Schema-less Structural Links in MediaDesc

Andrea Caloini
Human Media Res. Labs., NEC
4-1-1 Miyazaki, Miyamae-ku, Kawasaki, Japan
EMail: caloini@hm.cl.nec.co.jp
Voice: +81(44)856.2363 Fax: +81(44)856.2239

Abstract

The maintenance of links in hypermedia documents is a well known problem; research-level authoring systems address this problem by means of structural links, which require the definition of a formal schema of the document; this kind of approach discourages rapid prototyping and forbids bottom-up development — which is often preferred to a top-down approach by authors such as artists and designers. On the other hand, commercial authoring systems address the the maintenance of links in hypermedia documents very poorly.

On the contrary, MediaDesc[1] structural links can exploit regularities in the document structure without forcing the author to define a document schema. MediaDesc structural links can describe sophisticated navigation patterns for well-structured documents while, at the same time, authors are not forced to use a complicated document structure if they just need a trivial navigation pattern. Finally, we will show that MediaDesc authors can start writing a hyperdocument without defining a precise structure and switch to a more structured approach at a later stage with a reasonable effort.

1. Introduction

The maintenance of links in hypermedia documents is a well known problem [10], but in many cases a large percentage of the links can be described in terms of the structure of the document [4]; such links are often called *structural links* (e.g. [6]).

Several authors (e.g. [6][9][12]) have devised hypertext systems where the structural properties of hyperdocuments are exploited to provide a large number of links. However, in such systems structural links are defined in terms of the schema of the document, which must be defined *prior* to creating the links (and the document instance). Even if schema evolution is supported (such as in [9]), this kind of approach discourages rapid prototyping and it is often not welcome by authors accustomed to a bottom-up development — such as artists and designers (see also [11]).

Hypermedia systems built on top of a database can apparently provide structural links very easily relying on the database query language (e.g. [2]); however, they still require a schema and they actually provide relationship traversal, not with links — for example, [2] is unaware of sequential browsing and it requires the author to explicitly define a *next* relationship.

Finally, mainstream commercial authoring systems, such as “ToolBook” [13] and “Authorware” [1], do not address the problem at all because they only provide a very poor set of structural links (*next*, *previous*, *first* and *last*); more sophisticated linking patterns must be implemented by the author using the scripting language of such products.

Although a schema-based approach is likely to be effective for database-like applications (i.e. highly regular, medium/large sized collections of multimedia data), if the hypermedia document is rather similar to a book, or an article of a magazine, we believe that a more document-oriented approach is required.

In MediaDesc, we take an approach inspired by SGML [3]: a MediaDesc document is a sorted tree of components which can be labelled (tagged) to describe the logical structure of the document; MediaDesc structural links exploit such document structure to define the target of a link in terms of its relative position with respect to the source node. Unlike in

1. MediaDesc is an authoring system developed at NEC Central Research Labs. and currently commercialized in Japan.

1060-3425/98 $10.00 (C) 1998 IEEE

[6][9][12], MediaDesc structural links can be defined directly on the top of a document instance and a document schema is not required.

(Although we believe that a document schema can be a useful validation tool, we choosed not to force the author to use it, especially considered that the typical user of our system is not a computer specialist.)

In this paper we show how MediaDesc supports the navigation of SGML-like hypermedia documents composed of heterogeneous aggregates and homogeneous ordered collections as well as the navigation of more loosely structured documents. We also illustrate a number of other additional features, such as node-level link overriding, session-dependent history and context management, etc..

The rest of this paper is structured as follows: Section 2 illustrates our requirements for structural links and discuss how such requirements are satisfied in MediaDesc and in other systems, Section 3 describes in detail the MediaDesc document model and MediaDesc structural links, Section 4 provides some details about the implementation and the user interface, Section 5 draws some conclusions.

2. Requirements

In this section we will list some general requirements for structural links and then discuss how such requirements are satisfied in MediaDesc and in other systems.

R1: Some documents are best described by a loose structure or no structure at all, while others are worth being described by a very precise structure. A flexible authoring system should be able to accommodate several levels of structuring and exploit them as much as possible to provide structural links.

R2: Applying a structure to an unstructured document should be as painless as possible.

R3: A powerful authoring system should support *common navigation patterns*. Typical navigational patterns include browsing through homogeneous collections (e.g., in a tourist guide, a collection of *hotels*) and heterogeneous aggregates (e.g. among the *lounge*, the *room*s and the optional *swimming pool* of an hotel aggregate), indexes, guided tours etc.. The satisfaction of this requirement depends on the expressiveness of both the document model and structural links.

R1-R3 represent overall requirements; the following requirements R4-R8 describe more precisely some of the “common navigation patterns” we mentioned in requirement R3.

R4: Authors should be able to define (structural) links whose destination may not exist – either because the target node is an *optional component* of the document, or either because it *has not been created yet*.

R5: Whenever the target of a link does not exist (see R4), authors should be able to define *alternate link targets*.

R6: Sometimes documents are only partially structured — i.e. some parts are structured but some others are not; authors should be able to define structural links *originating from a non-structured* portion and pointing to a structured location in the document.

R7: Given two documents whose structure is partially or completely *isomorphic* (for example, the English and Japanese versions of a document), authors should be able to link node pairs located in corresponding positions.

R8: To increase the modularity and the maintainability of the document, node groups should be a valid target for a link.

Concerning R1, MediaDesc offers more flexibility than the systems described in [2][6][9][12] first because it does not require a document schema and second because the document itself can be more or less structured (see Section 3.3). Of course, we not claiming a document schema is useless: a document schema can be used to validate the link definitions as well as the document instance. However, since structural links can be defined without referring to a document schema, we choosed do not force authors to define a schema if they can do without it.

Requirement R2 does not apply to systems that cannot handle structured documents ([1][13]) and otherwise it appears to be addressed only by [9] — although [9] supports schema evolution, rather than the transition from an unstructured document to a structured one. In MediaDesc, this requirement is satisfied because structuring a document only requires to gather logically-related nodes into node groups and to assign them a label.

Concerning R3, the expressiveness of MediaDesc structural links is located somewhere between the bare next/previous/first/last links of commercial systems

[1][13] and the sophisticated functionality of systems like [6][9][12] (see also Section 3.3)

In the spirit of R1, requirements R4, R5 and R6 allow the creation of partially structured documents and call for a corresponding support at structured link level. As far as we know, these requirements are not satisfied by other systems.

Requirement R7 can be implemented by a query in systems like [2] but it is usually not satisfied in hypermedia systems not built on the top of a database.

Requirements R8 is usually satisfied both in MediaDesc and in other systems (although not in commercial systems like [1][13]).

3. Structural Links in MediaDesc

3.1 The MediaDesc document model

In this section we will describe the MediaDesc document model; for the sake of conciseness, we will focus on those aspects relevant to the topic of this paper. We will also compare the MediaDesc document model to the Dexter Model [8] (DM) and to ToolBook [13] (TB) since they are probably known by many of our readers.

A MediaDesc document (*scenario*) is a sorted tree of DM-composites whose leaves are displayable units called *scenes* (Figure 1); these correspond to TB-pages and to the commonly understood concept of 'node'. Non-leaf composites (*scene groups*) are not displayable ('instantiable' in DM terms); they usually describe complex real-world entities and they ease link definition and document editing.

A scene can contain zero or more *media*, which correspond to DM-atomic components, and to TB-fields, TB-graphic objects etc.. A scene also has a *background* [2] (similar to TB-backgrounds) that contains a background bitmap and zero or more media; a background (and its media) can be shared among several scenes.

In MediaDesc, *scenes* and *scene groups* have a mandatory *indentifier* (unique within the document) and an optional *label* describing their semantic type; such labels describe the document structure in a fashion similar to SGML tags (see below).

A MediaDesc *link* connects a media to a target scene; when the link is transversed, the target scene is displayed in place of the current scene. A link can be activated upon user input, by a timeout, or by a MediaDesc internal event (e.g. 'go to scene *X* when this media is over'). Links pointing to a scene group or to an external document are resolved into the *entry scene* of the target.

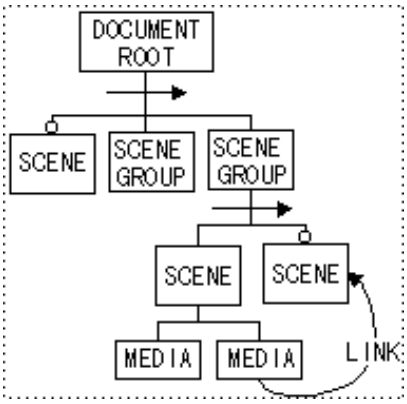

Figure 1: A Mediadesc document instance (simplified). A *document* is an ordered collection of *scenes* and *scene groups* — and so is a scene group. A *scene* has a list of *media*, a media can hold one or more *links* pointing to another scene. Documents and scene groups have an *entry scene* (marked by a circle in the figure); the entry scene is the scene which is displayed when a document/ group is the target of a link.

At runtime, whenever a link is traversed, the source node of the link is recorded on the *history stack*; the reader can therefore go back to any previously visited node.

Furthermore, the author can force the source node of some links to be recorded on another runtime stack (*context stack*) so that, later on, the reader can go back to the original context with a single mouse click. A typical application of the context stack is to bring back the reader to the original context after she has jumped to the help.

Before closing this section, we would like to briefly compare MediaDesc and SGML. Although SGML tags and MediaDesc labels are both used to mark up the structure of the document, MediaDesc labels are optional — i.e. the author can define a tree-structured document without assigning any label (e.g. Figure 4); on the other hand, MediaDesc does not yet support anything similar to the Document Type Definition of SGML or to the *schema* of [2][6][9][12].

2. Not represented in the figure.

3.2 Structural Links in MediaDesc

Formally speaking, a MediaDesc structural link (*scenario link*) is an expression which can be evaluated with respect to a set of source scenes (*source set*) and yields a set of target scenes (*target set*); usually the *source set* is either the current scene or the target set of another scenario link[3].

In general, the target of a scenario link is defined in terms of its relative position with respect to the source; such relative position is expressed in terms of the hierarchical structure of the document, the order of the scenes, and their labels.

In the rest of this section we report a list of all valid scenario links; whenever a specific scenario link satisfies a specific requirement, we will also reference the requirement. In the next section we will discuss the requirements we will not cover in this section.

Next (Prev/First/Last). Points to the next (previous/ first/last) scene in the current scene group; if a label is specified (e.g. *NEXT "toc"*), the next (previous/first/ last) scene with the given label is retrieved. If the *ignoregroups* option is specified, group boundaries are ignored and the document is treated as a flat sequence of scenes. If the target scene exists, it is unique. For example (see Figure 2)[4]:

target(*scene1*, NEXT) =*scene2*
target(*scene9*, NEXT) =NULL
target(*scene8*, FIRST "toc") =*scene5*
target(*scene1*, NEXT "toc" IGNOREGROUPS) = *scene5*.

Up. Points to the parent group of the current scene. If a label is specified (e.g. *UP "heading1"*), the document hierarchy is climbed until a group scene with the specified label is found (R6). If the source scene is located right below the document root, the target does not exist; otherwise it exists and it is unique. For example

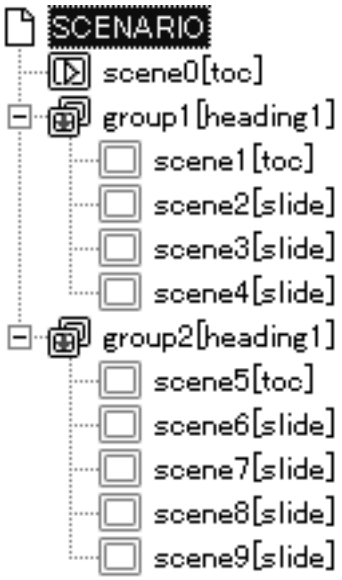

Figure 2: A MediaDesc slideshow (MediaDesc Outline Editor). On the right of scene icons the scena name (e.g. *scene0*) and the scene label (e.g. *slide*) are displayed. Both of them can be assigned by the author.

target(*scene1*, UP) = *group1*
target(*scene0*, UP) = *NULL*

Top. Points to the topmost parent of the current scene; the target always exists and is unique. For example

target(*scene1*, TOP) = *group1*
target(*scene0*, TOP) = *scene0*

Root. Points to the current document — i.e. one level above the scene pointed to by top; the target always exists and is unique. For example:

target(*scene1*, ROOT) = *SCENARIO*;

Down. Points to the first child of the current (group) scene; if a label is specified (e.g. DOWN "toc") the link points to the first child having the specified label. The target scene does not exist if the current scene is not a group scene, or it has no children, or it has no children with the specified label. If the ALL option is specified, all the children (with the given label) are returned. For example:

3. For readability, in the following we will say "the source scene" rather than "for each scene of the source set"; moreover, since most scenario links return a single target scene (for each source scene), we will often say "the target scene" rather than "each scene of the target set".
4. In the following we will use the notation *target(sourceSceneID, scenarioLinkExpression) = targetSceneID* to illustrate in a compact form examples of evaluation of scenario links; when writing scenario link expressions, we will write scenario link keywords in uppercase (e.g. NEXT) and author-defined labels in lowercase and between quotation marks (e.g. "toc").

target(*group1*, DOWN) =*scene1*
target(*scene1*, DOWN) =*NULL*
target(*group1*, DOWN “slide”) =*scene2*
target(*group1*, DOWN “slide” ALL) = {*scene2*, *scene3*, *scene4*}

IsLabel. If the current scene has the named label, returns the current scene, otherwise fails. For example:

target(*scene1*, ISLABEL “toc”) = *scene1*
target(*scene2*, ISLABEL “toc”) = *NULL*

Follow. Returns the target of a named scenario link; using follow we can define a scenario link in terms of another scenario link. For example:

DEFINE nextTOC =
NEXT “*toc*” IGNOREGROUPS

target(*scene1*, FOLLOW “nextTOC”) = *scene5*

Goto. Points to a named scene. Clearly, this is not a structural link, but we want to be able to mix structural and non-structural links in link expressions. The target function is trivial:

target(*whatever*, GOTO “scene0”) = *scene0*;

Scenario. Points to the first scene of a named scenario file (e.g. scenario “C:\myDir\scn1.mdc”). This link isn’t structural either.

The scenario links listed above can be combined into scenario link expressions using the *then* and *or* operators:

Then. The *then* operator concatenates two link expressions. The scene pointed by the expression expr1 THEN expr2 is obtained computing the target (*t1*) of *expr1* w.r.t. the source scene and, if *t1* exists, computing the target of *expr2* w.r.t. *t1*. For example,

target(*scene1*, NEXT THEN NEXT) = *scene3*
target(*scene8*, NEXT THEN NEXT) = *NULL*
target(*scene1*, UP “heading1” THEN FIRST “toc”) = *scene0*

Or. The *or* operator expresses alternate link destinations (R5). The scene pointed by the expression expr1 OR expr2 is obtained computing the target of *expr1*; however, if *expr1* returns no target, the target of the expression becomes the target of *expr2*. If both *expr1* and *expr2* return no target, the whole expression returns no target. For example, the expression NEXT OR FIRST implements a “circular next”:

target(*scene2*, NEXT OR FIRST) = *scene3*
target(*scene4*, NEXT OR FIRST) = *scene1*

In the following example, link *toTOC* points to the table of contents of the current group or to the table of contents of the parent group if the current scene itself is a table of contents.

DEFINE toTOC =
(ISLABEL “toc” THEN UP THEN FIRST “toc”)
OR (FIRST “toc”)[5]

target(*scene2*, FOLLOW “toTOC”) = *scene1*
target(*scene1*, FOLLOW “toTOC”) = *scene0*

Finally, scenes located in isomorphic scene groups or isomorphic documents can be linked using some special options of *up* and *down* commands (R7). More precisely, it is possible to record the labels on the path transversed while going upward in a document (for example *pool[0], hotel[3], city[2]*) and use it in a subsequent point of the same link definition to move downward in a different scene group. For example:

DEFINE link1 =
UP “heading1” RECORDPATH THEN
PREV “heading1”
THEN DOWN PLAYPATH

target(*scene5*, FOLLOW “link1”) = *scene1*
target(*scene6*, FOLLOW “link1”) = *scene2*
target(*scene7*, FOLLOW “link1”) = *scene3*
target(*scene9*, FOLLOW “link1”) = *NULL*

If the target of a link expression is a scene group, the entry scene of the group is (recursively) retrieved and it becomes the ultimate target of the link (R8). This resolution process only occours when a link expression is invoked directly by the user, not when it is invoked by a *follow* command.

At runtime, if the scenario link associated to a button returns a null target, the runtime player disables the button itself (R4).

In this section we illustrated in detail the features of scenario links and we also showed how they satisfy requirements R4-R8; in the next section we will discuss the remaining requirements.

5. Parenthesis are added for clarity only — actually, *then* binds stronger than *or*.

3.3 Discussion and examples

Concerning requirement R1, the structure of a MediaDesc document can range from a flat sequence of scenes (Figure 3) to a sorted and labelled n-ary tree (Figure 2 and Figure 6); of course, the richer the structure the better scenario links can be exploited.

If the document is a plain sequence of scenes (Figure 3), only *next*, *previous*, *first* and *last* scenario links can be used; these four links will suffice for small documents with unsophisticated navigation requirements (e.g. a slide presentation where slides are browsed sequentially with *prev*/*next* links and the front page is reached with the *first* link); we recall that this is the maximum level of sophistication supported by commercial authoring systems like [1] and [13]. If other types of

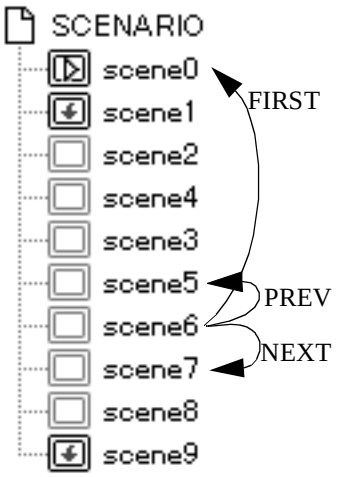

Figure 3: Various Structuring styles. The document reported in Figure 2 before applying any structure; only a limited set of structural links can be used.

links are needed, they can be created using the MediaDesc link editor (Figure 7) (whereas in [1] and [13] the author must write a script).

As the size of the document grows, we expect authors to move from a plain sequence of nodes to an unlabelled tree (Figure 4). According to our experience, authors feel more comfortable with the concept of *scene group* rather than with *labels*; moreover, scene groups improve the usability of MediaDesc's outline editor (Figure 2) and the link editor (Figure 7). If the scenes of a document are divided into logical groups, link commands *up*, *down*, *top* and *root* become available and the expressiveness of *next*, *previous*, *first* and *last* links increases. For example, now we can use the link *first* to reach the table of contents of a group (scenes 1 and 5) from within the group itself and we can still jump to the table of contents of the document with the link *up then first*.

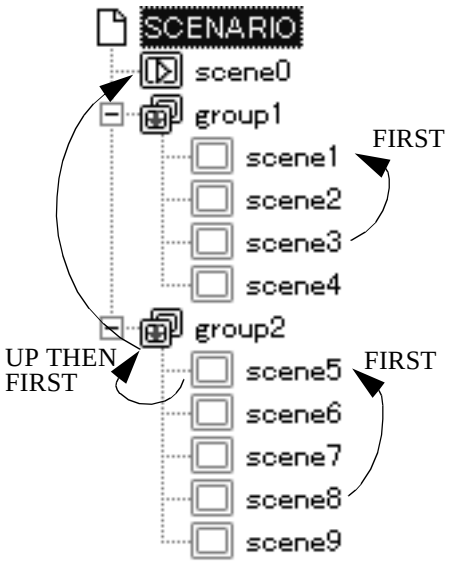

Figure 4: Various Structuring styles. The document reported in Figure 3 after gathering logically-related scenes into groups.

In theory, a similar expressiveness can be achieved assigning labels instead of creating groups (Figure 5). In this case, *next*, *previous*, *first* and *last* scenario links become capable of selecting the target scene depending on its label. For example (Figure 5), the link *previous "toc"* connects each scene with its table of contents and the link *first "toc"* points to the table of contents of the document. However, because groups are conceptually easier to manage, we do not expect this kind of document structure to be used very often.

To take full advantage of MediaDesc structural links, both scene groups and labels must be used (as in Figure 2); in this case it becomes possible to write more sophisticated and more readable links — such as those reported in the "hotel guide" sample described below. Requirement R1 is therefore satisfied.

Requirement R2 is also satisfied because scene groups and labels can be assigned at any stage of the authoring process with a limited effort.

Concerning requirement R3, we will demonstrate

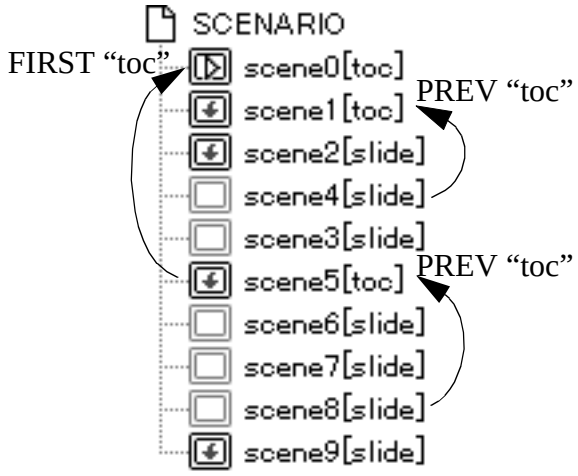

Figure 5: Various Structuring styles. The document reported in Figure 3 after labelling the scenes.

MediaDesc capabilities with an example. Suppose we created a small sample of an hotel guide (Figure 6). As it can be seen from the figure, the scenario is composed of an index node (*scene0*) and three hotels; each hotel is a scene group composed of a lounge, an optional swimming pool and list of rooms (one scene for each room type)[6]..

Navigation among the components of the hotel aggregate can be performed with the following links:

```
DEFINE toLounge =   FIRST “lounge”
DEFINE toPool =     FIRST “pool”
DEFINE toRooms =    FIRST “room”
```

to activate the above links, three buttons must be placed in each scene belonging to an *hotel* group (12 scenes).

The room list can be browsed (in a circular fashion) with the links:

```
DEFINE nextRoom =   NEXT “room” OR
                    FIRST “room”

DEFINE prevRoom =   PREV “room” OR
                    LAST “room”
```

to activate the above links, two buttons must be placed in each “room” scene (7 scenes).

Navigation from one hotel to another can be granted with the links (again in a circular fashion):

```
DEFINE nextHotel = UP “hotel” THEN
            (NEXT “hotel” OR FIRST “hotel”)

DEFINE prevHotel = UP “hotel” THEN
    (PREV “hotel” OR LAST “hotel”)
```

to activate the above links, two buttons could be placed in each scene belonging to an *hotel* group (12 scenes).

Finally, the list of the hotels contained in *scene0* can be generated with the link:

```
DEFINE hotels = ROOT THEN
                DOWN “hotel” ALL THEN
                DOWN “lounge”
```

i.e., go to the root of the document, select all the child scenes labelled “*hotel*” (*group1*, *group2*, *group3*) and for each of those, retrieve the “*lounge*” (*scene1*, *scene4*, *scene7*).

To make the above links available to the reader, the author must first create a button (media), then assign the link to the button, and finally make the button available to the proper scenes; the latter operation can be conveniently performed in MediaDesc using either back-

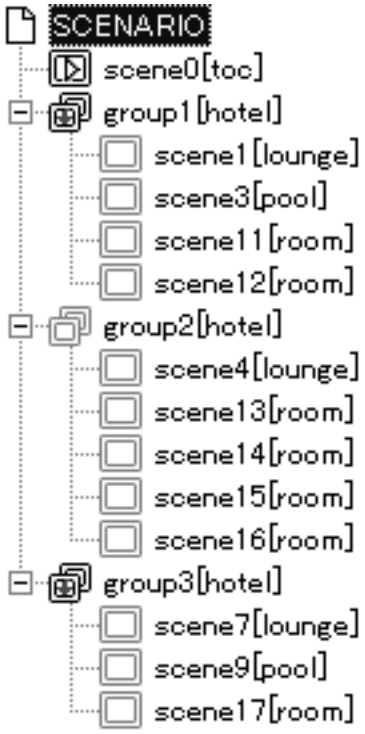

Figure 6: A hotel guide scenario.

6. The SGML DTD element declaration for such an *hotel* would be *longe*, *pool*?, *room*+ .

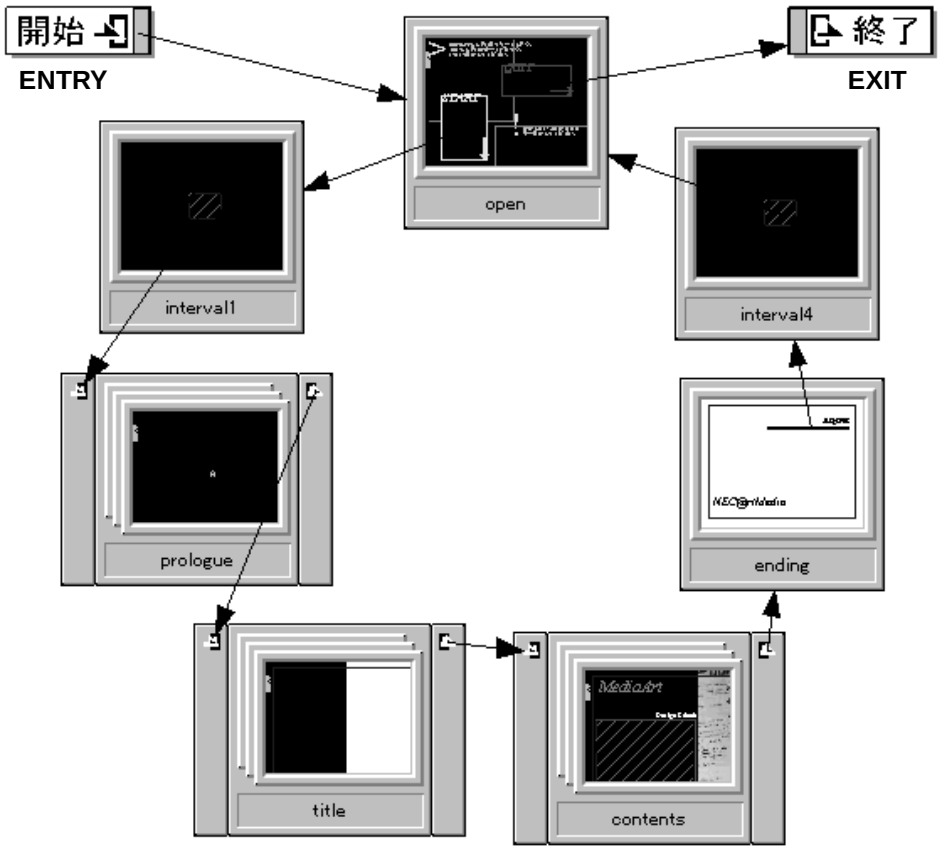

Figure 1: Mediadesc Link Editor; top level view of a scenario composed of four scenes (*open*, *interval1*, *interval6*, *ending*) and three scene groups (*prologue*, *title*, *contents*). The arrow originating from the "entry" icon in the top-left corner shows that scene *open* is the entry node of the scenario; the arrow originating from a media of scene *open* and pointing to the "exit" icon on the right shows how the user can leave the current document.

grounds or templates.

Using *backgrounds* (see section Section 3.1) it is possible to share a button among several scenes — i.e. "create" several links with a single operation. Indeed, in the above example 77 links can be "created" by means of just 8 scenario link definitions.

If buttons are added to the scenes using *templates* ([5]) rather than backgrounds, it becomes possible to override the link at scene level. In fact, when a media is added (removed/edited) in a template, *a copy* of the same media is added (removed/edited) in all the scenes referencing the template. Such copies can be edited individually, thereby overriding template settings (overridden attributes are properly flagged so that they will not be overwritten when the template is edited)

After the links and their buttons are set up, to expand the document is a trivial operation. For example, to add a new hotel it is enough to copy and paste an existing hotel, change the contents of the scenes (text, pictures etc.) and — as long as the structure of the new hotel does not depart from the original pattern — all the scenes are already properly linked.

As we have seen, a MediaDesc document can represent and link both heterogeneous aggregates and homogeneous collections; however, some other navigation patterns are not yet supported; for example, guided tours and navigation of direct acyclic graph documents (rather than just trees). Requirement R3 is therefore only partially satisfied — i.e. MediaDesc fares better than commercial systems but it is not yet as powerful as some

Figure 1: Scenario link input dialog box

research systems. However, considering the flexibility MediaDesc elsewhere enjoys (requirements R1, R2) we believe that the overall balance to be satisfactory.

4. Implementation

4.1 Generalities

MediaDesc is implemented in C++ for the MS-Windows operating system; the version currently released (1.0) includes a scene layout editor and a visual link editor; the features described in this paper will be included in the next release. The commercialized version supports several media formats (BMP, WMF, JPG, AVI, WAV, MCI, etc.) and the player can be used as a Netscape plug-in or a Microsoft ActiveX control.

4.2 Scenario Link Interface

Scenario links can be visualized graphically as arrows in the MediaDesc link editor (Figure 7) and the author can selectively choose which scenario links are displayed (e.g. display only *toRooms* links).

In Figure 8 is represented a prototype of the dialog box used to define scenario links[7]. The link definition is

7. Alas in Japanese.

displayed both in a textual form and in an iconic form (top left quarter of Figure 8); the author can edit the link definition directly using the keyboard, use a menu-assisted input (bottom left quarter of Figure 8). While editing a scenario link definition, the user can verify the behaviour of the link by selecting a source scene from the document tree in the right half of the dialog box; the target of the link is immediately displayed both in the document tree and in the small preview icon in the bottom right corner of the dialog box. Several source scenes can be checked very quickly without playing the document and without even creating the button that will be required at runtime to activate the link.

5. Conclusions

Many hypermedia applications follow a regular structural pattern and can their development can take advantage from a structured approach, especially for what concerns the management of hyperlinks. However, many hypermedia authors are definitely not accustomed to a structured approach and we believe that a user-friendly authoring system must provide the tools to take advantage of a formal approach without forcing the authors to use them.

In this paper we have shown how MediaDesc structural links can offer a fairly good expressive power without hampering those authors who do not need them; indeed we have shown that MediaDesc structural links can lend themselves to several authoring styles (structured, semi-structured, nearly unstructured) and that the transition from an unstructured document (such as a prototype) to a full fledged structured application has in MediaDesc a quite reasonable cost.

6. Bibliography

[1] Authorware 3J Reference Manual, Macromedia Inc., 600 Townsend San Francisco, CA 94103, U.S.A.

[2] Amann B., Scholl M., Rizk A., *Querying Typed Hypertexts in Multicard/O2*, Proceedings of the ACM Conference on Hypertext (ECHT 94)

[3] Brian M., *An author's guide to the Standard Markup Generalized Language*, Addison-Wesley, Workingham, England, 1988

[4] Caloini A., *Matching Hypertext Models to Hypertext Systems*, Proc. of the ACM Conf. on Hypertext (ECHT 92), Milano, Italy, Nov. 30 - Dec. 4, ACM Press 1992

[5] Caloini A., Tanaka E., *Extending Styles to Hypermedia documents*, Proc. of the Int. Conf. on Multimedia Computing and Systems (IEEE Multimedia '96), 1996 June 17-23, Hiroshima, Japan.

[6] Garzotto F., Paolini P., Schwabe D., *HDM – a model-based approach to hypertext design*, ACM Transactions on Information Systems, 11(1):1-26, January 1993.

[7] Goodman D., *The Complete HyperCard Book*, Bantam Books, New York, 1987.

[8] Halasz, F.G. and Schwartz, M., *The Dexter Hypertext Reference Model*, Comm. of the ACM, Feb 1994, Vol.37, n.2, pages 30-39.

[9] Kesseler M., *A Schema-Based Approach to HTML Authoring*, Fourth WWW Conference, Boston. (http://www8.informatik.uni-erlangen.de/IMMD8/staff/Kesseler/www4/technica/aschemax.htm)

[10] Ogawa R. and Tanaka E. and Taguchi D. and Harada K., *Design Strategies for Scenario-based Hypermedia: Description of its Structure, Dynamics, and Style*, Proc. of the ACM Conf. on Hypertext (ECHT 92), Milano, Italy, Nov. 30 - Dec. 4, ACM Press 1992

[11] Streitz N., Haake J., Hannemann J., Lemke A., Schuler W., Schütt H., and Thüring M., *SEPIA: A Cooperative Hypermedia Authoring Environment*. In Proc. of the ACM Conf. on Hypertext (ECHT 92), Milano, Italy, Nov. 30 - Dec. 4, ACM Press 1992, pages 11-22

[12] Schwabe D., Rossi G., Barbosa S.D.J., *Systematic Hypermedia Application Design with OOHDM*, Seventh ACM Conference on Hypertext (Hypertext '96), Washington DC, USA, March 16-20, 1996.

[13] ToolBook 5.0 online manual, Asymetrix Corporation, 110-110th Ave. N.E. Bellevue, WA, 98004, U.S.A.