SWAM — a structured WWW access model

Kam-Fai Wong\textsuperscript{a,\*}, Chun-Hung Cheng\textsuperscript{b}, Jaideep Motwani\textsuperscript{c}

\textsuperscript{a}Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
\textsuperscript{b}Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
\textsuperscript{c}Department of Management, Grand Valley State University, Grand Rapids, MI 49504, USA

Received 16 September 1998; accepted 18 July 1999

Abstract

As the number of World-Wide Web users increase, the information content of the WWW documents gets richer and richer. Today, it is not uncommon to find a Web document involving many levels of subdocuments; each, in turn, consists of a number of hyperlinks; and each hyperlink may reference any part of the web. These intermingled hyperlinks render the conventional browsing and authoring methods ineffective. Traditional browser and authoring tools do not provide an overall view of the structure of the document. As a result, readers may need to go through many hyperlinks to obtain the required information, and individual authors of a group editing project may not conform to the same authoring style. To overcome such problems, a Structured WWW Access Model is proposed. In it, browsing and authoring are performed on two levels. The users first view the structural graph depicting the overall structure of the document, and, by choosing the required nodes on the graph, the users can selectively access the desired document segments. As an example of its use, the FAT-RAT system was developed.

\#2000 Elsevier Science B.V. All rights reserved.

Keywords: Internet; WWW; Authoring tools; Browsing tools; Bilingual documents

1. Introduction

“The world has glimpsed a subset of hypermedia functionality and its potential for structuring and accessing information through the recent surge in World-Wide Web (WWW) activity. Yet, we lack guidelines and tools to design and develop hypermedia applications. This is especially true for commercial scale systems which involve frequently changing information. Without such design guidelines and tools, the ever-growing network of interlinked applications is becoming increasingly spaghetti-like and hard to maintain.” [1]

This statement reveals a practical problem that is affecting the society as a whole. The ever-growing popularity of WWW applications, compounded with the increasing amount of information involved in such applications, makes the situation worse.

A large WWW document commonly consists of hyperlinks (or anchors) referring to different subordinate documents that contain multimedia information. Furthermore, these documents may have hyperlinks to others. This, therefore, requires proper manage-
ment. However, there are still too many problems, including

D1: Readers do not have an overall picture of the document. Under the present browsing convention, the readers have only a local view of a document; they can only look into the hyperlinks associated with the page currently being accessed.

D2: A highly structured document renders the conventional browsing approach awkward. Readers often have to go down many pages (levels) to obtain the required information.

D3: A typically large WWW document has many hyperlinks pointing to different subdocuments. In a large corporation, these subdocuments are written by authors with different backgrounds, knowledge and skill sets; for example, in an advertising firm, there may be one by the graphic design artist and another by the project manager. Lack of coordination in the conventional authoring mechanism can lead to a poorly structured document, with different page formats and fonts, etc.

D4: Authors often find it hard to determine the hyperlinks and the connections between them due to the vast volume of information that has to be included in a large document.

To overcome these problems, Structured WWW Access Model (SWAM) is proposed. With the new model, authoring and browsing of WWW documents can be done in an orderly and repeatable manner. The development and maintenance of WWW documents are no longer the task of expert practitioners. Managers can maintain some degree of control by enforcing this discipline.¹

2. The structured WWW access model

The Structured WWW Access Model (SWAM) is proposed. With the new model, authoring and browsing of WWW documents can be done in an orderly and repeatable manner. The development and maintenance of WWW documents are no longer the task of expert practitioners. Managers can maintain some degree of control by enforcing this discipline.¹

2.1. Data submodel

In SWAM, a Web document has two parts: document structure (EMAP) and text. Document structure

is a graph-based representation of the overall organization of the document text. It shows the connections of different hyperlinks embedded in the document.

2.2. Operation submodel

Operation submodel defines how a SWAM-based document is accessed. There are two modes of access: authoring and browsing. In the process of authoring, the human editor first defines the overall structure of the document using the EMAP graph (thus solving problem D4). The individual nodes of the graph can, then, be assigned to different specialist authors who will produce the corresponding segments (solving problem D3). In the browsing process, the reverse of authoring takes place. The overall structure of the document (i.e., in EMAP form) is read (solving problem D1). The human readers can then selectively browse the desired segments of the document (solving problem D2).

3. The SWAM data model

The two-tier mechanism of SWAM is superior to the traditional approach. At the operation level, the effectiveness of this mechanism is strongly influenced by the data submodel (document structure) selected for describing the overall structure of the underlying document text.

3.1. Related work

Data models for hypermedia (i.e., WWW) documents² fall mainly into three classes:

1. In the concept-based model, the relationships between subdocuments are described. The description is similar to the ER diagram commonly used for database design. Many hypermedia data models have been defined in various research efforts, including the HDM model defined by Gazotto, Paolini and Schwabe [4]; it describes representation schemes, but provides little infor-

²One should not confuse the data models for hypermedia application with the same for hypermedia systems. Dexter [5] and Trellis [12] are examples of data models for hypermedia (more specifically, hypertext) systems.

¹This argument was suggested to us by one of the anonymous reviewers of the paper.
mation on the procedures for using those repre-
sentations in the design process. The RMDM
(Relationship Management Data Model) [7]
provides different schemas to represent relation-
ship between information objects (e.g., subdocu-
ments), the attributes in these objects, as well as
their keys. RMDM is the cornerstone of the RMM
(Relationship Management Methodology) for
structured hypermedia design. Lange [8] (EORM —
Enhanced Object Relationship Model) and
Oriented Hypermedia Design Model) studied
object-oriented approaches to hypermedia design;
their models emphasize both conceptual and
navigational structures and facilitate information
re-use.

2. In the spatial model [9], different colors are used
to identify different classes of information object
(i.e., provide visual cues) and relative distances
are used to represent the semantic closeness
between them (i.e., provide spatial proximity).
For example, an object placed on top of another
object implies that the former is the super class of
the latter.

3. In the browser-based model [2,6], a 2D graphical
network is used to represent the structure of a
hypermedia application. In this model, informa-
tion objects (e.g., subdocuments) are represented
in nodes, which are interconnected via arcs.

The conceptual model is certainly a good one for
WWW document authoring. Especially in an intranet-
based corporate environment where multiple authors
with different skill sets (e.g., an artist versus a man-
ger) may contribute to different parts of the overall
document. However, it is not directly applicable to
Web page readers, who may be naive users and have
little knowledge of the conceptual models. Similarly,
the spatial model has its origin in the browser-based
model. Spatial proximity and visual cues enable a
reader to browse easily through a document. In prac-
tice, for large Web documents with many hyperlinks,
the resulting spatial design diagram can be quite
messy. The browser-based model, on the other hand,
explicitly reflects the structure of a hypermedia docu-
ment. This provides a clear picture of the overall
organization of the document, making browsing more
effective. However, it cannot represent the semantic

relationship. This is a clear deficiency in most hyper-
media application designs. Nevertheless, in the
authoring of Web documents where structural rela-
tionship is more important than semantic relation-
ships, the browser-based model is adequate.

From the operation point of view, existing WWW
systems are ineffective. For example, Microsoft Front-
Page adopts a similar structured access approach, but
it is a closed system and can handle only its own
WWW documents. The KMAP system [3] also sup-
ports structured information access, but it is machine
dependent — it runs only on MacIntosh computers.
Furthermore, neither FrontPage nor KMAP provide
multilingual support.3

3.2. Definition of EMAP

The browser-based model (EMAP) was chosen as
the basis of the SWAM data model. An EMAP docu-
ment structure consists of one or more node(s) inter-
connected via link(s). A node can be either a
composite or simple node. The latter simply represents
a segment of the WWW document (i.e., the content),
which can be any type of multimedia data, that is, text,
image, graph, audio or video. Each simple node is
associated with a constructor and a destructor: func-
tions invoked upon activation and de-activation of the
node. A composite node is one or more simple node(s)
embedded with hyperlink(s). A hyperlink is repre-
sented by an anchor and link pair. The former defines
the position in the document (e.g., a keyword in a
text or a co-ordinate in an image) from which a
hyperlink emerges, and the latter is the reference
to the destination document (i.e., the URL). Option-
ally, an alarm can be associated to a link. This is a
function that is invoked whenever its node is modified
(Fig. 1).

3.3. EMAP — extensions to KMAP

EMAP is practically an extension to KMAP [3]. At
the modeling level, the extensions are as follows:

- A new link type has been introduced. An alarm
link implies that the parent node is dependent on

3 In particular, the lack of a Chinese interface hinders the
widespread acceptance of these systems in Asia.
the child, and whenever the content of the node is changed, the alarm is initiated.

- A node can be tagged to require that it must not be null valued. This is useful in an Intranet-based corporate environment where the editor-in-chief knows which pages can be left blank. This is achieved through the use of the destructor function.

- Node types are introduced. A node can be text, image, graph, audio or video. Also, a composite type is supported; for example, a text document that includes an image. With this feature, the editor-in-chief can define the overall structure of the document.

4. Examples: application of EMAP

4.1. Example 1: representing an HTML document in EMAP

Consider the WWW document shown in Fig. 2; its corresponding HTML source and EMAP are shown in Fig. 3. In the SWAM operation submodel, browsing is performed in a structured manner. One views the EMAP structure first, and, by pointing at the different nodes, can selectively access different document segments. Reading a WWW document in this fashion is more effective for the reader, because it gives a global overall picture of the document prior to drilling into the details. Furthermore, this one can selectively read certain parts of its segments.

4.2. Example 2: compiling a corporate-wide HTML document

Consider a large corporation preparing an annual report for its public shareholders. The report might cover three main issues: (1) financial status, (2) personnel information, and (3) customer statistics. The publication department must compile the report and mount it on the Internet.

The chief editor of publications decides that the report should have three sections. The content of each
would be provided by the department concerned. However, rather than leaving the structure of each section completely to the individual department, the chief editor first defines the format of each section:

**Annual report**

<table>
<thead>
<tr>
<th>Subsection 0</th>
<th>Introduction of the company (to be written by the publication dept.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsection 1</td>
<td>Financial status (to be written by the publication dept.)</td>
</tr>
<tr>
<td>1.1 Sales figures</td>
<td></td>
</tr>
<tr>
<td>1.2 Graph for sales</td>
<td></td>
</tr>
<tr>
<td>Subsection 2</td>
<td>Personnel information (to be written by the personnel dept.)</td>
</tr>
<tr>
<td>2.1 Turnover rate per month</td>
<td></td>
</tr>
<tr>
<td>2.2 Salary statistics</td>
<td></td>
</tr>
<tr>
<td>Subsection 3</td>
<td>Customer statistics (to be written by the public relations dept.)</td>
</tr>
<tr>
<td>3.1 Distribution of type of customers</td>
<td></td>
</tr>
</tbody>
</table>

Later, the editor changes his/her mind slightly and decides that the departments could have a small subsection to describe their work. This, however, is optional and limited in size (e.g., a maximum number of words). The structure was modified to:

**Annual report**

<table>
<thead>
<tr>
<th>Subsection 0</th>
<th>Introduction of the company (to be written by the publication dept.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsection 1</td>
<td>Financial status (to be written by the publication dept.)</td>
</tr>
<tr>
<td>1.0 Department description (optional, maximum 1000 words)</td>
<td></td>
</tr>
<tr>
<td>1.1 Sales figures</td>
<td></td>
</tr>
<tr>
<td>1.2 Graph for sales</td>
<td></td>
</tr>
<tr>
<td>Subsection 2</td>
<td>Personnel information (to be written by the personnel dept.)</td>
</tr>
<tr>
<td>2.0 Department description (optional, maximum 1000 words)</td>
<td></td>
</tr>
<tr>
<td>2.1 Turnover per month</td>
<td></td>
</tr>
<tr>
<td>2.2 Salary statistics</td>
<td></td>
</tr>
<tr>
<td>Subsection 3</td>
<td>Customer statistics (to be written by the public relations dept.)</td>
</tr>
<tr>
<td>3.0 Department description (optional, maximum 1000 words)</td>
<td></td>
</tr>
<tr>
<td>3.1 Distribution of type of customers</td>
<td></td>
</tr>
</tbody>
</table>

In defining the EMAP graph for the above example, the chief editor would tag the subsections 0, 1, 1.1, etc. as compulsory, and 1.0, 2.0 and 3.0 as optional. To
achieve this, a destructor function can be added to each of the compulsory nodes to check for a 'null document'. Similarly, a destructor function could be added to each of the optional nodes to ensure that the text is limited to 1000 words. The resulting EMAP structure is shown in Fig. 4.

5. The FAT-RAT system

Based on SWAM, a prototypical information system was designed and developed. This system consists of two parts: Forward Authoring Tool (FAT) and the Backward Authoring Tool (RAT). FAT is for creating and editing SWAM-based documents, and RAT is the tool for browsing. Using the structured design methodology, proposed by Yourdon [17], the top-level diagrams of the FAT-RAT system — the context diagram and the level 0 diagram — are shown in Fig. 5.

5.1. FAT — the authoring process

The behavioral model of the FAT subsystem is shown in Fig. 6. The user may make a request to edit a
SWAM-based document consisting of an EMAP structural graph and the associated HTML document. Editing can begin from either the EMAP or the HTML side (Processes 2.1 and 2.2). In the former, the user issues different editing commands (i.e., Request.EditEMAP) to the EditEMAP process. Editing may require access to existing documents and require the document to be written back to the store on completion. While an EMAP structure is on display, the user can select a node and edit its content or add/delete a constructor and/or a destructor. Also, when the node with an associated constructor function is accessed, the constructor function will be executed.

On the other hand, if the user begins to edit an HTML document, he/she may send editing commands to FAT, such as find, replace, delete word, etc. Once the HTML document is completed, one, both, or none of the following two situations may occur before the document is stored:

Fig. 5. The top-level design diagrams of FAT-RAT: (a) the system boundary; and (b) the division of labor; together with the statement of purpose and event list.
1. The current HTML document is not associated with an EMAP graph. In this case, a Translate command is sent to the TranslateToEMAP process (Process 2.3). This will produce the corresponding EMAP.

2. The EMAP node that corresponds to the HTML document is associated with a destructor. In this case, the destructor function is invoked before the document is stored (Process 2.5 on ExecuteFunction).

5.2. RAT — the browsing process

The behavioral model of the RAT subsystem is shown in Fig. 7. When a retrieval request (Request.Retrieval) is received from the user, RAT will first send a WebRequest to the external WWW server, which may be local or remote. The returned WebDocument is SWAM-based. Its EMAP and HTML parts are extracted, and the former is displayed graphically (Process 1.2, DisplayEMAP) to the user (i.e., VisualEMAP). The user can select any node or link on the EMAP. This will cause different operations (i.e., Request.EMAPOperation). If the operation is DisplayDocumentSegment, the relevant HTML document segment will be displayed through the DisplayWWWDocument process (Process 1.3). If the operation is a NodeOperation (hide or show a node), the selected node(s) on the EMAP will be manipulated in Process 1.2. This will lead to a change in the visual EMAP representation.

5.3. Implementation issues

5.3.1. Development environment

The development of the FAT-RAT system made heavy use of open system and graphics programming facilities. It was implemented on top of X-windows, making it machine-independent. Effectively, it can run on any UNIX platform that supports X-windows. On top of it, Tcl7.4/Tk4.0 [14] were used for the development of the graphical user interface and the

---

4 Microsoft FrontPage cannot handle pure HTML documents. This limits the acceptance of the tool.
EMAP editor. Furthermore, the connection to an external WWW server was achieved via sockets. For this purpose, the extended Tcl command set, namely, TclX [13], was adopted.

5.3.2. Chinese support

Conventional Tcl/Tk does not support double-byte languages, common in Asia. For this reason, a Japanese localization toolkit that supports both Japanese and Korean encoding (JIS and SJIS) was developed by Yoshiyuki Nishinaka [10]. There was, however, no such toolkit for Chinese. In view of this, a Hanzi Tcl/Tk toolkit was developed in a project\(^5\). It was, in fact, an extension of its Japanese counterpart. The extensions are the supports of BIG-5 encoding,\(^6\) the Cangjie and the Pinyin input methods.

5.3.3. HTML-to-EMAP translation

There is no document type restriction at the starting point of the authoring process. One can start from an EMAP structure or from an HTML document. The former supports the original idea of structural document design and is expected to be the normal one. One can associate HTML texts to the nodes by using the conventional point-and-click mechanism. In some cases (in both browsing and authoring), the input WWW document may not be based on the SWAM model. When this happens, FAT-RAT will first convert the downloaded HTML source into its corresponding EMAP graph, displaying it to the document segments of the selected nodes. An HTML-to-EMAP translator was developed for this purpose.

5.3.4. Alarms

An alarm can be assigned to a link to detect changes to the destination node. At the implementation level, this link, together with its source and destination nodes, is written to a system file. When the document segment associated to a destination node is updated, FAT-RAT will update the appropriate link entry in the

\(^{5}\)The Hanzi Tcl/Tk is a by-product of the FAT-RAT project, and is now available for public use at http://www.se.cuhk.edu.hk/.

\(^{6}\)BIG-5 is the de facto encoding standard for traditional Chinese character set commonly used in Taiwan and Hong Kong.
system file to indicate that this node has been modified. Concurrently, running in the background, is an alarm daemon that checks this system file periodically. As soon as a link is found to have been changed, the associating source node will be notified: (a) the node itself is tagged and will flash when displayed, and (b) if the email address of the author of the source node is registered, a warning message will be automatically generated and sent.

5.3.5. Constructor/destructor

Constructors and destructors are based on ideas borrowed from object-oriented programming. A constructor and a destructor are optional entries specified during the creation of a node.

5.3.6. Document store

The SWAM document produced using FAT-RAT can be stored in the local WWW document database. The IPOC information retrieval system [15] is used to provide this storage facility. IPOC supports Chinese text, and is word-, rather than character-, based. This ensures high effectiveness, as the basic semantic unit of Chinese text is a word, \(^7\) for example, *da men* (main gate) is composed of the characters *da* and *men*, which mean *big* and *door*, respectively, by themselves individually.\(^8\)

---

\(^7\) The average word length in Chinese is 2–3 characters.

\(^8\) The process of grouping characters in Chinese is termed ‘word segmentation’ [16].
6. An illustrative example

Fig. 8 shows a series of snap shots exemplifying the use of the FAT-RAT prototype for accessing a typical SWAM-based document: the home page of the Systems Engineering Department of the Chinese University of Hong Kong. At the start of the authoring process, the editor downloads the document by specifying the URL in the File menu. The top-level EMAP structure is displayed. The FAT-RAT user interface consists of two main panels: the graph (on the left), where the EMAP is displayed, and the control (on the right), which provides functional buttons for editing the EMAP currently on display. There are 10 editing functions (see Appendix A). Each is associated with a button on the control panel. By selecting the root node and clicking the Present button in the control panel, the home page of the department is displayed. Also, the related HTML source can be displayed by toggling the Text option in the menu bar.

The users can add new nodes to the EMAP at any time. The Add Single Node function is invoked for adding a new node, and then its properties can be introduced by invoking the Modify function. The users can modify the properties of a link; e.g., by adding an alarm. The alarm specification consists of two parts: the source (left) and the destination nodes (right). When the form is displayed, all the source
node(s) connected to the highlighted link are displayed on the left-hand side of the form and all the destination node(s) are displayed on the right. The users can select the nodes by using the selection option shown in the middle of the form.

**7. Conclusion**

The association of alarms to hyperlinks and constructors/destructors to nodes make the underlying SWAM model a powerful tool. Designers can use it to provide triggers or add active rules and make them reactive to events or conditions. These functions can enforce security, facilitate billing, etc. It combines both browsing and authoring in an integrated environment. The built-in HTML-to-EMAP translator enables structural browsing, even for non-SWAM-based documents. It supports the Chinese and Japanese languages; this makes it suitable for Asian applications.

**Acknowledgements**

We would like to thank the two anonymous referees and the editor, Professor Edgar H. Sibley for numerous insightful comments, which produced important improvements in the papers. This work is partially supported by the 1994 earmarked grant from the Hong Kong Government Research Grant Committee (No. CUHK256/94E) and the Strategic Grant Program of the Chinese University of Hong Kong.
Appendix A. EMAP editing functions supported by FAT-RAT

From top to bottom of the control panel (see Fig. 8a), the EMAP editing functions are as follows:

Add single node
This function enables the user to add a new node to the EMAP. The new node is initially dangling, that is, without a link. The name of the node is the ID number assigned by the system, but it can be changed by the user.

Add node
This function first creates a single node. Subsequently, it scans through the entire EMAP structure, identifies the nodes previously highlighted by the user, and adds links between these nodes and the newly created one.

Del node
This function deletes all the highlighted nodes in the EMAP and their associated links, except the root node.

Add link
This function adds a link between two highlighted nodes.

Modify
This function enables the users to modify the properties (e.g., the destructor) of a highlighted node.

Present
This function starts up an editor for the document segment associated to a highlighted node.

Clear composite
This function deletes all nodes...
associated to a composite node, except the root node.

Connect to server This function pings the specified server and checks whether it supports SWAM documents.

Send to server This function uploads a SWAM document to the remote server.

Open object This function starts up a new editor and presents the specified object in it.

References


Fig. 8 (Continued)
Kam-Fai Wong obtained his Ph.D. from Edinburgh University, Scotland, in 1987. After his Ph.D., he has performed research in Heriot-Watt University (Scotland), UniSys (Scotland), and ECRC (Germany). At present, he is an associate professor in Department of Systems Engineering and Engineering Management, and an associate director of the Center of Innovation and Technology (CINTEC) at the Chinese University of Hong Kong. His research interest centers on Chinese computing, parallel database, and information retrieval. He has published over 80 technical papers in these areas in various international journals, conferences, and books. He is a member of the ACM, CLCS, IEEE-CS, and IEE (UK) and a member of the editorial board of *Journal on Distributed and Parallel Databases*. During 1994–1996, he was the chairman of the ACM Hong Kong Chapter. Currently, he serves as the Asia Pacific Representative of the ACM International Membership Activities Committee and a distinguished lecturer of the ACM Lectureship series.

Chun-Hung Cheng received his Ph.D. in Management Information Systems from the University of Iowa in 1990. He started his teaching career at Kentucky State University. In 1994, he returned to Hong Kong, and is now an associate professor in Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong. His research interests are Information Systems and Operations Management. His research articles appeared in *Annals of Operations Research; Computers & Operations Research; Expert Systems, Information and Management; IIE Transactions; IEEE Transactions on Engineering Management; IEEE Transactions on Systems, Man, and Cybernetics; and Operations Research*, among others.

Jaideep Motwani received his Ph.D. in Operations Management at the University of North Texas. His research interests include operational strategy and performance, global strategies, application of emerging technology in operations and quality management, and the business value of IT. His work has appeared in various journals, including *Operations Research; IEEE Transactions on Engineering Management; European Journal of Operational Research; Expert Systems with Application; Omega; Journal of Operational Research Society; International Journal of Production Research; and Journal of Computer Information Systems.*