An Adaptive Presentation Model for Educational Hypermedia Systems

Amir Hekmatpour
IBM Manufacturing Technology Center
11400 Burnet Rd. MD 3940, Austin TX 78758, USA
rima@vnet.ibm.com

Authoring tools provide utilities for creating multimedia and hypermedia applications but they neither provide nor enforce efficient presentation models or methodologies. The basic design issues that make the quality and usability of information possible are left up to the developer. We present a three facet adaptive model for development and management of educational hypermedia systems. The model provides: spatial reconfiguration of the hypermedia objects based on activation frequency, information content customization based on author/teacher specified criteria, and hypergraphic network optimization based on usage. This paper presents the theory underlying the model, its architecture and hypermedia structure, its object–oriented content management, and introduces a set of quantitative measures for evaluation of the impact on development effort, as well as study time (the time it takes a student to study the courseware). Our analysis shows improved study time, reduced content skimming, fewer unnecessary hyperspace jumps, and reduction in presentation of irrelevant hypermedia information objects. The above gains in information delivery and content access are traded for increased development effort.

INTRODUCTION

Hypermedia systems offer great promise for capturing expertise and subsequently providing multifaceted access in support of a user engaged in a complex task. A primary issue in building such systems is how to structure the knowledge contained in them such that a user with a problem can find the most appropriate knowledge easily and naturally [Bareiss93, Okamoto94]. In addition, such a system could/should learn from and adapt to its environment and improve its interaction with the user [Thies94, Wittburg94]. Coordinating text, graphics, and images in a hypermedia environment is very important in effective presentation and management of the subject matter [Feiner90, Augenstein93, Hekmat94]. Some researchers have investigated the automatic generation of coordinated multimedia [Feiner91, Maybury91]. Others have studied controlling and managing the different elements composing a multimedia document [Merlet93, Hekmat94]. And still others have proposed architectures for adaptive online systems [Kaplan90, Peralta92, Boyle93]. Such studies have proved necessary due to inefficiency of the traditional book model online systems [Boyle93] and the ever increasing functionality of educational multimedia/hypermedia systems [Kong94].

Computer aided instruction (CAI) and computer based training (CBT) developers need tools and methodologies which incorporate the knowledge of experienced courseware designers, cognitive scientists, and even computer scientists to generate high quality courseware. Three important elements of designing online information are outlined by Donald Norman as: visibility, appropriate clues, and feedback [Norman89]. Basic elements of information development – such as style and grammar, graphic and layout design, have to be enhanced and extended for online hypermedia medium. Majority of reported CAI and CBT are text–based systems delivering a presequenced (linear) courseware. In most cases hypermedia information will be non–linear; which would require different courseware delivery and management [Hamond89, Rojas91]. Hypermedia information need not be textual; it may be presented using a myriad of multimedia tools – such as graphics, animation, sound, and video. As a result, cognitive and usability issues become paramount to the effective delivery of information [Okamato94]. The above issues and concerns are important in the design of educational systems, especially hypermedia systems. The model described in this paper addresses some of the issues discussed above and provides a methodology and formalism for hypermedia courseware development. A prototype of the proposed model is implemented in the current version of ExperMedia/2; an OS/2 multimedia expert system shell for computer–aided training and intelligent computer–aided learning application development [Hekmat93, Hekmat93.1].

This paper is organized into three major sections. First, we present the theory, the underlying architecture, and the object–oriented hypermedia structure which comprise the proposed presentation model. Next, the three adaptive features of the model are described. Finally, a set of quantitative measures are introduced and the impact of the model and its adaptive features on the authoring effort and the courseware study time are analyzed.

ARCHITECTURE AND ELEMENTS OF THE MODEL

The organization of online documentation depends largely on the kind of information being delivered. The main goal in deciding on a structure is to make it as simple and as useful as possible (Rojas910). Hypermedia is flexible and highly interactive. To take advantage of these qualities, we need to provide the user with an effective model of the structure, such as a visual representation. The unstructured topology of the information space inherent to hypermedia has been compared to the problems related to the GoTo statements in computer programming [Tomek93]. This problem of uncontrolled linking has been addressed by many researchers by introducing higher level aggregates that group related nodes and links. While formation of hierarchical structures makes information stored in large hypermedia systems easier to understand and use, the number of links emanating from individual nodes may still be overwhelming. The question then is how to present these links to the user so that he/she can make good use of them [Tomek93]. Graph–theoretical methods proposed by Botafogo could simplify the task of creating such hierarchies by automatically identifying node–link clusters [Botafogo92]. Such a model would present the current and relevant information in a consistent and effective manner, as well as provide clues to whereabouts of the user in the hyperspace. Some researchers have suggested configuring several parameters of the information display: rate of flow, language style, number and types of control buttons, degree of details, and others based on general user profile (e.g., interest, common questions, sociological–educational, background) [Peralta92]. While the above research provides valuable insight into the structural and organizational requirements of effective online systems, these do not provide specific educational hypermedia presentation models and development methodologies. A presentation model, a development methodology, and an online system architecture suitable for academic and industrial education, as well as public edutainment, is therefore required. We believe the proposed model addresses some of these requirements.

Partitioning of Courseware into Logical Pages

Identifying nodes of information that are highly related has many applications in information systems, and in particular in hypertext systems. Botafogo describes a technique for identifying ‘natural’ clusters in a hypertext system [Botafogo93]. In the model described in this paper, hypermedia information are partitioned into clusters of ‘study units’ presented as *Logical Pages* (*LP*). A *LP* is a structure for presenting all of the relevant and related information objects (text, graphics, images, audio, animation, video clips, ...) which the user should/could review when studying the subject matter covered in that page (Figure 1). These information objects have a physical layout as well as a logical layout. The physical layout is comprised of the parent window, x–y coordinates within the parent window, background color, font, scrolls, and system icons. The logical layout is comprised of hypertext and hypergraphic links to other objects in the *LP* as well as to other *LP*s, external applications, databases, and services.

Our *logical page* has some similarity with multimedia presentation structure described by Hardman [Hardman93], in that both allow the author to define (select) the structure of the presentation explicitly. Hardman’s structure is in fact a hierarchical view of the multimedia presentation and does not correspond to the final physical representation of the document (except at the lowest level of hierarchy). Even at the lowest level of hierarchy, Hardman's structural description does not clearly and simply correspond to the physical representation of the media, but rather the logical coordination of the media. Our *LP* methodology on the other hand, provides physical as well as logical correlation of the hypermedia information objects. Physical correlation is provided by proximate placement and presentation of information objects in a *LP,* as well as among related *LP*s. Logical correlation is maintained via hypertext and hypergraphic links in a *LP* as well among relevant *LP*s.

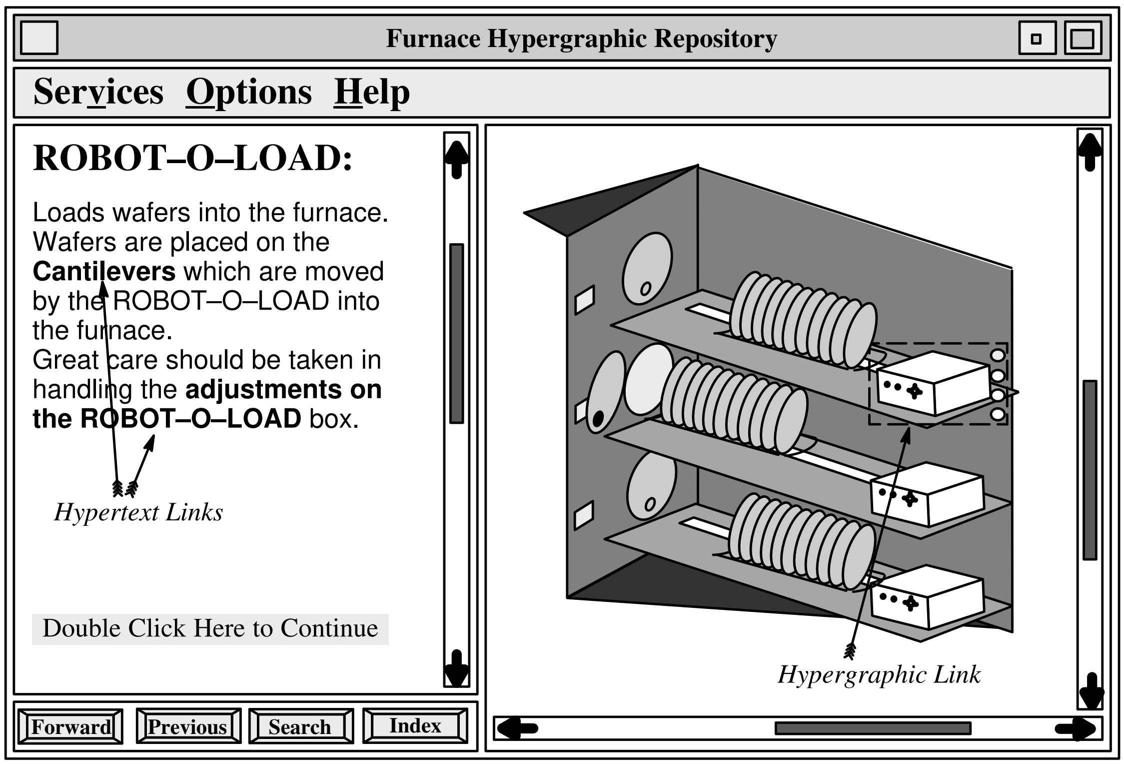

Figure 1. A sample Logical Page

The *logical page* characteristics can be summarized as;

1) It supports structure–based authoring, allowing the author to manipulate and define the structure of the document in an explicit manner.
2) It encourages and facilitates partitioning of the content into useful clusters (study units).
3) It allows the author to define appropriate and customized presentation and access methods for each study unit.
4) It presents the end–user with the structure as well as the contents of the current–topic and visual clues on accessing the contents. This reduces ambiguity and uncertainty on what is included in the current topic/subject and what is the proper access method and review order.
5) It supports display and presentation of multiple media simultaneously, providing user control for each media (e.g., push button, link, scroll bar, pop–up window)

Logical Page Architecture

A *LP* has one *primary window* (*PW*) and none, one or more *secondary windows* (*SW*), Figure 2. The *PW* is considered parent and owner of all *SWs* in that *LP*. In a *LP*, each window (*PW* or *SW*) has a unique window number (*WN*=1, 2, 3, ...) and a unique window ID (*WI* = alphanumeric_string) which are used for linking to and selecting the window. The *PW*'s *WN* is considered the *LP* number and the *PW*'s *WI* is the LP ID. The *SW*'s *WN* and *WI* are in fact treated as sub–topic number and ID within that *LP*.

In addition, all windows in a *LP* have a group number (*WG* = 1, 2, 3, ...). The *WG* allows grouping windows in a *LP* based on the type of information they contain, as well as the purpose or function they serve. Only one window of each group can be active (open, visible) in a *LP* at any time. When a new window is initiated in a *LP*, the *window manager* searches among the open windows in that *LP* to find one with a *WG* matching the new window being initiated. If a match is found, its contents are swapped with the contents of new window. If no match is found, the *window manager* opens a new window at the coordinate specified and displays its contents.

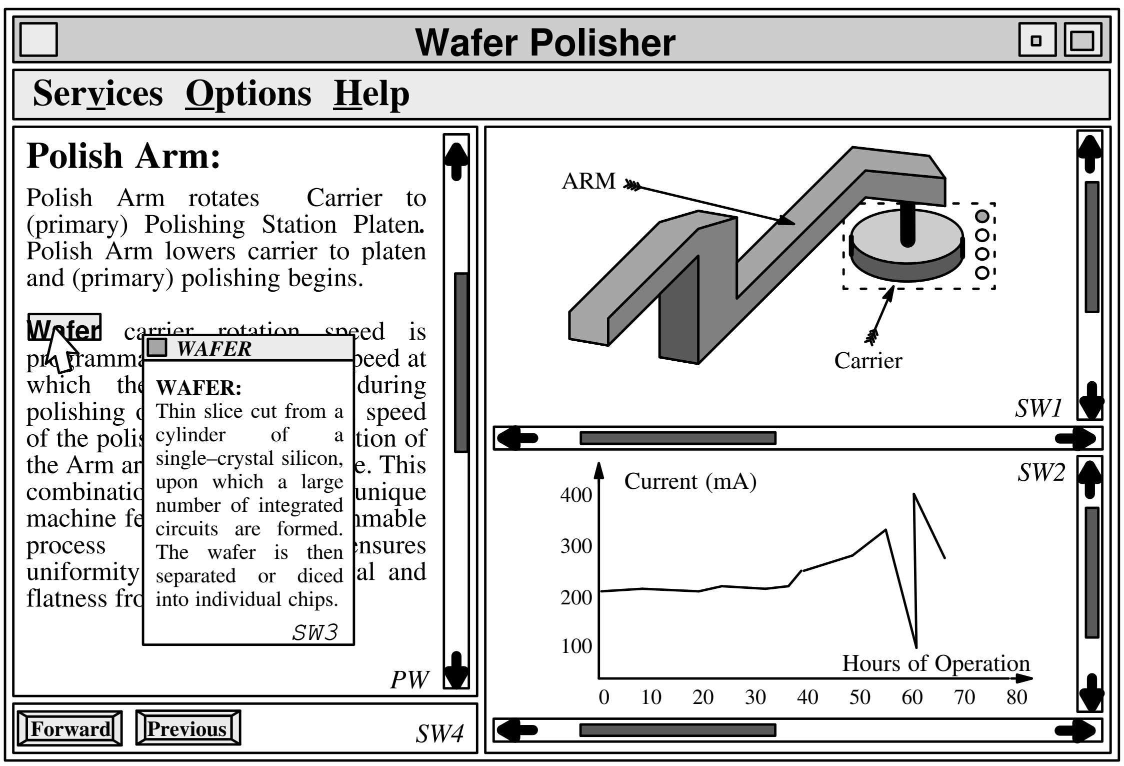

Figure 2. An Example of Logical Page Layout

To assure a complete and accurate display of a *LP* regardless of how it is activated (hypertext, hypergraphic, external linking, search, table of contents:TOC, or index selection), it can only be linked to via its *PW*. A *LP*, similar to a page of a book, may contain many different objects of interest, but can only be opened by a call to its *PW*. All other (secondary) windows in a *LP* are either dependent and automatically linked to the *PW* or are pop–up windows which are activated via hypertext or hypergraphic links. *SW*s are only accessible from within the *LP*. When a *PW* is opened, all its dependent *SW*s are opened automatically. A window can be assigned vertical and horizontal scroll bars, as well as title, background and foreground colors, font style and size, border style, coordinates, and system icons.

Logical Page Objects

To effectively present and manage an aggregation of hypermedia objects in our *LP* architecture, we have adopted an object–oriented approach and formalism as described in this and following sections. Wittenburg has also reported a similar model for adaptive management of mixed–media documents based on relational database technology and an object–oriented data model [Wittenburg94].

Each window in a *LP* (*PW* or *SW*) contains a set of hypermedia objects (e.g., text blocks, still images, video/animation clips, audio segments, graphics, charts, push buttons, hypertext links, and hypergraphic links). For example, in Figure 2, the *PW* contains three text objects (one title and two paragraphs) and a hypertext link to a pop–up window. The *SW1* contains a still image with a hot–spot. A hot–spot is a hypergraphic link which the user can select by double clicking on it, similar to a hypertext link. The *SW2* contains a chart object and the pop–up *SW3* contains two text objects.

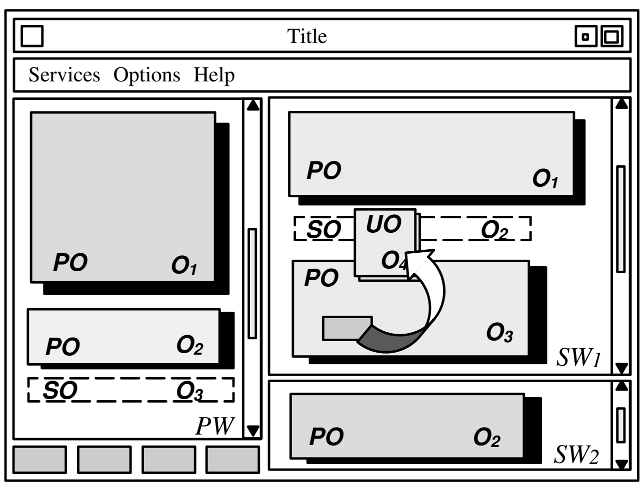

Figure 3. Hypermedia Object Types

Objects in a *LP* are either *primary objects* (*PO*), *secondary objects* (*SO*), *pop–up objects* (*UO*), or *hidden objects* (*HO*). *PO*s are automatically activated (displayed) whenever their parent window is displayed. *SO*s have a place holder in the window but are not automatically displayed (e.g., Object *O3* in *PW* and Object *O2* in *SW1* in Figure 3). *SO*s are displayed via direct (user selection) or indirect (hypertext, hypergraphic, or automatic link) activation. *SO*s are usually hypermedia utilities that require large amounts of computer resource (e.g., video, animation, audio) and therefore it is not practical to activate them by default every time the *LP* or their parent window is activated. The end–user should be made aware of these utilities and be allowed to decide when to activate them. *UO*s have no place holder in the window and simply pop–up in the pre–specified coordinates (or at the center of the parent window by default) when activated (e.g., *O4* in *SW1* in Figure 3). *UO* activation is usually via hypertext or hypergraphic links and is suitable for footnote and warning messages. *HO*s are *PO*s or *SO*s that are hidden (temporarily removed from presentation) based on some conditions defined at development time.

Object Attributes

In addition to some content, each object contains a set of attributes describing its type, display order, style, hide condition, and usage frequency as described below:

Object ID (OI) . . . . *A unique alphanumeric ID (object1, O_99, 101)*
Object Type (OT) . . . . *PO, SO, UO, and HO*
Object Content Type (OCT) . . . . *paragraph, still image, ordered list, ...*
Object Content (OC) . . . . *'Some text', 'c:\images\image99.tif'*
Object Content Method (OCM) . . . . . . *A method for initializing and updating the content*
Logical Page ID (LPI) . . . . *The LP containing the object*
Logical Page Style (LPS) . . . . *The style of LP containing the object*
Window Number (WN) . . . . *Object's parent Window number (e.g., 1234)*
Window ID (WI) . . . . *Object's parent Window ID (e.g., Monthly_Sales_Chart)*
Window Group (WG) . . . . *Object's parent Window group*
Window Type (WT) . . . . *Object's parent Window type (PW, SW)*
Display Order (DO) . . . . *The relative order of appearance of object in the window*
Display Order Method (DOM) . . . . *A method for initializing and updating the value of DO*
Activation Frequency Coefficient (AFC) *A number representing the frequency of object activation*
AFC Method (AFCM) . . . . *A method for initializing and updating the value of AFC*
Hide Condition (HC) . . . . *A variable–value pair (e.g., user_level=beginner)*

The *Activation Frequency Coefficient* (*AFC*) is a measure of usage or activation of an object in a window and is utilized in reordering of objects within a window to match the environment and user habits and needs. The activation of an object represents the likelihood that it would be relevant to the future users' need in that environment. The *AFC* is further described in the adaptive presentation section. An object attribute may have a method assigned to it. Such a method may automatically initialize or update the attribute's value based on some changes in the environment. For example, an object's *DOM* updates its *DO* based on the value of the object's *AFC* and some reordering threshold. The *AFCM* on the other hand increases the object's *AFC* value by Δ every time the object is activated. Value of Δ may depend on the type of activation (direct or indirect) and the source of activation (*PW*, *SW*, TOC, Search, ...). $\Delta = 1$ by default.

Object Sharing

The modular structure of the *LP* supports object sharing among pages. This means the same object instance (text, image, video,...) can be referenced in several different *LPs* but only defined once at authoring time and loaded once at run time. Each reference needs to define only those object attributes different from the first instance of the object (e.g., *LPI, WI, WT, WG, DOM, AFCM, HC*). The generic objects (e.g., safety procedures and warnings) and storage intensive objects (still images, video and audio clips) can be usually shared and therefore loaded only once for a module. This is very useful when an image, a video clip, an animation, or a text segment is referenced many times in a courseware.

Window Chaining in a Logical Page

When a *LP* is activated, its *PW* is automatically displayed. The *PW* in turn activates at least one *SW* (if there is one). Other *SW*s of the *LP* may get activated by the *PW* or other *SW*s. Figure 4 depicts three example scenarios. In scenario 1, the *PW* activates *SW1*, *and SW1* activates *SW2*, which in turn activates *SW3* (*PW–>SW1–>SW2–>SW3*). In other words, *SW1* is dependent and chained to *PW*, *SW2* is dependent on *SW1*, and *SW3* is dependent on *SW2*. In scenario 2, *PW* activates all secondary windows (*PW –>SW1:SW2:SW3*) meaning all secondary windows are dependent on and chained to *PW*. In scenario 3, *PW* activates *SW1* and *SW2*, and *SW2* in turn activates *SW3*. A window chaining scheme is selected at development time based on the relation between windows in a *LP*. In scenario 1, only *SW1* is directly related to *PW*, whereas in scenario 2, all three *SW*s are directly related to *PW* (i.e., *SW1*, *SW2*, and *SW3* are sub–sections of the topic covered in *PW).*

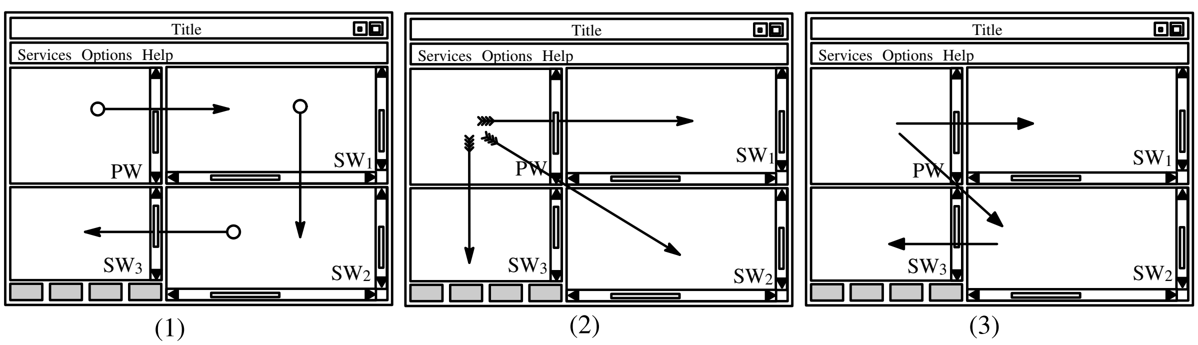

Figure 4. A partial List of Logical Page Window Chaining

Updating Window Contents

In some cases it is necessary to update contents of a window in a *LP* in response to activation of an object in the same or other windows. In other words, an object may update another object in the *LP* (in the same or a different window) or it may update the entire contents of a window in the *LP*. For example, the *PW* may contain names of people in an organization, the *SW1* may include the individuals' pictures, and

SW2 displays the employee profiles. In this case, when the user selects a name in the *PW*, the contents of *SW1* and *SW2* need to be updated. This can be handled in two different ways;

1) *PW* updates the content of one of the secondary windows, which in turn updates the contents of all other dependent and related secondary windows. See Figure 5a *(i)* and 5a *(j)*.

2) *PW* updates the content of all dependent and related windows directly. See Figure 5a *(k)*.

Of course, the initiating window (window containing the object which initiates the updating) does not have to be a *PW*. Figure 5b shows three examples of an object in a *SW* initiating update of the *PW* as well as other *SW*s.

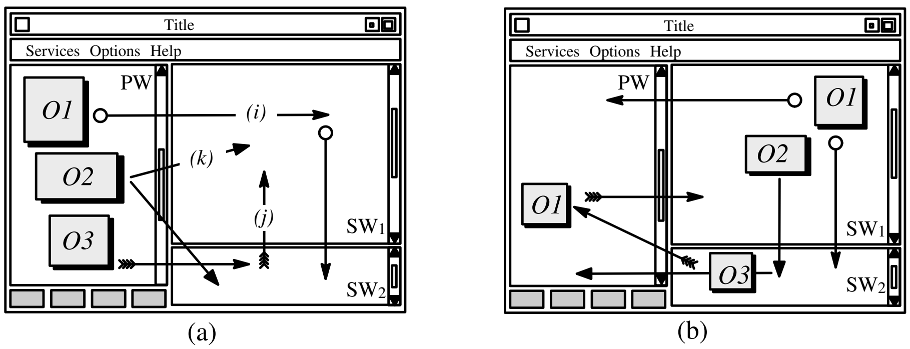

Figure 5. Logical Page Window Updating

Updating a window involves repainting or refreshing its contents or repainting the entire *LP*. Although it is possible to refresh only those objects which need to be updated rather than the entire contents of the window or the *LP*, our current implementation does not support direct object updating.

ADAPTIVE PRESENTATION

Hypermedia users often suffer from the 'lost in hyperspace' problem: distractions from too many jumps while traversing a complex network or too much skimming . Botafogo presents a solution to this problem by encouraging to create hierarchical hypermedia systems to make information stored easier to understand and use [Botafogo92]. Another solution would be to dynamically reduce unnecessary jumps and eliminate (or hide from users' view) irrelevant information objects. This would require to identify all relevant objects and to devise the shortest sequence to present all objects in the proper order. The scenario (list of relevant objects and the presentation sequence for a topic) may have to change over time to reflect the changes in the environment or as new information objects are added to the system. Such a system is considered adaptive if it changes in some way in response to its environment and users [Reiser85, Boyle93]. In the context of the presentation model described in this paper, such a system uses information about the users' goals (i.e., tasks performed in the environment), the active objects (i.e. visited *LP*s), and the strength of association between these objects (i.e. their *AFC*s) to dynamically add/delete/order the objects and modify their attributes. We can consider the objects in a window as an ordered list presented to the user for selection and/or review. Some of the objects are selectable and some are not. Initially these objects are ordered according to the author's description (initial objects' *DO* value). This initial ordering may not be a good representation of the deployment environment, although it might have been a good representation at one time or for some other environment. It is therefore necessary for the system to be able to dynamically adjust (adapt) to the environment. In some cases it becomes necessary to rearrange objects presented to the user, based on likelihood of usefulness, relevancy, usage, and even past experiences.

With a static presentation, users would need to examine on average $N/2$ objects (N= total number of objects in a window) to find a target object. With a properly ordered object list it would take ($M/(N/2) = 2M/N$) as much search (skimming) before the user reaches the desired object. Here M is the number of objects presented prior to the target object in the active window. Of course, it is desired that $M < N/2$ and eventually $M => 0$ as the system adapts to the environment. In other words, users of such a system would find all the relevant objects at the top of their corresponding window, thereby reducing (and eventually eliminating) search, scrolling, and skimming. Kaplan calls the $(M/(N/2))$ "error measure" and observes that error measure values less than 1 represent a net gain in efficiency for users [Kaplan90]. This measure compares the performance of an adaptive hypertext with systems employing a static contents list.

The three adaptive aspects of our presentation model described in this section – adaptive logical page object presentation, presentation contents customization, and adaptive hypergraphic functions – provide a mechanism for dynamically compiling, presenting, and managing online hypermedia courseware. The presentation model deals with the spatial structure of information. The content model deals with degree of details and customization of the information contents, and the hypergraphic model deals with the link network optimization.

Adaptive Logical Page Presentation

Our presentation model utilizes the object *AFC* (*Activation Frequency Coefficient*) to discriminate among objects of a *LP* and coordinate their presentation to the end–user. In other words, *AFC* is a measure of how frequently an object is used (accessed, or selected) in an environment. An object's *AFC* is a representation of its usefulness in the current environment and therefore its presentation (order, placement) should reflect that. In essence this rule makes the assumption that *the more an object was used in the past the more likely it is to be useful now and in the future*. Note that this is based on the assumption that all users of such a hypermedia courseware are performing the same or similar tasks in that environment. For example, all students in a science lab usually perform a set of similar tasks, and therefore their use of an online training or lab process documentation is based on a set of common goals. In other words, utilization of such a system depends largely on the set of tasks performed in that lab (classroom) and not individual students themselves. This is how we distinguish between adapting to the environment, rather than individual users/students themselves.

Whenever a window is activated, all its objects are displayed according to their *Display Order* (*DO*). *DO*s specify the order in which objects (or their place holders) are placed in a window. *DO* is a function of the objects' *AFC* and is updated based on the objects' *DOM* (*Display Order Method*). An object's *AFC* has an initial value of 0 and is increased by Δ each time the object is activated. Value of Δ may depend on the type and source of activation and is determined by *AFCM*. Application developer (courseware author) assigns the initial value and method for *DO*. The *Window manager* tracks object activation and updates the objects' *AFC* based on their *AFCM*. When a *LP* is activated, the *window manager* calculates object *DO*s and displays objects of each window according to their new *DO*s. Figure 6 shows two examples of the adaptation process of a *LP* to its final state.

Similar techniques in adaptive user interfaces have been reported by [Anderson87, Jerrams89, Reiser85]. In addition, Tomek proposes ranking of links based on the value of facts presented at the link and the relative value of links to the goal [Tomek93]. It assumes that the goal of the user is to select the next object so as to maximize his current knowledge. It then describes several quantitative measures that could be used to rank nodes, based on the context of the document, the goals of the user, the facts presented at a node, and the relative value of links and facts to the user goal. But it is not clear how to calculate the value of facts and links and how to select the value of coefficients. Tomek offers some good possibilities and an initial attempt in identifying and assigning values of facts and links, but it is not clear how to rank facts and identify their relative values in a complex system . In addition, a node, and therefore the facts presented at the node, may be linked to from several different other nodes. It then becomes necessary to assign different fact and link values for a node based on its incoming links. It is not clear how this can be accomplished in Tomek's proposal. Finally, the value of facts and information presented in an online document changes over time and varies from environment to environment. Tomek's static assignment of fact and link values does not allow system adaptation and reconfiguration based on their usage and environment. Our methodology does not discriminate between objects in a hypermedia system based on the value of their contents (which is subjective) and their link hierarchy (which is complex and unpredictable), but rather their initial perceived value (by the author) and the eventual actual value to the user (usage) and the deployment environment (relevancy).

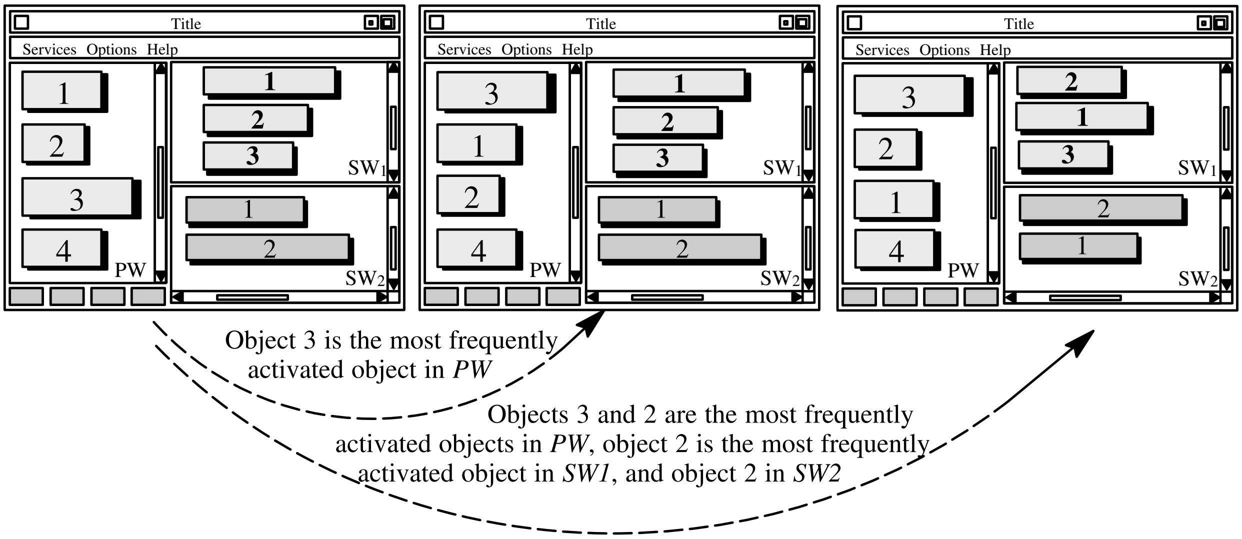

Figure 6. Object Presentation Adaptation

Customizing the Presentation Contents

In addition to updating the order of appearance of objects, objects may also be dynamically removed (hidden) from the *LP* based on some predefined conditions. In order to be able to hide an object it has to be of *Hide Object* (*HO*) type. A *HO* has a "*Hide Condition*" attribute which specifies a variable–value pair (e.g., user_level=advanced; video_card=no; test_score>90;). If a window contains an object type of *HO*, the *window manager* evaluates its *hide condition* whenever the window is activated or repainted. If the condition is true, the object is not displayed. If there are other objects in the window, those are shifted accordingly to fill up the gap. The *HO* feature is very useful in customizing the information presented to the user (contents of *LP*) based on some external criteria. For example, specific sections or segments of a *LP* as well as multimedia objects can be hidden from a user based on his/her level, job duties, needs, or changes in system configuration. These criteria could represent the user environment as well as the user's habits and usage trends. In other words, the system can adapt to the user and his/her environment by dynamically removing (hiding) specific portions of the information as the user progresses through the training. For example, in a training document, safety warning objects and detailed explanations suitable for beginners can be hidden from more experienced users (after they have been certified), based on their updated profile. A similar concept is implemented by Peralto–Fabi where several parameters of the information display (including the degree of content detail) are reconfigured based on the user profile [Peralta92].

Adaptive Hypergraphic Functions

A hypergraphic link is a selectable link assigned to an area (hot–spot) on an image. Each hot–spot can have several selectable functions as shown in Figure 7. When a hypergraphic link is selected (by clicking on its hot–spot area) its pop–up function menu is displayed. The user can then select any of the functions. Hot–spot functions allow the user to decide what form of additional information should be presented. In other words, hot–spot functions provide navigation between an object and its related information nodes in the hyperspace. Managing navigation between nodes and adapting to the navigation patterns in hypermedia information systems has recently received some attention under the topic of intelligent hypermedia systems (IHS) [Merlet93].

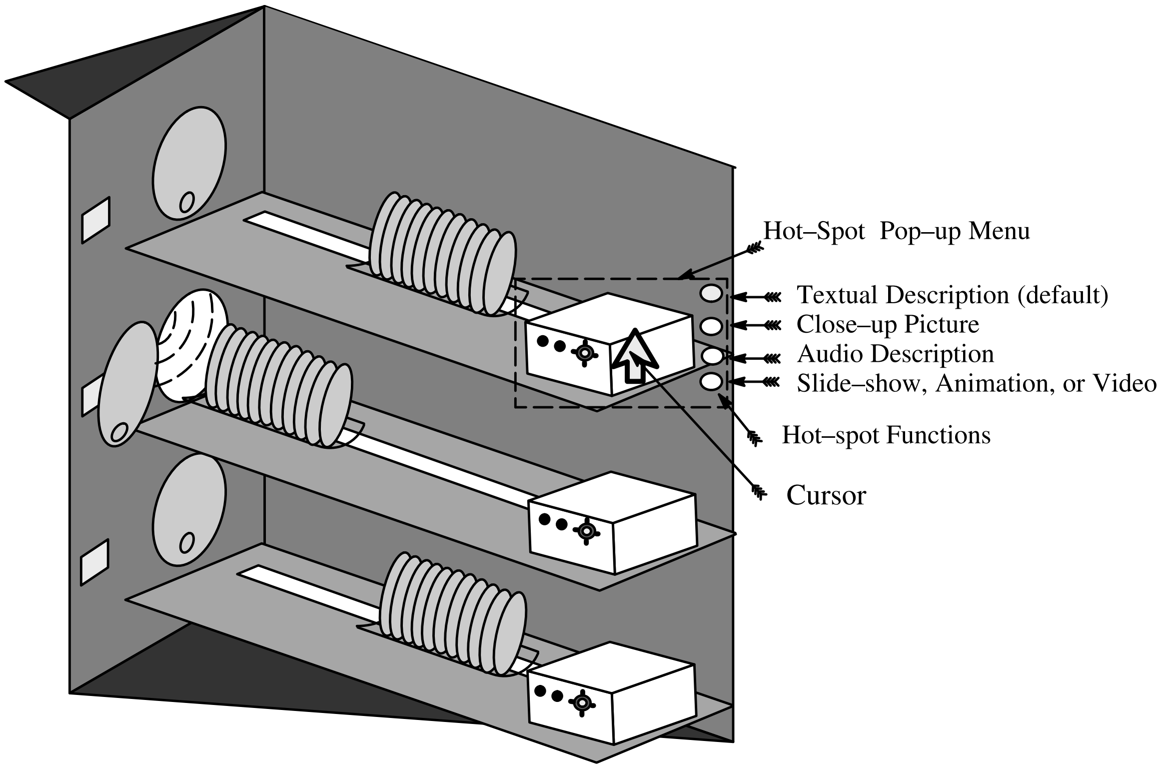

Figure 7. Hypergraphic Hot–Spot and Hot–Spot Functions

The hot–spot pop–up functions' appearance and default function selection adapt to the environment based on the frequency of usage and user habits. There are two adaptation scenarios:

(a) The most frequently activated function moves to the top of the function list (Figure 8a) .

(b) The most frequently activated function becomes the default function (Figure 8b) .

The default function is automatically selected if the user double clicks on the hot–spot, or hits the *Enter* key after selecting the hot–spot, or just clicks on the highlighted function button. Of course, the user may select any of the hot–spot functions, but the default is the most frequently activated function. Scenario (a) shown in Figure 8a has the disadvantage that the user interface presentation consistency is not maintained. In scenario (b) shown in Figure 8b, the user interface consistency is maintained but the default function position changes over time and in a sense adapts to the environment. For example, if audio is the most frequently selected hot–spot function for a specific hot–spot, then in scenario (a) the audio function button will gradually move to the top of the function list. In scenario (b), the default function highlight will shift to the audio button.

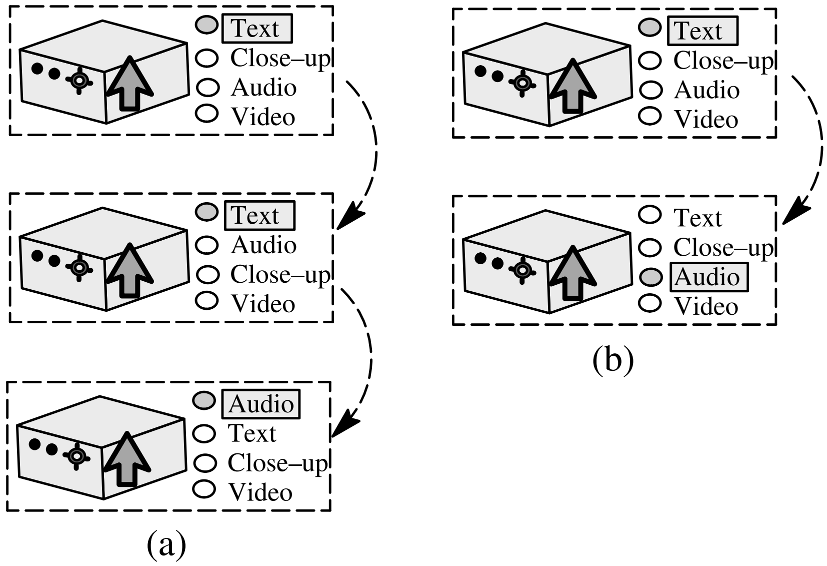

Figure 8. Hot–spot Pop–up Function Adaptation

IMPACT ON DEVELOPMENT AND UTILIZATION

In this section we present some preliminary analysis of the impact of our proposed model on development effort, as well as study time (the time it takes a user to study the courseware). We are still enhancing these quantitative measures, as well as defining additional qualitative measures to better understand and predict the impact of the proposed model on large scale development and deployment. The results reported here, although limited in scope, show positive improvements in delivery and dissemination of technical subject matters (e.g., online training, manufacturing procedures). In addition, they provide some useful metric for estimating the required development efforts and resources.

Impact of Partitioning and Logical Page Utilization on Study Time

A major concern with hypermedia courseware is the student getting lost, or countlessly paging through the document to find/compile the topics of his/her study subject. Such confusion and lengthy search will increase the required study time and may frustrate some of the students all together. To measure the impact of partitioning the courseware subject matter into customized and manageable study units (*logical pages*), we measured the time required to study hypermedia documents of various size. Figure 9 shows the study time for three versions of the same document at various stages of development.

The document, a semiconductor manufacturing training procedure, included over 500 objects (text blocks, images, graphs, diagrams, animation, still video, and audio segments) in its final form. The document was initially created in a text book style, with a table of contents and subjects followed sequentially. The table of contents and some limited hypertext links provided access to various parts of the document. This version of the document is considered a single *LP*. In the next phase, the document was partitioned to 5 *LP*s (each page containing a chapter). In the final phase, it was partitioned to 58 *LPs*, each *LP* containing a complete and independent study unit (a topic, section, or sub–section). Each *LP* contained on average 9 objects (min 5, max 14).

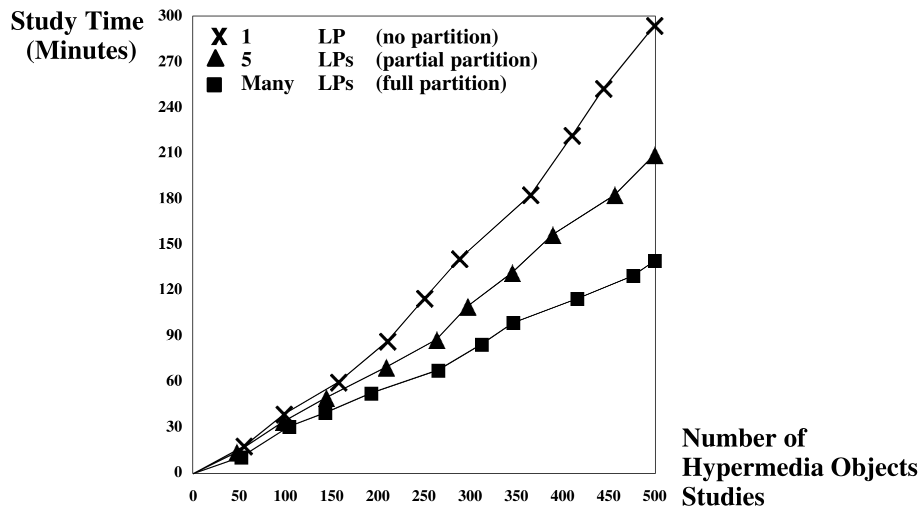

Figure 9. Average Study Time as a Function of Content Partitioning

A group of hypermedia application developers and end–users were asked to study the document to gain a good understanding of its contents (but not necessarily memorize the contents). They were asked to review the document in any order that made sense to them, but to make sure to cover the entire document. We then recorded the time it took the subjects to review (access and browse) each section of the document. From this we calculated average study time for varying numbers of objects at each of the three phases of development (no partition, partial partition, full partition). The result shown in Figure 9 indicates that study time for the fully partitioned module is linearly proportional to the number of objects in a given topic rather than total size of the document. This is a step towards making every interesting and useful unit of study equal distance from the user, regardless of the number of topics in the document. The improvement in study time associated with partitioning, use of *LP*s, and hypermedia navigation is consistent with results reported by other researchers [Rojas91, Stanton92]

Impact of Partitioning On Authoring Time

In addition to the impact of partitioning on study and access time, we were also interested in its impact on the development effort. It is clear that converting a sequential linear document to a fully partitioned and non–linear online hypermedia document would require additional effort. We were interested in establishing some benchmark on average development time as a function of final study time. Such an estimate would be useful in coordinating and managing large scale hypermedia development projects, as well project cost and time estimation. We should emphasize that although the numbers may not apply to other authoring systems and development/deployment environments, we believe the relative ratio of each implementation is similarly bounded (i.e., on average it takes an experienced developer two to two and half times as much effort to develop a fully partitioned version of a hypermedia document vs. a straight forward sequential version). This can be attributed to the additional effort required for analyzing and identifying clusters of information (*LP*s and their contents), their interdependencies and link networks, as well as appropriate *LP* style, window attributes and object characteristics. As shown in Figure 10a, the authoring time for sequential hypermedia documents increases proportionally to the size of the document, whereas the time for fully partitioned document based on our methodology is an increasing function of the document size and complexity.

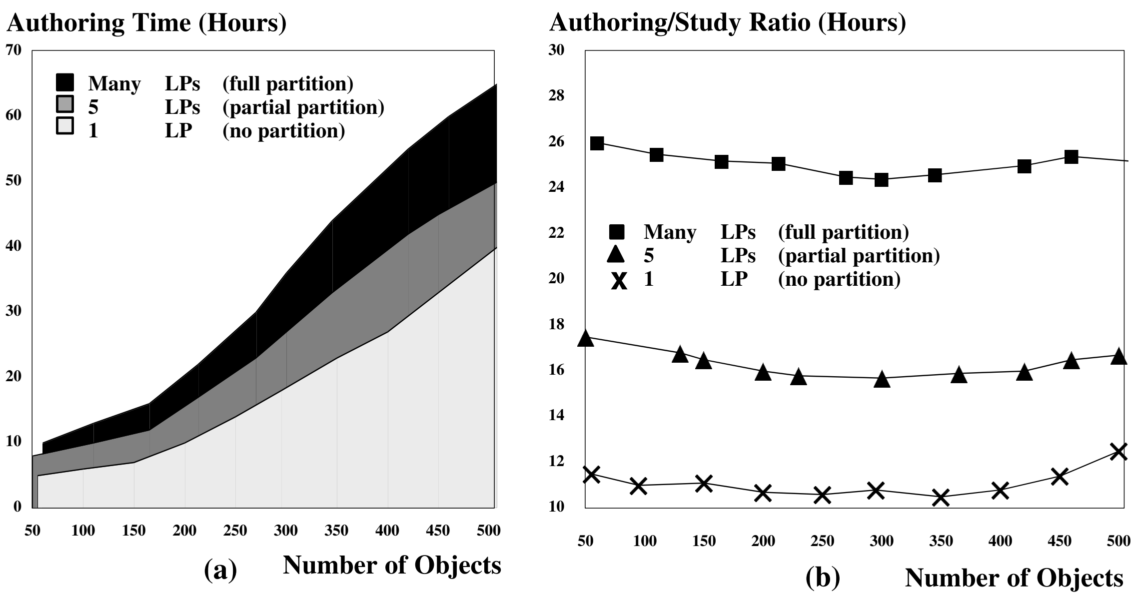

Figure 10. Development–Study Time Ratio as a Function of Partitioning

In addition, in our environment, using ExperMedia/2, it took experienced developers on average 25 hours of development per hour of online courseware, for fully partitioned documents, compared to about 10 hours for sequential linear hypermedia documents (Figure 10b). An hour of online courseware is the amount of courseware material covered by an average student in one hour. The development time does not include multimedia object creation (digitizing, scanning, recording, etc.). Figure 10b shows higher ratios for both small and large numbers of objects. We can list at least two reasons for an increasing authoring/study ratio as the number of objects increase. First, as the number of objects increases (increasing size of the document) the document load and save time, as well as compile time increases, although we suspect the real authoring ratio to remain constant. Secondly, increasing size of the document and number of objects increase the author's cognitive overhead and disorientation and require larger number of checks and rechecks to locate and link to objects and *LPs* of interest. The cognitive overhead impact is even more noticeable for a sequential development effort for large number of objects. The higher ratio for a small number of objects can be attributed to the initial fixed setup, coding, and knowledge engineering efforts.

Impact of Adaptation on Access and Study Time

Two factors impact the study time of online hypermedia courseware. A user has to locate the topic of interest, link to the topic, and then review its contents. Locating the topic of interest usually involves browsing the table of contents or topic lists, issuing a search or query command, and/or skimming through the document. Once the topic (*LP*) has been located, the user arrives at the topic via a hypertext or hypergraphic link or scrolling. Studying the subject matter usually involves a number of additional links (mouse clicks) or scrolls. In other words, a number of irrelevant topics may have to be initially bypassed to reach the topic of interest. Then a number of additional scrolls or links may also be required to bypass the irrelevant objects and to review all objects of interest in the topic. The number of required scrolls depends on the layout (logical page style), as well as on the subject matter organization.

Adaptation to the frequency of usage and reordering of the objects should over time reduce (ideally eliminate) the need for additional scrolling and mouse clicking for routine topics/subjects in an environment. In other words, users should be able to locate and reach topics of interest in a minimum number of scrolls and mouse clicks. In addition, the number of unrelated and uninteresting objects interfering with the study sequence should be minimized. One of the quantitative measures we have developed for measuring the effect of adaptation on studying a subject matter is the number of object scrolls and mouse clicks to complete a study.

For this analysis, we selected an orientation to a semiconductor fabrication facility. The document included over 800 objects and 74 logical pages. The document was originally developed by process technicians responsible for supervising and training the operators in the fabricator. We started from scratch with the goal of recreating the original document in 10 steps, developing additional logical pages in each step. At the end of each step, the document was compiled and half of the topics were selected randomly. For each of the selected topics, the document was initiated , appropriate selections were made to locate the topic and then the topic was browsed in its entirety (all hypertext and hypergraphic links were visited). Once all topics were visited, then the same topics were selected (in the same order) over and over until there were no more object reorderings (full adaptation is reached). In each case, the total number of objects scrolled, as well as the total number of mouse clicks (hypertext, hypergraphic, and external link selection) were recorded. The halfway point between the first trial and the last trial was selected as partial adaptation.

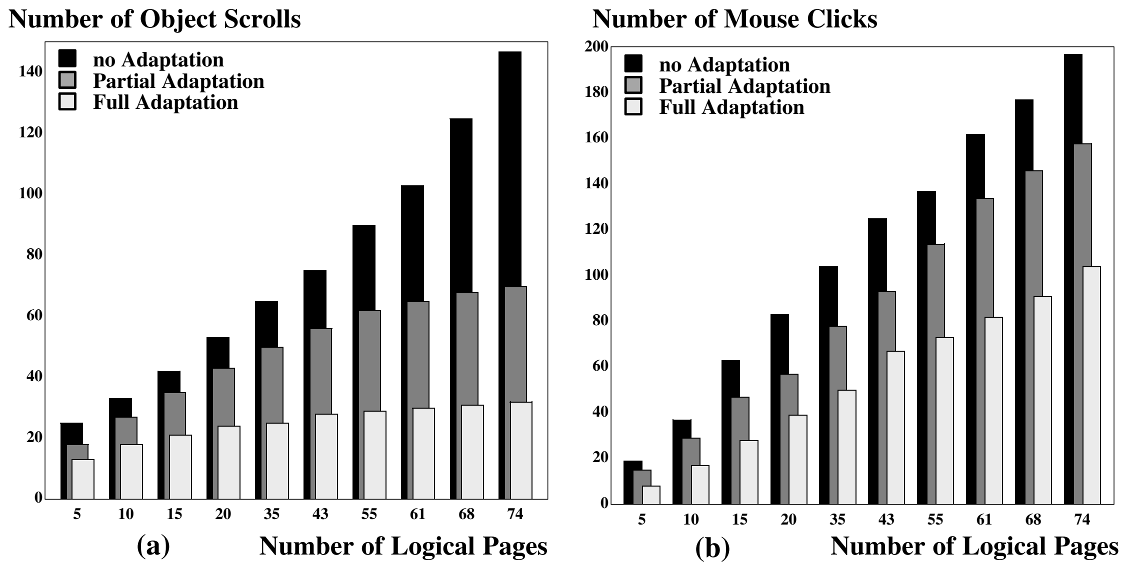

Figure 11. Impact of Adaptation on Object Scrolling and Mouse Clicking

Figure 11a shows the number of object scrolls for studying half of the topics (selected randomly) in the above document, before, during, and after complete adaptation. A complete adaptation is achieved when there are no or minimal object shifting as a result of reviewing the topics. Figure 11b shows the number of mouse clicks during this study. The results show about 50% reduction in object scrolling and mouse clicking for average size hypermedia documents and even larger reduction for larger and more complex documents. Reduction in scrolling and clicking reduces study time, reduces confusion and getting lost in the hyperspace network, and improves the comprehension and retention by reducing the irrelevant information objects.

SUMMARY

Although research in the field of hypermedia systems has provided valuable insight into the structural and organizational requirements of such online systems, it has not provided general purpose presentation models and development methodologies for educational systems. In this paper we propose a hypermedia presentation model for online hypermedia courseware. The proposed presentation model offers the end–user a range of engagement and study content generation options in a controlled and predefined structure. The courseware is partitioned into *logical pages.* The *logical page* supports object–oriented structure–based authoring of the courseware, allowing the author to manipulate and define its structure in an explicit manner. In addition to the presentation model and development methodology, the proposed model provides three adaptive features: adaptive object presentation, presentation contents customization, and adaptive hypergraphic functions. The object presentation deals with the spatial structure of information. The content customization deals with degree of details and customization of the information contents, and the adaptive hypergraphic deals with navigation between an object and its related information nodes in the hyperspace. Adaptive capabilities of our model depends largely on the set of tasks performed in the deployed environment (classroom, lab, manufacturing) and not individual students themselves.

Our analysis indicates that gains in information delivery and content access are traded for increased development effort. Information delivery gains (after full adaptation) are 10–20% reduction in users skimming too much information and 30–50% reduction in unnecessary hyperspace navigation. Information access gains can be summarized as 20 – 40% reduction in study time, 30–70% reduction in bypassing unrelated hypermedia information, and 20 – 60% reduction in distracting actions (mouse clicks, menu option selection, waiting for hypermedia objects to load/display). Information delivery and access gains are dependent on the size of the courseware and dynamism of the deployment environment. In our future research we plan to analyze the impact on the information quality and student retention.

Acknowledgment

The author would like to thank IBM management Mike Potter, Rufus White and Dr. Manny Athavale for encouragement and support of this research, and the anonymous JEMH referees for their comments and suggestions on the earlier version of this paper.

References

[Andeson87] Anderson M. P., McDonald J. E., and Schvaneveldt R. W., “Empirical user modeling: Command usage analyses for deriving models of users,” Proc. the Human Factors Society, 31st Annual Meeting, pp. 41–45, 1987.

[Augenstein93] Augenstein F., Ottmann T, and Schoning J., “Logical Markup For Hypermedia Documents: The TRAIN–System,” Proc. of ED–MEDIA 93, pp. 17–25, 1993.

[Bareiss93] Bareiss R., Osgood R., “Applying AI models to the design of exploratory hypermedia systems,” Proc. of the ACM Conf. on Hypertext, pp. 94–105, 1993.

[Botafogo92] Botafogo A. R., Rivlin E., and Shneiderman B., “Structural analysis of hypertexts: Identifying hierarchies and useful metrics,” ACM Trans Inf Syst v 10 n 2, pp. 142–180, April 1992.

[Botafogo93] Botafogo A. R., “Cluster analysis for hypertext systems,” Proc Annual Int ACM SIGIR conf on Research and Development in Information Retrieval, pp. 116–124, 1993.

[Boyle93] Boyle C., Teh S. H., “Multimedia intelligent documentation: MetaDoc V,” Proc. of the 11th Annual Int. Conf. on SIGDOC, pub by ACM, New York, NY, US, pp. 21–27, 1993.

[Feiner90] Feiner S. and McKeown K., “Coordinating Text and Graphics in Explanation Generation,” Proc. AAAI–90, pp. 442–449, Boston, MA, July 29–August 3, 1990.

[Feiner91] Feiner S. and McKeown K., “Automating the Generation of Coordinated Multimedia Explanation,” IEEE Computer 24(10), pp. 33–41, October 91.

[Hardman93] Hardman L., Rossum G., and Bulterman D., “Structured Multimedia Authoring,” Proc. of 1st ACM International Conference on Multimedia, pp. 283–289, Los Angeles, CA, 1993.

[Hamond89] Hammond N, “Instructional models for hypermedia,” Proc. of Hypertext II, York: University of York, 1989.

[Hekmat93] Hekmatpour A., Brown G., Brault R., Bowen G., Grant L., Millham E., “FTDD973: A Multimedia Knowledge Based System and Methodology for Operator Training and Diagnostics,” Proc. of Conference on Intelligent Computer–Aided Training and Virtual Environment Technology (ICAT/VET–93), NASA/Johnson Space Center, Houston, Texas, May 1993.

[Hekmat93.1] Hekmatpour A., Millham E., Grant L. “ExperMedia/2: An OS/2 Multimedia Expert System Shell for Domain Experts,” presented at the World Conference on Educational Multimedia and Hypermedia Conference (ED–MEDIA 93), Orlando, Florida, June 1993.

[Hekmat94] Hekmatpour A., “A Multimedia Methodology and Architecture for Computer–Aided Training and Certification,” Int J of Computers and Their Applications, v 1 n 2, pp. 67–78, Dec. 1994.

[Jerrams89] Jerrams–Smith J., “An attempt to incorporate expertise about users into an intelligent interface for Unix,” Int. J. of Man–Machine Studies, Vol. 31, pp. 269–292, 1989.

[Kaplan90] Kaplan G. and Wolff G., “Adaptive Hypertext,” Technical Report, Human Factors Center, IBM Santa Teresa Labs, San Jose, California, 1990.

[Kong94] Kong H. P., “Intelligent multimedia supported instructional system,” Expert Systems with Applications, v7 n3, pp. 451–465, July–Sept 1994.

[Maybury91] Maybury M. T., ”Planning Multimedia Explanation Using Communicative Acts,” Proc. of AAAI–91, pp.61–66.

[Merlet93] Merlet, Jean–Francois, “Human computer interactions through an intelligent hypermedia,” Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics Part 4 (of 5), pp. 268–271, 1993.

[Norman89] Norman Donald A., “The Design of Everyday Things,” Doubleday: NY, 1989.

[Okamoto94] Okamoto T., “Current situations and future directions of intelligent CAI research/development,” IEICE Trans on Information and Systems, v E77–D n 1, pp. 9–18, Jan 1994.

[Peralta92] Peralta–Fabi R., Peralta A., et. al., “Large interactive database: design and implementation,” Proc. of SPIE, V 1662, Image Storage and Retrieval Systems Conf., pp.93–99, 1992.

[Reiser85] Reiser B. J., Anderson J. R., and Farrell R. G., “Dynamic student modeling in an intelligent tutor for Lisp programming,” Proc. Ninth International Joint Conference on Artificial Intelligence, pp. 8–14, Los Angeles 1985.

[Rojas91] Rojas–Fernandez, Helen, “Online and Hypermedia Information Design,” Proc. of IPCC 91, Engineered Communication, IEEE Cat. NO. 92CH3087–4, PP. 28–32, 1992.

[Stanton92] Stanton N. A., “An Investigation of Styles and Strategies in Self–Directed Learning,” J. of Educational Multimedia and Hypermedia, Vol. 1, No. 2, pp.147–167, 1992.

[Thies94] Thies M. A., “Adaptive user interfaces,” Proc. of the 13th World Computer Congress, Part 2 (of 3), pp. 196–202, Hamburg Ger, 1994.

[Tomek93] Tomek I., Maurer H., and Nasser M., “Optimal Presentation of Links in Large Hypermedia Systems,” Proc. of ED–MEDIA 93, pp. 511–518, 1993.

[Wittenburg94] Wittenburg T. M., “Adaptive document management system for shared multimedia data,” Proc int conf Multimedia Computing and Systems, pp. 245–254, Los Alamitos, CA, USA, 1994.