Design Strategies for Scenario-based Hypermedia: Description of its Structure, Dynamics, and Style

Ryuichi Ogawa, Eiichiro Tanaka, Daigo Taguchi and Komei Harada
C&C Information Technology Research Laboratories, NEC Corporation
4-1-1 Miyazaki, Miyamae-ku, Kawasaki, Kanagawa 216, Japan

Abstract

This paper describes design strategies for scenario-based hypermedia, which presents media composite nodes according to timed scripts. In order to translate an author's story into timed scripts within a hypermedia framework, we present a design model with four different levels of scenario specifications. In these levels an author specifies details of 1) global structure described as the hierarchy of composite nodes with sequencing relationship between them, 2) detailed structure of a composite node described as a set of subnodes and navigation flow between them, 3) content specification of multimedia data, and 4) time and spatial presentation style of media data included in each node. Design strategies based on the model were applied to the authoring of a CD-ROM based English listening course for Japanese students. The design work was accomplished as a joint project with English teachers, and our scenario-based hypermedia system, *Videobook*, was used as the authoring platform. This paper reports the details of the design strategies in each level and discusses how they made the authoring efficient while promoting the quality of the course.

1 Introduction

Scenario-based hypermedia is a type of hypermedia application, in which the basic structure is based on a written story or a detailed scenario. Although it might be perceived as a special category for time-based applications using audiovisual data, its authoring support is fundamental for the development of qualitative applications for the following reasons. First, incorporating a story into hypermedia produces well organized applications in which readers can grasp the author's intention by following the story. Second, authors who can write stories are much more numerous than those who know what hypermedia is. Therefore, good hypermedia applications can be more easily developed when based on previously existing stories. Third, carefully written scenarios can be good templates for similar applications. For example, if we can write a good scenario for an English learning course, it can be used to develop other language courses with much less authoring effort. However, most hypermedia systems do not incorporate a methodology for scenario writing, even when accepting audiovisual data. The lack of a methodology is one of the major reasons why interactive multimedia application authoring is so hard and expensive.

When we develop a multimedia application based on a story, it is common to prepare several design specifications with different abstraction levels, such as data structure, navigation flow, presentation style, data content specification, etc. Scenario writing should be considered as a method to design an application by integrating such different levels of specifications. However, the current hypermedia technologies are dealing with issues related to only one particular level. For example, we developed a scenario-based hypermedia system, *Videobook* [7], which accepts timed scripts specifying presentation styles of media data. Although *Videobook* has been successful for presentation oriented applications, it is not ready to integrate more abstract specification levels. Other proposed techniques like path mechanism [12], guided tour [5,11], and the Trellis model [8] are also limited structuring methods which accept only a part of the specifications. What we really need is, therefore, a global design model, which consists of fundamental design levels to describe different specifications, and global design strategies based on the model.

From these viewpoints we present a design model

for scenario-based hypermedia. The model consists of four design levels: 1) global structure described as the hierarchy of composite nodes with sequencing relationship between them, 2) detailed structure of a composite node described as a set of subnodes and navigation flow between them, 3) content specification of multimedia data, and 4) time and spatial presentation style of media data included in each node. Design strategies based on this model were applied to the authoring of a CD-ROM based English listening course for Japanese students, using *Videobook* as an authoring platform.

The course is a typical example of a time-based, story-oriented hypermedia, where students go through learning materials according to a predefined path. The courseware specifications were described according to the design levels and then translated into nodes and links. The basic strategy was to modularize the large sized hypermedia structure, so that the authoring became small sized and manageable.

In the next section we describe the four level design model. Section 3 gives an overview of the English course and its authoring. Section 4 describes details of the design strategies and important lessons learned. Section 5 examines the quality of the English course and the overall efficiency of the design strategies, and then summarizes the strategies as one possible scenario writing method for hypermedia.

2 A design model for scenario-based hypermedia

Recent research work shows the importance of templates or predefined data structure for improving hypermedia quality [1,2,3,10]. We believe that scenario writing is another important strategy to make hypermedia structure well organized, so that authors can present a story, or help readers to reach an intended goal. From this viewpoint we define a scenario as a set of specifications to translate an author's story into a structured navigation and presentation flow in hypermedia framework. Through our experience of developing a couple of multimedia applications, we take the following types of specification as essential.

1) Global structure: The logical structure of ordinary stories is usually hierarchical, such as chapters, sections, and paragraphs. In this level we describe a given story as a hierarchy of modules. A module is a logical unit of a story which is later translated into a composite node in hypermedia framework. It must be semantically independent from other modules, and must not have tens of links with them. If the hierarchy is large we decompose it into subhierarchies. This hierarchy decomposition is a basic strategy to grasp and manage large sized hypermedia structure.

2) Detailed structure: This describes a module in the hierarchy in terms of units of presentation and relationships between them, i.e., nodes and links. Definition of nodes and links are major design issues of this level. Basically links are defined as state transitions rather than semantic relationships (in other words, they are 'goto' links), so timing or conditions to activate the transition are often specified. Also transition styles such as 'scene change,' 'superimpose,' or 'dissolve' may be specified.

3) Data content specification: This specifies multimedia data content (e.g. description of a video shot or an audio narration in text form) for each node. Since those data are created according to this specification, its quality can critically affect the authoring cost. Also, it affects the design work of other levels. For example, an author must often think what content to be presented while designing global structure, detailed structure, or presentation style.

4) Presentation style: This describes the presentation style of a node. It is essential for user interface design. At this level window layout and timing of presentation should be specified.

For example, *Videobook* provides a visual interface to edit a time-based, media composite node 'scene,' which specifies when and where media data and menu buttons are displayed (see Figure 1).

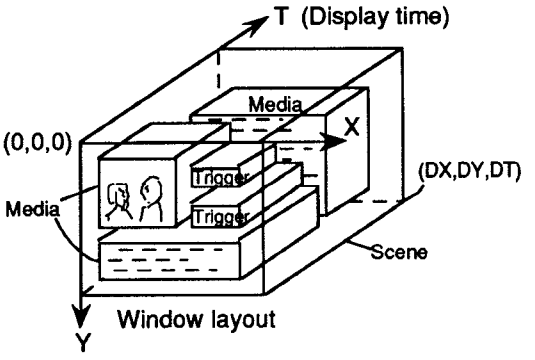


Figure 1: *Videobook*'s visual scene model

In the figure 'media' and 'trigger' are object types that represent media data and buttons, respectively. We can specify their location on the screen and presentation timing (x,y,t), and their display size and duration time of presentation (dx,dy,dt) as the layout of three dimensional solids. As for audio, only timing parameters are meaningful. Data synchronization is achieved in two ways: 1) media data and button pre-

sentation based on the solid layout, and 2) the use of an 'auto trigger,' a type of trigger which automatically activates an event at the specified time. Time-based, video-like browsing buttons are also provided as triggers to 'play,' 'stop,' 'reverse,' 'repeat' and 'skip' scenes.

The above four levels are the basis of our design model for scenario-based hypermedia. Note that numbers show only the abstraction level, and the actual design procedure does not necessarily follow this sequence. Also, through our experience, we believe that any good design model is useless without practical strategies to apply it. The major purpose of this paper is therefore to show how we can actually design a good application using the four levels. For further discussion, we introduce a case study: the authoring of a CD-ROM based English listening course.

3 Authoring of a CD-ROM based English listening course

3.1 Background

Language learning is a promising multimedia application domain because of: 1) the need for a good language learning courseware is great and becoming even greater, 2) plenty of learning methods (i.e. stories) and audiovisual learning materials have been accumulated, and 3) evaluation can be easily made when compared to other applications. The development of a CD-ROM based English listening course is an ongoing project, which incorporates a new learning method proposed by English teachers of Chiba University and uses *Videobook* as a hypermedia platform. This has been a good opportunity to test and improve the usability of *Videobook*, and to extract useful authoring strategies at each design level described above.

3.2 Basic features of the English course

The English course is designed for self learning, using a personal computer with a CD-ROM drive. Targeted users are Japanese whose TOEIC[1] score is within 200-400 (beginners) and who are not experienced computer users. 18 passages (English dialogues in audiovisual format) were selected from 5 topics they experience when traveling in the United States.

The basic structure of the course is based on what we call 'three round method.' In this method, students go through a set of related passages three times while answering three-level-graded questions. At each level, questions require gradually higher listening skills, requiring students to understand 1) key words, 2) details, and 3) the whole passage, respectively. If they cannot answer questions, they can get hints suggesting points to listen or explaining the situation. In Figure 2 the learning sequence based on the method is shown by arrows. In the figure, 'passage' denotes a set of learning materials related to one dialogue, and 'unit' denotes a set of passages having a common topic. The sequence shows that students take the easiest lessons (round1) for different passages of a unit, then intermediate lessons (round2) for the same passages, and so on. The sequence and materials are carefully chosen so that students are stimulated and never find questions too difficult[2].

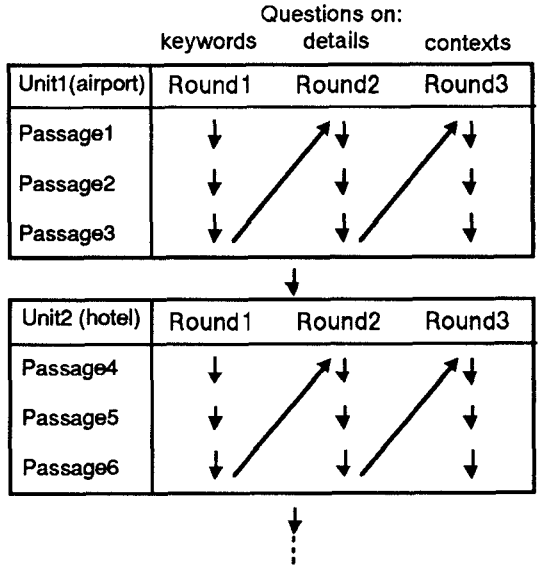


Figure 2: Three round method

Since the course was designed for practical use, the application size became large, as shown below:

Number of CD-ROMs	5 (One CD-ROM per unit)
Audio objects	2300 (English dialogues, narration, music)
Image objects	500 (scanned color photos, illustrations, graphics)
Text objects	600 (questions, answers, hints, glossaries, etc.)

[1] Test Of English as International Communication.

[2] Comparing to this course, Project Athena [6] allows students a more exploratory learning style, rather than guiding them along a systematic path.

Lesson time	30 minutes per day 15 hours in total

Expecting that the total number of nodes and links could exceed 10,000, we took the modularization of the application into small parts as one of the basic design strategies.

3.3 Authoring procedure

The whole authoring procedure of the course includes the following steps.

1) Planning: We had careful discussions with the teachers to determine fundamental specifications as described in **3.2**.

2) Courseware description: The original courseware was written by the teachers in ordinary document form. This was a most natural way, since they knew little about hypertext.

3) Translation: The courseware was translated into nodes and links. According to *Videobook*'s data model, nodes, media data in nodes, and links were described as scene, media, and trigger objects, respectively.

4) Audiovisual data creation: Audio, photos and illustrations were newly created according to the content specifications.

5) Integration on *Videobook*: Scene, media and trigger objects were created and stored into *Videobook*'s database, in parallel with the media content editing and the presentation style design.

6) CD-ROM writing: Debugged data content and the database were written to CD-ROM disks.

4 Details of design process

4.1 Original courseware documents

Before going into details, we look at the actual courseware description. Basically, four types of documents were written.

1) Global structure tree: This describes the global structure of the courseware as a hierarchy of functional modules, such as opening, question, answer, exercise, closing, etc. These modules are the units teachers employed to organize their method.

2) Detailed structure (textual): This describes the module's detailed structure as a sequence of data content specifications with their functional type (dialogue, question, answer, hint, instruction, etc.) and media type. It also includes markups to denote boundaries of the global structure and 'goto' statements due to user interaction. A simplified example is shown in Figure 3.

```
Q 1 [Markup for functional module Question1]
Dialogue (Graphic)      Photo of the passenger showing his passport to the immigration officer
Instruction (Audio)     Let's listen to the dialogue.
Dialogue (Audio)        Dialogue of passage1
Question (Text)         What did the passenger show to the immigration officer?
Question (Audio)        What did the passenger show to the immigration officer?
```

Figure 3: Detailed structure in textual form

3) Detailed structure (graphical): This is a supplement to visualize state transitions according to user interaction. Actually, it illustrates the transition of screen images by menu selection, so that it also shows a rough idea of the layout of windows and menu buttons.

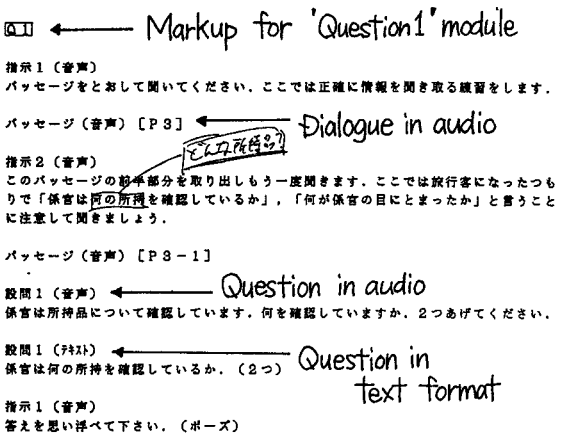
4) Content specification compiled for media data creation: In order to prepare color photos, illustrations, and audio data, their content specifications were written. The audio specification is a recompilation of the one used in the detailed structure document.

Figure 4 shows examples of document 2 and 3, written by the teachers[1]. Thanks to their previous experiments with the three round method, teachers already had a clear idea of the global structure and the detailed structure. They immediately started to describe the detailed structure both in textual and graphical format. The separation of document 2 and document 3 was the second best choice, as they wanted to describe them together. As for multimedia synchronization, they had a serious concern from an educational viewpoint. However at that moment they could not imagine how the actual synchronization would look like, and did not specify it explicitly.

From the above facts we derive three lessons: 1) if previous experience of courseware design is available, authors can start the design work at a detailed level, 2) authors want to specify media content and its visualization on the screen in parallel with the structure design, and 3) timing specifications can be neglected in the early stage of the structure design.

[1] Markups in Japanese were in the original documents, and markups in English were added to help the reader.

Detailed structure (textual)



Detailed structure (graphical)

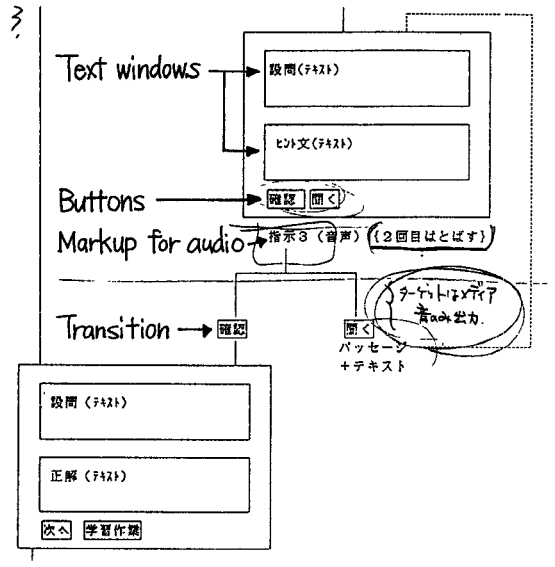


Figure 4: Example of the original documents

4.2 Global structure design

4.2.1 The whole structure

The whole structure is schematically described as Figure 5. This tree can be regarded as a directory of modules. Each module on the right side has one incoming link and one outgoing link for inter-module navigation[1], and the vertical order of the modules shows the link connections. According to the connections, students go through modules in the figure sequentially from top to bottom.

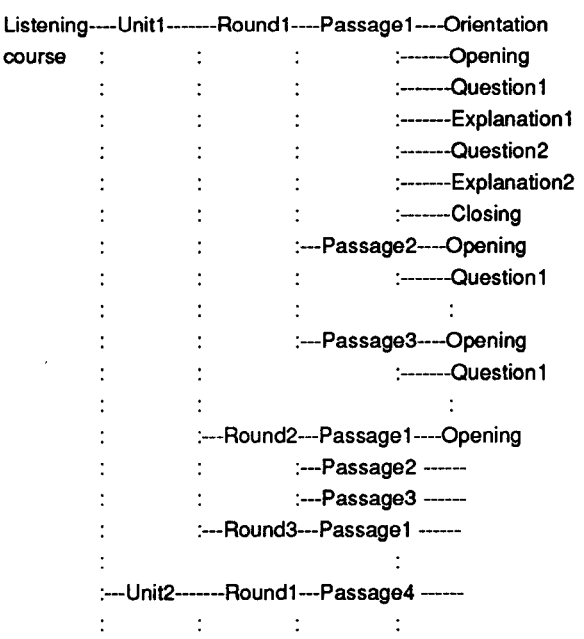


Figure 5: Global structure of the whole courseware

4.2.2 Separation strategies

In order to keep the authoring in a moderate scale, we need small subtrees. A naive strategy is to separate the tree of Figure 5. This can be called flow oriented, since the module sequence in the subtrees still reflects students' navigation flow. Although the strategy is simple, there are two problems: 1) authors do not necessarily design an application along the navigation flow, and 2) the strategy is not necessarily the best way to handle a large amount of data.

Now consider a subtree which gathers all three round lessons of one passage. We can call it data oriented, since modules sharing the same audiovisual data are clustered. The comparison is shown in Figure 6. In the data oriented structure, the outgoing link of module Passage1/Round1/Closing[2] does not connect to its neighbor Passage1/Round2/Opening, but Passage2/Round1/Opening included in the subtree of passage2 (not shown in Figure 6). Likewise Passage2/Round1/Closing connects to Passage3/Round1/Opening in the subtree of passage3. Then Passage3/Round1/Closing connects to Passage1/Round2/Opening in the subtree of passage1.

[1] This is not a mandatory constraint. We allow a module to have several inter-module links.

[2] Passage1/Round1/Closing is a simplified notation, regarding the subtrees as file directories.

Although the navigation flow became complicated, teachers preferred this structure, because passage-by-passage design was a more natural way for them. We also preferred it because multimedia data for one passage was localized into one subtree. Actually we created 18 MS-DOS directories for each subtree and stored the corresponding *Videobook* database and data content in it. One database stored 600 data objects, which is a manageable size for *Videobook* editors. Since revisions and updates of a specific passage often occurred, this localization was extremely useful. We could update everything by making changes to just one database, instead of editing three databases in the flow oriented structure[1].

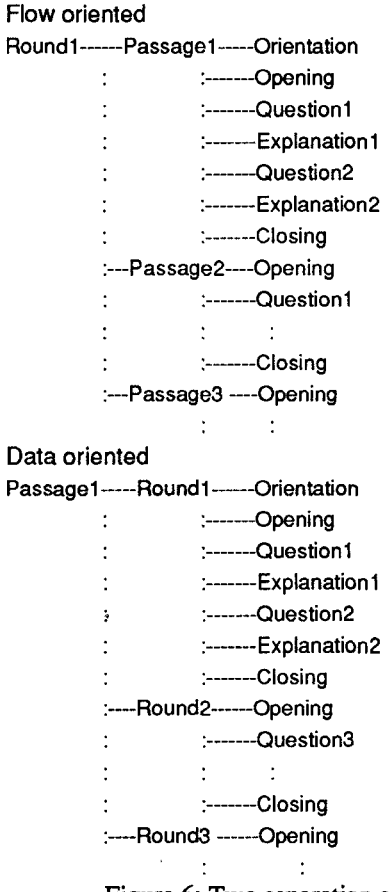


Figure 6: Two separation strategies

Another useful strategy was to design node names to represent their locations in a subtree. Figure 7 shows an example. If authors get used to the notation, they can easily see what node they are editing by its name. And they do not need a browser to search it.

[1] The updates were mostly revisions of multimedia data content, and local changes of presentation styles.

There is neither name collision nor semantic confusion, which easily occur when a hypermedia structure grows bigger with unorganized node naming. With the naming we again found advantages of the data oriented structure. For example, file names represented as p01* (* is an escape code) show that they belong to passage1. This made the file management quite easy.

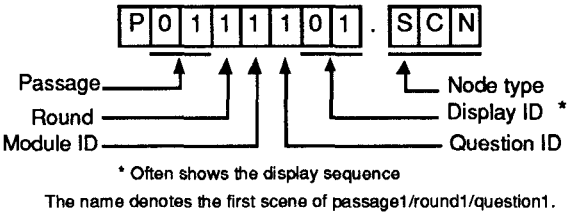


Figure 7: Naming strategy

Now we summarize the lesson of this level. When the global structure is large, its separation does not have to be based on the navigation flow. Data oriented separation can be a good solution, since it is a natural way of authoring and it also localizes multimedia data. However, when we present the global structure to students for navigation or as a browsing aid, it must be flow oriented, since what they actually must see is the learning sequence.

4.3 Detailed structure design

4.3.1 Definition of nodes and links

Basically, a node should be a complete, unbreakable unit of information. In our case, teachers took care to present one clear message to students by one audio instruction or English dialogue and one whole screen (not one particular window) synchronously displayed. Therefore we defined a node as a complete presentation unit, consisting of an audio object, and of windows and buttons synchronously displayed.

Then we interpreted a link as a mechanism to activate the transition from one node to another, and derived three link types:

1) User driven links: This is an ordinary link activated by users' button selection. For this application most links were of this type, and they were usually 'goto' links. Referential links were partly used to refer to glossary nodes and English dialogue nodes repeatedly.

2) Timer driven links: This link has a timing attribute and is activated automatically at the specified

time. It was mainly used to describe presentation oriented modules, such as orientation and opening.

3) Condition driven links: This link has a condition attribute and is activated only when a message given to the link matches the attribute. This represents 'if condition A then goto B' type of transitions. We employed it to accept students' text input for answering questions, and activate links according to their answers.

In the graphical detailed structure document teachers did not specify time-based transitions (they just specified menu buttons for students). Also they did not clearly distinguish referential type transitions from 'goto' type ones, making it hard to perceive what transitions are the main stream of the learning sequence. Therefore it is important to ask courseware authors what type of transitions they actually want.

4.3.2 Finding nodes and links

We examined both the textual and graphical detailed structure document and divided the textual specifications, so that each division included one complete unit of information. The example shown in Figure 3 is divided into three nodes as shown below.

Node1	Dialogue (Graphic) Instruction (Audio)	Photo Let's listen to the dialogue.
Node2	Dialogue (Audio)	Dialogue of passage1
Node3	Question (Text) Question (Audio)	What did the passenger ... What did the passenger ...

Figure 8: Division of detailed structure

Also we defined menu buttons to activate the transitions. In the above example, teachers specified buttons for transitions from Node1 to Node2, and from Node3 to the neighboring Node4 (not described in Figure 8). We took the transition from Node2 to Node3 as timer driven, and added the necessary buttons (see Figure 9). In the figure, (User) and (Timer) denote link types as described above.

Node1	Dialogue (Graphic) Instruction (Audio) Next Button (User)	Photo Let's listen to the ... Jump to Node2
Node2	Dialogue (Audio) Next Button (Timer)	Dialogue of passage1 Jump to Node3 after the audio dialogue is over
Node3	Question (Text) Question (Audio) Next Button (User)	What did the passenger ... What did the passenger ... Jump to Node4

Figure 9: Addition of links

4.3.3 Completeness of nodes

In the original specification (Figure 3) teachers specified that the audio dialogue should be accompanied by a photo image of the dialogue, which was to be displayed prior to the audio. However, in Figure 8, Node2 does not include the statement for the photo. That is, Node2 is not the complete presentation unit intended by the teachers, and makes sense only if it is accessed from Node1. This example poses an important problem of context dependence and node completeness. Usually a linear document has a context. When we derive nodes from the document, we must be careful what predefined information is necessary for a node to be complete.

In general, a node should be complete and context free. It makes hypermedia structure simple[1] and allows flexible linking. Also it helps hypermedia editors to see what information they are actually editing. Particularly when authoring is done collaboratively, keeping the structure understandable becomes the first priority issue. Considering these, we made nodes complete. In the above case, we added the dialogue photo to Node2.

On the other hand, making every node complete can pose efficiency problems. Data storage redundancy can be avoided by data sharing. Still, we might suffer from slow node presentation (we need to draw the same graphic node by node). We believe that the problem should be solved by introducing a higher level structuring mechanism: for example, a mechanism to describe context inheritance which excludes the redundant presentation. Since such mechanisms are not well investigated, we should first design a simple hypermedia structure equipped with complete nodes. By employing the data oriented structure shown in 4.2.2 and a data sharing strategy described in the next section, updating the redundant node description can be done systematically. And once the simple structure is obtained, converting it to a more sophisticated one is an easy task.

4.4 Data content specification design

The original intention of setting this design level was to allow authors to prepare data content specifications apart from hypermedia structuring. In our case, we needed specifications for color photos, illustrations,

[1] Here the term 'simple' means that we can expect the behavior of a hypermedia structure without considering a user's previous navigation history.

and audio data before the structuring. At the same time we observed that knowing what content was to be included in what node was essential for designing nodes and links. Also, we needed the information all through the authoring procedure. These facts suggest that the content specification should be incorporated to a hypermedia authoring environment in much more integrated ways.

Another important lesson is the need for the specification of data sharing, which tells what data content should be shared by what nodes. There are two motivations for data sharing: 1) complete nodes require repetitive description of the same data content, and 2) a large amount of audiovisual data can be a critical problem for the whole authoring. In our case, the original audio specification was written without considering the reuse of data for similar instructions, so that the total amount exceeded the storage limit (one CD-ROM per unit)[1]. We reviewed the specification to extract common (reusable) instructions, and found that 900 out of the 2300 audio data could be shared. Then we revised the detailed structure document to specify which node had common instructions. As a result, we saved 37% of audio data creation work and stored the whole unit data in one CD-ROM.

4.5 Presentation style design

4.5.1 *Videobook* in practice

The presentation style design of each scene (node) was done on *Videobook*'s scene editor, in parallel with the media data editing. The editor has a visual interface shown in Figure 1, where we graphically specify the size and location of solids with a mouse. Since teachers asked not to present unnecessary information, the display style became simple with less than two windows and three buttons centered on the screen.

As for presentation timing, teachers wanted windows or buttons to be synchronized with a specific phrase of instruction, like 'let's go to the next lesson.' Also they wanted an interval between instructions, so that students had time to think. We could easily edit the timing parameters by moving solids along the time axis and verify the results with a real time scene presentation function. One problem was the lack of a mechanism to keep the synchronization constraint between a phrase and a button, so we often had to fix it manually during the editing.

One teacher joined this design work. He found *Videobook*'s visual interface interesting but a little tricky. He also pointed out some of its problems, such as instability and the lack of macro commands. However, he liked the flexibility that allowed any scene to be edited in any order and the ability of testing various presentation styles. He also liked two timing control functions which he had not expected: 1) auto triggers as timer driven links (see **4.3.1**), to achieve automatic scene change, and 2) 'repeat' buttons to repeat the current scene presentation.

Another new issue posed by this application was the complexity of user-input handling. For example, the courseware required spell checking of students' typed answers. Also it required sophisticated algorithms to determine the next scene according to their input. In order to implement these, we introduced three new functions: 1) external programs as a type of media object, so that a scene could execute a program (e.g. a program to check the input and determine the next event to occur) at a specified time, 2) switch triggers as condition driven links (see **4.3.1**), and 3) a parameter register, into which an external program wrote a parameter (e.g. the next scene name) after its execution. Once the parameter was written, any following switch trigger with the same parameter could activate an event (e.g. jump to a scene).

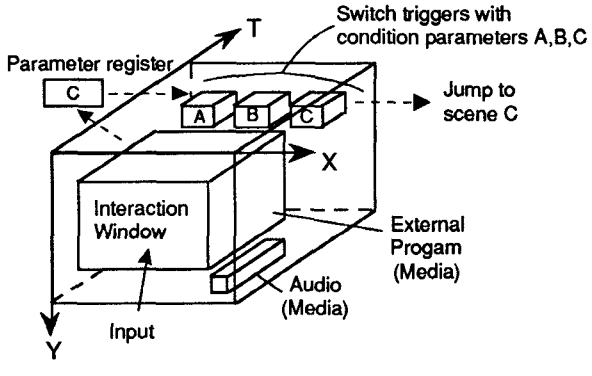


Figure 10: A 'if then' type scene example

These enabled us to describe 'if input is P then go to Q' type of scripts as a scene (see Figure 10), and worked well for this application. The integration of timer driven and condition driven events is an interesting issue and needs to be further investigated.

4.5.2 Lessons for the design strategy

What we learned about the design strategy was that most of the design work was done after we prepared

[1] The audio data was digitized in CD format. With *Videobook*'s CD editor we segmented the data and registered each segment as an audio type media object.

the data content. At first (when the audio and the image data were not ready), we tried to edit a scene's presentation style without them, and it did not work at all. The main reason was that we had no feedback, i.e., the actual appearance of each data content. After creating the content, we tested various style parameters such as window size, text layout and coloring, button layout, synchronization timing, etc., until fixing the final style format.

This fact does not deny the importance of the presentation style in the earlier design process. Actually the visualization of screen in the detailed structure document (see Figure 4) helped the node and link definition. Therefore the lessons are summarized this way: visualize the basic idea of the style before you start structuring, and determine the exact parameters after you get the data content in a good real-time presentation environment.

5 Discussion

5.1 Evaluation of the whole authoring

Since February 1992 the English course has been used for training NEC people. So far 45 students have finished the training and have shown good results. For example, 9 students whose previous TOEIC scores were within 200-360 (targeted area) showed 90 points improvement after 12 hours exercise (24 lessons in 3 months). In previous experiments students needed twice (or more) as much time to achieve the same improvement [9]. 22 students answered our questionnaire and expressed their satisfaction with the course, giving the average satisfaction score 4.4 out of 5. They liked its simple learning style and multimedia presentation. They never felt the user interface confusing and wanted to continue the lessons. As a whole, the English course has proved to be highly effective and fun to use.

Although it is hard to say to what extent the design strategies have contributed to the application quality, these helped us to develop it efficiently. The global structure design divided the application into small parts with less than 200 nodes, a manageable size for a hypermedia tool without strong database support. The detailed structure design made the hypermedia structure simple and understandable. This made the English course understandable to students. Also it made our collaboration go smoothly. The content specification design saved 37% of the audio data creation cost. The presentation style design, together with the scene presentation function of *Videobook*, allowed us to test various style possibilities and present multimedia in the most desirable way. The more important thing is that the design model helped our collaboration with the teachers go smoothly, because the four design levels were natural and understandable to them, too. Considering these, we believe that the four level design model and strategies have been quite successful for the English course authoring.

5.2 Summary of design strategies

Here we summarize the design strategies more generally. In the following we do not assume that authors are education experts or that they already have courseware documents.

1) Global structure design strategies

Describe a story as a tree structure of functional modules. If the tree is large, separate it into subtrees. Data oriented separation is a useful strategy, helping authors' natural design procedure and data management. A node naming strategy to represent the tree organization can also help the data management.

2) Detailed structure design strategies

Define a unit of presentation. Describe the unit as a set of content specifications. Also visualize the screen images of the units and specify transitions between them. Be sure to make the unit complete and context free. Examine if the transitions are user driven, timer driven, or condition driven, or if they are 'goto' type or referential type.

3) Data content specification design strategies

Design the content specification together with the detailed structure design. Extract a common, sharable part from the specifications. When the node and link structure is derived, specify what data should be shared by what nodes.

4) Presentation style design strategies

Determine the basic style early, for the detailed structure design. Keep the style consistent. Specify detailed parameters after creating the data content, in a good real-time presentation environment.

5.3 Future work

Most of the design work discussed in this paper has been done manually, except for the style design on *Videobook*. In order to support authors' scenario writing work totally, many issues need to be further investigated. Here we list just a couple of important issues we are working on.

1) A detailed structure design environment which helps authors define a presentation unit (node) and a

transition mechanism (link). It must be able to edit both data content specifications and transition flows visually, with screen images.

2) A presentation style design environment which provides good visual feedback by real time presentation, even if the data contents are not ready.

3) A template support mechanism to store well designed specifications as templates and reuse them for future applications. The templates are useful not only for data structure, but also for contents and presentation styles [4].

6 Conclusion

We have described design strategies for scenario-based hypermedia applications. The strategies are based on the design model which consists of four levels, specifying global structure, detailed structure, content specification, and presentation style of the application, respectively. Design strategies based on the model were actually tested in the authoring of a CD-ROM based English listening course for Japanese students, and the followings have been confirmed to be quite useful: 1) data oriented separation in the global structure design, 2) complete, context free hypermedia structure in the detailed structure design, 3) data sharing in the content specification design, and 4) real time scene presentation in the presentation style design. 45 students who actually used the course for 3 months showed that the course was effective and fun to use. In summary, the strategies succeeded in developing the course efficiently while keeping its quality. Since the model and the strategies are described in general terms, they can be a good framework to construct a general scenario writing method for hypermedia applications.

Acknowledgment

The authors express their sincere thanks to Prof. Yukio Takefuta, Dr. Chikako Ohnishi, and Dr. Hideo Takahashi of Chiba University, Masakatsu Yoshida and Takami Sato for their collaboration to develop the English course. They are also grateful to Rodrigo Botafogo for critically reading the manuscript.

References

[1] K.Catlin, L.Garrett and J.Launhardt, Hypermedia Templates: An Author's Tool, in *Proceedings of Hypertext'91*, 1991, 147-160.

[2] F.Garzotto, P.Paolini and D.Schwabe, HDM - A Model for the Design of Hypertext Applications, in *Proceedings of Hypertext'91*, 1991, 313-328.

[3] D.Jordan, D.Russel, A.Jensen and R.Roggers, Facilitating the Development of Representations in Hypertext with IDE, in *Proceedings of Hypertext'89*, 1989, 93-104.

[4] R.MacNeil, Generating Multimedia Presentations Automatically using TYRO, the Constraint, Case-based Designer's Apprentice, in *Proceedings of the IEEE Workshop on Visual Languages*, 1991.

[5] C.Marshall and P.Irish, Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers, in *Proceedings of Hypertext'89*, 1989, 15-26.

[6] J.Murray, Emerging Genres of Interactive Videodiscs for Language Instruction, in *Multimedia and Language Learning*, Institute for Academic Technology, University of North Carolina at Chapel Hill, 1990, 11-22.

[7] R.Ogawa, H.Harada and A.Kaneko, Scenario-based Hypermedia: A Model and a System, in: A.Rizk, N.Streitz and J.Andre, eds., *Hypertext: Concepts, Systems, and Applications*, Cambridge University Press, 1990, 38-51.

[8] P.Stotts and R.Furuta, Dynamic Adaptation of Hypertext Structure, in *Proceedings of Hypertext'91*, 1991, 219-231.

[9] Y.Takefuta, editor, *Working Papers in Language and Speech Science* 2, Chiba University, 1991.

[10] M.Thuring, M.Haake and J.Hannemann, What's Eliza Doing in the Chinese Room? Incoherent Hyperdocuments -- and How to Avoid Them, in *Proceedings of Hypertext'91*, 1991, 161-177.

[11] R.Trigg, Guided Tours and Table Tops: Tools for Communicating in a Hypertext Environment, *ACM Transactions on Office Information Systems 6(4)*, 1988, 398-414.

[12] P.Zellweger, Scripted Documents: A Hypermedia Path Mechanism, in *Proceedings of Hypertext'89*, 1989, 1-14.