

An Integrated Architecture for Tightly Coupled Design and Evaluation of Educational Multimedia

Selma Holmquist

Departamento de Cálculo, Facultad de Ingeniería

Universidad de Los Andes

Mérida, Venezuela

N. Hari Narayanan *

Intelligent & Interactive Systems Laboratory

Department of Computer Science & Software Engineering

Auburn University, Auburn, AL 36849, USA

Abstract. Multimedia technology is increasingly being used to create instructional environments for distance education. However, the design of educational multimedia is currently driven more by intuition than by empirically or theoretically derived design guidelines. In the absence of prescriptive design principles, iterative design – a cyclical process of design, test and redesign – becomes critically important for creating effective educational multimedia. This paper proposes a framework for the iterative design of a class of educational multimedia called Hypermedia Educational Manuals. The architecture of an authoring and evaluation platform designed to support this framework and ease the designer's task is presented. Two unique features of this tool are the automatic creation of a system structure definition of the multimedia system being authored, and the automatic incorporation of interaction-logging elements in the system. This is done by an authoring component that assists the designer with the design and implementation of educational multimedia. An evaluation component then parses both the system structure definition and interaction logs created while students work with the multimedia system, in order to generate statistical analyses and graphical visualizations of how students interacted with the multimedia. The integrated architecture of the tool embodies a tight coupling between the design of educational multimedia and the evaluation of its effectiveness. This coupling reduces the time and effort required in repeating evaluation-redesign cycles in order to iteratively refine the initial design. We also briefly describe an experimental demonstration of the utility of this tool.

Keywords: educational multimedia, iterative design, interaction data visualization, authoring, evaluation

* Address all correspondence to this author. Tel: +1 334-844-6312; Fax: +1 334-844-6329; Email: narayan@eng.auburn.edu, Post: 107 Dunstan Hall, Auburn University, Auburn, AL 36849, USA.

1. Introduction

Multimedia authoring tools are increasingly being used to create instructional environments for distance education. The use of multimedia technology allows designers of educational software to use a variety of different media types to convey information. With multiple means such as diagrams, animations, textual explanations, photos, video and audio available for presenting the same information, educational software designers can create different environments to meet the different learning styles of students. However, design guidelines derived from empirical studies of users [e.g., 1] or theoretical models of learning and information comprehension [e.g., 2] for creating multimedia systems that are effective in conveying information have only just begun to be reported in the literature. Effective educational multimedia should present information in such a way that students are able to not only create accurate mental models of the information and but also successfully apply this knowledge to new situations. To assure the quality of educational multimedia, it should be thoroughly evaluated to assure that it fulfills its purpose. In the absence of prescriptive design guidelines, iterative design, in which several cycles of design, evaluation and redesign are carried out to arrive at the final system [3], provides the only assured way of developing an effective product.

A framework for the iterative design of interactive educational multimedia systems is proposed in this paper. This framework is supported by an implemented software tool that provides integrated authoring and evaluation support to the designer. Two unique features of this tool are the automatic generation of a structural description of the multimedia system being designed and the automatic incorporation of interaction data logging elements in the system. This description and interaction logs are then used by an evaluation component to generate statistical analyses and graphical visualizations of student interactions with the manual. This component also generates redesign recommendations based on its analysis of

student actions. Thus, the architecture of this software tool embodies a tight coupling between the design of educational multimedia and the evaluation of its instructional effectiveness.

The iterative design framework that we advocate for educational multimedia systems is presented in the next section. A description of the architecture and features of the integrated software platform that we designed and implemented to support iterative design cycles follows. An experiment designed to provide a preliminary assessment of the utility of this tool is presented in section four. Finally, the concluding section reiterates contributions of this research and discusses related research and future directions.

2. A framework for iterative design of educational multimedia

Evaluation of interactive educational multimedia has to be based on how students interact with the system *and* on how much knowledge they obtain from it. Researchers on human-computer interaction have developed several ways to collect data on user interactions. These include experimenter's observations of user reactions to the system, video-taping users at work, automatically recording users' keyboard and mouse actions, conducting a pre-test (before a user works with the system) and a post-test (after a user has worked with the system), interviewing users, and having users fill out survey questionnaires. Even when only a small group of users are involved in system evaluation, the data thus collected can be quite large. This voluminous data must be analyzed in order to assess the usefulness of the system and decide how to improve its design. Once such redesign decisions are made and implemented, this new version of the multimedia system needs to be re-evaluated to confirm that the design changes do result in improved effectiveness. Sometimes, this redesign–re-evaluate cycle needs to be repeated several times. However, because the data collection and analysis processes are time and resource intensive, changes made to the system after a first round of evaluation are often not re-

evaluated. But without re-evaluating the system, the designer can not be sure that the effects of the changes made are the ones expected. Thus, there is a strong need for automated tool support for evaluation and redesign of multimedia systems, especially in the educational domain where the effectiveness (or lack thereof) of interactive software can have a significant impact on student learning.

Insert Figure 1 about here

Fig. 1 shows the iterative design approach to interactive educational multimedia that we espouse. It involves the following steps. First, a prototype is created and instrumented with automatic interaction data collection capabilities. Second, the prototype is tested with a representative sample of students. During this testing phase, the multimedia system records all student actions in interaction log files. Third, these interaction logs and a description of the educational multimedia system — a system structure definition that contains information about the internal structure and components of the multimedia system — are input to an evaluation system. Fourth, the interaction log data is processed in the context of the system structure definition to generate useful statistical information about students' interactions with, and visualizations of students' navigation patterns within, the prototype. Fifth, this data is interpreted in the context of information on student learning that is gathered during testing (e.g. experimenter observations, pre-tests, post-tests, questionnaires) to arrive at conclusions about deficiencies in the original design of the system and corresponding decisions about redesign. This completes one cycle of iterative design. This can be repeated as many times as necessary (for example, until post-test results indicate that the system helps students reach a desired level of learning). Clearly, this approach is both time and resource intensive. We believe that tool support, which automates as many aspects of this process as possible, is necessary to make the iterative design of educational hypermedia both feasible and attractive to designers of instructional environments.

There are several ways in which such a tool with novel capabilities can be designed. Powerful multimedia authoring environments are currently available in the market. But these provide no support for the evaluation of a multimedia system once it has been created. For instance, no commercial authoring tool provides facilities to make the system being designed automatically collect and save user action logs that can later be used by an evaluation program, despite the fact that interaction logs are a powerful source of data for evaluation purposes. Manually instrumenting an interactive multimedia system to collect log data can be tedious and time consuming. This provides one major opportunity for automation in the authoring phase.

Another area ripe for automation is log data processing during the evaluation phase. Log data can provide valuable insight on how the resources offered by an interactive multimedia system are being used [4]. Interaction logs contain a complete account of the actions users executed on the system, time spent in different parts of it, and the way users navigated through it. A capability to automatically instrument a prototype to collect interaction data during authoring, coupled with a capability to process, analyze and visualize statistical and navigational information extracted from user logs during evaluation, can eliminate the time consuming process of manually collecting and analyzing large quantities of interaction data. This can contribute to significantly reducing the time and resource requirements of iterative design cycles.

However, user interaction logs need to be analyzed in the context of the structure and content of the system being evaluated. This implies that an evaluation system needs not only action and navigation data collected from users, but also information on the interactive objects that users acted upon and the overall structure of the system within which they navigated. Therefore, a third opportunity for automation is the creation of a system description that contains information about the internal structure and contents of the

prototype. An authoring component capable of automatically generating such a system structure/content description, as well as transparently incorporating interaction data collection code in the system being designed, can thus provide a seamless integration between authoring and evaluation. To address these issues, we designed and implemented an integrated platform that combines an authoring component with an interaction data analysis component. Its architecture and features are described next.

3. An integrated authoring and evaluation platform for educational multimedia

3.1 Hypermedia Educational Manuals

In this paper we focus on a particular kind of educational multimedia, which we call Hypermedia Educational Manuals (henceforth referred to as manuals). Such manuals have an underlying structure based on a cognitive model of multimedia comprehension, developed by Narayanan and Hegarty [2]. A manual is made up of sections of information, each of which is designed to help the student achieve a particular learning objective. Each section may be divided further into sub-sections, sub-subsections, and so on. Each section or sub-section presents information using multiple media types and provides various interaction capabilities, which we call actions. For example, the first section of a manual about a complex machine may present a mouse-sensitive schematic diagram of the machine in which each component can be clicked on to trigger a spoken explanation of that component. Other examples of actions are playing an animation or following a hyperlink. Such actions present the student with more information when they are executed. Question sections containing multiple choice questions may be included in the manual. Questions provide learners with a means to organize and assert their newly acquired knowledge [5], and to reflect on their progress. Navigation between sections may be restricted, i.e. learners may be forced to follow a certain path through the sections (linear navigation), or made completely flexible (non-linear

navigation). The purpose of a manual is to help students teach themselves in such a manner that they can, step by step, build a correct mental representation of the information presented by the manual.

3.2 The Authoring Component

Designing hypermedia and multimedia systems has been an active area of research and commercial development. The emergence of hypermedia design models such as the Amsterdam Hypermedia Model [6], Object Oriented Hypermedia Design Model [7], and others has encouraged research towards authoring systems with which to implement these models. Two examples of such authoring environments are CMIFed [8], which supports structure-based composition of multimedia presentations and the synchronization of media items, and IMMPS [9], an interactive multimedia presentation development system in which presentations are based on message-passing among windows. These systems provide support for the implementation of temporal and spatial constraints between the different media types that make up a multimedia system. In addition to such research prototypes that provide powerful authoring capabilities to designers, several general-purpose commercial authoring tools are also currently available. Some examples of such commercial systems are ToolBook™ by Asymetrix for Windows-only multimedia production, AuthorWare™ by Macromedia, a high-end tool built for complex, cross-platform training applications, and Director™ by Macromedia, considered by many to be the best tool on the market for multimedia producers.

These authoring tools provide designers with the ability to create multimedia and hypermedia systems, but provide no support for the evaluation of systems after they have been created. In order to evaluate the effectiveness of an educational multimedia system one requires, at the very least, three kinds of information. The first is information on how students interacted with the system – which sections did

they view or not view, how did they navigate through the system, and what actions they executed. The second is information about the structure (syntax) and content (semantics) of the multimedia system. This descriptive information provides the context to meaningfully interpret student actions and navigation patterns. For example, suppose it is found that students are repeatedly going to a section and pressing a button that plays an audio file explaining the operation of a component in a manual explaining a machine. This may warrant redesign suggestions such as presenting that information visually rather than aurally (since the transient nature of audio may explain the repeated action) and making it more directly (i.e. with fewer actions) accessible from other sections. In order to come up with such redesign suggestions, student actions need to be interpreted in the context of the system's structure (i.e. in which part of the system does the play button appear?) and semantics (i.e. what does the button do?). The third is information about how much students learned from interacting with the system. Any tool intended to aid the designer in evaluation needs to be able to automatically collect the first two kinds of information and utilize these in its computations. The third type of information, on student learning, depends on the domain and will be different for different multimedia systems. Evaluating student learning is not an activity that can be fully automated. However, evaluating students' interactions with educational multimedia, in the context of the structure and contents of the multimedia system, is an activity that can be significantly aided by interaction data collection and analysis software. In order for such software to generate redesign recommendations, it needs access to a description of the structure and contents of the multimedia system being evaluated as well as logs of students' interactions and navigation actions.

Without automated tool support, the designer will need to manually create a description of the multimedia system for such evaluation to take place. Furthermore, to collect information on students' actions, the designer will need to place, in the event handler of every interactive object, code necessary for logging the corresponding action. Navigation action logging code will have to be added to all sections of the manual and all navigation objects (e.g. "forward" and "back" buttons). Depending on the size and

complexity of the multimedia system, manually including the code for action and navigation log generation can be both error prone and time-consuming. The designer has to also ensure consistency between log information produced by each interactive element, and information about that interactive element in the system description. Clearly, manually instrumenting the system is very cumbersome. This indicates the necessity of an authoring component that, while allowing the designer to easily create educational multimedia prototypes, *also* automatically generates information about the structure and content of the system being authored *and* automatically incorporates action and navigation logging code with every interactive object and interactive media element in the system. We distinguish between an interactive object, which a user can interact with but does not carry educational information (e.g., a button or a menu item), and an interactive media element, which presents information as well as allows a user to interact with it (e.g., a map that a user can click to zoom and pan). Any good commercial authoring tool will allow the designer to easily create educational multimedia. However, these other capabilities are lacking in commercial tools. Therefore, we extended a general-purpose commercial multimedia authoring tool (Macromedia Director™) to create an Authoring Component with all the above capabilities, as part of an integrated platform.

This Authoring Component allows a designer to do the following.

- Create sections and subsections of a manual.
- Create interactive objects (e.g. a button) and interactive media elements (e.g. a clickable picture) of various kinds.
- Specify user actions and system responses for the interactive objects and media elements.
- Specify the kind of log information interactive objects and media elements will generate when acted upon by a user.
- Specify which interactive objects and media elements belong to which sections or subsections.

- Import content (media elements such as pictures and animations) into each section and subsection.

Meanwhile the Authoring Component automatically generates the following.

- Code associated with each interactive object and media element to acquire and save log information when a user acts upon it.
- Code associated with each section and subsection to create and save navigation log information when a user visits it.
- Code needed to manage (create, open, write and close) interaction log files.
- A navigation map, which is a mouse-sensitive graphic of the sectional structure of the multimedia system so that a user may visit any section by clicking on the graphic, if the designer requests such a map.
- A system structure definition that contains information about structure and content of the authored system.

By extending a popular commercial multimedia authoring platform to create this Authoring Component, we avoided the duplication involved in trying to build yet another multimedia authoring tool from the ground up. Instead, it becomes possible for designers to create the basic manual structure using the Authoring Component while maintaining access to the extensive media generation and editing capabilities of Macromedia Director™ for creating and importing content. With the Authoring Component, the designer can create different sections and subsections of the multimedia system, together with the interactive objects and media elements that will appear in each section. Then, using Director's capabilities, the designer can add animations, transitions, audio files and any other multimedia element. A designer would first create the underlying manual structure and interactive objects using the Authoring Component so that the system is automatically "described" and "instrumented", and then flesh out the

manual by exploiting the extensive multimedia editing and orchestration capabilities of Director to create and import content.

Because the Authoring Component was created using Director and its programming language Lingo, it is useful at this stage to explain terminology specific to Director before further describing the system.

Director creates multimedia presentations called *movies*. A movie is a procedural specification of a multimedia presentation, including the interactive objects and media elements that the presentation consists of. The interactive objects and media elements of a movie are called *cast members*. There are many different kinds of cast members such as bitmaps, text, and sound. The *stage* is a data structure that shows what is happening in a movie (which interactive objects and media elements are visible) at a particular time. A representation of a cast member on the stage is called a *sprite*. A sprite contains information on when, where and how a cast member appears on the stage. Director uses Lingo to control different aspects of a movie. Lingo code consists of one or more statements grouped in a function like structure called a *handler*. Handlers can be added to a movie, a media element in a movie, a screen in a movie (or frame) and to many other multimedia elements.

The Authoring Component creates, using Lingo, movies that become the sections of a manual. In this sense, the Authoring Component is implemented as a movie that creates other movies. Its principal component is a Main Movie that contains Lingo objects providing a set of functionalities described in the following paragraphs. It also contains a Section Shell Movie that is used by the Main Movie to create manual sections and subsections, and a Manual Shell Movie that generates a movie to coordinate the visual presentation of the entire manual. The Authoring Component also includes a set of small movies that create pop-up windows that provide information to the designer or request information from him/her during the authoring process.

3.2.1 User interface

The top-level menu of the Authoring Component contains three options. The *File* option allows creation of a new manual, opening an existing one, and saving a modified one. The *Section* option allows creating a new section to be added to the manual, creating subsections of a section, opening an existing section or subsection, adding interactive objects and media elements to a section or subsection, and saving changes made to it. The *Navigation* option creates a mouse-sensitive navigation map for the manual.

3.2.2 Creating the system structure definition

The Authoring Component automatically generates a file, called a “system structure definition”, containing a description of the structure and content of the manual. In order to do this, it keeps track of the sections, subsections, interactive objects and media elements that the designer is creating. It then generates a tree representation of the manual. Each node of this tree is an object with properties *parent*, *name*, *number*, *subsections*, *elements* and *description*. *Parent* points to the parent node and *subsections* point to child nodes. *Name* and *description* are descriptive strings obtained interactively from the designer. *Elements* are interactive objects and media elements contained in that section or subsection. *Number* is a numerical identifier (such as 1-5 for the fifth subsection of section 1) that indicates the location of the section or subsection within the manual. In this way, the entire manual is described by a recursive tree structure, as shown in Fig. 2. At the end of a session (when the designer exits the program), the Authoring Component translates this tree structure into a set of nested lists and writes these into a system structure definition file.

Insert Figure 2 about here

3.2.3 Adding interaction data logging code

Interaction data needs to be collected whenever a student enters a section or a subsection, and whenever he or she interacts with an object or media element. For this, the Authoring Component exploits the fact that Director provides predefined handlers that are executed whenever a particular event occurs.

Each section of the manual is implemented as a movie. In Director, the *StartMovie* handler executes whenever a movie starts playing. Therefore, the Authoring Component creates, for each section of a manual, a *StartMovie* handler and adds to this handler additional code necessary for data logging.

Another predefined handler in Director is the *EnterFrame* handler. It is executed whenever a movie enters a frame with an associated *EnterFrame* handler. Therefore, the Authoring Component finds the first frame of each subsection of the manual and creates for each of these frames an *EnterFrame* handler with additional data logging code for that subsection.

Whenever the designer elects to add a new interactive object or media element to the manual, he or she is prompted for a name of the element and asked whether it needs to be logged. If so, the Authoring Component asks additional questions regarding the event (mouse-up or mouse-down) that will generate the log and the type of action the interactive object supports (see Fig. 3). Then it automatically creates a data logging handler for the new element and opens a media editing window for the designer to create or import content.

Insert Figure 3 about here

The net result is that whenever a student interacts with a manual, a log file containing triples representing each action executed is produced. An action triple contains the type of the action, a number assigned to

the action, and the time at which it was executed. The type of an action describes its semantics. Action type includes *Section*, *Subsection*, *Navigation*, and the types shown in Fig. 3 (b). The number of an action represents its location within the nested sectional structure of the manual. For instance, two consecutive triples (*Navigation*, 4-6, 15.55) and (*Section*, 5, 15.65) appearing in a log file indicate that the student executed the sixth action in section four at 15.55 seconds after he or she started interacting with the manual, which was a navigation action, and then entered the fifth section of the manual 0.1 seconds afterwards.

3.2.4 Generating a navigation map

A navigation map is a mouse-sensitive graphic depicting the structure of the manual. It appears on every screen of the manual, allowing students to easily navigate to different sections of the manual. Each button of a navigation map, depicting a section or subsection, can be in one of three states: active, inactive, or pressed. When a section is active (e.g. *The Problem*, *Algorithm*, *Pointers*, *Records* and *Exit* buttons in Fig. 4), clicking on it will take the student to that section. A pressed button indicates the section the student is currently in (e.g. *Questions* button in Fig. 4). An inactive button (e.g. *Program* button in Fig. 4) indicates a section in the manual the student is not yet allowed to visit because of navigation restrictions.

Insert Figure 4 about here

To create a navigation map for a manual, the Authoring Component needs to know the order of its various sections and any navigation restrictions that the designer wants to impose. The section order is obtained from the designer through a dialog box in which all sections are listed and the designer can drag sections in the desired order. To obtain navigation restrictions for each section in the manual, the

Authoring Component asks the designer to specify all sections that will become accessible to a student once he or she has visited the current section. Once both these types of information have been obtained, the Authoring Component automatically creates a navigation map that will appear on every screen of the manual, and adds to this map all necessary handlers (Lingo code) to enforce the navigation restrictions specified by the designer and to log interaction data.

3.3 The Evaluation Component

The evaluation of an educational multimedia system should be based on how students interacted with it *and* what they learned from it. The pre-test and post-test paradigm from experimental psychology can be applied to gauge the extent of student learning. Collection and analysis of interaction logs will provide a picture of how students interacted with the system. It is this latter process that can be automated so that the designer can be presented with understandable explanations and visualizations of student interactions and navigation patterns. He or she can then correlate interactions with learning (or lack thereof) in order to evaluate the efficacy of the educational multimedia system being evaluated.

The second part of the integrated architecture described in this paper is therefore an Evaluation Component that parses the system structure definition created by the Authoring Component and the interaction logs generated when students interact with the multimedia manual. The resulting data is analyzed and presented to the designer as statistics and visualizations. The designer can view both graphic and numeric representations of information on how students interacted with the manual. This information includes actions they executed, their navigation patterns, and their answers to multiple-choice questions if such questions were included in the manual. The designer can not only see how each student interacted with the system, but also obtain statistical and analytical information on groups of

students, and semantic interpretations of their navigation patterns. We first describe the navigation pattern extraction and analysis capabilities of the Evaluation Component, followed by a discussion of the various explanations and visualizations that it generates for the designer.

3.3.1 *Extracting and analyzing navigation patterns*

The Evaluation Component bases its student navigation analyses on the extraction and interpretation of navigation patterns obtained from the interaction log files. These files contain all actions and navigation steps executed by each student in the form of a string of triples of the form (*action section-identifier time-stamp*). A pattern extraction algorithm (shown in Fig. 5) searches these strings for patterns that occur multiple times. Those patterns that occur above a threshold frequency (specified by the designer) are selected as “navigation patterns of interest”. The list of these patterns together with the number of times they were executed form the output of the pattern detection algorithm. Fig. 6 shows how the Evaluation Component displays the patterns it detects. This figure illustrates a navigation pattern that occurred 4 times. The pattern starts in the first subsection of Section 2, from which the student(s) went to the main page of Section 1, and from there to subsections 5 and 4.

Insert Figure 5 about here

Insert Figure 6 about here

An inference algorithm analyzes *both* the raw log files and patterns found by the pattern detection algorithm. It uses production rules to generate recommendations on how to improve the design of the manual. These rules encode a designer’s assumptions of relations between particular student actions and

possible flaws in the manual design. These assumptions, coded into production rules in Lisp with pre-conditions and post-conditions, are really heuristics that need to be tried out, refined or rejected by the designer as the design-redesign process is under way. The pre-condition of a rule may be, for example, a match with a particular navigation pattern. The post-conditions are redesign recommendations. The rules consider diverse information from the interaction logs. For example, different log data such as the time spent by the students in their first visits to sections, the number of students that executed certain actions, parts of the manual that students never viewed, etc., as well as the extracted navigation patterns, are used in the pre-conditions of rules. The post-conditions of all rules that fire during an analysis session are added to a list. This list of suggested redesigns is displayed to the designer. The recommendations may suggest changes to media types, information content, or structure of the manual. The final choice as to which of the redesigns suggested by the Evaluation Component are implemented rests with the designer.

There are three kinds of rules: behavior-based rules, media-based rules and question-based rules.

Behavior-based rules analyze the behavior of students (their navigation patterns, the actions they executed, and the time they spent in sections and subsections) in order to detect potential problems and suggest redesigns. An example of such a rule, translated into English, is:

“If students are seen to go back and forth between one information presentation element in section A and another in a following section B repeatedly, the corresponding information in section A should be moved to, replicated in, or made directly accessible from Section B”.

Media-based rules analyze how students interacted with media such as audio clips and animations. An example of such a rule, translated into English, is:

“If students are playing an audio clip more than a certain number of times (threshold set by the designer), segment the audio into shorter files if it is too long, improve the quality of the audio if it is poor, improve the comprehensibility of the message by rewording if it is too difficult to understand, or consider replacing audio with on-screen text”.

Question-based rules analyze how students answered multiple-choice questions, which sections and subsections they visited before answering, and the correctness of the answers provided, in order to provide recommendations about redesigning questions and sections containing information relevant to questions. An example of such a rule, translated into English, is:

“If students are answering a question incorrectly, then going to section or subsections relevant to the question and returning to again incorrectly answer the question, in a repeating pattern, consider revising the content of those sections to make them easier to understand”.

The system currently contains 19 rules, 10 of which are behavior-based rules, 4 are media-based rules and 5 are question-based rules. A rule typically recommends several possible changes, but the choice of which, if any, to implement rests with the designer. The system maintains a separate file or rule-base so that rules can be easily added, deleted or modified. At present, automatic consistency checking to see whether a newly added rule or a revised rule is consistent with other rules in the rule-base is not done. This is part of our future research.

3.3.2 The designer's interface

The interface of the Evaluation Component provides a variety of ways to view the interaction and navigation data, which are organized into a top-level menu shown in Fig. 7. The designer first chooses to view information on an individual student or on a group of students, as well as the log file(s) and system definition file to be used in the analysis. Then he or she can elect to conduct any of several data analyses and view different data visualizations. The available analyses and visualizations are listed below and described in more detail in the following paragraphs.

Insert Figure 7 about here

- **General Information:** provides information on the manual and on rules that the Evaluation Component uses to analyze navigation patterns. The designer can view the following information.
 - Sectional structure of the manual.
 - Navigation restrictions.
 - Rules used to analyze navigation patterns.

- **Information on Sections:** provides information on how students interacted with the sections of the manual. The designer can view the following information.
 - (Sub)Sections seen by the student(s).
 - Time spent in each (sub)section.
 - Actions executed in any (sub)section.
 - Percentage of time spent in a section with respect to the total time spent with the manual.
 - Number of times the student(s) visited a section.

- **Information on Navigation:** provides information on the way students navigated through the manual. The designer can view the following information.
 - A graphical representation of the navigation paths.
 - Frequency with which a visit to a section was followed by a visit to another section.
 - Frequently occurring navigation patterns.
 - A rule-based analysis of the navigation patterns.

- **Information on Questions:** provides information on how students interacted with multiple-choice questions in the manual. The designer can view the following kinds of information.
 - The answer given to each question by a student, whether it was correct or not, the number of attempts before getting it correct, and sections viewed before each answer. For a group of students, the percentage of students that answered each question and the percentage of students that answered correctly on the first try are also provided.

The designer can view the structure of the manual in the form of a hierarchical tree layout. A graphical representation of the navigation restrictions of a manual can also be viewed. In this display each section is represented as a node. A connection between two nodes indicates that a student can navigate from one node to the other. Fig. 8 shows such a display, indicating a manual with 7 sections that a student is restricted to visiting in order (except section 7). The figures that follow show data collected on this particular manual.

Insert Figure 8 about here

The designer can view specific actions of a student and statistics on these actions in any section or subsection of the manual. Fig. 9 shows a sample of section visit statistics. Furthermore, one can choose to view specific actions executed by a student during a particular visit to a section or subsection. Fig. 10 shows such a display, of the actions of a student “SUNY” during the second visit to section 3 containing multiple choice questions.

Insert Figure 9 about here

Insert Figure 10 about here

The time chart visualization option displays a pie chart indicating the percentage of the total time a student spent on each section (Fig. 11). The frequency graph display shows a bar graph showing the

number of times each section was visited (Fig. 12). The Evaluation Component computes and displays the total time each individual student spent with a manual, and the mean amount of time a group of students spent with a manual together with the standard deviation. The designer can also view the percentage of students (from a group) that executed any specific action in a manual (Fig. 13).

Insert Figure 11 about here

Insert Figure 12 about here

Insert Figure 13 about here

The Evaluation Component can display the navigation of an individual student or a group of students in one of three ways: as a diagram, as a table, or as a graph. The navigation diagram (Fig. 14) shows the paths between sections a student followed when viewing the manual. It also shows the number of times the student followed each path.

A navigation table is shown in Fig. 15. The number in a cell of the table indicates the number of times a student went from the section in the corresponding row to the section in the corresponding column. If

group analysis is chosen, each table cell will instead show the percentage of students that navigated from the section on a row to the section on the corresponding column.

The navigation sequence graph shows the complete navigation path followed by a student (Fig. 16). It is different from the navigation diagram not only in the way the navigation path is presented, but also in the fact that a section visited n times, will appear n times in the graph. The navigation sequence graph also provides information on the amount of time the student spent in each visit to a section.

Insert Figure 14 about here

Insert Figure 15 about here

Insert Figure 16 about here

3.4 Integrated implementation of the Authoring and Evaluation Components

The overall architecture of the integrated platform is shown in Fig. 17. Several different software packages and programming languages were used to develop the integrated tool. The Authoring Component was developed using Macromedia DirectorTM and its programming language Lingo. This application was selected because of its widespread use and the programming flexibility provided by

Lingo. The computation layer of the Evaluation Component, responsible for parsing the system structure definition and log data files, extracting patterns, doing numeric computations, applying production rules and generating visualizations, is implemented in Macintosh Common Lisp. The designer's interface was created using FaceSpanTM. It is a Graphical User Interface (GUI) development environment for Macintosh computers. AppleScript, the scripting language of the Macintosh Operation System, and AppleEvents, the system level messaging facility between applications, were used to provide communication links between the interface and computation layers of the Evaluation Component. Communication between the Authoring and Evaluation Components is achieved through file transfer. Thus, it is possible for authoring and evaluation to take place at different locations, with file transfer done through a network.

Insert Figure 17 about here

4. An experimental demonstration

We designed and conducted a pilot study to test the effectiveness of the integrated authoring and evaluation platform for educational multimedia. In this experiment, a prototype manual was created with the Authoring Component and tested with a group of students. This prototype was then redesigned, based on the interaction data analyses, visualizations and recommendations provided by the Evaluation Component which were interpreted in the context of pre-test and post-test results of the first group of students. The redesigned manual was re-evaluated using a second group of students.

The experiment was conducted in six stages. In the first stage a prototype of a Linked List Hypermedia Educational Manual was created. It was designed to instruct sophomore-level computer science undergraduates about the concepts of records, pointers and linked lists, and how to implement find, insert and delete routines for Linked Lists in the programming language Ada. In the second stage of the experiment, a first group of 13 students from the course “Fundamentals of Computer Science II” at Auburn University took a pre-test on the topic of Linked Lists, interacted with the manual, and then completed a post-test and a user satisfaction questionnaire. In the third stage of the experiment, data collected from this group was evaluated. This evaluation included assessing the students’ performance in the tests, considering their opinions from the questionnaire, and analyzing their interactions using the Evaluation Component. The fourth stage involved redesigning the manual, based primarily on the visualizations and recommendations generated by the Evaluation Component. In the fifth stage of the experiment, the revised manual was tested with a second group (which matched in level and experience with the first group) of 10 students from the same class. The same questionnaires and tests were used. In the sixth stage, data collected from this second group (pre-tests and post-tests, questionnaires and interaction logs) were again analyzed with the help of the Evaluation Component and results were compared to those obtained in stage three.

The results of this experiment showed that the second group obtained a better understanding of Linked List topics than the first group. The average pre-test and post-test scores that students in the first group obtained were 19.15% (standard deviation 26.25%) and 75.85% (standard deviation 10.29%) respectively. Clearly, their knowledge increased after interacting with the first prototype of the manual. The low average score and high standard deviation of the pre-test indicated that the first group was very uneven regarding their prior knowledge of Linked List topics. The substantial decrease in standard deviation from the pre-test to the post-test indicates that students started out with different levels of prior knowledge about Linked Lists, but their knowledge reached a much more even level after working with

the prototype. The average pre-test and post-test scores that students in the second group obtained were 6.9% (standard deviation 17.46%) and 86.3% (standard deviation 4.57%) respectively. Clearly, this group of students' knowledge increased after interacting with the revised prototype. The low average score and high standard deviation of the pre-test indicated that the second group was also very uneven regarding their prior knowledge of Linked List topics. The substantial decrease in standard deviation from the pre-test to the post-test indicates that students started out with different levels of prior knowledge about Linked Lists, but their knowledge reached a much more even level after working with the revised prototype.

More importantly, the average post-test score of the second group was 10.45% higher than that of the first group. This difference is statistically significant (as indicated by the F-test results $F(1,22)=8.01$, $p<0.01$). The second group's test score improvement (the difference between pre-test and post-test scores) after working with the revised manual was 22.71% higher than the corresponding improvement shown by the first group after working with the original manual. This difference is also statistically significant (as indicated by the F-test results $F(1,22)=5.79$, $p<0.025$). Thus, this experiment demonstrated that the authoring, log data capturing, analysis, visualization and interpretation capabilities provided by the integrated tool supports the evaluation and redesign phases of iterative design, and thus improves the effectiveness of iterative design cycles. A complete account of the experimental procedure and results can be found in [10].

5. Conclusion

The central focus of this research is to design and implement a tightly coupled authoring and evaluating environment for educational hypermedia. The phrase "tightly coupled" refers to the fact that the

authoring component creates prototypes that include (transparently to the designer) necessary provisions for evaluation. These provisions include supporting the creation of multimedia systems with a well-defined structure, automatically generating a file containing a structural description of the system, and automatically instrumenting the system to capture interaction log information. The evaluation component is tightly coupled to the authoring tool in the sense that it can analyze the user interaction information generated by the prototype in the context of its structural description provided by the authoring tool, present the results to the designer, and suggest potential changes to the system.

We espouse an iterative design framework for educational multimedia. In this framework, an initial design is created and then refined through several iterations of an evaluate-redesign cycle. This is similar to the task-based hypermedia design approach proposed by Paterno and Mancini [11] in that both are intended to enhance the usability and effectiveness of the resulting system when compared to purely system-oriented design approaches. The contribution of our research is to enhance iterative design and refinement of educational multimedia through instrumented authoring and automatic, formative evaluation. An integrated platform has been designed to help the designer tightly couple the twin processes of authoring multimedia prototypes and evaluating them. Automated tool support thus provided helps speed up designing and evaluating prototypes of educational multimedia. This improves the practical feasibility of the iterative design approach, and allows evaluation/redesign cycles to be carried out more efficiently.

The purpose of educational multimedia is to effectively convey information to learners. Our research has shown that changing the structure, navigation and contents of an educational manual in iterative design cycles that tightly couple authoring and evaluation can indeed result in improved instructional effectiveness. Furthermore, the semi-automated support for this process that the integrated authoring and evaluation platform provides improves the efficiency of iterative design cycles. This is achieved by

reducing the manual effort involved in various stages of design and evaluation, i.e., instrumenting an authored system to collect log data, generating a structural description of the authored system to provide a context for data analyses, analyzing the data, visualizing the data, extracting interesting navigation patterns, and interpreting those to generate redesign recommendations.

Though our target domain is educational multimedia, we believe that this approach is applicable to interactive multimedia information presentation systems in general. The integrated authoring and evaluation tool can be applied to any hierarchically structured multimedia system in a task domain in which the designer can predict productive interaction patterns, detect unproductive (or problematic) ones, and quantify the effectiveness of a design. This latter requirement allows the performance of redesigns to be measured and compared against a target (e.g. an expected comprehension or recall score in post-tests) to decide when to conclude iterative design.

Another contribution is to automating the analysis of data on user interactions with multimedia.

Analyzing and presenting information extracted from user interaction logs in multiple and interrelated ways provide the designer with a better understanding of how the various resources offered by a multimedia system are accessed by users. Furthermore, intelligent analysis of navigation patterns generates recommendations on changes to the structure and media elements of the multimedia system being tested.

Prior research has addressed automated log data analysis. Yamada and Sugita [12] used log data to determine the number of users that reached different levels of information in a hypermedia system. This information was used to determine the value of a hypermedia evaluation metric. Lawless and Kulikowich [13] used log files to analyze users' navigational paths through a hypertext environment. Information from the log file was manually processed to detect groups of users with similar navigation. This research

helped identify navigational profiles of hypertext users. In a similar study, Recker [14] created clusters of users with similar navigation strategies. Lee and Heller [4] described the use of keystroke log files to identify usage patterns of an interactive multimedia system installed in a museum. The authors found, by analyzing the log data, that film and video resources were not being accessed. Siochi and Ehrich [15] developed a technique called Maximal Repeating Patterns to extract patterns of usage from recorded logs. Maximal Repeating Patterns are substrings that occur at more than one position in a string and are of maximal length possible. They are extracted from log files and reviewed by designers to find problems such as ambiguous or missing functions in a user interface. Chen and colleagues [16] developed algorithms to extract frequent traversal patterns from log data of Web navigation. Chavero and colleagues reported [17] a tool called Diagram Generator that graphically presented the paths and times of students navigating through an instructional hypermedia program. This tool displayed sections of the hypermedia system as nodes of a graph, and drew circles around the nodes with radii proportional to the amount of time students spent in each section. Navigation paths were shown as lines connecting the nodes. By comparing the navigation paths and viewing times of different students together with their performance on subsequent tests, designers could draw conclusions on ways to improve the hypermedia system.

In each of these cases, log file analysis is restricted to either deriving a specific type of information, or extracting and displaying traversal patterns. Log data contains rich information on navigation, time, preferred actions, ignored actions, and much more. Therefore, one goal of our research has been to advance the state of the art in the area of automatic log data processing. Our research goes farther in exploiting the richness of interaction logs by presenting the derived information (which includes temporal and action information in addition to navigation patterns) in a variety of interrelated formats (e.g. tables, graphs, charts, and patterns). Such multiple views that highlight interrelations better help the designer in drawing conclusions about the efficacy of the system under test. This is coupled with rule-

based analysis of navigation patterns to generate recommendations on how to improve the system. All of these serve to provide the designer with a much richer palette of information and a comprehensive picture of how users interacted with a multimedia system.

Putting the integrated authoring and evaluation tool to prolonged use in iterative design cycles of several multimedia design projects is the primary focus of future work. Other future research avenues include refining the pattern interpretation rules, supporting the designer in generating and customizing new rules by providing a natural language interface to the rule base, and incorporating automatic consistency checking.

Acknowledgments: This research was supported by the Office of Naval Research under contract N10014-96-11187 to Auburn University. We thank the anonymous reviewers for providing comments, which greatly helped improve the quality of presentation.

References

- [1] P. Faraday, A. Sutcliffe, Providing advice for multimedia designers, in: Proceedings of the ACM Human Factors in Computing Systems Conference (CHI'98), ACM Press, New York, 1998, pp. 124–131.
- [2] N. H. Narayanan, M. Hegarty, On designing comprehensible interactive hypermedia manuals, *International Journal of Human Computer Studies* 48 (1998) 267–301.
- [3] D. Hix, H. R. Hartson, *Developing user interfaces: ensuring usability through product & process*, John Wiley & Sons Inc., New York, 1993.
- [4] S. Lee, R. S. Heller, Use of a keystroke log file to evaluate an interactive computer system in a museum setting, *Computers and Education* 29(2/3) (1997) 89-101.

- [5] C. Trudel, S. J. Payne, Self-monitoring during exploration of an interactive device, *International Journal of Human-Computer Studies* 45 (1996) 723–747.
- [6] L. Hardman, D. C. A. Bulterman, G. van Rossum, The Amsterdam Hypermedia Model: Adding time and context to the Dexter model, *Communications of the ACM* 37(2), (1994) 50–62.
- [7] G. Rossi, D. Schwabe, A. Garrido, Designing computational hypermedia applications, *Journal of Digital Information* 1(4) (1999) Electronic journal available at <http://jodi.ecs.soton.ac.uk>.
- [8] G. van Rossum, J. Jansen, K. S. Mullender, D. C. A. Bulterman, CMIFed: A presentation environment for portable hypermedia documents, in: *Proceedings of the ACM Multimedia '93 Conference*, ACM Press, New York, 1993, pp. 183–188.
- [9] T. K. Shih, R. E. Davis, IMMPS: A multimedia presentation design system, *IEEE Multimedia* 4(2) (1997) 67–78.
- [10] S. Holmquist, An integrated architecture for tightly coupled design and evaluation of hypermedia educational manuals, Ph.D. Thesis, Auburn University, 1999.
- [11] F. Paterno, C. Mancini, Engineering the design of usable hypermedia, *Empirical Software Engineering Journal* 4(1) (1999) 11–42.
- [12] S. Yamada, J.-K. Hong, S. Sugita, Development and evaluation of hypermedia for museum education: Validation of metrics, *ACM Transactions on Computer-Human Interaction* 2(4) (1995) 284–304.
- [13] K. A. Lawless, J. M. Kulikowich, Domain knowledge, interest, and hypertext navigation: A study of individual differences, *Journal of Educational Multimedia and Hypermedia* 7(1) (1998) 51–69.
- [14] M. M. Recker, A methodology for analyzing students' interactions within educational hypertext, in: *Proceedings of the World Conference on Educational Multimedia and Hypermedia (ED-MEDIA '94)*, Association for the Advancement of Computers in Education, Charlottesville, VA, 1994, CD-ROM.

- [15] A. Siochi, R. W. Enrich, Computer analysis of user interfaces based on repetition in transcripts of user sessions, *ACM Transactions on Information Systems* 9(4) (1991) 309–335.
- [16] M.-S. Chen, J. S. Park, P. S. Yu, Efficient data mining for path traversal patterns, *IEEE Transactions on Knowledge and Data Engineering* 10(2) (1998) 209–220.
- [17] J. C. Chavero, J. Carrasco, M. A. Rossell, J. M. Vega, A graphical tool for analyzing navigation through educational hypermedia, *Journal of Educational Multimedia and Hypermedia* 7(1), (1998) 33--49.

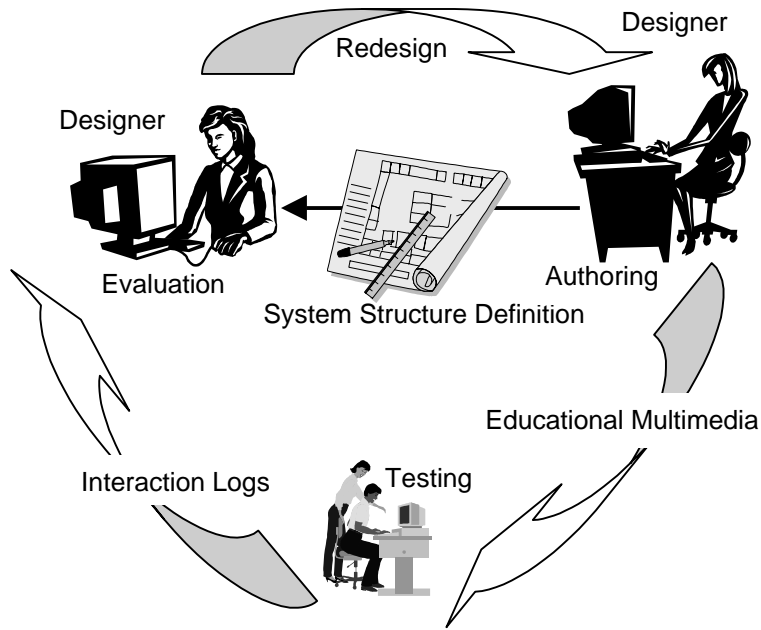


Fig. 1. Iterative design framework

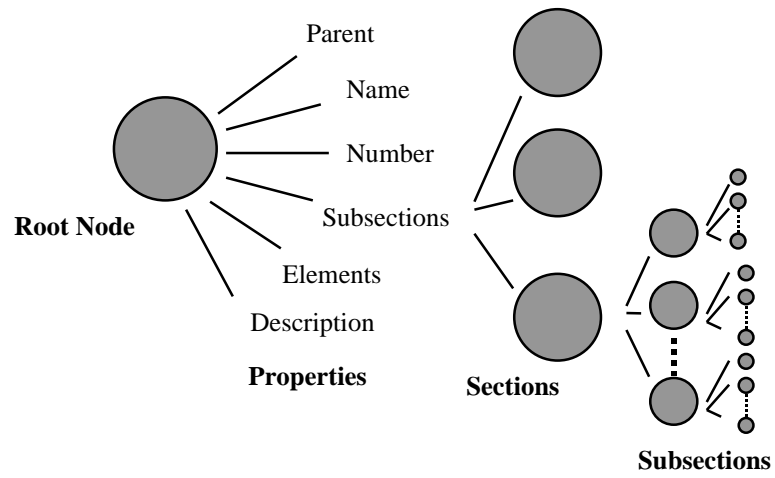


Fig. 2. Tree representation of a manual

(a)

Enter a name for the new media:

Provide logging data code for this element?

yes
 no

(b)

Collect log data on:

mouseUp
 mouseDown

Type of action:

Action Label
 Animation Navigation
 Audio Text
 Diagram Video

Fig. 3. Designer prompts for a new interactive object

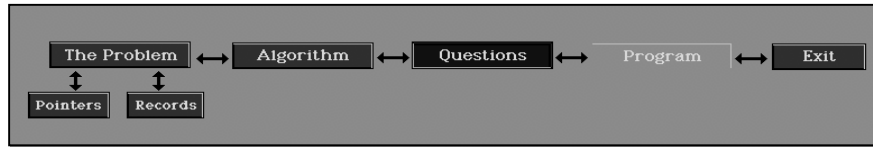


Fig. 4. A navigation map

Step 1: Segment each user action string based on “first visits” to sections; i. e., each substring thus obtained will start with a first visit to a section or subsection, and end with a triplet that precedes the next visit to a section or subsection that has not been visited before. Let S be the set of substrings obtained from all users as a result of this segmentation.

Step 2. $P = \emptyset$

For each string $s \in S$, do

For each possible pattern p consisting of at least 2 consecutive triples $\in s$ do

If $p \notin P$ then

Count = the number of occurrences of p in all strings $\in S$

If count > a threshold set by the designer then

Add p with this count to P

End if

End if

End for

Delete s from S

End for

Fig. 5. Pattern detection algorithm

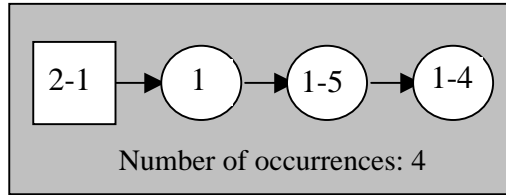


Fig. 6. A navigation pattern

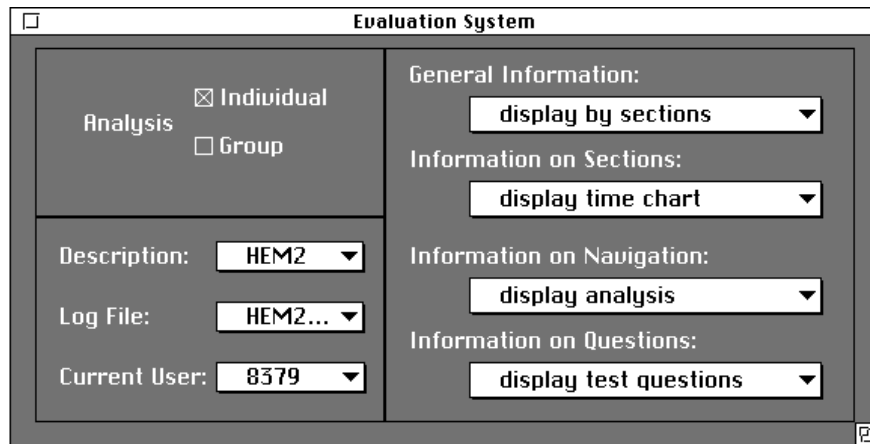


Fig. 7. Top-level menu of the designer's interface

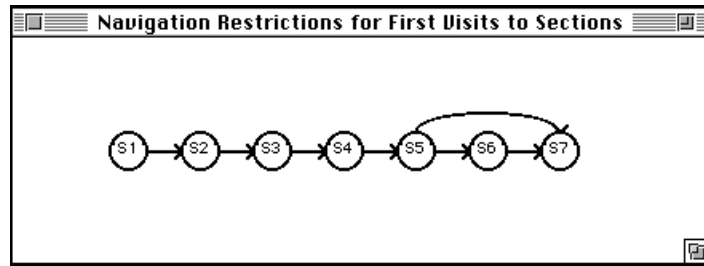


Fig. 8. A visualization of navigation restrictions

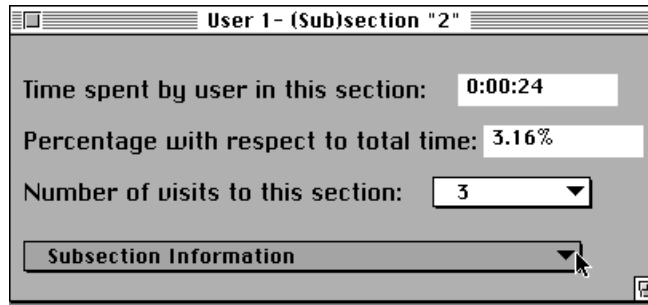


Fig. 9. Section visit statistics

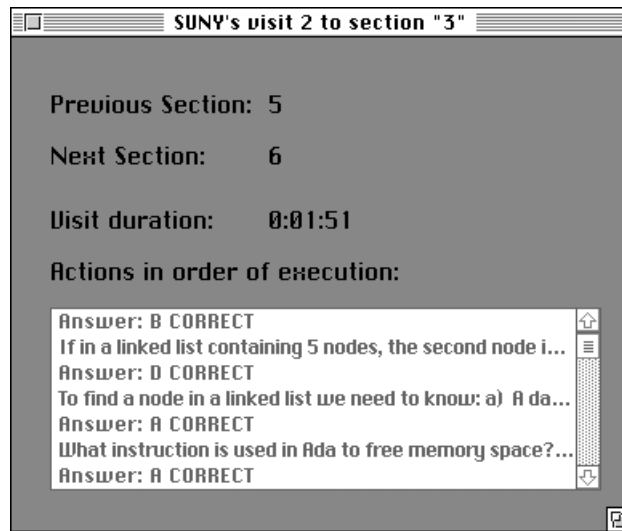


Fig. 10. User actions during a section visit

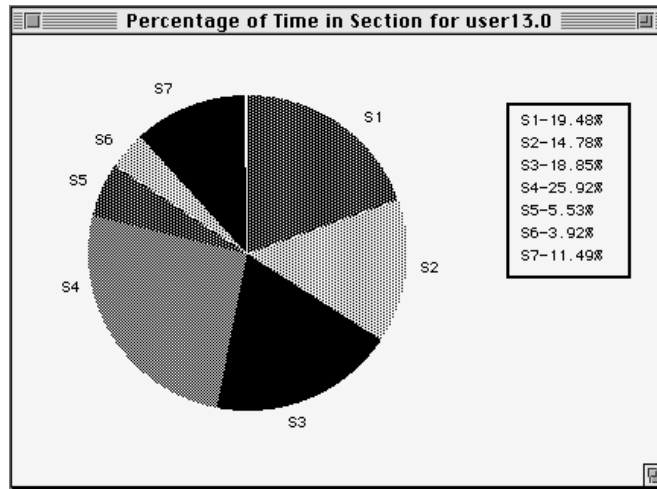


Fig. 11. A time chart

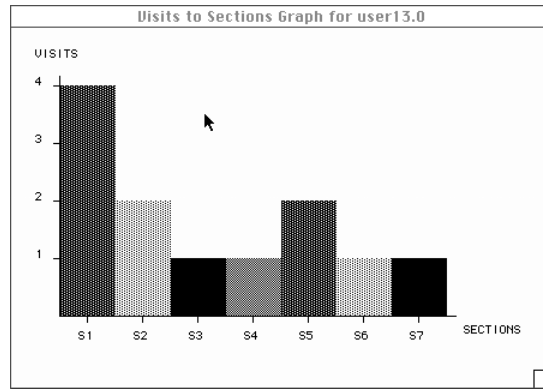
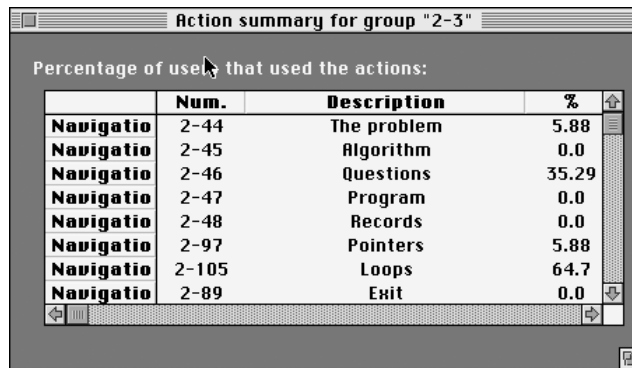


Fig. 12. Section visit frequencies



The screenshot shows a window titled "Action summary for group '2-3'". Below the title bar, the text "Percentage of use that used the actions:" is displayed. A table follows, listing various actions with their corresponding numbers, descriptions, and percentages. The table has four columns: "Num.", "Description", and "%". The first column is partially obscured by the word "Navigatio" repeated in each row. The table includes a vertical scrollbar on the right side and navigation arrows at the bottom.

	Num.	Description	%
Navigatio	2-44	The problem	5.88
Navigatio	2-45	Algorithm	0.0
Navigatio	2-46	Questions	35.29
Navigatio	2-47	Program	0.0
Navigatio	2-48	Records	0.0
Navigatio	2-97	Pointers	5.88
Navigatio	2-105	Loops	64.7
Navigatio	2-89	Exit	0.0

Fig. 13. Action display

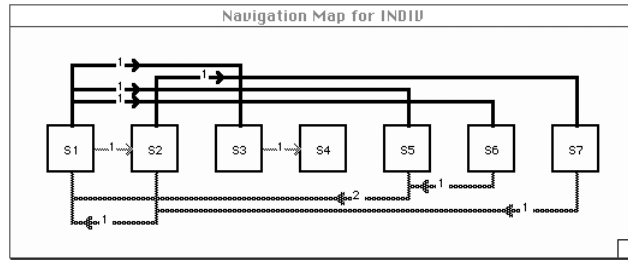


Fig. 14. A navigation diagram

'group Navigation Table

		to						
		S1	S2	S3	S4	S5	S6	S7
from	S1	0	14	0	0	2	0	0
	S2	3	0	13	2	5	2	0
	S3	0	4	0	9	1	3	0
	S4	0	1	1	0	11	4	0
	S5	0	5	2	1	0	6	0
	S6	2	2	1	5	1	0	0
	S7	0	0	0	0	0	0	0

Highlight repeated Nav.

Highlight Forward Nav.

Highlight Backtracking

Fig. 15. A navigation table

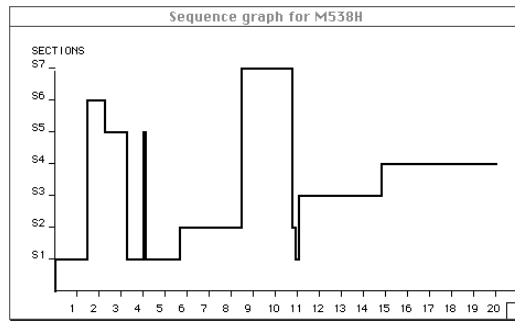


Fig. 16. A navigation sequence graph

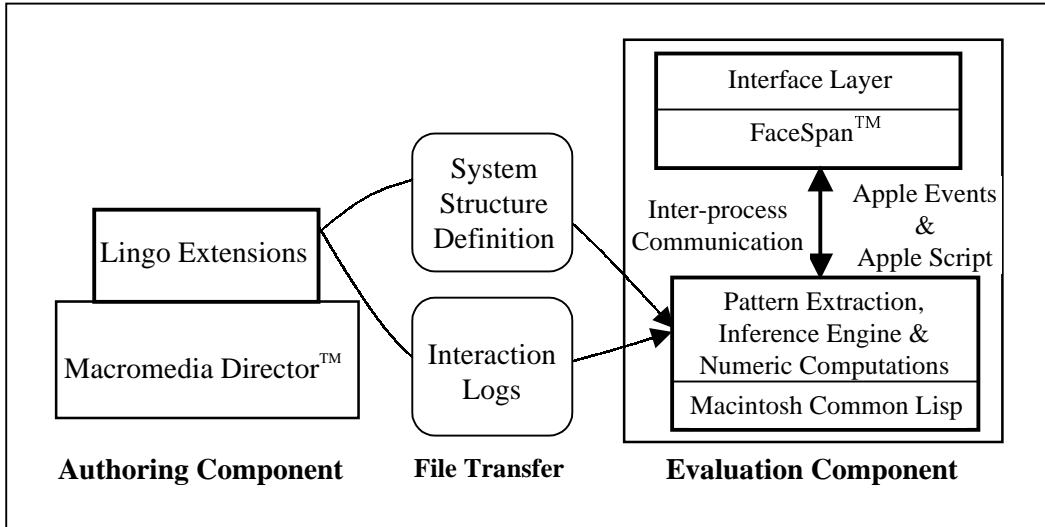


Fig. 17. System architecture

List of Figure Captions

- Figure 1. Iterative design framework
- Figure 2. Tree representation of a manual
- Figure 3. Designer prompts for a new interactive object
- Figure 4. A navigation map
- Figure 5. Pattern detection algorithm
- Figure 6. A navigation pattern
- Figure 7. Top-level menu of the designer's interface
- Figure 8. A visualization of navigation restrictions
- Figure 9. Section visit statistics
- Figure 10. User actions during a section visit
- Figure 11. A time chart
- Figure 12. Section visit frequencies
- Figure 13. Action display
- Figure 14. A navigation diagram
- Figure 15. A navigation table
- Figure 16. A navigation sequence graph
- Figure 17. System architecture