready for use in the writing phase. In particular, the location of the page from which the e-mail is requested is extracted from the subject. Fragments of the message can be copied from this panel to one of the writing panels. Each message can be deleted, retained in the current list or stored in a mailbox. Attachments can be opened or transferred.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Field</th>
<th>Domain</th>
<th>R/X/F</th>
<th>DKey</th>
<th>HKey</th>
<th>Cardinality</th>
<th>CollPage</th>
<th>EntPage</th>
<th>FormIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book</td>
<td>ISBN</td>
<td>Varchar(20)</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>Title</td>
<td>Varchar(30)</td>
<td>R</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>Date</td>
<td>Date</td>
<td>R</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>PageLength</td>
<td>Number2</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>Author</td>
<td>Author</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>Publisher</td>
<td>Publisher</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>ID</td>
<td>Number2</td>
<td>R</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Name</td>
<td>Varchar(20)</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Nationality</td>
<td>Varchar(10)</td>
<td>R</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>BirthDate</td>
<td>Date</td>
<td>R</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>DeathDate</td>
<td>Date</td>
<td>R</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Photograph</td>
<td>image</td>
<td>F</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>Wrote</td>
<td>Book</td>
<td>X</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Publisher</td>
<td>ID</td>
<td>Number2</td>
<td>X</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Publisher</td>
<td>Name</td>
<td>Varchar(20)</td>
<td>X</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Publisher</td>
<td>Address</td>
<td>Varchar(50)</td>
<td>X</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Publisher</td>
<td>Phone</td>
<td>Varchar(20)</td>
<td>X</td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Publisher</td>
<td>Publishes</td>
<td>Book</td>
<td>X</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. A Schema for Collaborative Web Site Generation
appropriate XML elements or normalised relational forms is easily identified from the schema.

- The relational writing component also has a panel which lists the partially completed forms created either from an input form (with the same metadata) or as a new form which can be created at any time. At any time a form can be accepted or rejected. When it is accepted, it is passed to the live relational database for insertion or updating and an entry is made in the Contributions table. If it is intended to merge data from two sources into one form, then each is brought in and saved in turn. In this way two Contributions records are created.

- The XML writing component also has a panel which lists the incomplete XML documents. At any time, new blank documents can be created for any entity type which is mapped to XML in the repository. Data can be added from the input in one of two ways – tagged or untagged. The former inserts one of the recognised tags from the schema around the text as it is inserted, the latter assumes that the tags will be inserted separately. At any time the XML document can be saved into the appropriate directory. Thus the moderation program can also be generated automatically given the schema, because the only site specific data it needs are live repository metadata which determines the input forms and their fields, the live repository structure and the relationships between the two.

5. Conclusions and Future Work

The paper has presented the outline of an architecture for the declarative specification of web sites which publish information and invite visitors to add further information. The architecture has been constructed to be regular allowing each aspect to be generated automatically from the declarative data model, while the appearance of the pages can be quickly and consistently modified by the use of stylesheets. Furthermore, the web site maintenance software, in this case a program used to moderate the submitted data, is also generated systematically.

Although there are an extensive number of web tools for collaboration, tools and techniques for supporting information sharing web sites of this kind are less common. The present work has some similarity to the mushrooming field of grid computing [5], although the present work is clearly significantly less ambitious. Collaboration in terms of the web is mostly intended to support the sharing of activities rather than information in this way. For instance the PIÑAS project [6] is building software to support a web community jointly writing a document, while Matskin shows how agents can be used to support collaborative advertising [7]. All of the 44 products surveyed by Bafoutsou and Mentzas [8] support activity sharing of one kind or another rather than information sharing. Products such as Sitescape [www.sitescape.com] are also targeted towards supporting on-line collaboration activities. The work discussed seems therefore to be introducing novel support for an increasingly important information sharing process.

There are many enhancements which need to be made to this architecture. In particular, it is intended to enhance the moderation of e-mail using information extraction techniques. The main goal however, remains the construction of a web site design tool which permits the declarative specification of families of web site. Sports sites, E-commerce systems and Virtual Art Galleries constitute three such families. These families share some common structures, but also include individual elements. The intended tool will allow the site designer to choose a family and then include such site as elements as required and then have the site generated automatically.

Acknowledgements

The author would like to thank Nicola Laciok, Derek Wright, Morten Barth-Nilsen, Alistair Fay, Susan Halfpenny, Lina Kokaly, Stephen Rhodes, Sun Ho Beck, Susan Eltringham and David Kerr for their excellent work on various components of the system. I would also like to thank Karen Renaud for her useful comments on a draft of the paper.

Bibliography


