DOCUMENT RESUME

ED 427 719 IR 019 270

AUTHOR Maurer, Hermann; Scherbakov, Nick
TITLE Course Development Environment for Hyperwave.
PUB DATE 1998-11-00
NOTE 7p.; In: WebNet 98 World Conference of the WWW, Internet and Intranet Proceedings (3rd, Orlando, FL, November 7-12, 1998); see IR 019 231. Figures may not reproduce clearly.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programming); *Computer Software Development; Computer Uses in Education; *Courseware; Educational Technology; *Instructional Design; Interaction; Learning Modules; *Multimedia Materials; Navigation (Information Systems); Training; *World Wide Web
IDENTIFIERS Client Server Computing Systems; *Course Development

ABSTRACT

A new Courseware Development Environment (CoDE) is currently being developed for Hyperwave World Wide Web servers. CoDE provides instructional designers with an easy-to-use yet powerful environment to develop online training; students have the ability to access this training using a standard Web browser. Functionality of the Hyperwave server is extended by: (1) a Multimedia Editor, providing the capability for creating highly interactive learning elements with functionality that cannot be achieved in plain HTML format; (2) a Page Editor, providing a convenient way for authoring pages by means of reusing existing elements; and (2) a Structure Editor, providing a way to impose a navigable structure on top of existing collections of pages, thus creating a learning unit. Each of these components, described in detail in this paper, are based on a common tool, called "Tree View," that provides a unified access to a local drive or a Hyperwave server. Figures present diagrams of page template components, saving a learning unit to a Hyperwave server, and accessing a learning unit, as well as screens of activating a macro cell and editing a learning unit. (AEF)

Course Development Environment for Hyperwave

Hermann Maurer
Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology
Schießstattgasse 4a, A-8010 Graz, Austria
Email: hmaurer@iicm.tu-graz.ac.at

Nick Scherbakov
Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology
Schießstattgasse 4a, A-8010 Graz, Austria
Email: nsherbak@iicm.tu-graz.ac.at

Abstract: The paper presents a new Courseware Development Environment (CoDE) which is currently being developed for Hyperwave WWW servers.

1. Introduction

The paper presents a new Courseware Development Environment (CoDE). CoDE provides instructional designers with an easy to use yet powerful environment to develop online training. Students have the ability to access this training using a standard web browser. The course development environment is based on the Hyperwave [Maurer 1997] information server. Functionality of the server is considerably extended by the following components:

1. *Multimedia Editor* provides a possibility to create highly interactive learning elements with functionality which cannot be achieved in plain HTML format (like simulations, animations, answer-judging, vector graphic, dynamic screen transitions synchronized with sound clips, etc.). Such elements are called MM applets in the rest of this paper.
2. *Page Editor* provides a convenient way for authoring pages (i.e. basic elements visualized on student screen as indivisible chunk of learning information) by means of reusing existing elements. It should be especially noted that the Page Editor is not designed as an alternative to any of existing HTML editors. It uses a "template authoring paradigm" where existing HTML Editors can be seen as optional add-ons to provide basic elements of which a final product is assembled via an advanced GUI.
3. *Structure Editor* produces learning units by means of imposing a navigable structure on a top of existing collections of pages. It is based on the main concept of typed composites. Thus, there is a number of predefined templates (composite types) where users can simply insert existing pages or other composites to define sophisticated navigable structures.

This paper contains requirements and definitions for these three tools.

2. Data Elements of Online Training

Learning material won't be edited by instructional designers in the form of HTML pages. Instead, instructional designers edit the structure of learning units (learning unit -> page -> MM Element). It should be especially noted that all the three levels are optional and may be omitted if an author has a preferable alternative to implement the same task.

Thus, for example, authors may use MM Editor to create educational animations, but they are allowed to use any other tool like e.g. Macromedia Flash to perform a similar task. The resultant file, whether it is internal MM Editor format or another one, is available for reuse on the next level - Page Editor.

Similarly, authors may use Page Editor to assembly a page out of existing elements (HTML fragments, texts, images, movies, animations, etc.), but they are also allowed to use their favorite HTML Editors to create a complete HTML page.

Finally, authors are provided with an advanced structuring tool for automatic generation of a navigational structure by means of inserting/removing elements into/from learning units. Nevertheless, the authors are free to embed navigational elements (buttons, text anchors, etc.) into original HTML pages if they prefer so.

An addressable element is an object that can be requested by a student's browser with an HTTP request and is therefore "addressable" using a URL.

There exist two types of Addressable Elements: Learning Unit and Page.

- A Learning Unit is a collection-like unit that contains other Learning Units and Pages. Learning Units and Pages can occur in different Learning Units at different positions.
- A Page, the "atomic" element of learning units, is in fact not atomic because it can contain various media units and MM applets as it's paragraphs.

3. Authoring/Editing the Data Elements

3.1. MM Editor

The Editor provides a possibility to create highly interactive learning elements with functionality that cannot be achieved in plain HTML format. Functionality: the MM Editor works with so-called media objects. The following list of Media Objects is currently considered for implementation:

- Vector graphic (WMF and HMW formats);
- Still images (GIF and JPG formats)
- Movies (MPEG)
- Texts (plain ASCII)
- Sound (AU format)

The MM Editor allows to combine the MM objects into an executable form by means of the following operations:

- drag and drop objects into an editing area,
- resize a selected object,
- alter colors,
- edit content (texts and HMW format).

The MM Editor essentially adds a value to such imported objects by means of:

- Animating them on the screen;
- Inserting interactive elements like pauses or the possibilities for user's input;
- Allowing branches and loops;
- Dramatic transitions of the whole screen or its parts;
- Inserting conditions Analyzing User's input (Question/Answer Dialogue);
- Synchronizing active elements on the screen.

Technically the MM Editor is available in two forms - as Java Applet and Java Application. As a Java Application the Editor allows to select MM Objects or completed MM applets from a local drive or from a HyperWave server using unified "treeview" browser. The application will allow to save the resultant applets on a local drive or upload it to a Hyperwave Server.

As a Java Applet the Editor has considerable limitations on using local drive. It allows to select MM Objects or completed MM applets from HyperWave server only. The applet allows to upload resultant file into a Hyperwave Server. Additionally, the applet can be initiated in a special "edit only" mode which allows to edit existing MM applets residing on hyperwave server.

3.2. Page Editor

The Editor provides a convenient way for authoring pages (i.e. basic elements visualized on student screens as indivisible chunk of learning information) by means of reusing existing elements.
The Page Editor works with o-called page templates and page elements. A page template is a collection of cells. Each template's cell belongs to a particular type and defines:

- the operations which an author can apply to the cell;
- the position of a page element (or elements) on the user's screen;
- the appearance of the page elements on the user's screen.

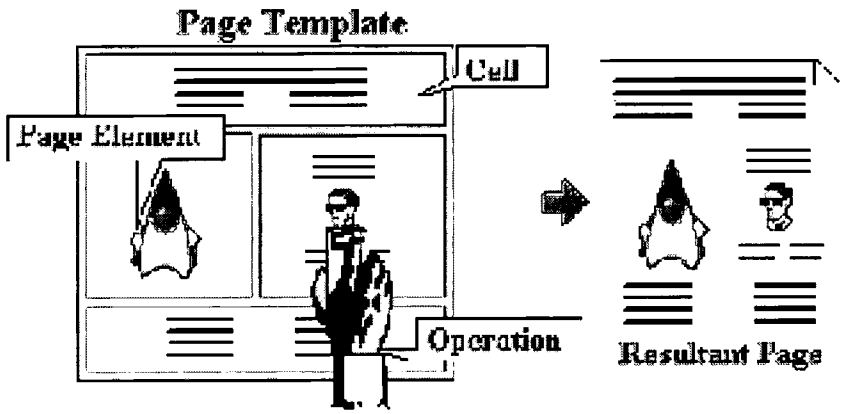

Figure 2: Page Template Components

The Page Editor supports the combination of page elements into a valid HTML document by means of operations which are applied to the template's cells. Thus, the main way of authoring is filling a page template with existing page elements.

There are the following cell types:

- Singular cell;
- Multiple cell;
- Macro cell.

The *singular cell* support the following authoring operations:

1. Drag and drop a page element from a local drive into the cell. Putting a new element into a particular singular cell replaces a previously defined element with the new one.
2. Double click the cell to define a new HTML or textual fragment.
3. Delete an element from the cell.

The *multiple cell* support the following authoring operations:

1. Drag and drop a page element from a local drive into the cell. Putting a new element into a particular multiple cell creates a sequence of elements.
2. Double click the cell to define a new HTML or textual fragment.
3. Move an element within the multiple cell onto a new position (change an order of elements);
4. Delete an element from the cell.

The *macro cell* contains a special macro which is activated when the cell is double clicked.

Normally, the macro requests a number of parameters from the author and generates a valid HTML fragment using the current values.

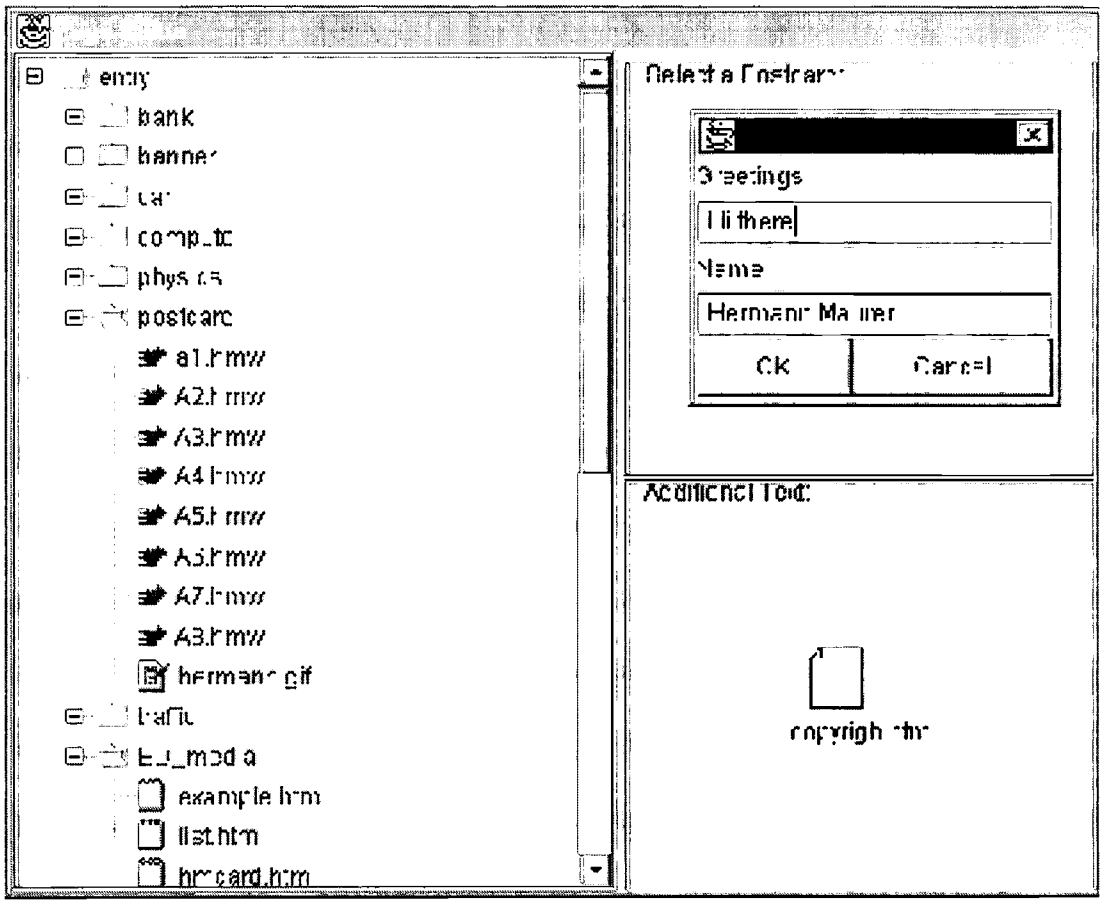

Figure 2: Activating a macro cell

Page elements are created with other editing systems. Generally, any MM format can be incorporated into a page created by means of the Page Editor. There are so-called *native* Page Editor formats and *author defined* format. The native MM formats do not require any additional definition and can be previewed by the page editor.

Author defined formats should be additionally described using a special notation. The following list of native formats is currently considered for implementation:

- HTML Fragment;
- Still images (GIF and JPG formats)
- Movies (Mpeg and Vivo formats)
- Sound (AU, WAV and Vivo formats)
- Plain Text fragment (plain ASCII)
- MM Applet in native MM Editor (see Section 3.1) HMW format.

The Page Editor adds a value to textual page elements by means of:

- Defining relative positions of page elements on the screen;
- Assigning a type to template cells (ordered list, title, footnote, etc.);
- Assigning a desired color and font attributes to text fragments.

Technical Details: the Page Editor is available in two forms - as Java Applet and Java Application. As a Java Application the Editor allows to select Page elements from a local drive or from a HyperWave server using unified "treeview" browser. The application allows to save the resultant page on a local drive or upload it to a Hyperwave Server. The Page Editor also allows to edit existing pages residing on a hyperwave server.

As a Java Applet the Editor has considerable limitations on using local drive. It allows to select Page elements from HyperWave server only. The applet allows to upload resultant file into a Hyperwave Server. Additionally, the applet can be initiated in a special "edit only" mode which allows to edit an existing page residing on a hyperwave server.

3.3. Structure Editor

The Structure Editor provides a convenient way to impose a navigable structure on a top of existing collections of pages, thus creating a learning unit.

Functionality: The Structure Editor works with a so-called Learning Unit templates, pages and existing Learning Units.

Each Learning Unit is a Hyperwave collection having a number of additional attributes which are automatically assigned by the Structure Editor. Elements of a learning unit are members of the corresponding collection which are also automatically provided with a special set of attributes.

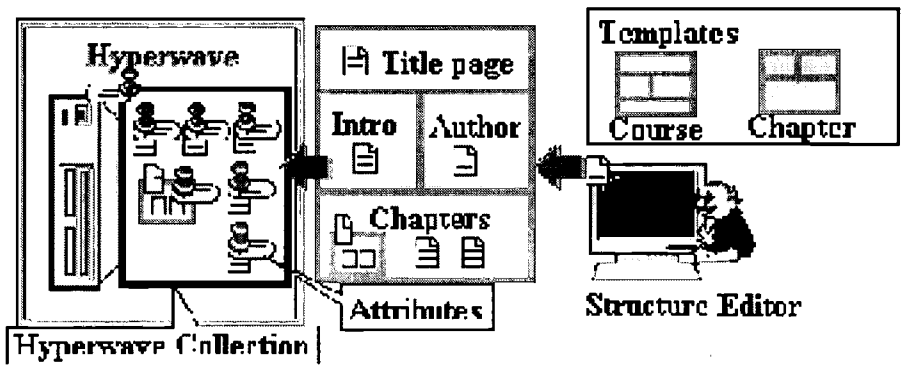

Figure 3: Saving a Learning Unit into a Hyperwave Server

Whenever a user accesses such learning unit with an ordinary WEB browser a special server-site Java script is run to visualize the unit in a form of interrelated HTML pages. The attributes attached to the collection and member documents are essentially used to control such vusualization.

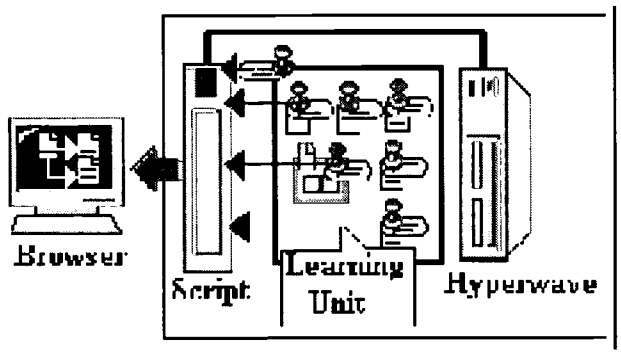

Figure 4: Accessing a Learning Unit

The main way of authoring is dragging existing pages and/or learning units onto a selected template. Pages are created with arbitrary other editing systems, i.e. they can, but need not , have been prepared using the Page Editor described in 3.2. All Pages must be in valid HTML format.

The Structure Editor allows to combine pages and existing course units into a new learning unit by means of the following operations:

- drag and drop objects into a template cell,
- answering questions generated by the template (e.g., would you like to create a reference to this element from the table of contents).

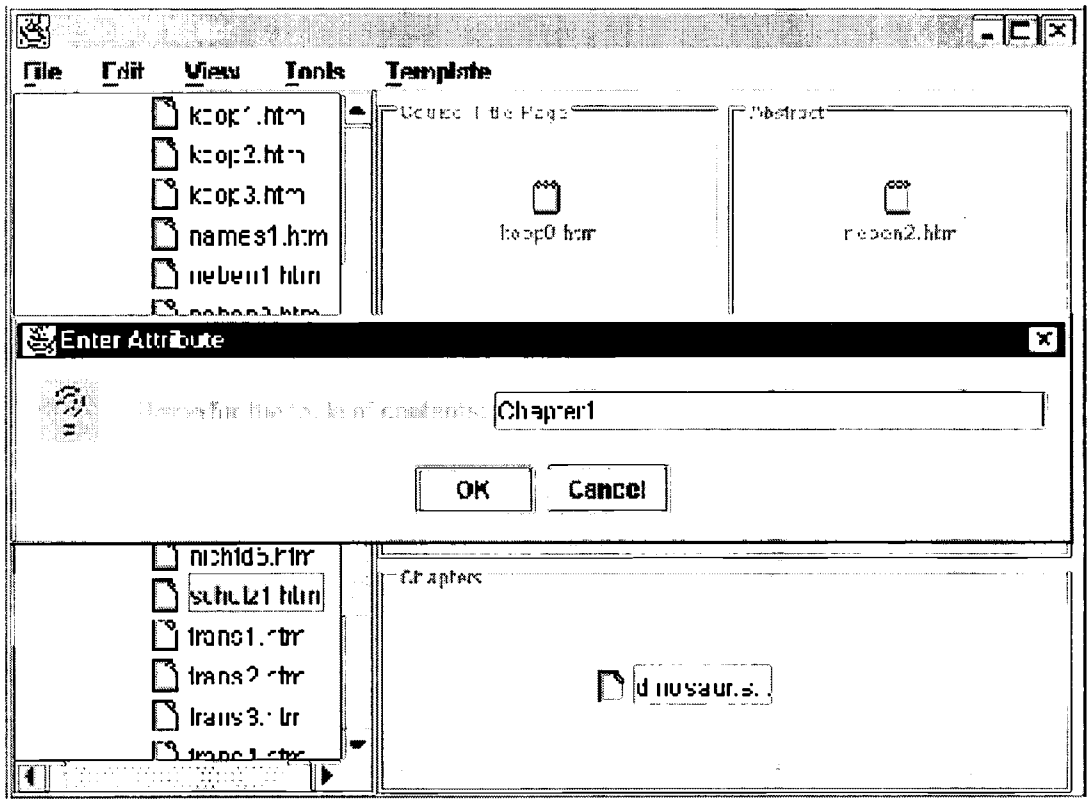

Figure 6: Editing a Learning Unit

Technical Details: The Structure Editor is available in two forms - as Java Applet and Java Application. As a Java Application the Editor allows to select elements from a local drive or from a HyperWave server using unified "treeview" browser. The application allows to save the resultant page on a local drive or upload it to a Hyperwave Server. The structure Editor can be also used to modify the structure of learning units that reside on a Hyperwave server. As a Java Applet the Editor has considerable limitations on using local drive. It allows to select Pages and/or existing Learning Units from HyperWave server only. The applet allows to upload resultant Learning Unit into a Hyperwave Server.

All the three editing components described above are based on a common tool that provides a unified access to a local drive or a Hyperwave server. This common tool, which is called "Tree View" in this paper, is used as an independent window for browsing existing materials by all three editors, it is also used for physical copying resultant files onto a local drive or uploading them into a hyperwave server.

4. References

[Maurer 1996] Maurer, H. (Ed.) *Hyperwave: The Next Generation Web Solution,* Addison Wesley (1996).

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").