INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

A MULTIMEDIA AUTHORING TOOL FOR WEB BASED LEARNING

THOTTAM RANGANATHAN SRIRAM

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

OCTOBER 1998

Abstract

A Multimedia authoring tool for Web based Learning

Thottam Ranganathan Sriram

Internet is a network infrastructure that connects millions of computers and users worldwide. Increasing interest and access to the worldwide web have made the web as a potentially suitable medium for Computer Aided Learning (CAL). A web based presentation (courseware) is a collection of individual web pages having spatial and temporal objects in it. Spatial objects need to be organised in a two dimensional space, and temporal objects need to be sequenced. Our aim in this thesis is to provide an authoring toolkit which supports the author as much as possible to think in high-level domain oriented terms from which we generate a set of low-level spatial and temporal constraints automatically.

A syntax-directed translation and a grammar are used for this purpose. We have developed a grammar that is suitable for specifying the spatial and temporal constraints in a page. A Concept Graph Model, which is a directed acyclic graph is used to organise a web based presentation of the courseware. The authoring and presentation tool kits developed as part of this thesis work are evaluated by applying them to a real life problem. In this application, the author has developed a courseware on Assembly Language for a first year course in Computer Science. The input for the authoring tool, in this case, were made available by the author as Microsoft Word document and as slides in Microsoft Power Point. The authored material is used by about twenty five students as a way of evaluation. The conclusions are, *it is relatively easy to develop courseware using our authoring toolkit* and *incremental development of the courseware is simple.* The authored courseware is found by students to be *very useful* in Computer Aided Learning.

Acknowledgments

I wish to take this opportunity to thank all the people who have contributed in making this dissertation possible.

Thanks are due to Dr. T Radhakrishnan and Dr. Manas Saksena who not only helped supervise this dissertation, but were a constant source of help and encouragement. I wish to thank Dr Rajaraman of Indian Institute of Science, Bangalore for his helpful discussions on the Concept Graph Model.

I wish to thank the monitors of Computer Science department, Concordia University for their patience and valuable suggestions during my implementation. I would like to thank Mr. Stan Swiercz for his valuable suggestions during the development process.

I would like to thank Ms. Toopana Pathmanathan for her untiring work in helping to create the materials for the courseware and the presentation itself.

I would like to thank Monkiewicz Halina for her help as a graduate program secretary; she made my life a lot easier in the last few months. Thanks are also due to Bowen Edwina for her help.

I would like to thank my friends and colleagues in the multimedia lab for their useful discussions and suggestions. Thanks are due to my colleagues and room mates Bhaskar and Venkat for their support and understanding. I would like to take this

oppurtunity to thank Vitaly Iourtchenko for his valuable suggestions through the difficult phases of implementation.

I appreciate the friendship of all Concordia students past and present for providing a lively work place. My special thanks are due to my supervisors Dr T Radhakrishnan and Dr Manas Saksena for providing us with an excellent lab to make this research possible.

I wish to thank my family and friends in India who have been very supportive of my endeavours and have been a constant source of encouragement. Special thanks are due to my parents who must have done it right for me to reach this position in life. Finally, I thank my wife Mahima for her moral support, patience and understanding, for her love, during the difficult phases of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Making a multimedia presentation is a complex process. The main problem we are concerned with is how to simplify the process of planning, building, delivering, and maintaining an interactive multimedia presentation for Computer Aided Learning (CAL). Authors from different domains, viz. authors who are experts in computer programming, and authors who are not experts in computer programming, would wish to present their material on the computer using the web. There should be enough support given to authors irrespective of their domain expertise, to let them create interactive multimedia presentations with ease.

Authors think of a presentation in terms of a collection of concepts in a hierarchical manner. There are two different ways in which presentations can be classified, firstly as a collection of concepts in a hierarchical manner and which can then be refined as individual concepts, or as individual concepts first and then group them as a collection of concepts in a hierarchical manner. These two approaches are called top-down and bottom-up respectively.

It has been well recognised that interactive multimedia presentation is facilitated through the use of low level web based programming language. As the web is a

widely used medium, it is unfair to expect non-computer expert authors to be cognizant of the various computer primitives to deliver quality presentation on the web.

In this dissertation we address the problem of authoring a web based presentation. Our first goal is to reduce the complexity in planning, building, delivering and maintaining a multimedia presentation. Our second goal is to achieve highly interactive multimedia presentation. The result of our research is a layout model, event model and Concept Graph model of presenting a multimedia presentation. This approach simplifies the process of authoring as much as possible, thus enabling non-expert authors to create quality web based courseware.

1.1 Motivation

We consider the problem of reducing the complexity in developing Web based presentation. The problem of simplifying authoring web based presentation is motivated by the need to enable both expert and non-expert authors to deliver and maintain quality web based courseware. In the following sections, we address the motivating forces behind our research.

Web pages were static in nature till the advent of dynamic HTML and Java. Highly interactive web pages can be developed using Java applets and dynamic HTML, but this would involve in-depth understanding of various underlying concepts and good programming abilities. Resources should be laid out in a page in an organised manner during presentation. In interactive pages, the contents of the page keep changing regularly, thus affecting the layout of the page. It will be beneficial for authors if they will be able to specify core aspects of the layout, and maintain them throughout the presentation immaterial of the changes in content of the resources. The second issue is interaction between resources in a page. An interactive presentation should be able to react to user inputs dynamically,i.e., authors should be able to define events and

its corresponding actions easily.

Another issue is maintenance of these pages. Authors should be aided in maintaining and incrementally developing the pages. This is an arduous task if done manually, as the authors have to edit the code for the pages manually to realise the changes.

The final issue is making a structured and organised presentation. There should be ways in which authors can collect the pages in an organised manner to make a document. As documents refer to a collection of pages, there should be ways in which users can interact with a document. Authors should be able to incrementally develop a document. The choices to be made are,

- To select an effective medium where authors can present their presentation.
- To achieve separation of concerns, i.e., to let authors think and work mostly in their domain of expertise, which we call the *conceptual layer*, and to develop an authoring tool to do the translation from the conceptual layer to the syntax layer.
- To devise methods to aid authors as much as possible to plan, build, deliver, and maintain well structured and interactive presentations.

1.2 Our Approach

Figure 1 shows the various layers in making a multimedia presentation as a whole. As shown in Figure 1 we choose the Internet as the medium for presentation. The choice of Internet is evident, as it is a network infrastructure that connects millions of computers worldwide representing millions of users. Apart from the above, its availability in different operating system environments, its level of penetration into all walks of human kind, and its "exponential growth" over the last years make Internet the popular medium for presentation.

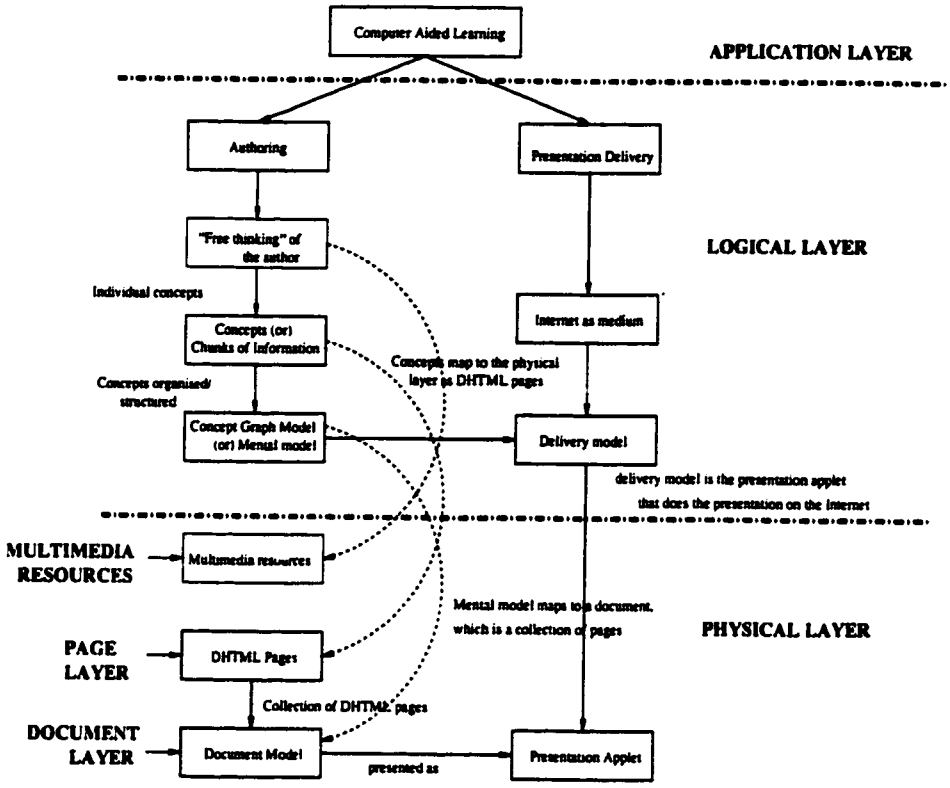

Figure 1: Layers in a presentation

The various tasks, the task of the author in his/her domain of expertise, the task of the authoring tool, and the application are layered as shown in Figure 1 to achieve separation of concerns. There are three different layers, viz. *the Application layer, the Logical layer, and the Physical layer.* Authors operate in the logical layer where the model that they operate on is called as the *mental model.* In this layer authors view a presentation as concepts and collection of concepts linked by associations. Concepts are chunks of information which have spatial and temporal property associated with them. A presentation for Computer Aided Learning comprises of two parts, the authoring part and presentation or delivery. In the physical layer, the mental model of the author is transformed into computer operations and realised for delivery. There should be ways in which the mental model of the author can be transformed and represented in the physical layer. This mental model of the author in our case is a graph structure, where nodes represent concepts and edges represent associations.

We transform the mental model of the author into another graph structure in the physical layer which we call as the document model. The chunks of information or concepts map to the physical layer as Dynamic HTML pages, and the collection of concepts or the mental model of the author maps in to a Document.

In the presentation delivery part, we are concerned with the delivery of the document as well as the Dynamic HTML pages. As one of the essential features of CAL is interactivity, the presentation delivery has to be interactive. The delivery model in the logical layer is mapped into the physical layer as a presentation Java applet, that can do the presentation on the Internet.

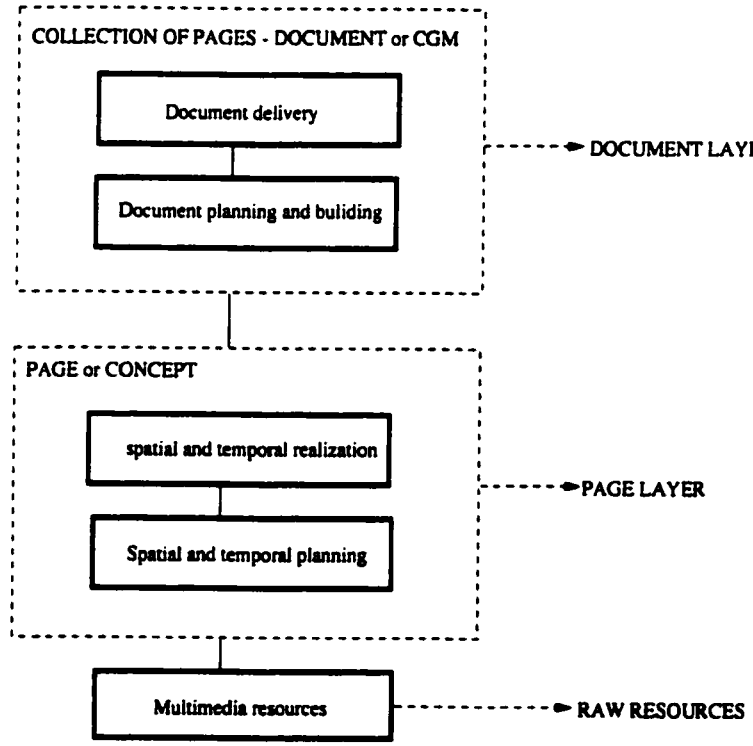

Figure 2: Block diagram of physical layer

As shown in Figure 3, a multimedia presentation consists of a page layer to let authors specify the spatial and temporal aspects of individual concepts of a presentation and a document layer to group these pages. Spatial aspects define the physical placement and alignment of resources within a page, and temporal aspects define the dynamic

properties of the resources in a page. The page authoring model should be able to interpret the definitions and requirements for the spatial layout of a page to generate the necessary low level code. As shown in Figure 3 various other commercially available web authoring tools and HTML generating tools can be used to generate HTML. We propose a spatial and temporal grammar for content independent layering of resources and specifying the set of interactions between resources in a page.

The next hierarchical step in a presentation is to collect these concepts in an orderly manner to form a document. As making a presentation and subsequent maintenance of it are complicated and done manually, we propose a model based approach for this purpose. We propose a *Document Model* which is a directed acyclic graph, where nodes represent concepts and edges represent precedence relation. We provide a *document planning* environment where authors can drag and drop the concepts, and define edges between the concepts to plan a document. Once the document is planned, the *document building tool* builds the appropriate code to realize the document, and the *document delivery tool* delivers the document on the browser. The document delivery tool presents the Document Model, which allows controlled and monitored navigation for users. Individual concepts are supported by custom Java and Java Script libraries at run-time to achieve spatial layering and desired temporal behaviour.

We consider an architecture consisting of two components, an authoring component and a presentation component. The author uses the authoring component to create the presentation from his/her logical model. The authoring component is further divided into two parts, the page authoring and the document authoring. The page authoring defines the spatial and temporal aspects of a page, and the document authoring defines the relation and the hierarchy between the pages.

The presentation component comprises of two parts, the document presentation and

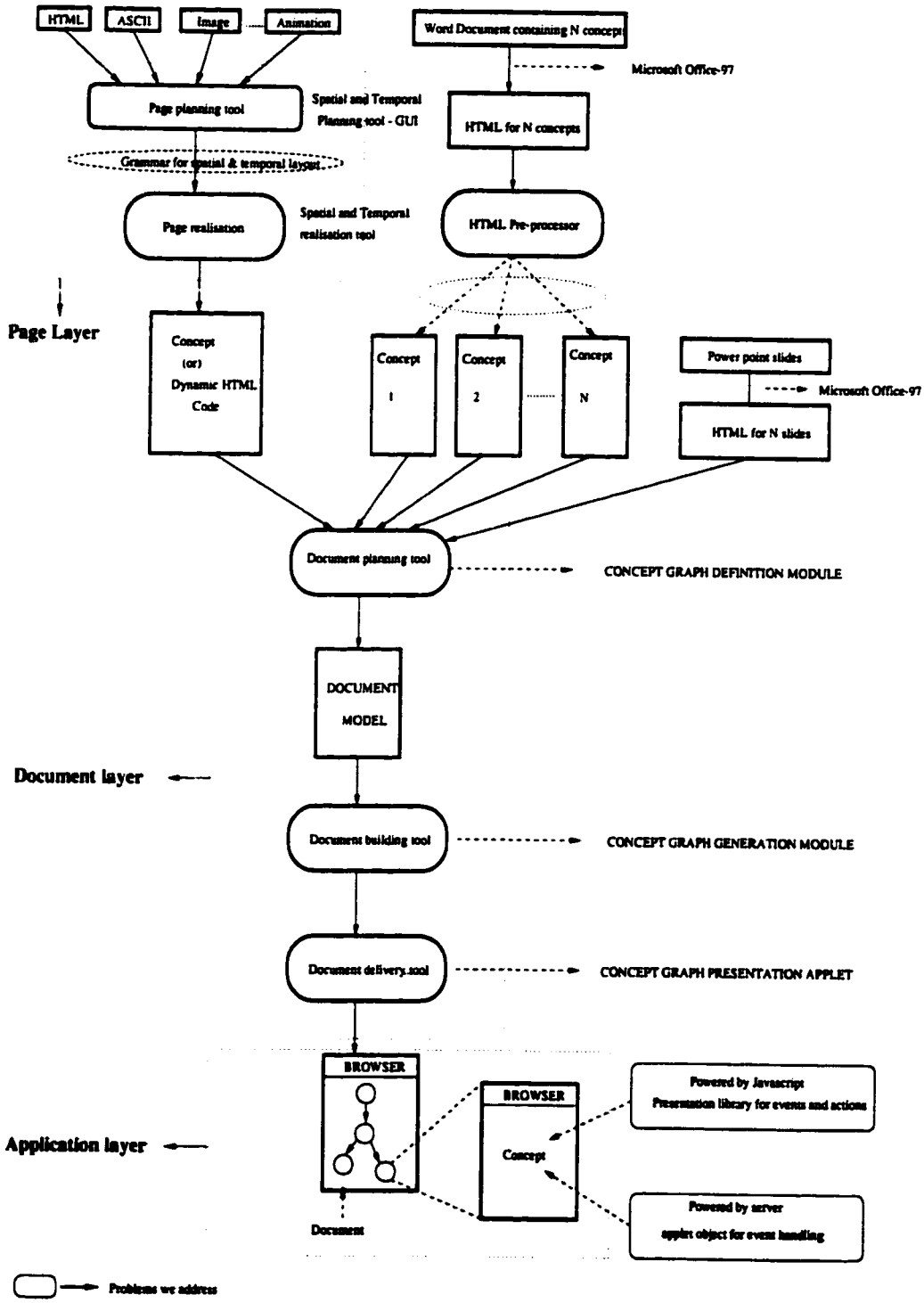

Figure 3: Page layer, document layer, and application layer in presentation

the page presentation. The page presentation realises the spatial and temporal aspects of a page, and the document presentation realises the document.

In our design of the authoring component, we emphasis on simplifying the authoring process as much as possible. The page authoring comprises of three definitions, the resource definition, the layout definition and the event definition. The resource definition defines the nature and type of resource. The layout definition defines the spatial aspects of a page, and the event definition defines the interaction between the resources in a page. In the design of the document authoring, a graphical user interface is developed to create, modify and maintain a concept graph. The nodes of the graph represent the pages, and the edges the relation between the connecting pages.

In the design of the presentation component, we emphasis on delivering an active presentation. The page presentation uses concepts of dynamic HTML to layer the resources in a page during presentation time. To achieve an interactive presentation a client-server model of event handling is developed. The document presentation realises the authored document on the web using a document presentation applet.

1.2.1 Authoring component

A web page is a collection of various multi-media resources. Traditionally, these resources cannot be dynamically manipulated while or after loading of a page using static HTML, making the web static in nature. But, the advent of Java, Java Script and dynamic HTML have made the web more dynamic and interactive. Dynamic HTML makes it possible to determine and change properties of resources during page loading or during the course of a presentation. The code required to generate such layout on a page is complex, usually written in Java and Java Script. This thesis describes a grammar based approach for simplified specification of spatial layout of resources in a page by non-computer expert authors.

Resources in a page can interact among themselves using dynamic HTML. Such interactions can be used to achieve effective communication among resources in a page. For example, a text display might be required to run in synchronisation with an audio resource, changing the text display for every newly loaded audio. It would be highly beneficial for authors, if they can define the interactions between the resources in a page easily, rather than realizing the interactions using low level programming.

Static mark-up languages do not provide the designer the capability to control precisely the layout of the page when its parameters are modified. A solution to this problem is to use constraints to specify the core aspects of the design layout. The constraints capture the semantics of the design. An authoring tool takes as input a set of constraints representing the plan for the presentation and automatically generates the corresponding output to realise the same.

Authors need supporting tools to plan, build, deliver, and maintain a full presentation. Although most tools suggest various means and methods to assist authors in developing individual concepts, it will be beneficial if authors can be given an authoring environment where they can organise, build and present a collection of concepts as a document. The organisation of concepts can be done both in a top-down as well as in a bottom-up approach, i.e., authors make concepts first and then organise them to form a presentation, or make a skeleton presentation and fill the empty nodes with concepts.

We suggest an approach called the Concept Graph Model (CGM) which is particularly suited for Computer Aided Learning. In CGM, nodes represent concepts and edges represent precedence relation between the various concepts. For example, if there is a forward edge from Node A to Node B, then Node A should be learnt before

Node B. Apart from the above, we also aim at simplifying the maintenance and modification of the presentation. In the current scenario, it is too difficult to maintain a large collection of pages with multiple interconnecting links. The deletion of one page affects the references to that page in all other pages and such references have to be corrected manually. In our model, the authoring tool automatically takes care of this and thus simplifies the maintenance of a document.

1.2.2 Presentation component

Once the presentation has been delivered, the presentation system generates events that should be handled, evaluated, and executed. Figure 4 shows the event model which describes three stages in event processing, viz. event handling, event evaluation, and action execution. Events can be generated in two ways: by the user or by the system. User-generated events occur when the user interacts with the browser by moving the mouse, clicking a mouse button, or pressing keys on the keyboard. System-generated events occur when the state of the system changes, such as an error occurring after a specified time period or a page finishing loading. The event model defines a handling module to realise events, and an evaluation module to evaluate the event and the action, and action execution module to execute the corresponding action.

Java applets are programs that run on the browser that bring interactivity in web pages. It is possible to present a *document model* with the help of a Java applet, that a person can use to interact with a presentation. The document presentation system uses this Java applets to realise the authored document. It is very easy for casual users to get lost or loose track of their relative location when browsing through a presentation in the Internet. The document presentation system provides controlled and well directed navigation for users. Each interaction with the graph could trigger a web page displaying the content of the page.

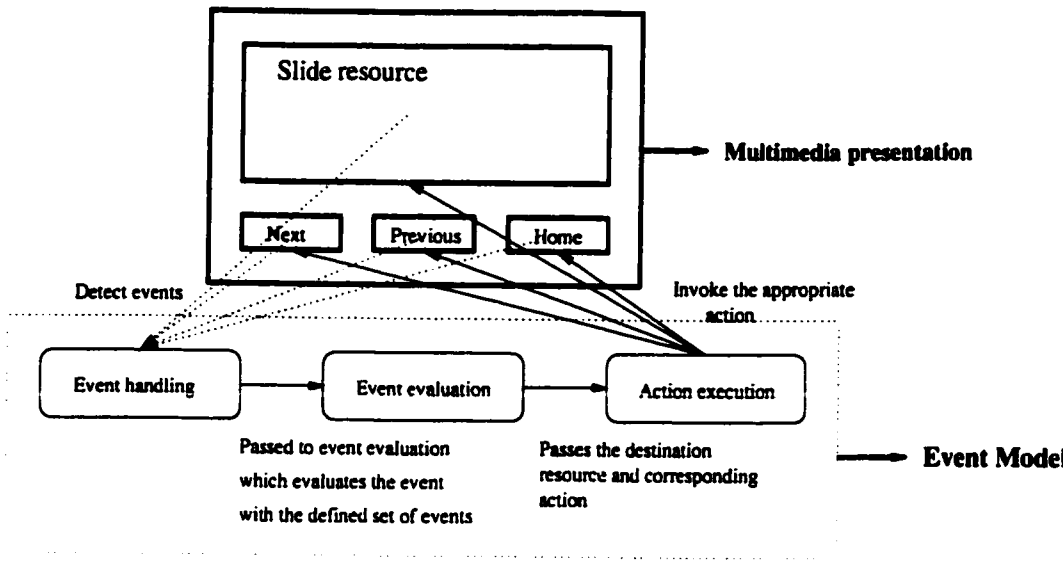

Figure 4: Event processing architecture

1.3 Contributions

The main contributions of the dissertation are itemised below:

- We propose a Web authoring tool that can be used to reduce the complexity involved in generating and presenting interactive courseware.
- We present an approach using two models, the authoring model and the presentation model. We propose a grammar based approach for mapping from a representation of the content and context of a presentation, to forms that specify the media objects to be realised.
- We address spatial and temporal aspects of a presentation during authoring and delivery. We have implemented a grammar that can be used to specify spatial and temporal behaviour of a page.
- We have implemented a page authoring module to automatically generate low level code to realise the specified spatial and temporal properties of a page.
- We develop a graph editor which is used to create a document. The graph editor provides easy drag and drop environment for authors to create their

presentation.

- We develop a presentation applet that can be used to present a Concept Graph. We demonstrate how the interactive applet can be used to achieve an interactive multimedia presentation.

1.4 Organisation

In this thesis, in the page layer we suggest a grammar based approach for planning and realization with respective to both spatial and temporal aspects of a page. In the document layer, we propose a model based approach called as the Concept Graph Model (CGM) to help authors plan, build, deliver, and maintain a presentation. Chapter 2 discusses related work in detail. In Chapter 3 we describe the grammar based approach in detail. We also specify ways of defining custom events besides the standard events in order to achieve sophisticated interaction within a page. We analyse the various issues involved in a multimedia presentation more in detail. In the page authoring part we discuss the spatial and the temporal aspects of a presentation. In the spatial aspects we go more in to detail and discuss about absolute positioning versus relative positioning. In the temporal aspects of a presentation we discuss about static multimedia, hyper media, passive multimedia, and active multimedia presentation. In Chapter 4 we discuss about implementation of the authoring tool. In Chapter 5 we present a real-life example of a slide show for an undergraduate computer science course in assembly language programming that has been developed using our tool. Finally, we state the advantages of such an approach and the trade-offs that we make in simplifying the existing complex structure.

Chapter 2

A summary of related work

In this chapter we present a summary of related work in the literature. We compare the related work based on various aspects of a multimedia presentation like, spatial and temporal planning, event architecture, methodology, interface support and intended application. In the methodology, we look into support given to authors to plan their presentation using both top-down and bottom-up techniques. A single concept is considered to have associated spatial and temporal components with it. In the temporal aspects of a presentation, we classify it further into static, hyper-media, passive, and active multimedia type presentation depending on the type of events the presentation supports. We discuss related work and compare our work on temporal aspects of a presentation more in detail in the subsequent chapters. In the spatial aspects of a presentation, we discuss layout of components using absolute positioning and using relative positioning. We classify the contents of a page into two categories, viz. static contents, contents that never change during the presentation (run-time) and dynamic contents, contents that change during a presentation (at run-time).

We compare and contrast the various approaches taken by researchers to solving the problem of making a multimedia presentation. We also discuss about the user interface provided to aid authors in building their presentation. We discuss two modes of interface, the *opaque and the transparent interface.* Though we propose a GUI

based user interface as our future work, we recommend a transparent interface as a default support as it assists both types of authors, viz. expert authors and non-expert authors.

2.1 Temporal model for interactive multimedia scenarios

Hirzalla, Falchuk, and Karmouch [1] talks about the modes and methods of representing multimedia documents. Their paper does not focus on layout and quality of presentation. Its main focus is on the temporal aspects of a presentation.

The model used in their approach is the enhanced time-line model. This paper clearly shows the limitations in time-line model in expressing asynchronous events, as the traditional time-line model requires a complete a priori specification of all temporal relations. The author distinguishes between the hyper-media, passive multimedia and active multimedia presentations. To summarise, Hyper-media implies store-and-forward techniques where user actions such as mouse-selections on hot-spots, cause the system to retrieve a new page of data which could be an image, text, video etc. Passive multimedia implies a fully synchronised document that "plays itself back" in time, synchronising all media objects together. Active multi-media implies that there are hyper-media-type choices presented to users during the playback of a multimedia document which allow the user's interaction to "drive" the playback. Hirzalla et.al [1] suggest a new type of media object called *choice* which has an associated data structure with it. The fields associated with the object are : *user_action*, *region* and *destination_scenario_pointer* using which good amount of interaction could be achieved. The choice media objects can be placed directly on the traditional time-line model there by giving choice points during presentation.

The paper also talks about passive multimedia and active multimedia. The author

concentrates on active multimedia presentations, with asynchronous user events. Subsequently the paper discusses the enhanced temporal model where media on a time-line is split into three units representing the start, body and the end of the media. To incorporate asynchronous events the author suggests a *time-line tree* to define the alternate paths of a presentation. An example presentation of a car is demonstrated and is also later compared with other work.

The author clearly demonstrates the complexities of the previously reported [6] petri-net model for large documents. The same holds for the Buchanan et.al [5] firefly model and the representation becomes more difficult for active documents with the use of the firefly model.

Evaluation : This approach [1] discusses new ways of using the traditional time-line model to create active multimedia presentation. This research has two parts, one which talks about the time-line tree and the other about the enhanced time-line model. Though the enhanced model is an improvement to the time-line model, it does not support fully active multimedia presentation. Also, the paper does not address the spatial aspects of a presentation. The aspects discussed in the paper are mostly towards events generated by time-dependent resources. Events generated by resources during the course of a presentation are not discussed. These are unlike our work, but we use some of the terminologies from this paper.

2.2 Constraints for the Web

Borning, Lin, and Marriott [2] suggests an architecture in which both the author and the end-user (viewer) can impose page layout constraints. The final appearance of the document is thus in effect the result of negotiation between the author and the viewer. They discuss two types of negotiation models. In one model, both the web-authors and the web viewer engage in solving run-time constraints. The author uses a solver and the viewer uses a different solver to solve the constraints. In this

case a compact representation of the constraints, the contents of the page, additional layout information and applets, are shipped over the network for each page. In the second model, the web author again uses a powerful runtime constraint solver, but the viewers role in constraint solving is restricted. The authoring tool compiles a Java program representing a plan that satisfies the author's constraints and the predetermined kinds of constraints that the viewer may impose.

The paper discusses constraint-based page layout to specify the core aspects of the design layout. As the authors mention, the constraints capture the "semantics" of the design and those aspects that must hold true for the layout to be aesthetically appealing. An example constraint-based layout discussed is superior in performance but is dependent upon the competence of the user. In other words the constraint is opaque and is not well suited for both expert authors and non-expert authors.

The model as suggested by the author has three main components: *the document authoring model, the viewing tool and the constraint solver.* The grammar based approach suggested by the author is very powerful in expressing the various layouts. The paper also addresses constraint-based applets about which not much detail is furnished. The authoring tool is used by the designer to construct the constraint style sheets and document contents. The viewing tool integrates constraints from the designer with those of the viewer, to place the contents in the layout. The constraint solver is used by the author while laying out and testing the page, viewers use the constraint system while viewing the page and interacting with constraint-based applets.

Evaluation : The constraints discussed in this paper are intended for resizing the browser or a frame, moving objects around in the window, and interacting with applets. The paper proposes an elegant user interface as its future work. Though the model is powerful in expressing various spatial layouts with the help of the document

authoring tool, the grammar focuses on finer details of layout, hence manual editing of the generated code is too complex. The tool generates approximately 27 constraints for an example layout presented in the paper. The same example could be expressed in 5 constraints using our model. Temporal aspects of a presentation are not addressed in this paper.

2.3 Grammar-based articulation for multimedia document design

The research described by Weitzman, and Wittenburg [3] focuses on the media realization phase and describes a formalism called "Relational Grammars", for encoding design knowledge along with a methodology - parsing, syntax-directed translation, and constraint resolution - as a realization procedure that may encode the same content differently under different circumstances. The paper addresses graphic constraints (e.g., font specification), spatial constraints (e.g, relative positioning), and temporal constraints (e.g, sequence of presentation).

The input to the parser is a set of content objects, as well as the domain-independent relations that hold between them. A rule language used has forms for specifying the primary elements of a rule and their types, the rule language has three kinds of forms: *relational constraints, attribute assignments, and OUT forms.* The relational constraint such as *author-of element2 element1* is a requirement that the object matching rule element2 must stand in the *author-of* relation to the object matching rule element1. Attribute assignments associate attributes of the left-hand side of the rule with those of the right-hand side. The OUT forms serve the role of style specification in the articulation process. Abstract specifications like fonts are used here.

A variety of approaches and methods to model multimedia presentations have been

proposed in the literature. The interactive and spatial capability of these models are often lacking or unsatisfactory. Weitzman et.al [3] suggests a parsing and syntax-directed translation that uses relational grammars. Although their work deals with the spatial aspects of a multimedia presentation, little support is provided to the temporal aspects of a presentation. Our ideas are similar to their ideas in the spatial aspects of the presentation. This paper specifies spatial aspects of the presentation based on the automatic layout generation from the specified relational constraints between the objects. The model also provides ways in which authors can specify fonts and other properties in the layout. The browser is capable of supporting different fonts, colors and other properties. In our model HTML files are treated as resources that can be grouped to make pages. We use the existing features of HTML for supporting different fonts, colors etc of the presentation. As the objects in our model could be HTML documents, support of fonts and other presentation items are encapsulated within the HTML domain. Unlike our work, this paper does not address active temporal aspects of a presentation. On the contrary it suggests a simple extension of the spatial constraints to support temporal aspect. The user interacts with a bar on top of the page in order to control the presentation. The temporal aspects of a presentation are given as an extension to the :OUT forms.

Evaluation : An interesting feature in this work is the relational-constraint between objects. Though this feature was thought about in our initial phase of research, our implementation issues forced us to deviate from this. We propose this in our future work. We use a completely different event handling mechanism than in Weitzman et.al [3]. Our approach can handle and support a rich composition of events. We address events in a broader spectrum with User events, Intra-object events, Inter -object events, application events and custom events. In contrast, in this paper Weitzman et.al [3] concentrates only on one problem of a presentation, and does not address all the issue of a presentation. In our model we suggest a Concept Graph Model (CGM) approach for assembling the pieces of a presentation in order to make

a full presentation.

2.4 Structured Multimedia Authoring

Hardman, Rossum, and Bulterman [7] discusses a rich hyper-media document model allowing structure-based composition of a multimedia presentation. The authoring environment therein presents three main views namely, *hierarchical view, channel view and the player view.* The hierarchical view allows the author to define the structural relations among media items making up the presentation. The channel view is used to add synchronisation constraints between any two data nodes. The player view maps the logical document to a particular presentation environment, and controls the playing of a presentation making use of the other two views. The author clearly pinpoints the drawbacks of a time-line based model [10]. The paper talks about making the implicit structure of multimedia documents explicit which is part of our goal also. As mentioned by the author, an authoring environment should provide means for creating an empty structure which can later be filled. For example, a sequence structure is created and then filled in with individual data nodes comprising dynamic or static data, or from collections of nodes. Instead of this top down approach, the author may also work bottom up, creating small clips and combining them into complex scenes.

An authoring environment for multimedia according to Hardman et.al [7] needs to provide support for viewing and manipulating presentation via their structure, supporting both top down and bottom up construction. Constraints between media items should be defined directly between those items and not via a time-line or separate script. In the model suggested in this paper, a multimedia presentation has a hierarchical structure whose leaf nodes are the data nodes which are played in the presentation, and whose non-leaf nodes are composite nodes containing a collection of other composite nodes and/or data nodes. The author explains the creation of a hierarchical view, channel view and player with the help of an example.

Evaluation : The CMIF model presented in Hardman et.al [7] suggested a novel approach of hierarchical view, channel view and player view. It offers traditional time-line type of visualisation called the channel view. The CMIF model does not address the active temporal events (user interactions) of a presentation, but talks about passive events like start and end. This is in contrast to our work.

2.5 Presentation Support for Distributed Multimedia Applications

Bates [11] discusses a variety of presentation aspects like presentational, compositional and colloborational. The thesis describes a scripting language with object oriented design for handling events, and supports a variety of media items and events. The focus is on collaborative and distributed work.

The thesis work explains how event monitoring is achieved using firing invocation and call-back module. There is an *event monitoring module, firing module, invocation module and call-back module.* The event monitoring module gets the events and passes it on to the firing module which executes the proper invocation and also handles proper call-backs. In the event functions discussed in the thesis work, the author talks about *Event registration, Event monitoring, and Event notification.* Event registration is used to inform an object of events to monitor for, Event monitoring is the algorithm used to detect the events, and event notification is to inform interested clients when the events occur. This is very similar to the process of generation, evaluation, and consumption of events discussed in our model.

Evaluation : Our presentation model for event handling resembles this work at the architecture level, but we differ in our implementation and the choice of presentation medium (WWW). The author, Bates [11], proposes a full fledged scripting language which is powerful but complex to learn.

2.6 Events in Interactive Multimedia Applications

Vazirgiannis, and Boll [12] suggests a similar approach based on events. That paper suggests an event processing scheme for the execution of Interactive Multimedia Applications (IMAP). The model suggests an "event evaluation and detection" module, that handles events delivered by the event generation module. For each event, the *event evaluation module* checks for defined events and executes the action part of it, if defined, and consumes it if not defined.

This approach introduces the notion of events in the specific context of Interactive Multimedia Application (IMAP) as a means to represent the happenings that are of interest to an IMAP. The author does an elaborate classification and modelling of events : events caused by the interaction of a user with the IMAP, Intra-object events, Inter-Object events, Application events, and User-defined events. The author proposes an Object-oriented modelling of events. Apart from the above mentioned classification of events, the author suggests the classifications of events into two layers namely, the generic and application-specific events. Generic events are the template events for the IMAP definition. The application-specific events are specialisation of these generic events. These application-specific events are defined on the basis of objects belonging to a specific IMAP. Application-specific events are defined during authoring time by the application designer.

Events are not analysed with respect to their semantic aspects or their consumption aspects, but are represented by a set of tuples so called *scenario_tuples*. Each tuple represents a fundamental or autonomous functionality in the framework of an IMAP and includes events that will trigger this functionality. The list of actions that will be executed when this tuple is triggered are also provided as part of the scenario-tuple. The event processing scheme for the execution of an IMAP presents the generation, evaluation, and consumption of events. Even before an IMAP is executed the events

of the scenario tuples are announced to the presentation system. The event evaluation and detection module evaluates the events delivered by the event generation module and detects occurrences of IMAP application-specific events. For each events that is received, the event evaluation checks if the event matches with any event expression in the scenario tuples or if it was a part of a composite event. If one or more event expression matches, the corresponding action executions are triggered.

Evaluation : Our approach to the temporal aspects of presentation mostly resemble with that of theirs. On the contrary we propose a grammar based approach for event definition and a client-server approach for event handling. Though very complex events are discussed in this paper, we limit ourself to a simple class of events :user events Vazirgiannis et.al [12], inter-object events, intra-object events and custom events supported by the browser. We view the objects in a presentation as clients and the event evaluation and detection module as the manager or server. Events are dispatched to the server, which analyses, takes the necessary action and sends an appropriate message to the corresponding client. A simple event definition grammar is proposed in our work and its trade-offs are discussed.

2.7 Specifying Temporal Behaviour in Hyper-media Documents

Buchanan, and Zellweger [5] addresses three main problems in multimedia authoring tools: the difficulties in representing asynchronous behaviour, difficulties in creating documents as per the author's needs, and the difficulties in maintaining documents over their lifetimes. The paper Buchanan et.al [5] proposes a system called the Firefly, whose goals are to support rich synchronisation as well as asynchronous behaviour.

The Firefly document model of Buchanan et.al [5] consists of three parts: media items, temporal synchronisation constraints, and operation lists. The media type

specifies the specific kind of medium the media item represents. Events, as defined by Buchanan et.al [5], represent time-points at which the display of the media item can be synchronised with other media items. Events are classified into synchronous and asynchronous events. Synchronous events are those whose temporal placement is known in advance, and asynchronous events are those whose time of occurrence cannot be determined until the media item is displayed. It also defines a term called *procedures* that operate on the media item and its events. The procedures allow the underlying media items to participate in the firefly system. The procedures are classified into *user-interface procedures, analysis procedures, and control procedures.* User interface procedures support the creation and editing of media items and events. Analysis procedures provide information such as the duration between two events or the ordering of events. Control procedures affect the display behaviour of a media item. All media items provide control procedures to start, end, pause, and resume the display of a media item. Authors interact with existing media editors to create and edit media items, and to mark points of interest, called events, within them. In firefly's graph notation, square nodes represent the start and end-events, and circular nodes represent the internal events. Edges represent the temporal adjacency of two events; the length of the edge is proportional to the duration between the events. Asynchronous media items float above the start node.

The firefly's runtime architecture consists of three components: *the viewtime system, the event handler, and the media items* in the document. The viewtime system initiates the display of a document when it receives a schedule from the scheduler. The view-time system registers the documents asynchronous events with the event handler and starts the document clock. When the view-time system encounters a *Send-Message* operation, it invokes the senders *Generate-Message* control procedure to determine what message should be sent. Messages may be sent to specific media items or broadcast to all active media items. When media items detect the occurrence of an asynchronous event, it notifies the *EventHandler*, which maintains a list

of active and inactive asynchronous events. If the event is inactive, it is ignored, or else the Event-Handler notifies the View-time system, which merges the auxiliary schedule for that event into the schedule.

Evaluation : The firefly model is a novel approach in multimedia synchronisation. The model gets too complicated for large systems with lots of asynchronous events, which is clearly demonstrated with an example in Bates [11], where an example presentation is taken and the firefly model is compared with his Ph.D thesis work.

2.8 Web-CT: An environment for Building WWW-Based Courses

Goldberg, Salari, and Swoboda [15] describes Web-CT as an easy-to-use environment for creating WWW-based courses that are otherwise beyond the ability of the non computer programmers. Web-CT allows the course authors to create a course and then to add a wide variety of tools and features to the authored course. Some example tools are, bulletin-boards, student self-evaluation, navigation tools, timed quizzes, electronic mail, and automatic index generation. As mentioned by the author, the advantages to students in using the Internet and the World Wide Web to make course material includes, location and time independent delivery of the course material, ability to serve a large number of students at a potentially reduced cost, and a simple and familiar interface. The author says that the proliferation of WWW-based courses is high in departments where there is high degree of technical familiarity. In contrast, other departments that are less technologically focused are either not taking advantage of the WWW or are not exploiting its full power. The other departments are equally enthusiastic, but are limited without the technical expertise to use CGI's to create static web pages (no interactivity), or are forced to hire a consultant to do the work for them.

The Web-CT presents an environment that allows educators, with or without technical expertise, to create sophisticated WWW-based courses. WEB-CT uses the Web as its GUI for both building and presenting WWW-based courses. The advantage is that these servers will be maintained centrally in an organisation and will be used by members of that organisation. WEB-CT addresses three aspects:

- Presentation tool - that allows course designers to determine the layout, colors, text, counters etc for the course page.
- Student tools - tools that can be integrated into any course.
- Administrative tools - that aid delivery of a course.

Evaluation : Web-CT focuses mostly on the support tools that can aid authors to present a variety of material on the Internet. The author describes a presentation tool that lets authors build a presentation. Though this supports some features of layout, it supports proto-typed layouts and does not give much options to the author. The work clearly does not concentrate on HTML features and leaves it to external authoring tools. The tool concentrates on other aspects of a presentation like, quizzes, chat rooms, self-evaluation module etc which are not the focus of our work.

2.9 Microsoft Scriplets

Microsoft is also in its early alpha release for a set of object libraries called *"scriplet"* Microsoft [18] to allow DHTML pages to be created as self-contained objects. The scriplet provides effective communication between the objects in a page, which are treated as controls. The scriplet also deals with standard events and custom events handled by the event handler. It provides methods like raiseEvent(..) that lets authors to raise a custom event, which could be handled in a customised way. The above software product is in its early alpha and is supposed to go through revisions before made ready for public use.

Scriplets as mentioned in [21] further enrich the Dynamic HTML object model, and makes it interesting not only for web developers, but also to Visual Basic and Visual C++ developers. Scriplets are simply HTML pages that contain client-side script, which conforms to a standard scripting architecture. This architecture, similar to Visual C++ and Visual Basic, defines what routines and variables are public (that is exposed to the control's container on the page) and private (hidden from the control's container). When a scriplet is loaded into memory, it is loaded into an instance of the HTML parsing engine that is itself wrapped inside a COM container. This container is placed inside the HTML parsing engine of the original HTML page that contains the reference to the scriplet.

As mentioned in the reference [21] at present there are no authoring tools that fully support Scriplets, though they are to be expected in future. Since Scriplets are more like a full fledged Dynamic Object Library, our grammar based system could use scriplets and become richer with the underlying support of Scriplet library.

Evaluation : One major disadvantage with that of scriplets Microsoft [18] is that it is supported only on Internet Explorer 4.0. Scriplets do not aim at making authoring easy and if it is released as a stable base, it could be used to improve the functionality of our authoring environment by using the libraries provided by scriplets and thus providing better service to authors.

2.10 Dreamweavers dynamic HTML without scripting

Dreamweaver is a visual Web authoring tool meant for professionals and it allows for creating style sheets, absolute positioning, time-line animation, and Dynamic HTML without coding or scripting. It provides full control of HTML code with real-time

display, easy drag-and-drop tables and frames, browser targeting/error reporting, extensible behaviours, and server file-locking; automatically generates appropriate HTML code and scripts for both Netscape and Internet Explorer. Dreamweaver is a professional authoring tool for Web page creation, design and management.

Dreamweavers integrated Dynamic HTML, Java Script and Cascading Style Sheets (CSS) features are expected to boost the productivity of web developers. The reference [20] points to six reasons as to why Dreamweaver is bound to revolutionise the web development. Firstly, the author talks about "Round-trip HTML". By round-trip HTML the author refers to opaque and transparent interface, meaning that Dreamweaver imports and exports the HTML without modifying it and gives full control over the underlying code. Dreamweaver provides a drag and drop for creating the pages. Dreamweaver lets author create many great effects using advanced HTML and Java Script without any programming needed.

Evaluation : Dreamweaver [20] suggests a time line based approach for dynamic presentation. The product is still in its early alpha release. Our approach is similar to that of Dreamweavers in functionality. The design of dreamweaver's event handling is not known and kept as a trade secret.

2.11 Comparison table

Related research papers	Spatial and temporal planning	Event architecture	Grammar approach	User interface classification	Illustration
Temporal model for interactive multimedia	Passive temporal	Extended time-line	Model based	Opaque	Prototyped
Constraints for the Web	Spatial	Nil	Grammar based	Opaque	Prototyped
Grammar-based articulation for multimedia documents	Spatial & Passive temporal	Nil	Grammar	Opaque	Prototyped
Structured multimedia Authoring	Spatial & Passive temporal	Time-line	Model based	Opaque	Real-life
Presentation support for Distributed multimedia Application	Active & passive temporal	Good	Scripting language	N/A	N/A
Events in Interactive multimedia application: Modelling and Implementation Design	Active & passive temporal	Client Server Model	N/A	N/A	N/A
Specifying Temporal Behaviour in Hyper-media documents	Spatial & passive temporal	Firefly	Firefly model	Opaque	Real-life
Web-CT	Active & passive	N/A	N/A	Web-based	Real-life
Microsoft Scriplets	Active & Passive	OO Library	Scriplet Language	N/A	N/A
Dreamweaver dynamic HTML without scripting	passive temporal	Time-line	N/A	Transparent	Prototyped

Chapter 3

Authoring Model

A web page is a collection of multimedia resources positioned in a two dimensional space, according to the needs of the author. The tree diagram for a multimedia presentation is as shown in Figure 5. These resources have to be interacting within themselves and responding to user inputs in an active multimedia presentation. In this chapter we formally describe the page layer and the document layer which facilitate authors in planning, building, and realising a page and a document in a multimedia presentation. The page layer assists the author in spatial and temporal planning and realisation of a page. Subsequently, the document layer aids authors in planning, building, and realising the document, which is an organised collection of pages.

We begin with a description of the *layout model* in Section 3.1 and demonstrate a content independent layout of resources in a page. In Section 3.2 we present the *event model* of the page layer to achieve an active multimedia presentation. In Section 3.3 we describe the *document model* which is used to present a document in an orderly manner to accomplish an organised presentation.

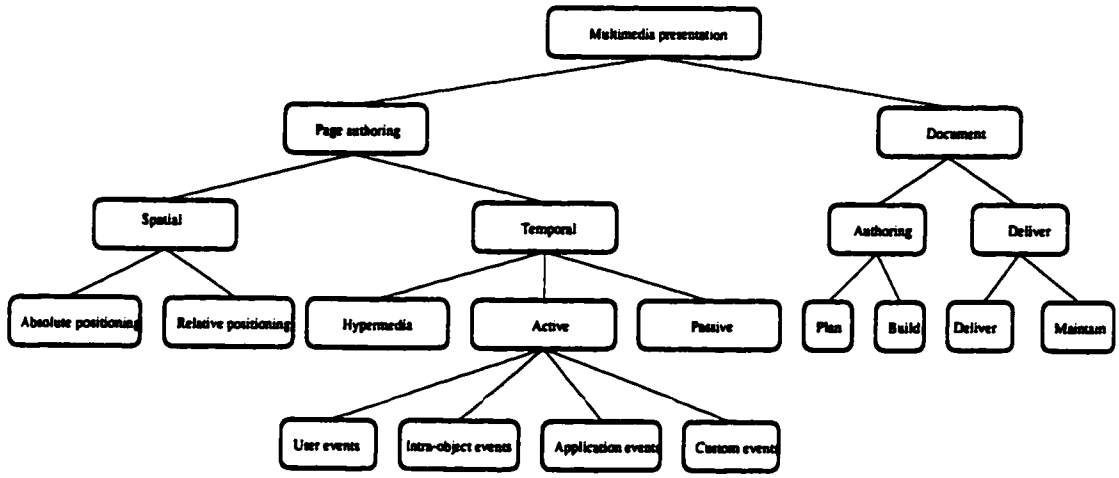

Figure 5: Presentation tree

3.1 Layout model

Authors should be provided enough support to achieve the following.

- to allow mental planning as easy as in using "pen and paper"
- to create resources using the software tools he/she is familiar with. Such tools may include, Microsoft Word, Microsoft Power Point, Paint Brush, Imaging tools, Audio tools etc.
- to edit and modify individual objects, but maintain the layout
- to organise and synchronise resources into a page
- to perform these tasks as easily as the author does in conventional medium

Resources in this context refers to primitive multimedia resources like images, audio, HTML, etc. Authors should be provided support to align various multimedia resources as per their needs and requirements to build a page. The contents of a page can be aligned using two ways of positioning, viz. ***Absolute positioning*** and ***Relative positioning.*** Absolute positioning uses hard-coded co-ordinate values to realise the layout. Though this is an effective way of positioning resources, it has its disadvantages. The layout generated using absolute positioning is specific to the contents of

a page and thus dynamic changes to the contents of the page disturb the layout considerably. Most Window based applications that support drag and drop of resources for layout, generate absolute positioning.

In the other form of layout using relative positioning, resources are aligned relative to the position of other resources. Though this poses some constraints in the layout of resources, it is best suited in cases where the contents of the pages change frequently and also dynamically. We choose relative positioning of resources in a page as the contents of our pages change dynamically. The choice of relative positioning along with the dynamic capabilities of dynamic HTML allow us to extract the core layout features and maintain it irrespective of the content of the page.

For example, consider the layout where five resources have to be placed in a web

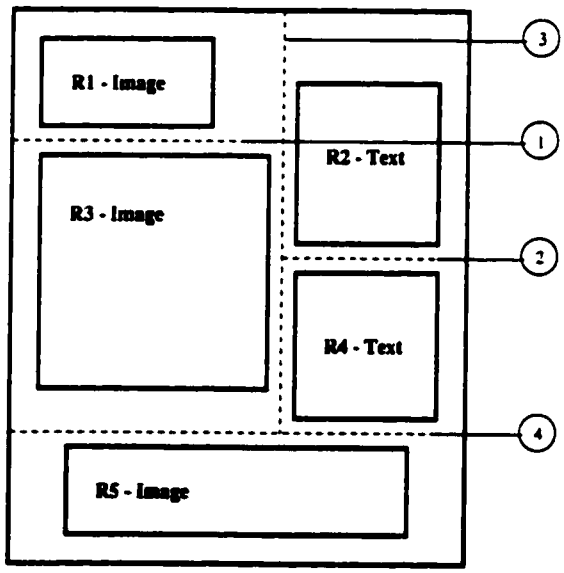

Figure 6: Layout segments of a page

page as shown in Figure 6. Though this layout can be achieved using a WYSWYG authoring tool, it generates absolute positioning to place the resources accordingly. This approach is highly Dis advantageous in a situation where the contents and sizes of the resources keep changing frequently affecting the position of other resources. In

Figure 6 position of resource R2 is dependent on the size and position of resource R1. Dynamic changes to R1 should reflect on the subsequent layout of R2 and other dependent resources. Hence, we suggest authoring of relative positioning of resources on a page using dynamic HTML and thus produce a content and size independent layout of a page.

3.1.1 Layout grammar

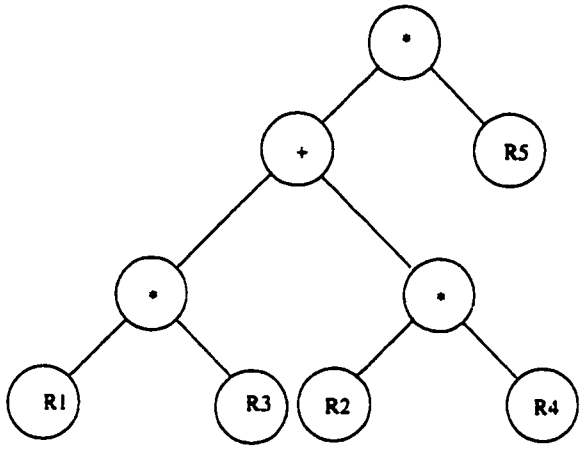

Figure 7: Tree diagram for the layout in Figure8

We develop a grammar based approach to define relative positioning of resource in a web page. The advantage of a layout grammar in contrast to absolute positioning is that, it abstracts the relative positioning in a layout at a higher level where the layout is maintained independent of the resources being layered. Thus, dynamic changes to the resources do not affect the layout of the page. The layout grammar defines the relative positions of the resources with respect to each other. An authoring tool interprets the grammar specified by the author and generates the corresponding dynamic HTML code. The proposed layout grammar is used to specify the physical relation with regard to position between the resources in a web page. The grammar is formed in a tree pattern, where the intermediate nodes represent relation between the resources and leaf nodes represent the resources. Figure 7 describes the tree for the layout shown in Figure 6. We define two types of alignments in our layout

grammar,viz.

- *parallel - one after other in a left to right scan.* Two resources R1 and R2 are said to be in parallel (R1 + R2), if R1 appears to left of R2.
- *serial - one below other in a top to bottom scan.* Tow resources R1 and R2 are said to be in serial (R1 * R2), if R1 appears on top of R2.

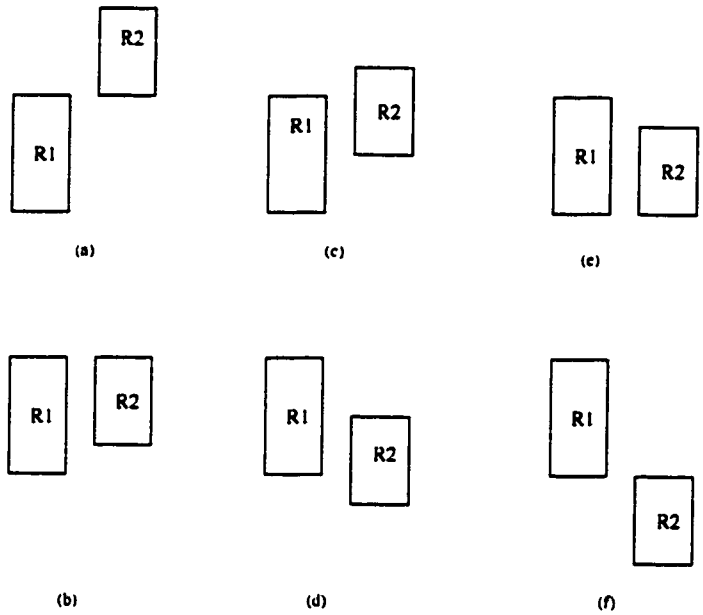

Figure 8: Parallel alignment properties (a)Till Top (b)From Top (c)Till Center (d)From Center (e)Till Bottom (f)From Bottom

Within the *parallel* and *serial* alignment, the resources can have different combinations of alignment properties as shown in Figure 8, 9, 10. There should be some balance between the complexity of the grammar provided and the functionality of the layering. We consider some of the essential features of layout that will be useful for authors in designing their web page. Various such essential alignments, and alignment properties of resources in a page are considered and a simple layout grammar is suggested.

Components in a page are divided into two major types

- **Atomic components** - Defines a single component or resource

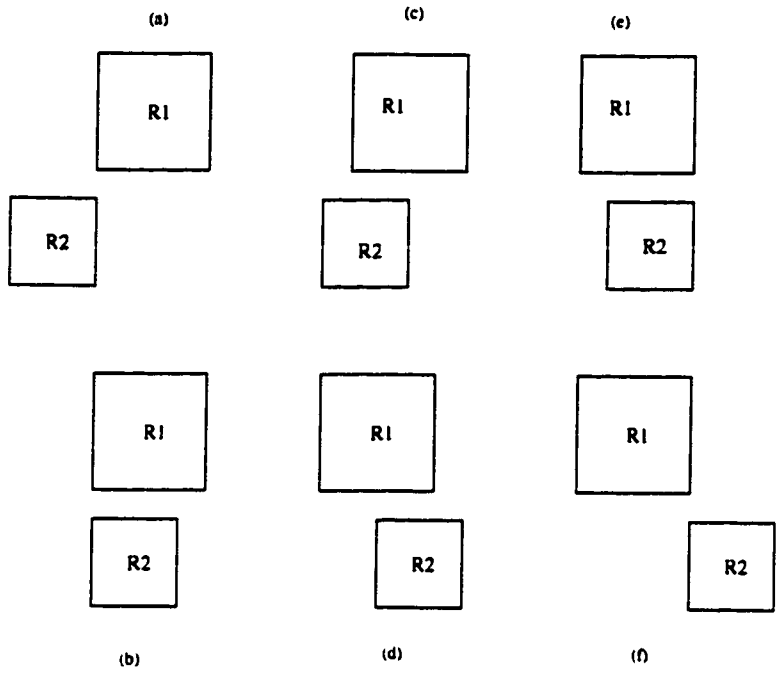

Figure 9: Serial alignment properties (a)Till Left (b)From Left (c)Till Center (d)From Center (e)Till Right (f)From Right

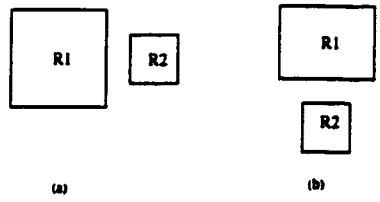

Figure 10: Centered properties (a)Parallel centered (b)Serial centered

- **Composite components** - Defines a collection of atomic components based on a grammar

Atomic components can be visualised as indivisible multimedia resources. When two atomic components are packed, it becomes a composite component. Figure 11 describes the BNF definition of the layout grammar. In Figure 11 *panel name* is the destination panel. *component1* and *component2* can be either atomic or composite components. *Operator* specifies the alignment between the participating components, viz. *parallel or serial.* The *alignment property* parameter specifies the property of alignment with reference to the *reference panel/component.* However, any atomic or composite component can be specified as a reference and the alignment will be done with reference to that panel/component.

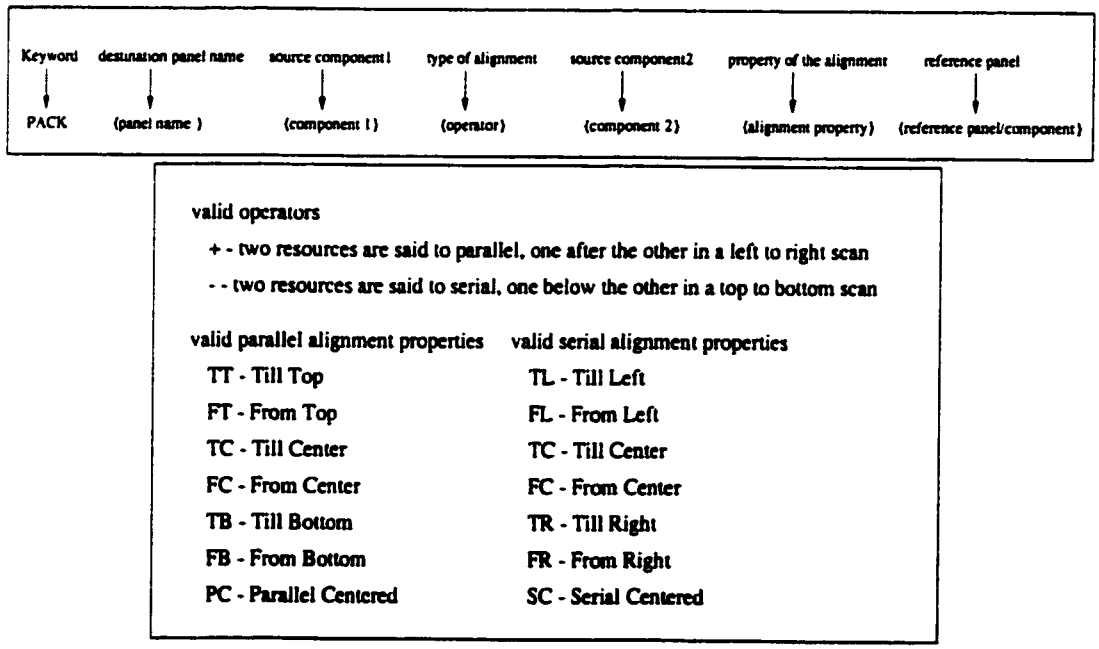

Figure 11: BNF definition of the layout grammar

The BNF definition of the grammar can is as follows.

```
BNF := PACK <panel_definition>
<panel_definition>    := START <name_of_resource> | <panel_name>
<panel_name>          := <name_of_panel> <resource_name1>
```

```
<resource_name1>       := <resource_name3> <type_of_alignment>
                          <resource_name2>
<resource_name2>       := <resource_name3> <alignment_property>
                          <resource_name3>
<resource_name3>       := <name_of_panel> | <name_of_resource> |
                          default
<name_of_resource>     := char[MAX];
<name_of_panel>        := char[MAX]
<type_of_alignment>    := '+' | '*'
<alignmnent_property> := 'TT'|'FT'|'TC'|'FC'|'TB'|'FB'|'PC'|
                          'TL'|'FL'|'TC'|'FC'|'TR'|'FR'|'SC'
```

This grammar is used by authors to align resources in a page. For example, to align two resources R1 and R2 in parallel, with the alignment property of R1 set to top of R2, the author specifies

```
pack PANEL1 R1 + R2 FT R1;
```

In the above syntax, pack is a keyword and PANEL1 is the destination panel name. R1 and R2 are the two source resources, '+' is the type of alignment and FT (FromTop) is the alignment property. The sixth entry R1 is the reference resource for the alignment. This produces a result, where PANEL1 is the encompassing object comprising the two resources R1 and R2. Future references to R1 and R2 as a whole can be made by means of the name PANEL1.

3.1.2 An example layout

Let us consider a complex layout as shown in Figure 6. There are five resources R1, R2, R3, R4, and R5. The corresponding spatial and temporal specification as per the grammar for the layout will be defined as follows.

```
Pack START R1
```

```
Pack Panel1 R1 * R3 FL default
Pack Panel2 Panel1 + R2 FC R1
Pack Panel3 R2 * R4 FL default
Pack Panel4 Panel1 + Panel3 FC  R1
Pack Panel5 Panel4 * R5 PC default
```

The above mentioned specification as per the grammar generates the output as in Figure 6 where FL means *FromLeft*, FC means *FromCenter*, FT means *From Top* and PC means *PageCenter* (Figure 11). The first statement defines that R1 is at (0,0) the left top of the screen. The second statement defines the relative position of R3. This defines the relative position of R3 to be in serial with R1 and sets the aligned property of R3 to the left of R1. The composite resource thus formed by combining both R1 and R3 is named Panel1. Now, R2 and R4 are aligned in serial with the alignment property of R4 set to the left of R2. These two atomic resources are combined into a composite resource and named Panel3. As the syntax for Panel3 does not define the x,y position of either Panel3, R2 or R4, we have to explicitly link Panel1 and Panel3 to identify the x,y position of R2 or R4, due to implementation constraints. We use the second statement to establish this relation. In this we say that R2 is parallel with Panel1, to fix the position of the first resource in a composite resource. Then Panel1 and Panel3 are aligned in parallel with alignment property of Panel3 set to the center of R1. The composite resource, Panel4 comprising of two composite resources Panel1 and Panel3 is formed. R5 is aligned in serial with this composite resource Panel4.

The same layout if specified as follows will generate a different result. For example,

```
Pack START R1
Pack Panel1 R1 + R2 FC default
Pack Panel2 Panel1 * R3 FL default
Pack Panel3 R3 + R4 FT default
Pack Panel4 Panel1 * Panel3 FL R1
```

```
Pack Panel5 Panel4 * R5 PC default
```

generates the output as in Figure 12

In the previous example, resources R1 and R3 were combined in serial to form a

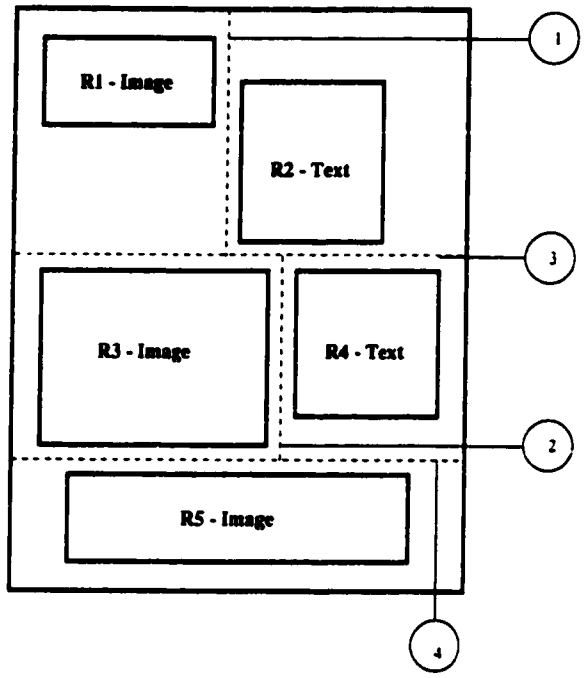

Figure 12: Changed layout with new specification

panel called Panel1. Subsequently, resources R2 and R4 were packed in serial to obtain Panel3. Then Panel1 and Panel3 were put together in parallel to generate Panel4. The layout thus generated is shown in Figure 6. To demonstrate the flexibility in positioning various resources, the layout is changed as shown above. Now, resources R1 and R2 are combined in parallel to form Panel1. Resources R3 and R4 are combined in parallel to form Panel3. Now, Panel1 and Panel3 are combined in serial generating Panel4. The new layout thus generated is as shown in Figure 12.

3.2 Event model

Resources in a web page are independent of each other. An event from one resource cannot have an effect on another resource using static HTML. Though dynamic HTML allows interaction between resources in a page, such interaction is achieved with the help of Java Script and Java code. An event model is proposed to overcome this static nature of web pages and to assist authors who may not be programming experts to make more interactive web pages.

We propose an event model using which authors can easily define, build and deliver active multimedia presentations on the web. We aim at supporting active multimedia presentations, with support to a variety of events. We also address the problem of synchronising resources during the course of a multimedia presentation. Though Hardman et.al [7] discuss about synchronisation of resources using multiple views, they support synchronisation of time-independent resources with time dependent resources. Most of the interaction models focus on the traditional time-line model [1] for temporal aspects of a presentation. The primary disadvantage with the time line model [1] is its inability to support active temporal aspects of presentation. There is a necessity for an authoring tool to enable authors easily specify the various interactions between the resources in a page and thus generate interactive web pages.

3.2.1 Interaction definition - A grammar based approach

Resources in a page have to be able to send as well as receive events. Events occur as a result of user reactions. For example, clicking a button is an event as is moving the mouse over a link. There are two types of events, viz. *system generated events* and *custom events.* System generated events are click, focus, keydown, keyup, load, unload etc. Similarly, custom events are ChangeResource, PrevResource, ChangeColor, Show, Hide etc. Appropriate event handlers should be defined to react to events. In the existing structure, authors have to handle events themselves and also define

action for the corresponding events.

We propose a grammar based approach for event specification. The architecture of such a design is as shown in Figure 13. Each resource on the page will have a client object associated with it. Every page loads a manager object during load time of a page. The manager object has the event table describing the interaction between the client objects. The page realiser module displays the page and waits for event to occur. On occurrence of events, the event handler handles the event and dispatches it to the manager object. The manager object maps the event to its event table and triggers the corresponding action. Each client object will report events and make use of the manager methods which will eventually dispatch the events to the corresponding destination client objects, thus impacting the resource controlled by it.

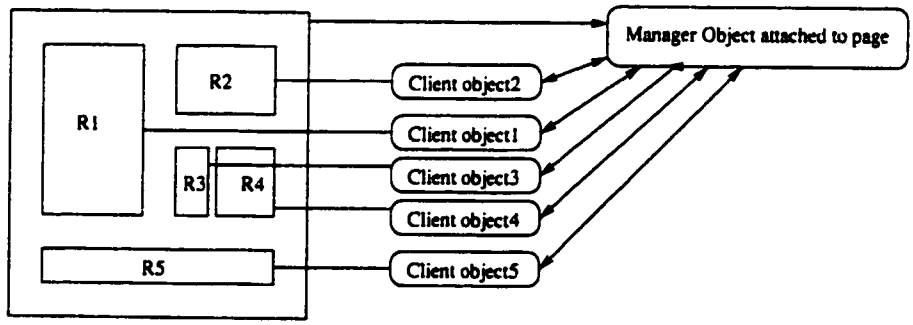

Figure 13: Client-Server structure of interaction

3.2.2 An example interaction

To demonstrate the interaction between resources in a page, let us consider two button objects, a previous button and next button and an image object in a web page. Let us assume that for every click of the button object the contents of the image object has to change accordingly. This can be achieved using Java Script and dynamic HTML concepts. On the contrary, for non-computer expert authors, the above task

is difficult as it requires deep programming skills in Java Script, understanding of dynamic HTML etc.

The Java Script code required to realise the above mentioned presentation would be as mentioned below.

```
<LAYER ID=Resource2 top=0 left=0>
<A HREF="#" onclick=Prev("Resource2")>
<CENTER> <IMG SRC=text/prev.gif BORDER=0> </CENTER> </A>
</LAYER>
<LAYER ID=Resource3 >
<A HREF="#" onclick=Next("Resource3")>
<IMG SRC=text/next.gif BORDER=0> </A>
<BR><BR> </LAYER>
<LAYER ID=Resource1 src="image1.jpg" width=600 HEIGHT=400>
<BR><BR> </LAYER>
<SCRIPT>
var list = new list();
list.Next = Next;
var i=0;
list.HANDLE=new Array();
for(i=0;i<100;i++)
list.HANDLE[i]=null;
list.HANDLE["Resource1"]=new Array("image1.jpg","image2.jpg",
"image3.jpg","image4.jpg","image5.jpg");
list.HANDLE["Resource1count"] = 0;
function Prev(n) {
if(list.HANDLE[n] != null) {
 list.HANDLE[n+"count"]--;
 if(list.HANDLE[n+"count"] < 0)
```

```
  list.HANDLE[n+"count"]=list.HANDLE[n].length-1;
 document.layers[n].load(list.HANDLE[n][list.HANDLE[n+"count"]],
600); }
else
 alert("Error: Resource not defined "+n);
}
function Next(n) {
if(list.HANDLE[n] != null) {
 list.HANDLE[n+"count"]++;
 if(list.HANDLE[n+"count"] > list.HANDLE[n].length-1)
  list.HANDLE[n+"count"]=0;
 document.layers[n].load(list.HANDLE[n][list.HANDLE[n+"count"]],
600); }
else
 alert("Error: Resource not defined "+n);
}
</SCRIPT>
```

The three layers define the three resources, viz. the previous button image, the next button image, and the series of five images that are to be displayed one after the other. These five images are stored in an array. Every click on the next or previous button, makes a call to the corresponding function which displays the next or previous image from the list accordingly. Though this code is a brief version of the real Java Script code, it is complicated.

To alleviate the author from the above task, we propose a grammar based approach for easy definitions of interactions between the resources in a page. The specifications as per the grammar shown below would automatically generate the Java Script code shown above in order to achieve an interactive presentation.

```
% Resource name and the resource file and its properties
```

```
R1 <type of resource> <options> "previous.jpg" width=200
   height=200
R2 <type of resource> <options> "next.jpg" width=default
   height=default
R3 <type of resource> <options> "image1.jpg&image2.jpg&
   image3.jpg&image4.jpg&image5.jpg" width=300 height=300
% Event definition
% Source resource - event - destination resource - action
R1 Click R3 PrevResource
R2 Click R3 NextResource
```

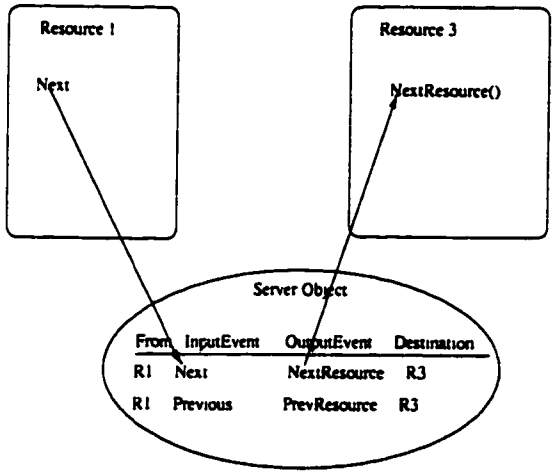

Figure 14: Example Client-Server interaction

The interaction between the resources are specified along with the input in the specification file as per the grammar. The source resource generating the event is specified first, and the event to generate in the destination resource is also specified. Once this specification file is generated, corresponding event mechanism is generated by the authoring tool to generate the events and also to capture the event at the destination with the help of the client-server mechanism. A server applet loads itself during load time of the page and updates itself of the various resources and the interactions between them using the specifications as per the grammar for interaction. A table for the interaction mechanism is created and the server waits for events to

happen. On occurrence of an event from a client the server consults its table and triggers the destination event or action on the destination client. Figure 14 shows an example where client object R1 fires an event Next to the server, which maps it in its table and triggers the NextResource event on the resource R3. Since, the two way communication between Java Script and Java is possible the above effect could be easily realised by an authoring tool relieving the author of hazards of programming and concentrate in his area of expertise.

3.2.3 Interaction bottlenecks

The availability of resources and the speed with which they can be down-loaded is highly unpredictable in the Internet. This raises questions on the reliability of the communication between resources in the event of failure or delay in the Internet. Although the problem is beyond the scope of this thesis, we provide some discussion of the interaction between the resources via the Internet. Resources invoke an event and wait for an acknowledgement from the destination resource before they invoke the appropriate action so that they can synchronise themselves with the other resource. For example, in a slide show an audio applet at the end of first audio segment will trigger a "NextResource" to the slide applet and will wait for an acknowledgement from the slide applet before switching itself to the next audio file. This protocol brings some amount of reliability and synchronisation in the interaction between the resources. In case of a failure, the source resource waits for a defined amount of time (time out), and proceeds after displaying a standard error message on the destination resource.

3.3 Document Model

A presentation is a collection of concepts delivered in an organised manner. Presentation on the web can be classified in to two main parts, viz. presenting individual pages, and presenting a collection of pages called document. Individual pages refer to

Figure 15: Precedence relation

concepts, where each concept has its own temporal and spatial constraints. Authors should be assisted in planning the collection of these pages to make an organised presentation. In this thesis, we propose a model which we call the "Document Model" to realise a document. The document model uses a directed acyclic graph where nodes represent pages and edges represent precedence relation between the pages. If there is a directed edge going from node A to node B (Figure 24) then node A should be covered before node B. An example concept graph is shown in Figure 16. The graph contains eight nodes or concepts and relations between them. There are two ways in which the user can read till node G: the path A,B,C,D,G or A,B,E,F,G. Navigation of a concept graph is well structured and directed.

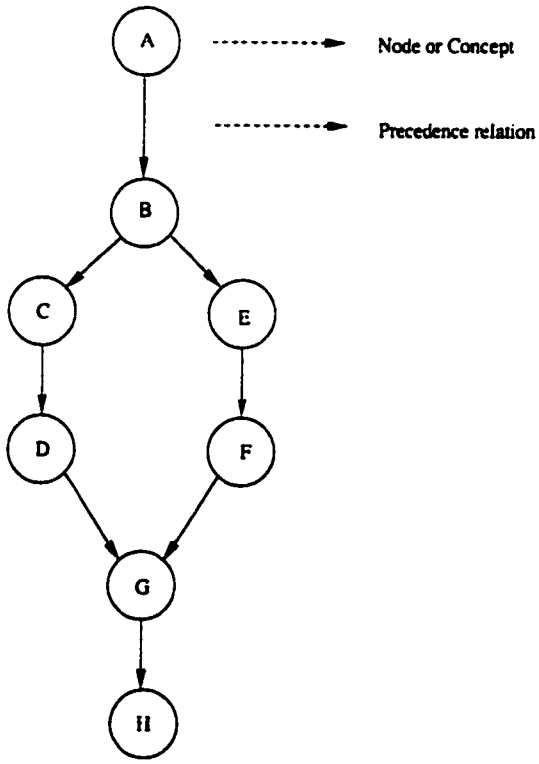

Figure 16: Concept Graph

3.3.1 Planning, building and delivering

The knowledge structure of the author could range from a simple linear structure to a complex graph structure. This knowledge structure of the author should be transformed into an appropriate computational structure by the document tool with minimal intervention from the author. The tool should be able to store the complex structure, build the equivalent document structure, and deliver the presentation as viewed by the author in his mental model. The knowledge structure is a graph structure that contains chunks of information linked by associations. The nodes represent the concepts or chunks of information, and links refer to precedence relation between the linked concepts. The various stages of a presentation as shown in Figure 5 can be mapped as follows,

- Planning - mapping of mental model to its equivalent document model
- Building - Adding semantics to the computer model
- Delivering - Realising the computer model in the form of a document

We consider this process of transforming the mental model of the author to the graph structure as the planning stage of a presentation. Authors are assisted in both approaches, viz. top-down or bottom-up to plan a presentation. Authors could make the concepts and plan the graph structure or else, make the graph structure and then fill the concepts.

Once the mental model of the author has been transformed successfully into its equivalent document model, the document building tools, interprets the grammar from the planning phase and generate an equivalent web code.

Finally, the document delivery tool interprets the code generated by the build stage and presents the document model as a multi-media presentation. Subsequent to this, maintenance of the presentation is made easier using the Concept Graph Model, as

authors can easily drag and drop new pages into an existing presentation, or simply delete existing nodes from a presentation. The building module reacts to the deletion and reflects it on the delivery.

Chapter 4

Implementation Design

In this chapter we present the implementation of the web authoring tool. The authoring part is a stand alone application that aids authors in planning and building a presentation. The authored product is delivered as an interactive multimedia presentation. The browser is the medium over which the presentation is realised.

The requirement is to design a software to support authors irrespective of their domain knowledge to build interactive web pages. The software should be able to take as input the knowledge structure of the author and create the necessary web pages automatically. The author interacts with the software and specifies his requirements for the layout of the resources and the interaction that is desired between the resources also called objects in a web page.

In Chapter 3 we presented the grammar that specifies the layout of resources and the interaction between the resources in a web page. The input to the page authoring part is as follows:

- The resources that are to be displayed in the page
- The spatial and temporal specifications as per the grammar containing the resource definition, layout definition and the event definition of the resources in

the page

The output from the page authoring part has the following,

- Dynamic HTML code required to display the page
- Java Script code to layout the resources as defined in the specification file as per the grammar
- Java Script code and Java applets to handle events

As mentioned in Chapter 3, the next step is to collect these pages in an organised manner to form a document. We developed a graph editor with a drag and drop user interface that can be used to create a document. The input to the document authoring part consists of the following two parts:

1. HTML pages
2. Graph drawn using our graph editor

The output from the document authoring is a file that has a suitable representation for the graph.

1. Representation of the graph as a graph file

This graph file can be interpreted by the document presentation module to display the graph on the browser.

We organise this chapter as follows. Our design of the authoring tool is presented in Section 4.1. We have already described the grammar to author the pages and documents in chapter 3. In this section we describe the process of converting the page specifications into Java Script and dynamic HTML code, and the process of translating a document representation to its presentation Java applet form. As the implemented authoring tool generates Java Script and dynamic HTML code to display a page and a document on the browser, we present an introduction to dynamic

HTML and Java Script in Section 4.2. In Section 4.3 we describe the implementation of the authoring tool. In this section we present the implementation of page-authoring and document-authoring modules. In Section 4.4 we demonstrate the implementation of the page presentation module and the document presentation module.

4.1 Design of the web authoring tool

We present the the architecture, object diagram and activity diagram of our design. We describe the various subsystems in the architecture of our design. Subsequently, we present the object diagram explaining the various objects in our design. Finally, we present our dynamic model with an activity diagram.

4.1.1 Architecture

Figure 17 shows the various subsystems and the interactions between them. In our architecture we identify three main subsystems namely, *Authoring part subsystem*, *Presentation part subsystem*, and the *Presentation medium.* The authoring part subsystem is further subdivided in to the *Graphical User Interface subsystem*, *Page subsystem* and the *Document subsystem.* The Page subsystem aids authors in planning and building pages of a document. It includes a *lexical analyser, syntax analyser, semantic analyser, and code generator.* The document subsystem consists of the *document generator and document representation.* The document generator is the graphical user interface that aids authors in defining the document. The document representation stores the defined document in computer concepts.

The presentation subsystem is made up of two different subsystems, viz. *presentation content* and *Event handler.* The presentation content is the material that is displayed on the presentation medium. The presentation content can be page or a document. To realise the presentation contents the presentation subsystem has an

event handler and Java Script library. The presentation part contains the representation of the page and the document respectively. With the help of custom libraries and the event handler, the presentation part realises the presentation. The browser generates events that are handled by the event handler. The events could be generated by the system or by the user interacting with the browser.

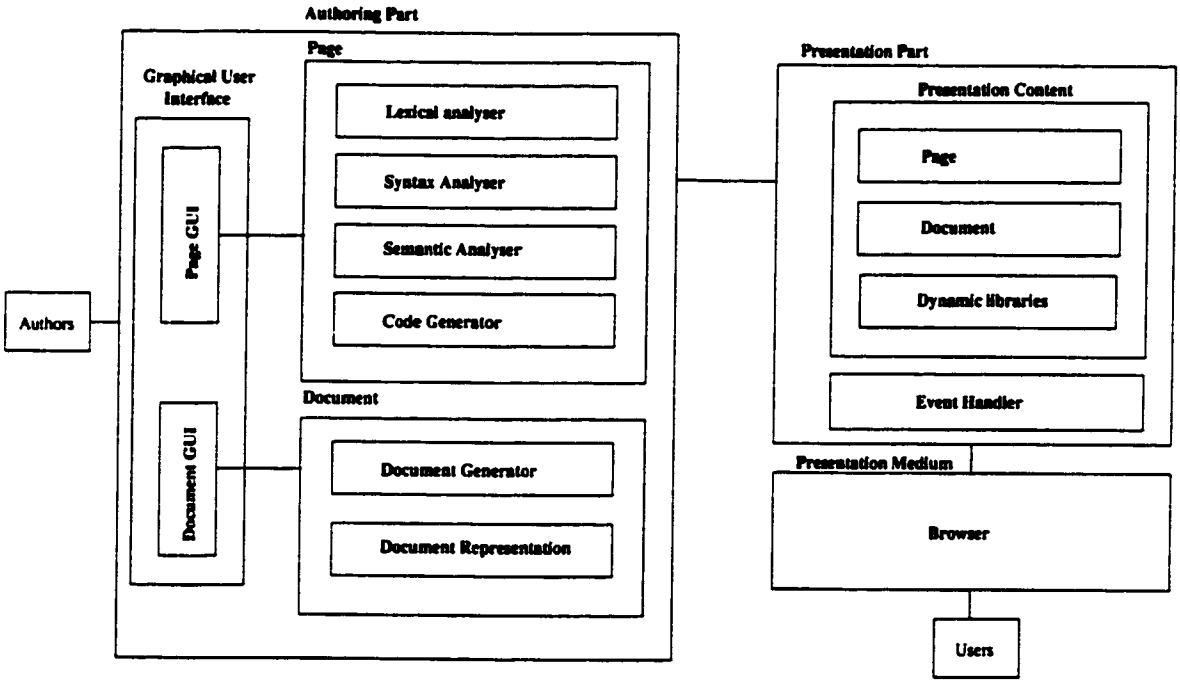

Figure 17: Architecture of authoring tool

4.1.2 Object design

In this section we explain our object diagram. As shown in Figure 18, the graphical user interface generates spatial and temporal specifications as per the grammar, for a page and a document respectively. The document grammar defines a document. The page specifications as per the grammar is used by the page authoring tool to generate the code required to realise a page. The page authoring tool consists of a lexical analyser, syntax analyser, semantic analyser, and a code generator. A document is a

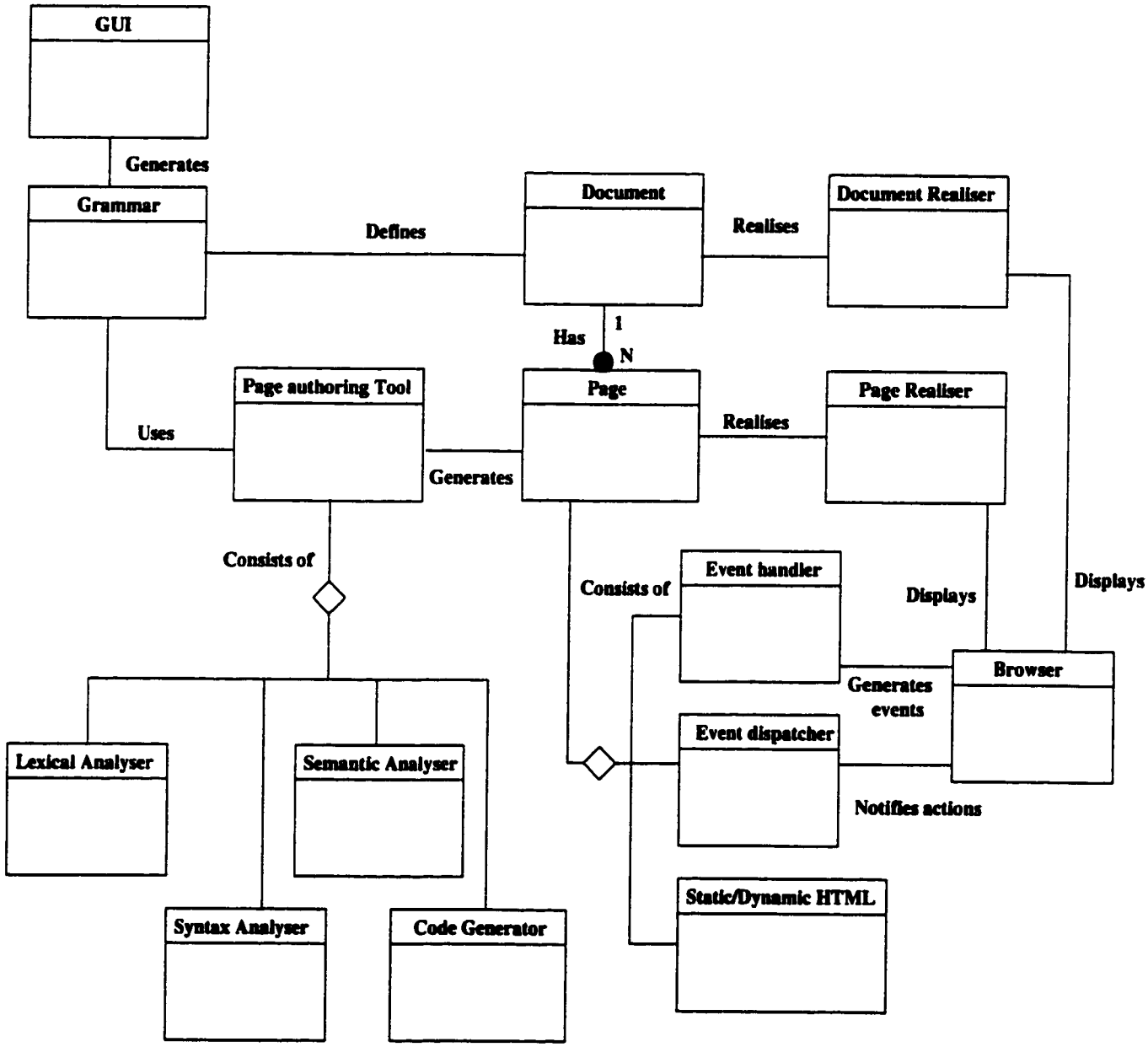

Figure 18: Object diagram

collection of pages and so has a one to many relation with page. Each page consists of an event handler, event dispatcher, and its dynamic HTML code. The document realiser realises the document and displays it on the browser. Similarly, the page realiser realises the page and displays it on the browser. The browser generates events that are handled by the event handler in a page. The event dispatcher analyses the event and notifies the necessary action to the browser.

4.1.3 Dynamic modelling

In this section we describe the dynamic model of the authoring tool. The dynamic model shows the time-dependent behaviour of the system and the objects in it. We consider some of the scenarios in this section. Let us consider the scenario where the author wants to create a page and identify the various steps involved. The various steps are as follows.

- the author specifies the spatial and temporal specifications as per the grammar for the page
- the file is checked for valid filename
- the author inputs the options using the user interface
- the specification file as per the grammar is read
- on success, it is checked for syntax
- on success, the relevant data structures are created
- generate dynamic HTML code for the specifications
- display the generated dynamic HTML code on the browser

The above mentioned scenario specifies the various steps in creating a page. Figure 19 explains the activity diagram for this scenario. We describe the scenario in creating a document as follows. Figure 19 explains this scenario.

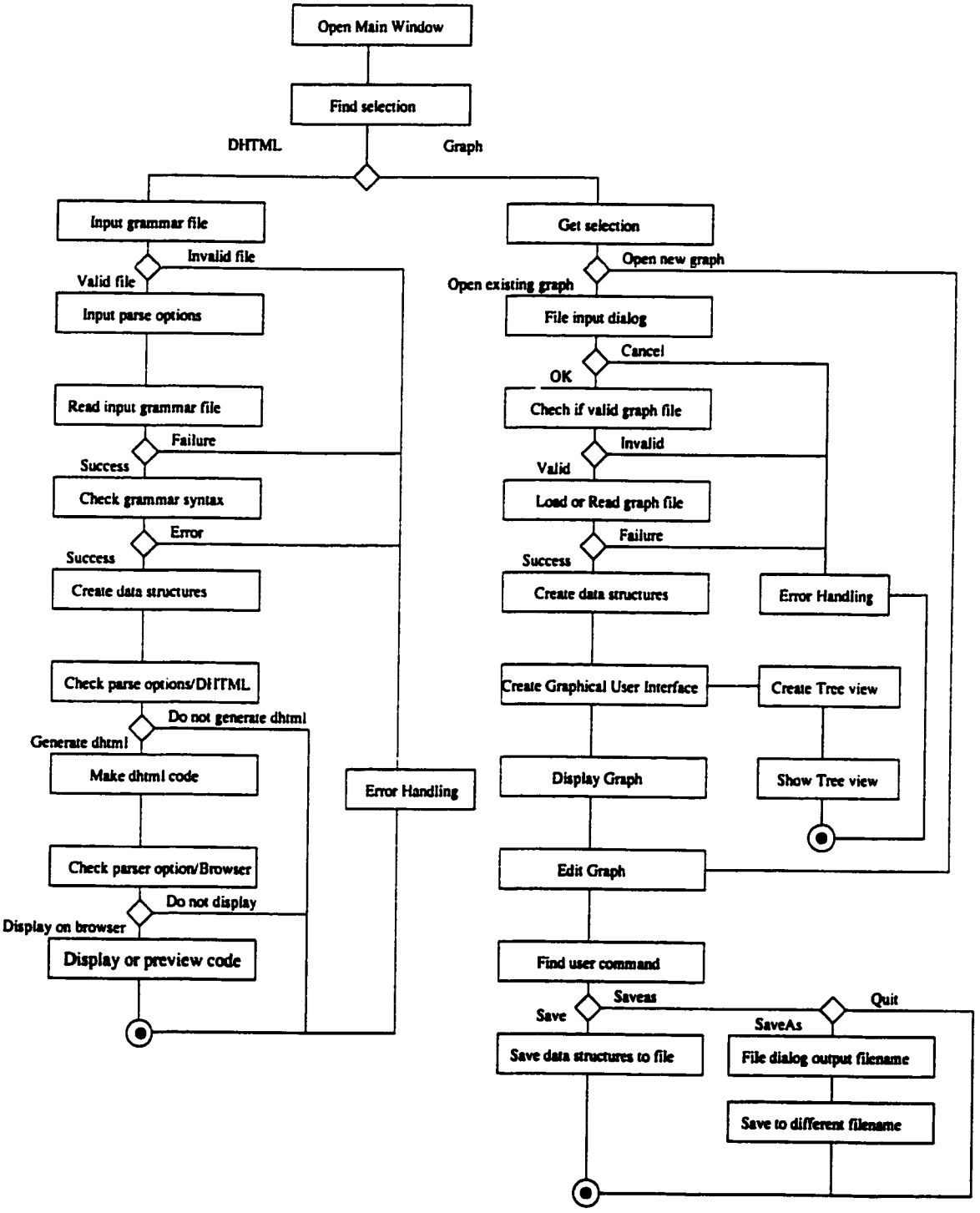

Figure 19: Activity diagram

- open input graph file
- check file for valid input files
- read input graph file
- create data structures from the file
- create the user interface
- display the graph
- allow user to edit the graph
- save the graph in a new file

4.2 Dynamic HTML and Java Script

Navigator version 4 from Netscape [19], which is part of the Communicator product suite, includes three new areas of functionality that taken together constitute Dynamic HTML. The three components of Dynamic HTML are *style sheets*, *content positioning*, and *downloadable fonts*. Used together, these three components give a greater control to the author over the appearance, layout, and behaviour of web pages.

Style sheets allow specification of the stylistic attributes of the typographic elements of a web page. With content positioning, we can ensure that pieces of content are displayed on the page exactly where we want them to appear, and we can modify their appearance and location after the page has been displayed. With down loadable fonts, we can use the fonts of our choice to enhance the appearance of text. Then we can package the fonts with the page so that the text is always displayed with the chosen fonts.

Using content positioning, no longer are we constrained to use sequential content

laid out linearly in web pages. By specifying positions for blocks of HTML content, we can decide what contents goes where on the page, instead of leaving it up to the browser to lay it out for us. We could, for example, place one block of content at the top-left corner of the page, and another block at the bottom-right corner. Blocks of content can share space too, so images and text can overlap. We decide precisely where each part of the content will appear, and Navigator 4 will lay our page out exactly as we want.

Java Script [25] is Netscape's cross-platform, object-based scripting language for client and server applications. Java Script lets us create applications that run over the Internet. Client applications run in a browser, such as Netscape Navigator, and server applications run on a server, such as Netscape Enterprise Server. Using Java Script, we can create dynamic HTML pages that process user input.

Using Java Script, we can change the layout of our page dynamically, and we can modify the page in a variety of ways after the user has opened it. We can make content vanish or appear, and we can change the color of individual parts of our page. We can incorporate animation into our web pages by moving and modifying individual parts of our HTML page on the fly.

Used together, content positioning and Java Script allow us to create web pages that can be more interactive.

4.3 Implementation of the authoring tool

The implementation of the web authoring tool has two parts, the *Authoring part* and the *Presentation part.* The authoring part is a stand alone Java program that takes as input a spatial and temporal specification as per the grammar and generates the corresponding web code, viz. ".html" file. These ".html" files are collected using the

authoring tools Graph Editor, which is a WYSWYG editor that supports full drag and drop creation of a directed acyclic graph.

In the presentation part, the presentation applet delivers the presentation taking as input the data file generated by the document authoring module. In the presentation of a page, to achieve an interactive and dynamic presentation, the page presentation is powered by Java Script libraries and Java applets. The implementation of the authoring tool is done using Java 1.1.6 and Swing 1.0.3 and requires Netscape version 4. Swing 1.0.3 is used to develop the user interface. Apart from the above, the implementation uses concepts of dynamic HTML, Java Script and JavaApplets.

4.3.1 Page authoring

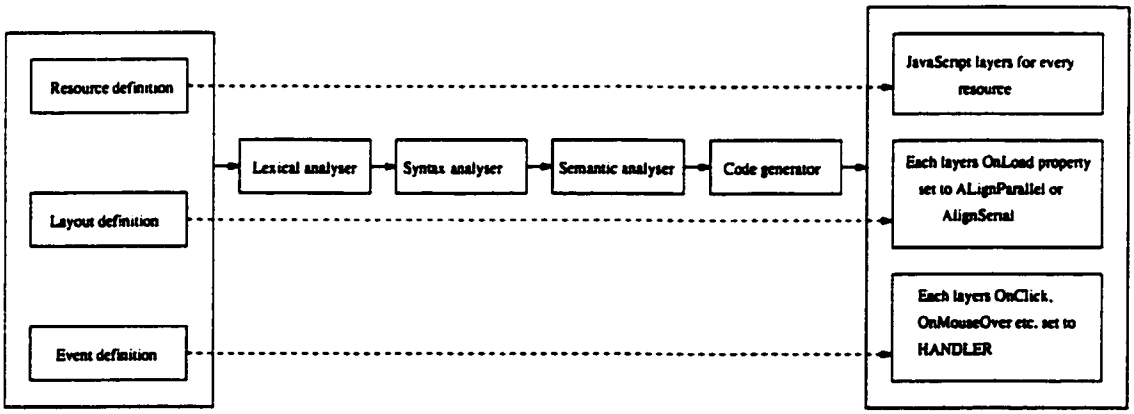

Figure 20: Block diagram of page authoring

In page authoring, a specification file as per the grammar is given as input to the authoring tool and a dynamic HTML output file is automatically generated. Figure 20 shows the block diagram of the authoring process. As shown in Figure 20 the specification file has three types of definitions, viz. *resource definition*, *layout definition*, and the *event definition*. The resource definition defines the type of the resource, its properties etc. An example resource definition is a shown below.

```
% RESOURCE DEFINITION
NAME=Resource1, TYPE=IMAGE, PROPERTY=inline, FILENAME=''prev.gif'',
                WIDTH=default, HEIGHT=default, ALIGNMENT=PC;
NAME=Resource2, TYPE=IMAGE, PROPERTY=inline, FILENAME=''next.gif'',
                WIDTH=default, HEIGHT=default, ALIGNMENT=default;
NAME=Resource3, TYPE=HTML, PROPERTY=inline, FILENAME=''image1.jpg&
                image2.jpg&image3.jpg&image4.jpg&image5.jpg'',
                WIDTH=600, HEIGHT=400, ALIGNMENT=PC;
```

In the above definition, The NAME parameter identifies the resource. The TYPE specifies the type of the resource, viz. HTML, IMAGE, AUDIO, etc. The third parameter defines the property of the resource as inline or as hyperlink. The FILENAME defines the file for the corresponding resource. The other two parameters define the WIDTH and the HEIGHT of the resource. This can be either set to default, in which case the natural height and width of the image will be assumed. The last parameter defines the alignment of the resource in a page. This can be default, in which case the resource will be alignment as per the layout definition, or page centered in which case it will be always aligned at the center of the page.

This resource definition is passed through the lexical analyser, parser, semantic analyser, and the code generator to generate the dynamic HTML code. The lexical analyser segments the syntax in to tokens and passes the tokens to the syntax analyser. The syntax analyser verifies the syntax and forms the data structures to store the definitions. In our implementation, the semantic analysis part is done during code generation. Hence, the code generates the corresponding code to realise the resources. The code for the above mentioned example will be as follows.

```
<LAYER ID=Resource1 top=0 left=0>
<A HREF="#" onclick=HANDLER("Resource1","CLICK")>
<CENTER>
<IMG SRC=prev.gif BORDER=0>
```

```
</CENTER>
</A>
</LAYER>
<LAYER ID=Resource2 onload=AlignParallel("Resource1",
                        "Resource2","FT","Resource1")>
<A HREF="#" onclick=HANDLER("Resource2","CLICK")>
<IMG SRC=next.gif BORDER=0>
</A>
</LAYER>
<LAYER ID=Resource3 src=image1.jpg onload=AlignSerial(
   "PANEL1","Resource3","FL","PANEL1") Width=600 HEIGHT=400 >
</LAYER>
<SCRIPT>
var list = new list();
list.Next = Next;
var i=0;
list.HANDLE=new Array();
for(i=0;i<100;i++)
list.HANDLE[i]=null;
list.HANDLE["Resource3"]=new Array("image1.jpg","image2.jpg",
     "image3.jpg","image4.jpg","image5.jpg");
list.HANDLE["Resource3count"] = 0;
</SCRIPT>
```

Following the resource definition, the spatial and temporal specification file contains the layout definition for the various resources. The layout grammar is described in Chapter 3. The layout grammar defines the serial and parallel relation between the resources in a page. An example layout definition is as follows.

```
% LAYOUT DEFINITION
pack START Resource1 11;
```

```
pack PANEL1 Resource1 + Resource2 FT Resource1 2;
pack PANEL1 PANEL1 * Resource3 FL PANEL1 2#
```

The layout specification shown above passes through the various phases of the authoring tool as shown in Figure 20. The code generator generates the code requied to realise the layout of the resources. The code generated is as shown above. Each layer has a OnLoad property which is executed during load time of that resource. This property of dynamic HTML is used to change the various properties of the resource during load time of that resource. In the code segment shown above, the layer definition for Resource2 has its OnLoad function pointing to a custom library function called AlignParallel. The Java Script function that does the alignment is as follows. The code demonstrates the parallel alignment between two resources identified by str1 and str2 with the reference resource identified by str3..

The final entry in the spatial and temporal specification file is the event definition. The event definition defines the interaction between the resources in a page. As a continuation to the above mentioned example, let us assume that a click on the *prev* button should take the image resource to its previous image and the *next* to its next image. To achieve the above, the click event on the *prev* and *next* button image are mapped to previous and next on the image resource. The specification as per the grammar for the event definition is as follows.

```
% EVENT DEFINITION
Resource1 CLICK Resource3 Prev;
Resource2 CLICK Resource3 Next;
```

The code generated to achieve the interaction between the resources is a shown below. The onclick event of the resources are caught and passed on to the event handler. The event handler posts the events to the server applet of the page. The server applet maps the event to its event table and executed the corresponding action.

```
<Layer Name="SERVER" visibility=hide>
```

```
<Applet NAME="SERVER" code="Server.class" height=50
        width=200 MAYSCRIPT=true>
<Param Name=EventDefinition Value="Resource2 Next
       Resource1 Next Resource2 Next Resource3 Hide
       Resource2 Prev Resource1 Prev Resource2 Prev
       Resource3 Show ">
</Applet>
</Layer>
```

In the code generated, every event might be a user generated event or a cascading event. A cascading event implies one event generating another event. For example, a hide event on once resource can generate a show event on another resource. The server applet has the event table passed as a parameter during creation.

4.3.2 Document authoring

In this section we describe the document authoring system. The document authoring system consists of a graphical user interface to create a simple data file for the presentation tool to realise the document during presentation. A sample data file created by the document authoring tool is as shown below. The string at the beginning of the file identifies a specification file. The various nodes in the graph are defined and the properties are specified. The last line in the file defines the edges between the nodes in the graph. An entry <1,2> means that there is a directed edge from node1 to node2.

```
GRAPH-FILE-VER1.0
1 First.html   "First Node Heading"   289 23  RECTANGLE
2 Second.html  "Second Node Heading"  212 95  CIRCLE
3 Third.html   "Third Node Heading"   348 91  CIRCLE
4 Fourth.html  "Fourth Node Heading"  150 137 CIRCLE
5 Fifth.html   "Fifth Node Heading"   243 151 CIRCLE
```

```
6 Sixth.html   "Sixth Node Heading"   398 148 CIRCLE
7 Seventh.html "Seventh Node Heading" 309 149 CIRCLE
<1,2> <1,3> <2,4> <2,5> <3,6> <3,7>
```

4.4 Implementation of Presentation tool

The presentation tool realises the output of the authoring tool. There are two types of presentation, viz. *page presentation* and *document presentation.* The page presentation aims at achieving the dynamic layout of resource in a page, and the interaction between the resources in a page. The document presentation presents the document on the web.

4.4.1 Page presentation

The page presentation is done with the help of a custom Java Script library to handle events and to layer the resources. As mentioned in page authoring, the onload property of every resource is set to either "AlignParallel() or AlignSerial" depending on the type of layout specified. This is a custom Java Script function that is used to place to resources in the appropriate positions inorder to realise the layout.

The code below shows a brief example of the AlignParallel function. The function takes as parameter two source panels that are to be aligned (str1 and str2). It also takes as input the alignment property and the reference resource. The functions checks for valid resource names and then computes the left, top, right, bottom for the second resource relative to the first resource. As this method is called every time a new resource is loaded, the layout is computed dynamically every time there is a change in the content of the resource.

```
function AlignParallel(str1,str2,align,str3) {
var resource1 = window.document.layers[str1];
var resource2 = window.document.layers[str2];
```

```
var reference = window.document.layers[str3];
if((resource1!="undefined")&&(resource2!="undefined")&&
                              (reference!="undefined")) {
 resource2.left=resource1.clip.width+resource1.left;
  var padding=0;
  switch(align) {
    case "FT":
      resource2.top=reference.top+padding;
      break;
    case "FC":
      resource2.top=(reference.clip.height)/2+reference.top+padding;
      break;
    case "FB":
      resource2.top=(reference.clip.height+reference.top)+padding;
      break;
    default:
      resource2.top=reference.top+padding;
      break; }
}
else
  alert("Error: Undefined panel/resource found in declaration...");
}
```

The above mentioned code is used to realise the layout of a page. To achieve interaction between the resource in a page, events generated on all resources are directed to a "HANDLER()" function. The handler function takes as input the name of the resource receiving the event and the event. After checking for valid resource name, the action for the event is executed. After executing the action for the resource, the event to generate in another resource is to be found. To do this, the name and the resource and the event is passed to the server applet in the default option. the server

applet maps the event with its event table and triggers the corresponding action on the destination resource.

```
function HANDLER(name,event) {
var resource1 = window.document.layers[name];
if(resource1 != "undefined") {
  switch(event) {
    case "Next":
      Next(name); break;
    case "Prev":
      Prev(name); break;
    case "Show":
      ShowLayer(name); break;
    case "Hide":
      HideLayer(name); break;
    default:
      document.layers["SERVER"].document.applets["SERVER"].
              PostMessageToServer(name,"CLICK");
      break; } }
else
  alert("Error: Panel/resource not found - "+name); }
```

In the current implementation, authors have to manually hand code custom events and actions. Custom events can be added as a case statement in the switch case for events in the HANDLER function. The required action can be defined in a JavaScript function and should be called in case if the appropriate event occurs.

4.4.2 Document presentation

The document presentation applet takes as input the document data file created by the document authoring tool. The data file is read by the applet and appropriate

data structures are formed. The document presentation tool then displays the graph on the web browser. The graph presented is a directed acyclic graph, where the nodes represent pages and the edges represent the precedence relation between the pages. The code below shows the document presentation applet defined in a html file.

```
<APPLET code="ShowTutorial.class" width=800 height=500>
<param name=FileName value="main.gph">
</APPLET>
```

As the user interacts with the graph by clicking on the respective nodes, the applet displays the contents of the node by opening another browser window.

Chapter 5

Application - A case study

In this chapter we demonstrate a real-life application of our web authoring tool in the area of Computer Aided Learning (CAL). A web based presentation is developed for the COMP 228 course taught at Concordia University for the first year undergraduate students. COMP 228 is a course on Computer Organisation and Assembly Language programming being taught at Concordia University. This is an introductory course in assembly language programming. The two main objectives of the course are:

- To introduce a specific Computer System, and a small subset of its Assembly Language features.
- To introduce the concepts of Computer Organisation in a generalised way giving reference to the particular system learnt.

The course being taught at the University has almost 300 students registered in it. The text-books followed in the course until June, 1998 are *Assembly Language for the IBM PC, K.R. Irvine* and *Computer Organisation and Architecture: Designing for performance, W.Stallings.* Apart from the text-books the course has abundant material available on the web. Some of the useful links are given in the course outline page in the departments home page at http://www.cs.concordia.ca/comp228/. Apart from these material, the course also has a set of slides containing the course notes. The slide show is non-interactive and is created laboriously every time when the

course material changes. To supplement the existing material with more user-friendly presentation, the Web authoring tool developed as part of this thesis work is used to develop an interactive slide show for the COMP 228 course. The slide show has eight modules comprising approximately fifteen slides each. The inputs to authoring tool are:

- slides for all the modules
- audio files corresponding to the modules

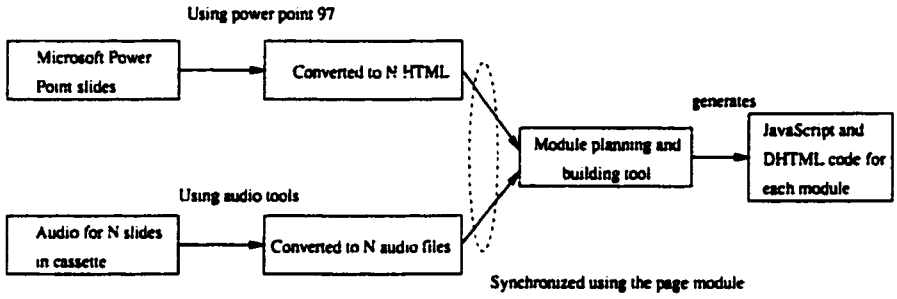

Figure 21: Slide show page planning and building

The Web authoring tool comprises of an *authoring part* and a *presentation part.* The assembly language part of COMP 228 is organised into eight modules. The eight modules are General Introduction[6 slides in Power Point], Register structures and Internals [17 slides], Assembly Language Basics [14 slides], Four addressing modes [10 slides], Stacks [12 slides], Arrays and Arithmetic [10 slides], Subroutines [16 slides], and Macros [8 slides]. The audio files explaining each slide contained in a module is provided by the author along with the Microsoft Power Point slides. Using the facilities available in Power Point, the slides are converted into HTML files and customised. The audio for the whole presentation given by the author is split into smaller units corresponding to each slide and saved in different files using suitable audio editors. In our case, we use soundtool and gaintool available on SunOS. Once this is done, the set of slides and audio are layered accordingly and sequenced as described in chapter 3, so that each slide is linked to its appropriate audio file using the authoring tool. The authoring tool generates the necessary Java Script and DHTML code needed to

realise the page representing the corresponding module. Figure 21 shows the block diagram of the page planning, building and realisation phase in a slide show presentation.

Once the individual modules of the assembly language course have been developed, they are organised and grouped as per the "Concept Graph". Figure 22 shows the block diagram of document planning and building. The collection of pages or modules are linked using the Concept Graph Model and the document presentation applet is generated. The applet does the interactive presentation on the browser for the individual modules with audio and slides synchronised.

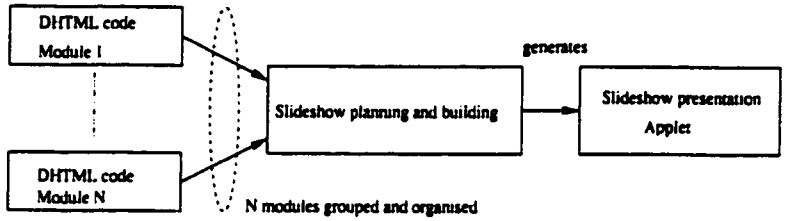

Figure 22: Slide show document organisation and building

We organise the rest of this chapter as follows. In section 5.1 we discuss the creation and presentation of the course material for the assembly language part of COMP 228. Subsequently, in section 5.2 we evaluate the usefulness of the authoring tool and presentation tool using real-life authors and users.

5.1 Authoring the assembly language course

We applied our newly developed tool in a specific example of Computer Aided Learning (CAL). The course material consists of an interactive slide show which is made with the help of our authoring tool. Out of the eight modules used for teaching this part of the course, slide show for four modules is used for evaluating the authoring tool. The four selected modules are then presented to students with the help of our presentation tool.

An example author who is a final year undergraduate student in Computer Science is selected. The professor creates the raw material required for the presentation using the tools and packaged the professor is familiar with. The example author uses the materials created by the professor for the slide show. Selected students of the course are requested to read the material presented with the help of the presentation tool and are supplied with a questionnaire for evaluation. The performance of the students is studied and their feedback is analysed and the authoring tool is incrementally refined for better performance. The feedback from the author is studied and the authoring tool gets fine tuned as per the requirements of the author. The following issues were examined when the authoring tool was put to real-life use:

- Versatility and feasibility of the authoring tool
- Ease of use by an author
- Learning curve required to master the authoring tool
- Incremental development, and maintenance of the authored material
- Usefulness on the authoring tool in producing a satisfactory end-product (courseware)

5.1.1 Slide show page definition

The material for presentation is created by the author. The slides are composed using the material provided in Microsoft Power Point. The slides for each module is converted to HTML files using the features provided in Microsoft Power Point. Microsoft Power Point saves the slide as several HTML files, each corresponding to one slide within a module. There are two versions of slide show that are generated automatically by power point, namely graphical version and text version. Power Point generates 2*N sets of HTML files for a single module comprising of N slides where, one set is for the graphical version and the other set is for the text version. The

audio for each module is recorded in Digital Audio Tape (DAT) in the Audio/Visual department, Concordia University using DAT recorder. The digital audio is played using a digital audio player. This digital audio player is connected to the server and the audio converted to audio files using suitable audio tools. The audio recording for a module is further split into smaller units corresponding to each slide and saved in different files.

The specification file as per the grammar for the presentation is created using a text editor. The resources are defined as per the syntax of the grammar and the layout is specified accordingly. Using the event definition module of the grammar, the N HTML-files corresponding to the N slides within a module are synchronised with the N audio files so that the audio part starts after the corresponding slide is displayed. Once the specification file as per the grammar has been created, the specification file is fed as input to the authoring tool to generate the low level DHTML code to realise the presentation. Figure 27 shows the "screen shot" of the authoring tool used to generate DHTML code from the input specification file as per the grammar. The checkbox option in the GUI for generating DHTML code is checked and the specification file as per the grammar is given as input. The GUI prompts the user to input the destination file name. On entering the destination file name and the user pressing the "ok" button, the authoring tool automatically generates the low level DHTML code to realise the grammar based specifications.

5.1.2 Slide show document definition

Authors have to be provided enough support to collect the automatically generated DHTML files to make a courseware. In the document definition phase authors are provided with a graph editor to build their Concept Graph. The graph editor provides a drag and drop environment for authors to build their Concept Graph. Authors can freely drag and drop the various DHTML files in the graph editor and define a precedence relation between them during the creation of a concept graph. The precedence relation between dynamic HTML concepts (nodes) can be drawn as directed edges

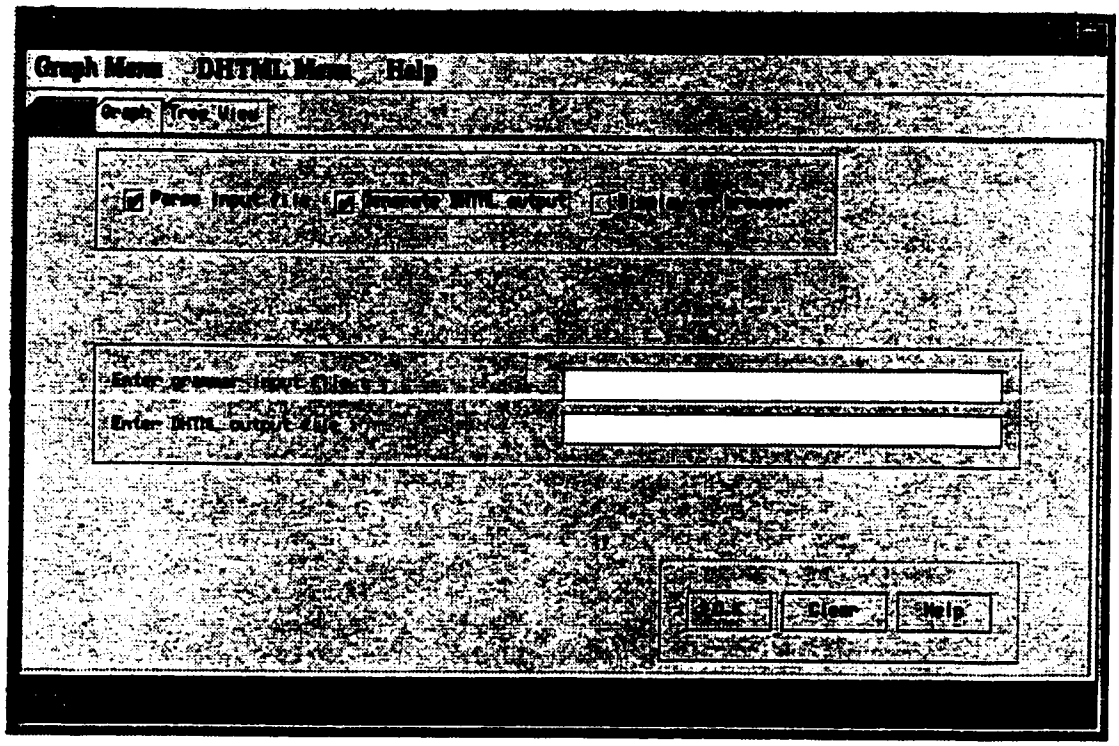

Figure 23: DHTML page of Web authoring tool

using the graph editor. Figure 24 shows a screen shot of the graph editor's output. Once the concept graph has been drawn using the editor, it is saved as a ".gph" file by the editor. A concept graph once created and saved can be reloaded and edited freely at a later time.

5.1.3 Slide show presentation

The world wide web is used as the medium of presentation. The slide show presentation is two fold.

- concept graph presentation (document presentation)
- presentation of individual concepts (page presentation)

A presentation Java applet is used to deliver the concept graph on the world wide web. The graph file (".gph") generated by the graph editor is used as input by the

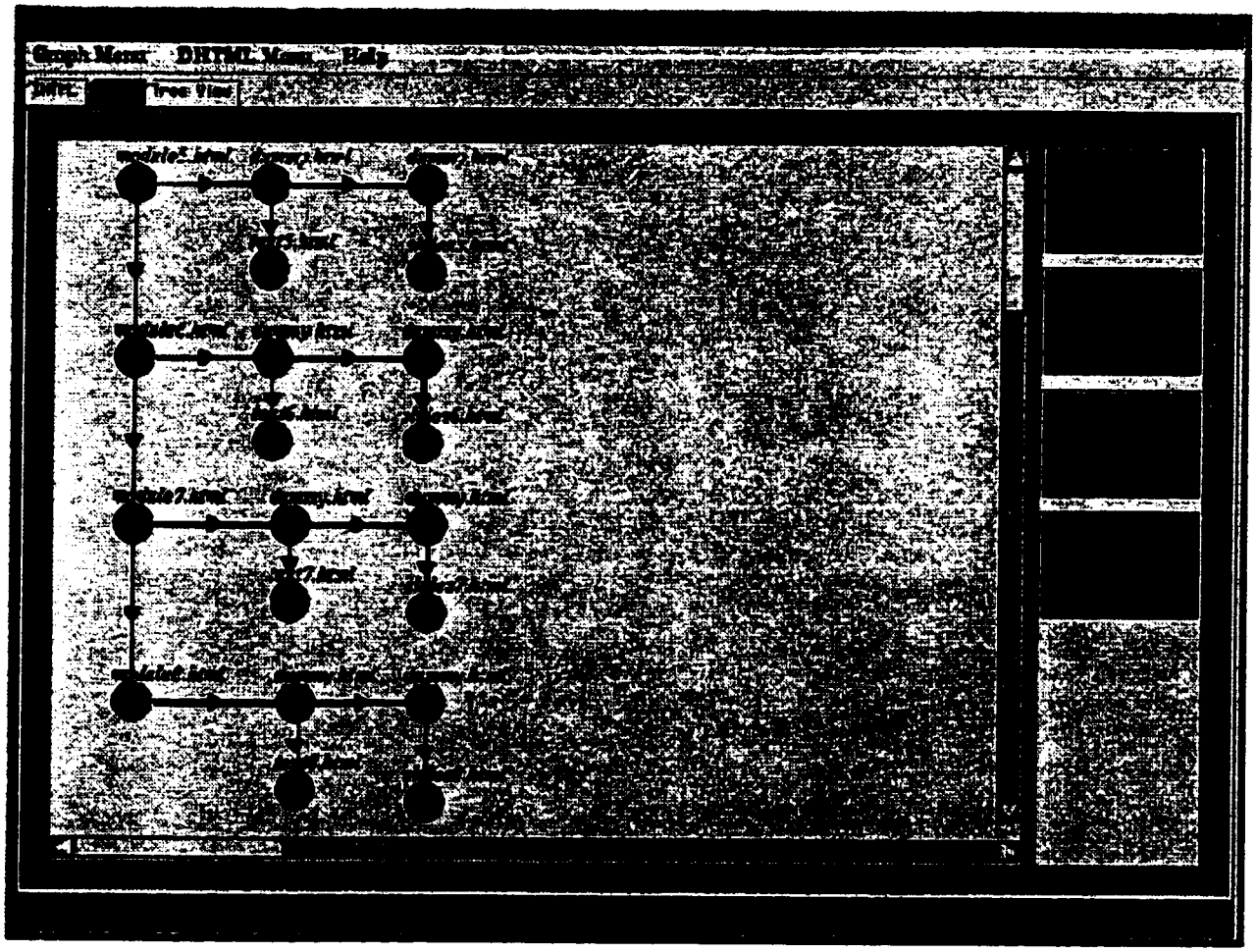

Figure 24: Graph editor of Web authoring tool

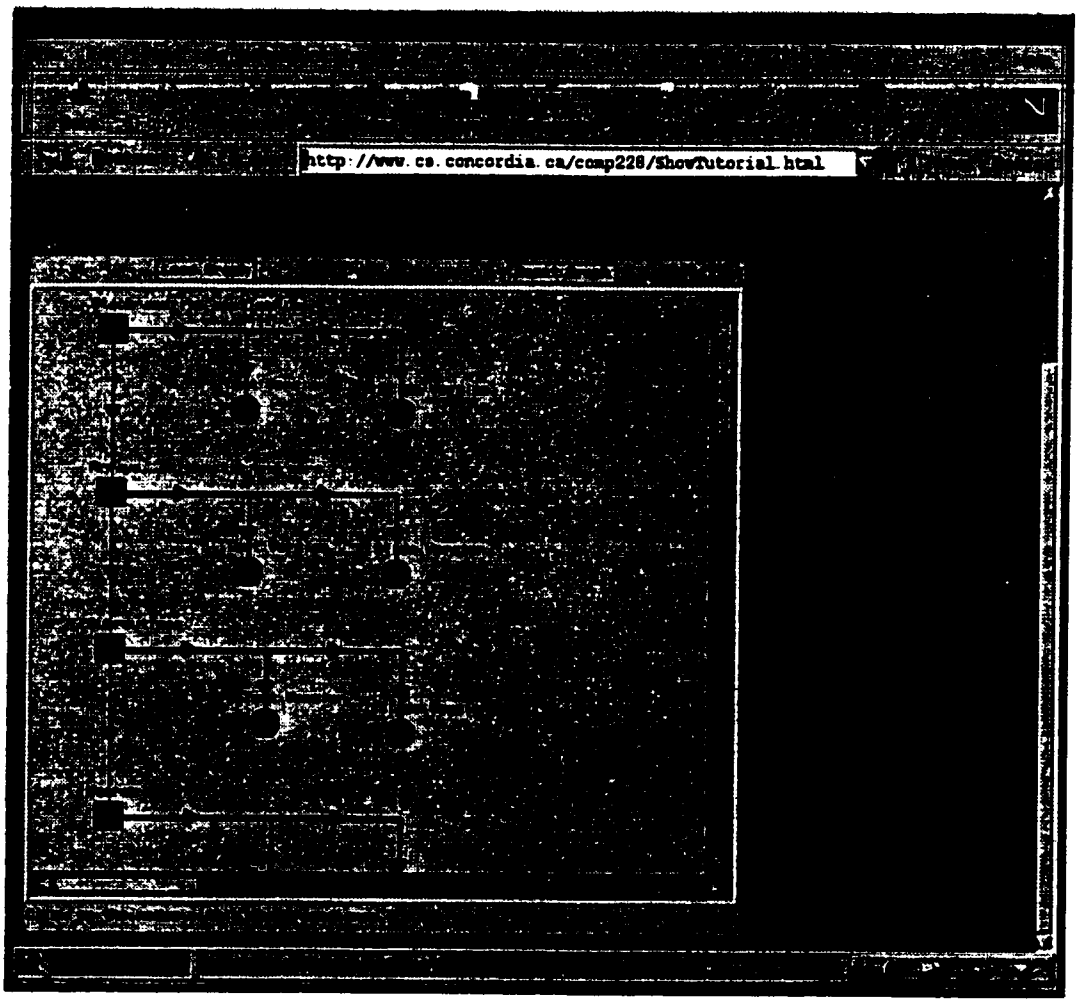

Figure 25: Document presentation

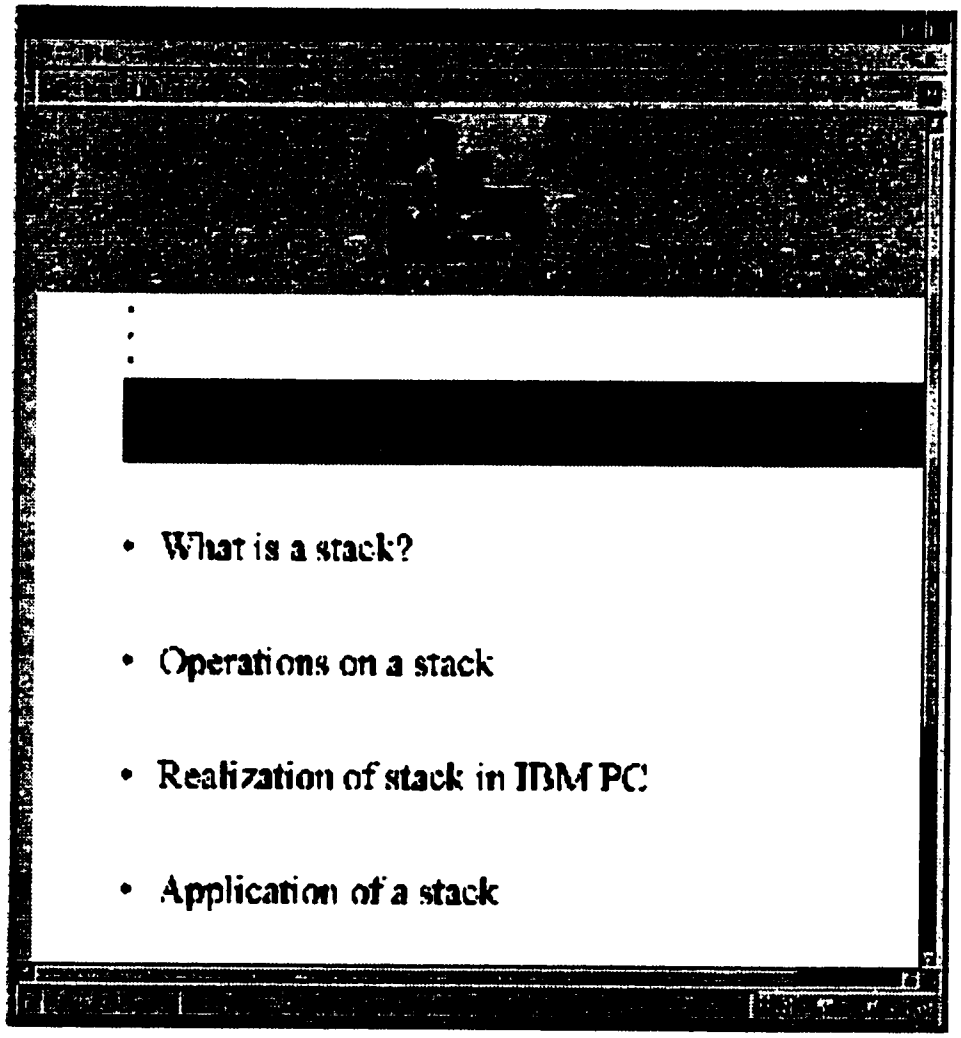

Figure 26: Page presentation

presentation applet. The applet reads the input graph file and displays it on the web. A screen shot of the presentation generated by the applet is shown in Figure 25. The presentation applet monitors the user action and reacts accordingly. A click on any node lets the user to travel to the respective web page describing the specific concept. The presentation applet also notifies the user of the pre-requisites that should have been covered before reaching that concept in the courseware.

Each concept is presented in an interactive way with the help of custom written Java Script and Java libraries, and using DHTML features. The definitions defined for interaction in the slide show module definition is realised using the libraries. Figure 26 shows a screen shot of a page during presentation.

5.2 A brief survey with end-users

The assembly language course material presented on the web was put to real-life use for an undergraduate course being taught at the department of Computer Science, Concordia University. The term real-life implies the following:

- Real lessons
- Real students
- Significant quantity of material (not just one sample lesson)

For this application we have developed audio presentation for four out of the eight modules. All the eight modules was presented on the web as slides and text, and modules five, six, seven and eight were available with audio and slides.

To evaluate the performance of the authored product, a real-life author is asked to use the authoring tool to prepare the lessons for the course. Once the course material is made ready for presentation, real-life students were asked to use the presentation and their feedback was analysed. In our case study, one real-life author and twenty five real-life students are requested to evaluate the authoring tool and the authored product. As part of the case study to evaluate the product, a questionnaire is given to

students and their feed back analysed. However, to evaluate the author, a structured interview is carried out with the author and the feedback analysed.

5.2.1 The questionnaire given to students

A questionnaire is given to the set of students who are asked to use the authored product and their feedback is analysed. The questions are prepared such that it can conclude on three aspects of the authored product:

1. Usefulness of the courseware
2. Availability of information in an organised manner
3. Quality of material and presentation

In the list of questions given below, the first and the sixth questions provide information on the usefulness of the authoring tool. Questions two and three help us conclude on the availability and the organised way of presenting information. The questions four and five are pertaining to the quality of the material and the presentation.

1. Was this new method of presenting information useful to you?
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate
2. Ease with which the presentation aspects could be changed or controlled
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate
3. Were you able to navigate through the presentation easily
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate
4. Was the audio presentation easily comprehensible
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate
5. Was the visual presentation easily comprehensible
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

6. Was the material presented useful to you
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

7. Would you like to refer to slides while reading the text, and vice versa.
   (a) Yes (b) No (c) No Comments

8. In following the material content of this module, how would you classify your pre-requisite knowledge.
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

9. Any other comment or suggestion that you would like to make
   (Responses to this question are not included in the tabular summary)

5.2.2 The questionnaire given to authors

In the evaluation for the authoring tool we conduct a structured interview with one example author. Author in this case is different from the professor. An interview was conducted with the real-life author to evaluate the tool. In this interview the following were examined:

- Feasibility and ease of using the authoring tool
- Learning curve required to master the authoring tool
- How easily incremental modifications and maintenance were made to the authored end-product

Questions are asked to the author about the feasibility of the authoring tool. The ease with which the authoring can be done is analysed and useful feedback taken for improvements and modifications. A major difficulty in learning new tools is a steep learning curve. Proper feedback is taken from the authors to evaluate the learning curve in mastering the authoring tool. The author is asked to do incremental development and also do maintenance of the courseware with changing input data. The

ease with which the maintenance and modifications could be done to the courseware with the help of the authoring tool is evaluated.

1. Ease with which the presentation can be authored
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

2. Use of Graphical User Interface for authoring
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

3. Efficiency of the automatically generated dynamic HTML code
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

4. Manual editing of the automatically generated dynamic HTML code
   (a) Very Easy (b) Easy (c) Fair (d) Impossible

5. Pre-requisite knowledge required to learn the tool
   (a) Excellent CS (b) Very Good CS (c) Good (d) Average CS (e) NO CS (CS: Computer Science knowledge)

6. Has supported the author to achieve his goals
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

7. Choice of grammar structure
   (a) Very hard (b) Hard (c) Fair (d) Simple (e) Easy

8. Tool support provided to authors to make incremental changes and subsequent maintenance
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

9. User friendliness of the authoring tool
   (a) Excellent (b) Very Good (c) Good (d) Fair (e) Inadequate

10. Achieved stated objectives in a scale of 1-5 (1.excellent and 5.poor)

    - 1. The authoring tool is easy to use

- 2. The learning curve required to master the authoring tool is minimum
- 3. Incremental modifications and maintenance is easy

11. Time taken to learn the authoring tool

12. Educational background of the author

13. Time taken to prepare module 5, 6, 7, and 8 in their final form

14. How easy was it to edit and correct an existing presentation?

5.2.3 Evaluation results

The results of the questionnaire based evaluation are summarised as follows. We draw our conclusions from students feedback based on the relationship between questions and conclusions as shown in Figure 27.
The conclusions of the evaluation are shown in a tabular column. Twenty five students were asked to evaluate the presentation. The following conclusions were made:

- 98% of the students gave a feedback that the tool was *very useful*
- 84% of the students felt that it was an *excellent* way of presenting information in an organised way. 14% felt that it was a *good* way of presenting information in an organised manner.
- 90% of the students felt that the quality of the material was *excellent.* 10% of the students felt that the quality of the material presented was *good.*
- 68% of the students didn't have very good knowledge of the material presented. 32% of the students had very good back ground on the material presented.

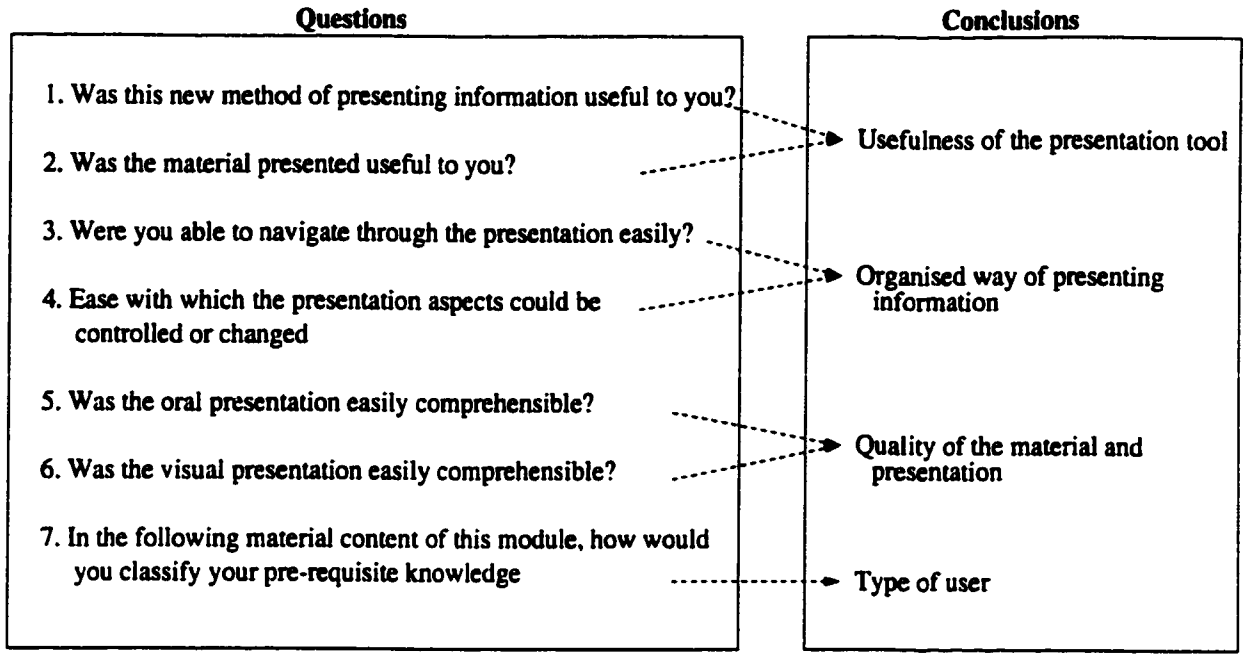

Figure 27: Evaluation of the authoring tool

(This is because the topics of module 5, 6, 7, and 8 are not covered yet in all sections of COMP 228)

- 72% of the students wanted to refer to slides when reading text, and to refer to slides when reading text. 12% of the students didn't want to refer to slides when reading text, and vice versa. 16% had no comments.

Questions	Excellent	Very Good	Good	Fair	Inadequate
1	17	8	0	0	0
2	9	15	1	0	0
3	11	12	1	1	0
4	14	5	6	0	0
5	12	11	2	0	0
6	12	10	3	0	0
7	18-Yes	3-No	4-No comments	0	0
8	3	5	9	5	2

[7. Would you like to refer to text while reading slides and vice versa]

5.3 Interpretation of the survey results

In the preparation of courseware for COMP228 course, three types of people were involved:

- Professor - who created the material for the text and slides, audio for the slides, and also the concept graph
- Author - a senior undergraduate student who has taken this course before and worked closely with the professor in preparing slides, text and the audio
- Student - the end-user of the authored presentation

The raw material for the presentation was provided by the professor to the author. The raw materials in this case, included slides, audio files, and the text. The slide show web pages were created by the author as follows:

Step 1 The author created a spatial and temporal specification file as per the grammar using a text editor.

Step 2 The specification file was given as input to the page authoring tool to generate the necessary low-level HTML code required to display the page.

Once the individual web pages have been created, the author creates a document using the steps defined below:

Step 3 A graph editor is developed as a part of this thesis work. Using this graph editor, the author drew the concept graph as structured by the professor. The graph is saved in an output file.

Step 4 The author specifies the name of the output file from step 3 as input to the document presentation applet.

Step 5 The presentation is viewed on the browser and tested.

In our case the author was provided with the raw materials for the presentation by the professor. To prepare the raw materials, the author took approximately 10 hours per module. Once the raw materials were prepared it took less than 30 minutes for the author to generate the web pages for each module by following the steps given above.

As part of our evaluation, only one author was considered to evaluate the authoring tool. Twenty five students were considered to evaluate the authored presentation. A structured interview with the author lets us conclude the following:

- that the authoring tool was easy to use,
- very little pre-requisite knowledge was required,
- the authoring was versatile to achieve desired goals,
- incremental development and modifications of the authored material was easily done.

The evaluation results from the end-users lets us conclude on the usefulness and quality of the authored product. As mentioned in the previous section, 95% of the students felt that the authored presentation was very useful. 90% of the students felt that the quality of the authored product was very good.

The overall wvaluation of the case study lets us conclude the following:

1. It is relatively easy to develop courseware for an author using our authoring toolkit.
2. Incremental development and modification of the courseware is simple using our authoring tool kit.
3. The authored courseware is very useful.

In our case study due to non-availability of time on the part of the professor, the role of the professor and the author was played by different persons. These two roles could be played by a single person aswell.

Chapter 6

Conclusion

Increase in computer use combined with a rapid expansion of Web access among the student population and a significant development in associated technologies have created a need for web-based instruction. Until the advent of dynamic HTML and Java, Web pages were static in nature. Though dynamic HTML and Java have made the Web more dynamic in nature, making use of the this feature is not easy for non-computer expert authors. Another issue in using the Web as a medium of presentation for learning lies in organising the courseware. As there is vast amount of material in a typical courseware, different users would have varying needs to start or position at different places in the navigation.

To achieve an organised presentation of the courseware, we have made use of a model called the Concept Graph Model (CGM). The CGM is relatively easy for an experienced author to create a courseware. Using the CGM end-users will be able to position themselves in a large presentation.

In this thesis, we describe a solution using which the authors can specify the spatial and temporal constraints in organising a page using higher-level descriptions. We have developed a grammar based approach using which the spatial and temporal constraints of a web page can be specified in a simple manner. Starting from this spatial

and temporal specification file as per the grammar as an input, the authoring toolkit automatically generates the necessary low-level web code required to display the web page.

We choose the widely accessible and available Internet as the medium for presentation. The software (our Web authoring tool) was developed on a stand-alone Sun sparc workstation (loon). The authored product (namely the four modules) are also developed on the same machine. At the end, the authored product was easily ported (in about 2 hrs) to the Internet platform. The portability of our software was thus evident. The authoring tool was also designed for protability to Internet platform. However, this porting is not done yet. If it were made available, one could try collaborative authoring as well.

The authoring toolkit is evaluated by a real-life author and the authoring toolkit fine tuned. The authored courseware is evaluated by real-life users and their feedback is analysed.

6.1 Limitations and future work

We suggest two ways for authors to create active courseware. In the first method, authors themselves use low level programming languages to make their web pages. In the second method, authors use higher-level descriptions and let an authoring tool generate low-level web code. The advantage in the first method is increase in power and the disadvantage is complexity and the learning curve. In the second method, the advantage is simplicity in usage, but the disadvantage is reduced power due to abstraction.

Most of the commercially available tools provide inbuilt support for the Web. Though most of the tools automatically generate web code, the problem of integrating these

web code generated by the different tools still persists. We also face this problem in our creation process. To generate HTML files from Microsoft Word and Microsoft Power Point is simple, but it takes some more effort in order to port them to Unix and access it as a resource in the authoring tool.

In the current version, the synchronisation between the audio and slides is course grained. A click on the "Next" button starts the audio as well the next slide. Erratic clicking on the next button of the audio can render the audio and the slides out of phase. A hand shake mechanism can be provided between the audio and the slide to achieve better synchronisation between them. Fine grained synchronisation between audio and active resources would be a nice future work.

We are mostly concerned about events within a web page. It will be an interesting work to explore the possibility of inter page events during a presentation. As an example, during a slide show a next slide event can generate a event in another web page requesting it to display the corresponding text material for that slide.

Though custom events and actions can be defined by the author, in the present implementation it involves manual intervention of the author to write some low-level code himself to define custom actions and events.

The concept graph model used in our document presentation does not cover all possible cases in a presentation. Though this model has its own weaknesses, it is useful in our application. The limitations of the concept graph model are beyond the scope of this thesis.

In our presentation user details are lost after an user navigates through the courseware. It will be great advantage to users in the presentation can have persistent information. This feature will enable each user to start from the point he/she left the

presentation.

It will also be an excellent idea to add typical class-room type facilities to the software. These facilities can include chat rooms, news groups, on-line quizzes, bulletin boards etc.

Appendix A

Structure of our implementation

We present the list of files in our implementation and a brief description of their functions.

- CMainFrame.java - Main window of the graphical user interface
  - CMainMenu.java - main menu definition
  - dhtmlPanel.java - GUI for dynamic HTML creation
  - graphPanel.java - GUI for document creation
  - treePanel.java - GUI to view the graph as a tree
- MainProgram.java - Input to parser and creation of data structures
  - ResourceStructure.java - structure of Resource definition
  - LayoutStructure.java - structure of Layout definition
  - EventStructure.java - structure of Event definition
- CGraphAlgorithm.java - GUI used to draw a graph
  - CGraph.java - reads an input graph file
  - GraphCanvas.java - holds the graph GUI
- ShowTutorial.java - to present the document on the web

- CNodeIndex.java - structure of nodes in the graph
- CNodeToFile.java - Converts from node number to its corersponding URL
- CoordinateArea.class - allied class for GUI
- FramedArea.class - allied class for GUI

- Page presentation
  - Tool.js - JavaScript support libraries
  - Sound.java - sound applet
  - Server.java - Server applet for event handling

- Parser files created by JavaCC
  - Parser.java
  - ParseException.java
  - ParserConstants.java
  - ParserTokenManager.java
  - Ascii-charStream.java
  - Token.java
  - TokenMgrError.java

- Miscellenous files
  - CGraphDialog.java - dialog for graph options
  - Global.java - to maintain global variables
  - MyFileDialog.java - customised file dialog
  - GraphPopup.java - dialog box that to enter node options
  - ExecuteCommand.java - JNI interface to invoke Netscape
  - ExecuteCommand.c - JNI interface to invoke Netscape

Bibliography

[1] Nael Hirzalla, Ben Falchuk, Ahmed Karmouch, *A temporal model for interactive multimedia scenarios, IEEE Multimedia, vol.2 No.3, Fall 1995. pp.24-31*

[2] Alan Borning, Richard Lin, and Kim Marriott, *Constraints for the Web, ACM Multimedia 97, Seattle Washington USA*

[3] Louis Weitzman, Kent Wittenburg, *Grammar-based articulation for multimedia document design, Multimedia Systems (1996)4:99-111*

[4] Louis Weitzman, Kent Wittenburg, *Automatic Presentation of Multimedia Documents Using Relational Grammars, ACM Multimedia 1994, SanFrancisco, CA, USA*

[5] M.Buchanan and P.Zellweger, *Specifying temporal behavior in hypermedia documents, Proceedings of the ACM Conference on Hypertext, ACM Press, NY, Dec. 1992,pp.262-271*

[6] T.D.C Little A. Ghafoor, *Synchronization and storage Models for Multimedia Objects, IEEE JSAC Vol.8, No.3,pp 413-427,Mar 1990*

[7] Lynda Hardman, Guido van Rossum, Dick C.A.Bulterman, *Structured Multimedia Authoring, CWI:Centrum voor Wiskunde en Informatica*

[8] R.Rossum, J.Jansen, K. Mullender, D.Bulterman, *CMIFed:A Presentation Environment for Portable Hypermedia Documents, proc. of ACM Multimedia 93, ACM press, CA, pp.183-188, August 1993.*

[9] Dick C.A.Bulterman, and Lynda Hardman, *Multimedia Authoring Tools: State of the Art and Research Challenges, CWI: Centrum voor Wiskunde en Informatica, Kruislaan, Amsterdam*

[10] Director version 2.0, *MacroMind 1990 (dynamic media authoring tool for the Apple Macintosh)*

[11] John Bates, *Presentation Support for Distributed Multimedia Application, University of Cambridge*

[12] Michael Vazirgiannis, and Susanne Boll, *Events in Interactive Multimedia Applications: Modelling and Implementation Design, International Conference on Multimedia Computing and Systems, June 3-6, 1997, Ottawa, Canada.pp 244-251*

[13] Michael Vazirgiannis, and T. Sellis, *Event and Action Representation and Composition for Multimedia Application Scenario Modelling, ERCIM Workshop on Interactive Distributed Multimedia Systems and Services, BERLIN, 3/1996*

[14] Michael Vazirgiannis, Y.Theodoridis, and T. Sellis, *Spatio Temporal Composition in Multimedia Application, In:Proc. of IEEE-ICSE 96 International Workshop on Multimedia Software Development - BERLIN, 3/1996*

[15] Murray W.Goldberg, Sasan Salari, and Paul Swoboda, *World Wide Web- Course Tool: An Environment for Building WWW-Based Courses, Fifth International World Wide Web Conference, May 6-10, 1996, Paris, France*

[16] Murray W.Goldberg, and Sasan Salari, *An update on Web-CT - a tool for the creation of Sophisticated Web-Based Learning Environment, Proceedings of NAUWeb'97 - Current Practices in Web-Based Course Development, June 12-15, 1997, Flagstaff, Arizona*

[17] A.Kameas, and P.Pintelas, *The Functional Architecture and Interaction Model of a GENerator of Intelligent TutORing Applications, J.Systems Software, 1997;36;233-245*

[18] Scriplet Technology, *http://207.68.156.61/msdn/sdk/inetsdk/help/scriptlets*

[19] Dynamic HTML in Netscape Communicator, *http://developer.netscape.com /docs/manuals/communicator/dynhtml/index.htm*

[20] Dynamic HTML without scripting,

[21] WROX Developer's Journal Volume 2.3, *http://www.wrox.com/*

[22] James Rumbaugh, Michael Blaha, William Premerlani, Fredrick Eddy, and William Lorensen, *Object-oriented modelling and design, Prentice hall*

[23] Alfred V. Aho, Jeffrey D. Ullman, *Pronciples of Compiler Design, Addison Wiley*

[24] UML Notation Guide, *http://www.rational.com/uml/html/notation/*

[25] Java Script Guide, *http://developer.netscape.com/docs/manuals/communicator /jsguide4/index.htm*