Hypertext aspects of the Grif structured editor : design and applications

Vincent Quint, Irène Vatton

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche
1992
25 *ème anniversaire*
N° 1734

Programme 3
Intelligence artificielle, Systèmes cognitifs et Interaction homme-machine

HYPERTEXT ASPECTS OF THE GRIF STRUCTURED EDITOR : DESIGN AND APPLICATIONS

Vincent QUINT
Irène VATTON

GROUPE DE RECHERCHE
GRENOBLE

Juillet 1992

RR-1734

July 1992

Hypertext aspects of the Grif structured editor: design and applications

Les aspects hypertexte de l'éditeur structuré Grif : conception et applications

Vincent Quint
Irène Vatton

Abstract:
This report presents the experience gained in developing and using the hypertext functionalities of the Grif system. Grif is a structured document editor based on the generic structure concept: each document is represented in the system by its logical structure which is an instance of a generic structure. This notion of logical structure encompasses both hierarchical structures (as is usual in structured documents) and non–hierarchical links (as is usual in hypertexts).

The document model on which Grif is based is presented, focusing on the different types of links. Various applications using these links are also described. It is shown that the approaches of electronic documents and hypertext, which are often opposed to each other, can be combined for building more powerful integrated systems.

Résumé:
Ce rapport présente l'expérience acquise lors du développement et de l'utilisation des fonctionalités hypertexte du système Grif. Grif est un éditeur de documents structurés fondé sur le concept de structure générique : chaque document est représenté dans le système par sa structure logique qui est une réalisation d'une structure générique. La notion de structure logique englobe à la fois les structures hiérarchiques (comme c'est l'usage dans les documents structurés) et les liens non–hiérarchiques (qui sont à la base des hypertextes).

On présente le modèle de document sur lequel s'appuie Grif, en mettant l'accent sur les différents types de liens. Plusieurs applications utilisants ces liens sont également décrites. On montre que les deux approches des documents électroniques et des hypertextes, qui sont souvent opposées l'une à l'autre, peuvent être combinées pour construire des systèmes intégrés d'une plus grande puissance.

Hypertext aspects of the Grif structured editor: design and applications

1 Introduction

Is there any difference between a traditional document and a hypertext? The answer to that question is not clear if we consider the recent evolution of systems dedicated to these two media. On one hand, document preparation systems include more and more functionalities inspired from the field of hypertext [4]. On the other hand, some hypertext systems use now principles coming from electronic documents, especially from structured documents [1].

These trends have already been noted by other authors (for instance, Stotts and Furuta [21]) and several systems examplify them. Dynatext, from Electronic Book Technology, transforms a structured document represented in SGML into a hypertext and allows the user to browse through it. Guide, which is well known as a hypertext system, uses such concepts as nesting and inheritance in exactly the same way as structured document systems [3].

The evolution of hypertext towards a more structured model is not surprising. The first step in the development of hypertext has put the emphasis on the *flexibility* of the structure: the objective was to allow links to relate freely any types of data. Then a new trend appeared, based on the first experiments; many researchers considered that a formal representation of the structure was necessary for taking advantage of all possibilities offered by hypertext [19]. Some authors proposed to integrate the results obtained from the study of nets and graphs [20], other ones noticed that links often make a tree structure and that methods for handling trees could be useful in hypertext.

At the same time, the interest for structured documents was growing and the SGML standard [9] was widely accepted. In structured documents, hierarchical structures are very important and, as many hypertext systems also consider this type of structure (for instance, influential systems such as NoteCards [7] and Guide [2]), it seems natural to combine the main features of hypertext and structured documents into the same system. That is what we have done in the system Grif [13]. Grif is well suited for this type of work, as it offers a unique formal model for both hierarchical and non–hierarchical links. This model makes it possible to browse trough a hyper–document very efficiently and it allows authors to structure their hypertext more rigorously.

This paper reports on the experience we have gained in developing and using hypertext functionalities in the context of a system for structured documents. The next section presents the main principles of the system Grif, focusing on the logical structure of documents, independently of any hypertext feature; the graphical aspect of documents is also briefly discussed. Section 3 describes the various types of links available in the document model. Section 4 is dedicated to the user interface: it gives the viewpoint of the end–user who is manipulating links. After this presentation of the features of Grif in the domain of hypertext, several typical applications are presented in section 5 and a conclusion terminates the report.

2 Document structure

Structure is the main characteristic of Grif. As opposed to desktop publishing systems, the purpose of Grif is not (only) to produce a graphical representation of a document, with the sole intent being to print. Although it can print documents,(1) the main objective of Grif is to prepare documents; i.e. to capture as much information as possible for producing various forms of a document, and not only printable forms. For that purpose, it considers documents abstractly and represents them internally as abstract structures. This approach has a number of advantages, which are not presented here in detail (see [1]), one of them being that, with a high-level representation, it is possible to perform a number of operations on a document (printing or displaying being only one of them).

Even if a document is represented *internally* in an abstract way, it is shown to the user in a concrete fashion, as it were printed, thus making the user interface quite intuitive.

2.1 Logical structure

In Grif a document is considered in the same way as in SGML [6][9] or ODA [10]: it is represented as an abstract (or logical) structure which combines such elements as title, author name, abstract, chapter, section, note, paragraph, etc. These types are not hard-wired; a user can define the types he needs. With the SGML version of Grif, the user can write a DTD (document type definition) in the SGML language; with the native version, he uses the S language [16], which is functionally equivalent to SGML, with some additional features. A DTD defines a type of document. It specifies all types of elements that can be used in a document of that type and all possible structural relations between these elements. For instance, the definition of a report says that the document must contain a title, an author name and a variable number of chapters and that a chapter must contain a heading and at least two sections, etc.

DTDs are normally written by document designers, who analyse the needs of an organisation concerning the documents produced and who formalise the result of this analysis in DTDs. These DTDs are then used by those who write and edit documents.

The Grif editor is driven by a DTD. It uses it to help the user produce a structurally correct document, that is a document with a structure that matches the DTD. It *guaranties* that the final document is structurally correct, permitting the user to perform only those editing operations that can lead to a correct document.

This approach may be considered as a strong constraint. In some cases, this constraint is necessary, because the document being written must conform a given editorial standard. In other cases the constraint may be very weak, especially if the DTD makes many elements optional, allows different orders between elements and uses many choices. But even in this latter case, a logical structure defined by a DTD is very useful: the DTD makes it possible to

(1) Obviously, this paper was written with Grif and printed by Grif.

perform a number of treatments, just because the structure of the document is formally defined.

2.2 Physical structure

Among the treatments made possible by the logical structure and the DTD is the presentation. The graphical aspect of a document is generated by the editor from the logical structure, by applying presentation rules to the elements. These presentation rules are gathered together as presentation models, which define the graphical aspect of a document type. Several presentation models may be defined for the same DTD, thus allowing the end–user to see the same document with different layouts, depending on the screen he is using or the type of work he is doing.

In Grif, presentation is taken in a broad sense [14]. It includes not only font selection, character size, spacing, justification, indentation etc., but also component numbering [8], automatic text generation, and multi–view display.

Presentation models are produced by document designers and applied by end–users. A specific language is used for writing presentation models, the language P [16]. For the SGML version of Grif, DSSSL [11] is considered as a potential successor of P.

2.3 Mixing structures

The approach presented above for editing documents is also taken for such components as tables, graphics or equations included in a document. These objects are also defined by DTDs and presentation models specify their graphical aspect. This allows the same system to be used for editing the document itself and all its structured components, whatever their nature. It also brings a high level of integration: it is possible to edit tables, graphics or equations at their normal place within the document and to include elements of different natures within each other without any constraint: a report may contain tables which contain graphics or equations, if such is allowed by the DTD. In addition, elements defined in a given DTD may be used in other DTDs: a footnote, defined in the DTD report, may be included in a table, defined by another DTD. More details on this modular approach can be found in [15].

2.4 Attributes

In the Grif model, the logical aspect of a document is not only represented by its elements (their type and their position in the logical structure), but also by the *attributes*. As in SGML, an attribute is information associated with an element which provides additional semantics to that element. In accordance with the approach taken in Grif, attributes must be declared in the DTD.

The DTD gives each attribute a name and a type chosen among the following: number, enumerate, name, text or reference. It also specifies which attributes can be associated with which type of element (several attributes can be associated with the same type of element). An attribute can be associated with a unique type of element, with several types or even with all types defined in the DTD. As an example, an attribute named Quality and of type enumerate, with values MainAuthor and CoAuthor, can only be associated with elements of type Author: it specifies the role of an author and would have no meaning for other types of element. An attribute named Language can be associated with a number of types of element, for specifying the language in which the content of an element is written.

3 Hypertext structures

The logical structure presented in section 2.1 is basically a hierarchical structure. But this tree structure is not always monolithic. When a document is too big, it is possible to split it into various parts, and so to have a first–level tree structure which assembles the parts, and a second–level structure which defines the organisation of each part. This way of dividing the contents of a document into several chunks and to assemble the chunks in a tree is a first hypertext aspect of Grif which is presented in more detail in section 3.2.

There is another way of considering hypertext in Grif, which is no longer hierarchical: Grif allows links to be established between an element in a document and another element in the same document or in another document. In addition, these links can play different roles. Of course, the generic approach brought by the DTD is also used for links and chunks, which, like other elements, are specified in DTDs. Thus the document model is consistent and its hypertext aspects homogeneous with the rest of the model. Even if this report puts the emphasis on these aspects, they are integral part of the Grif model.

Grif hypertext links are presented here according to four criteria: their type, their scope, their role and their presentation.

3.1 Attributes and elements

As stated above (see section 2.4), different types of attributes are available in Grif. One type of attribute is especially interesting in hypertexts: the type reference. It represents a link between the element with which it is associated and another element. This link, as with all links available in Grif, is directed. The origin is the element with which the attribute is associated and the target is the element designated by the attribute. The type of this second element is specified in the DTD. It can be restricted to a unique type, but it can also be a choice among a list of types or even any type.

Thus, for each reference attribute, the DTD specifies the allowed type(s) of the origin element (by associating this attribute only with this (these) type(s) of element) and the allowed type(s) of the target element.

All links in a document cannot be represented by reference attributes, simply because there is no element with which the attribute could be associated. For instance, a reference to

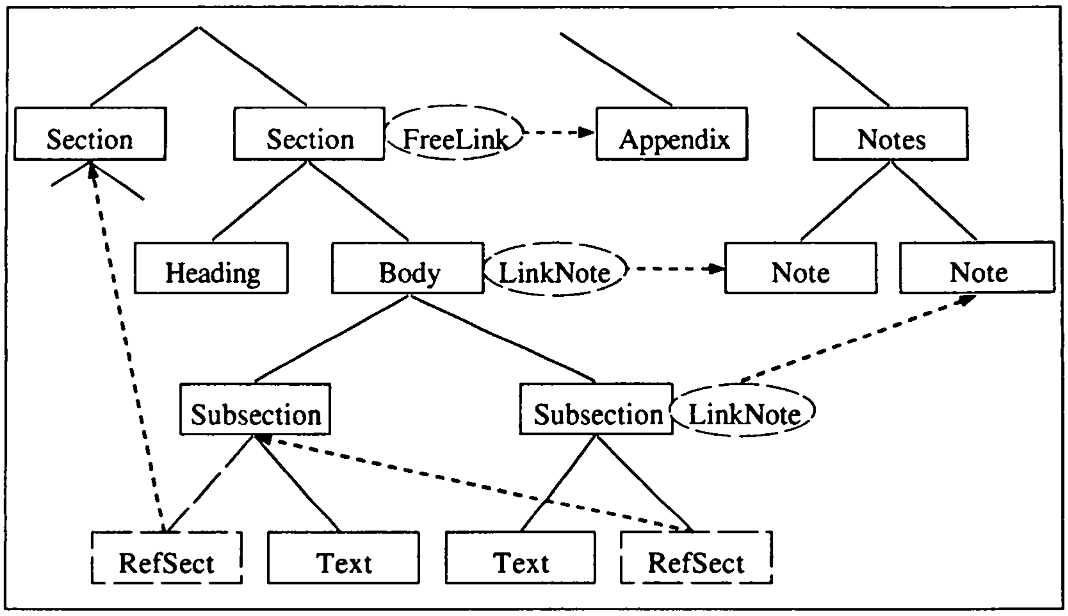

Fig. 1: links in a structured document

a figure (such as "see fig. 3") is not associated with an element; it is itself an element. The reference is *represented* by the character string "see fig. 3", but if there were no reference there would be no such character string. The SGML solution is to create a dummy (empty) element, just for carrying a reference attribute,(2) but another solution is offered by Grif: references are not only attributes, but also elements. Thus, in the document logical structure, the character string "fig. 3" is in fact an element which *is* a reference.

Reference elements are terminal elements in the hierarchical structure of a document, like character strings or pictures. They represent the origin of a link. Like any other element, they are specified in the DTD which indicates where they can appear in the hierarchical structure and what type of element can be the target. The specification of the target is done in exactly the same way as in a reference attribute, so the same flexibility is obtained for the type(s) of the target of a reference element.

Fig. 1 shows an example of these links (in the picture, elements are represented by rectangles, attributes by ovals, relationships of the hierarchical structure by lines, and links by arrows; the links and their origins are in dashed). This figure represents a small part of the logical structure of a document defined by the following DTD (keywords of the S language are in bold):

(2) This is possible in Grif, for compatibility with SGML.

```
ATTR
  LinkNote = REFERENCE(Note);
STRUCT
  Section  (ATTR FreeLink = REFERENCE(Any)) =
                BEGIN
                Heading = Text;
                Body = LIST OF (Subsection);
                END;
  Subsection = LIST OF (BasicElem);
  BasicElem  = CASE OF
                Text;
                RefSect = REFERENCE(AnySection);
                RefFig = REFERENCE(Figure);
                END;
  AnySection = CASE OF
                Section;
                Subsection;
                END;
```

This is only an excerpt of the DTD; the rest defines the types Note, Figure, and other types and attributes which are necessary to make a complete document.

This part defines a reference attribute LinkNote that can only designate an element of type Note. As the definition of LinkNote is not associated with the definition of any structural element, this attribute can be associated with any element of a document built with that DTD. This allows the user to associate notes with any type of element; in Fig. 1, the user has associated this attribute with an element of type Body and with an element of type Subsection.

In the DTD, the attribute FreeLink is associated with the type Section, but it can designate any type of element. The user has associated this attribute with the second section, in order to link that section with the appendix.

According to the DTD, a Section must contain a Heading and a Body, which must contain a sequence of Subsections; a Subsection must contain a sequence of basic elements which may be either a character string (Text), a RefSect or a RefFig. A RefSect is a reference element that can designate either a Section or a Subsection. In Fig. 1, the first subsection of the second section contains a RefSect, as well as the next subsection, and each RefSect points a different type of element.

3.2 Internal and external links, chunks

As we have seen, a link can have the form of an attribute or an element, and both forms of links can be either internal or external. This notion of scope is related to the notion of document: an internal link has its origin and its target in the same document whereas an external link relates elements located in different documents. The question now is to define a document. In the field of electronic documents this notion is rather simple and intuitive. In hypertexts, it is not so obvious.

Grif distinguishes two types of documents: basic documents and compound documents. Compound documents are made up of basic documents; they are presented in section 5.1. A basic document is a piece of information (a chunk), that is accessed as an atom by the system: the editor loads it entirely even if it needs to access only one of its elements. In the Unix version, each basic document is stored in a different file and there is a file for each document. Usually, a short report (like this one) is represented by a unique document, but a book or a dissertation is typically divided into several documents, one for each chapter.

As previously stated, the content of a chunk is considered as a structured document. This implies that each chunk must follow the model specified by a DTD and that the information that constitutes a hypertext must be divided in such a way that each piece is consistent with a DTD (different DTDs may be used by different chunks). Again, this constraint may be useful in some cases, and when it is not necessary, a very general structure may be defined for the chunks; for instance a sequence of elements of any type. More details on the problem of designing DTDs for modular documents can be found in [15].

Constraints imposed by DTDs on links concern only the types of the origin and the target. They are not related to the notion of document. For instance, if two documents are built according to the same DTD, external links between elements of these documents can be set in the same way as internal links. With the DTD of section 3.1, a RefSect can point to an element in the same document or in another document, provided that element is of type Section or Subsection; an attribute Freelink can designate any element in any type of document, as there is no constraint on the type of the referred element.

3.3 Cross–references and inclusions

Until now, a link has only been considered as a relationship between two elements, but no behaviour has been associated with it, only a name (FreeLink, LinkNote, RefSect, etc.). Concerning the behaviour of the system, links are divided into two main categories: *cross–references* and *inclusions.*

A cross–reference only establishes a relation between two elements and does not modify these elements; in the above example, LinkNote and RefSect are typical cross–references.

An inclusion is an actual copy of the target element, which takes the place of the origin element. The target element is considered as being included at the position of origin, while it remains at its own position. This copy is "alive" in the sense that all changes made in the target element are automatically reflected in the copy. This is different from a "dead" copy, produced with the Copy and Paste commands of the editor. Inclusions allow to share information: the same information is available at the origin and at the target. In addition, the same element can be included several times, in several documents.

Links of type inclusion are classified into two categories, which are related to the way the copy is done:

full expansion:	A copy of the whole target element takes the place of the origin element. These links may have two different behaviours: the copy may be done only when printing, or when editing and printing. In the first case, the editor does not copy the target element, but displays it in a different window, on the request of the user. In both cases, when printing the document, the copy of the target element takes the place of the origin.
half expansion:	Only some types of elements contained in the target are copied (these element types are explicitly indicated in their DTD).

Some examples of use of these links are given in section 5.1.

There is a restriction to the use of inclusion links, in that they can only be used as an element, not as an attribute. This is to avoid the conflict between the actual content of the origin element and the copy of the content of the target element: reference elements have no content *per se*.

An important difference between cross–references and inclusions lies in their definition in DTDs. Only cross–references defined in the DTD can be used in a document (section 3.1 gives examples of such definitions). On the contrary, inclusions do not need to be specified in the DTD: inclusion is an operation offered by the editor (see section 4.1). Nevertheless, the consistency of the document with its DTD is preserved: the user can include —create a reference element of type inclusion that points to— an element only if that element has a type allowed by the DTD at the place where it is included (created) in the logical structure of the document. In Fig. 1, it is possible to include a Subsection between the two Subsections of the Body of the second Section, but a paragraph can not be included at that position, because the defintion of the DTD prevents it:

```
Body = LIST OF (Subsection);
```

Nesting of inclusions is possible, even when targets are in different documents: in the above example, the included Subsection can come from another document and it can itself include a text from a third document.

3.4 Presentation of links

The image of a document displayed on the screen or printed on paper is elaborated from its logical structure, following a presentation model. Links, as part of the logical structure, also allow to control the graphical aspect of documents. We have already seen (section 3.3) that inclusions may be displayed in different ways, but presentation models allow to go further.

The difference between cross–references and inclusions is the same for presentation models as for DTDs: nothing specific needs to be done in presentation models for inclusions, whereas the aspect of cross–reference must be specified. The presentation rules applied to inclusions are those related to the target element. This does not mean that the copy looks exactly like the target: if the environment of the inclusion is different from that of the target, its graphical aspect is usually also different, because most presentation rules use inheritance and the inclusion inherits values of presentation parameters from its own environment. Presentation models very often use inheritance for such presentation parameters as font, character size, line spacing, line length, justification, indentation, etc.; thus, these parameters often receive different values for the copy and for the target. Guide uses the same approach [3] for inclusions.

Contrary to inclusions, the graphical aspect of cross–references needs to be explicitly specified in presentation models. This is especially important for reference attributes which would not otherwise be visible. The presence of a reference attribute can be made visible to the user by presentation rules. In a presentation model, different rules can be attached to different attributes(3) so that each attribute can be visually distinguished from each other. These rules can generate a character string (''This element has attribute FreeLink'', for instance), an icon, or graphic (a frame around the element, a bar in the margin, a coloured rectangle in the background of the element, etc.). They can also specify whether the element and/or the generated components is to be visible or not in different views (see section 4); they can modify the value of some presentation parameters set by the type of the element, etc.

Among the presentation parameters, position and dimension have a special interest for reference attributes. As a matter of fact, reference attributes are very useful in complex structures representing graphics. Many drawings illustrating technical reports cannot be represented by a hierarchical structure, but by a set of graphics elements (rectangles, ovals, arrows, segments) and short character strings, with a number of geometrical relationships between these elements: each character string is centered within a rectangle or an oval, each arrow or segment is connected to two rectangles or ovals, etc. In fact, in Grif, such a drawing is represented as a list of elements (graphics elements and strings), each with one or more reference attributes representing these relationships. The names of these attributes are, for instance, HorizontallyCentered, VerticallyCentered, AttachedToTop, AttachedToBottom, etc. In the presentation model used for displaying this complex structure, presentation rules are associated with these attributes for defining the position and dimension of elements relative to the target element of the link attribute. The advantage of this approach is that whenever the user moves or resizes an element, the system adjusts all elements appropriately.

Concerning the contents of the origin element of a link, there is a difference between reference elements and reference attributes. The content of a reference attribute is the content of the element with which it is associated, but, as a reference element is an empty terminal element, it has no content. Therefore a content can be specified for reference elements in presentation models, for making these elements visible. This content can combine

(3) This applies to all types of attributes, references, and also to numbers, names, strings, etc.

constants (character strings, icons, graphics) and variables (restricted to character strings) which depends on the target. As an example, in order to display elements RefFig as "Fig. *n*" (where *n* is the number of the referred figure) and the elements RefSect as an icon (which is contained in the file `IconSect.pic`), the presentation rules should be:

```
FigureNumber:
     Content: Text'Fig. ' Value(FigCounter, Arabic));
Figure:
     Create(FigureNumber);
RefFig:
     Content: Copy(FigureNumber);
RefSect:
     Content: Picture 'IconSect.pic';
```

These rules say that a Figure generates a component called FigureNumber. This component contains a fixed character string ("Fig. ") followed by the value of the counter FigCounter expressed in arabic digits. The rule for the computation of that counter are out of the scope of this report and are not indicated here (see [8] for further details). The content of all elements RefFig is a copy of the component FigureNumber generated by the target element.

3.5 Summary of links

When introducing links in Grif, we had two objectives:

- to extend the system with usual characteristics of hypertext systems,
- to keep the consistency of the model, and especially apply the concept of a generic structure to links and chunks.

For reaching the first objective, we have defined links and chunks in the most general way, avoiding restrictions on their use. So, using reference attributes, the origin of a link can be associated not only with an arbitrary character string, like in many hypertext systems, but also with any element in the logical structure of a document, even with a large element, such as a section or a chapter. In addition, reference elements can be used for anchoring links where there is no element, for instance between two elements.

The target of a link can be an entire basic document (a node or a chunk in the usual terminology of hypertext) or any element whitin a basic document; link can connect elements in the same basic document or in different basic documents. The advantage is that information can be divided into chunks considering only its internal consistency and without paying any attention to the links connecting these chunks. Links can then be established without any constraint and the addition of new links does not require to restructure information.

An element can be the origin of several links; an element can also be the target of several links. Moreover, due to the hierarchical structure of basic documents, an element can be the origin (or the target) of a link, while some parts of it can also be the origin (or the target) of other links.

Links are bidirectional: the system allows users to move from the origin to the target, but also from the target to the origin. It is thus possible to know whether a given element is the target of some links and what are these links.

Links may be used for representing cross-references and inclusion of elements as well.

With all these characteristics, links are very flexible and we think that our first objective is met. But the second objective is somewhat contradictory with the first one, as the problem is to restrict the flexibility, in such a way that only those possibilities chosen by a document designer be available to an author. Our approach to that problem was to offer the broadest range of general mechanisms to the document designer (that is in fact the first objective) and to provide him with appropriate means to restrict the use of these mechanisms, in a DTD, according to the needs of each application.

Due to DTDs, it is possible to modelise (i.e. to type) links very strongly, by assigning the type of their origin element and the type of their target element. But it is also possible to modelise a hypertext without any control, by allowing any type for the origin element and any type for the target element. All possiblities are also offered between these two extremes: the origin and/or the target can be more or less strongly typed. In addition, typed links and untyped links can be defined in the same DTD.

The definition of links in DTDs brings a number of advantages: links and anchors can be modelised and their graphical aspect can be specified in accordance with their type. Various processings may be associated with types, thus defining complete applications based on typed links (the index application presented in section 5.3 is a typical example).

4 User interface

The Grif editor uses a direct manipulation style of interface. The graphical representation of documents, generated according to presentation models, is displayed on the screen and the user interacts with the system through this representation. Mouse, windows, menus, dialogue boxes are the basis of the interface. The editor is interactive: each command issued by the user is immediately executed and its result is simultaneously reflected in the graphical representation of the document. In this respect, it is very similar to many hypertext systems. A difference with some hypertext systems is that the same environment is used for authoring a document and for reading it; even if some functions of the system are not used by the reader.

The graphical representation of a document is usually made up of several views. A view is an image that shows either only certain elements or all of them. It is defined in the presentation model. The elements displayed in a view can be selected according to various criteria: their type, their structural position, the attributes associated with them, the value of these attributes, etc. Each view is displayed in a different window and the user can open or close any view at any time. As an example, for a document like this one, the presentation model defines a view for the whole text, another for the table of contents, another for the bibliography and yet another for figures. Thus, the user can simultaneously see the bibliography and the part of the text where he wants to cite a bibliographic item. He can also see the table of contents of the document and any part of that document at the same time.

A feature that makes multiple views especially useful is synchronisation: clicking on the image of an element in any view causes the other opened views to scroll in such a way that the element be displayed in all views where it can be displayed. This element is also highlighted in all these views, by reverse video or colour background. Of course this mechanism can be disabled selectively for some views should the user not want them to scroll automatically. An example of synchronisation is the use of the table of contents: clicking on a section heading in the view of the table of contents allows one to move quickly to a specific part of a document. This feature is not considered as a hypertext feature, but as a feature related to the logical structure of documents: there is no hypertext link between the heading in the table of contents and the same heading in the body of the document. In fact, there is only one heading in the logical structure of the document, even if this element is displayed in several views.

4.1 Establishing links

There are essentially two different manners to create a link in Grif, one for inclusions and another for cross–references. The difference lies in the creation of the origin; the designation of the target being the same in both cases.

Cross–references are subdivided into two categories, elements and attributes. Reference elements are created in the same way as any other element, by calling the Create menu, which gives the list of all types of elements allowed by the DTD before or after the selected element(s). When a reference element is chosen in that menu, the editor asks the user, with an explicit message, to designate with the mouse an element of the type specified for that reference. When the user has clicked on any part of such an element, the reference appears in the image of the document, in all views where it is visible, with the aspect specified by the rules of the current presentation model. In fact, the only difference with the creation of a non–reference element is that the user must click on the target.

In some cases, it may be difficult to click on the target, for instance when creating in the first section a reference to the last section. The solution to this is provided through the use of views. Any view may be used for designating the target, for instance the table of contents, in this example; if the target can be displayed only in the same view as the origin, it is possible to open a second instance of the same view, that is displayed in a separate window. The two views can be scrolled independently (''desynchronised'') and it is then possible to display the origin in one window and the target in the other one.

Reference attributes are created in the same way as other attributes. The only difference is that the user must specify the target element in the same way as for a reference element.

The user must first select the element with which the reference attribute has to be associated (origin element). Then he calls the Attribute menu, which displays the names of all attributes that can be associated with that element, according to the DTD, and chooses the attribute required. If it is a reference attribute, the system requests that the target be selected and, when this has been done, the origin element is redisplayed in all views, with the presentation specified by the rules of the new attribute.

Inclusions are created by a special command, Include. This command displays the same menu as the command Create, but the result is different: instead of creating an actual element of the type chosen in the menu, the editor creates an inclusion link towards an element of that type designated by the user, and a copy of that element is displayed at the position of the inclusion. The copy is made according to the type of inclusion (partial or complete), and only if the copy must be made by the editor (see section 3.3).

An existing link of any type can be modified at any time. For any modification, the origin element must first be selected. Then, it can be deleted, copied or pasted like any other element. When a link is pasted, the new copy has the same target. The target of an existing link can also be changed, by the command Change Content and by clicking another target element of a type compatible with the link.

To conclude, creation and modification of links are performed in the same way as creation or modification of other elements or attributes, except for the designation of the target.

4.2 Using links

Grif also contains typical features of hypertexts for browsing through a document. Double click is the most common way of using links, whatever their type (element or attribute, internal or external link, cross–reference or inclusion). By double clicking an element that is the origin of a link, the user asks for the target of that link: the editor displays (if it is not already displayed) and highlights the target element. This is similar to the scrolling that occurs when synchronising different views. If the target element is not visible in any opened view, the editor opens automatically a view where this element can be displayed. In the case of an external link, if the basic document containing the target is not loaded, the editor loads it and then opens a view for showing the target element. This unique mechanism for following a link makes the user interface simple and consistent, although there exists a variety of links in the document model.

To allow the user to take advantage of links, they must be clearly visible. In Grif, the links themselves cannot be displayed —the graph constituted by all links of a hyperdocument cannot be drawn—but the origin of all links can be clearly identified. In addition, the system does not impose any graphical aspect to the origin element of a link, because there are many types of links and these types must be visually identified. This explains the need to ensure that the graphical aspect of the origin of a link be specified in presentation models. The document designer is then free to choose the presentation parameters that make the link explicit.

It is also possible to follow a link in the reverse direction, from the target to the origin. The user first selects an element by a simple click and then issues the Search command, with the option "Origin of link"(5). If the selected element is not the target of any link, the editor takes into account the first higher level element in the logical structure that is the target of a link. Then it highlights (and scrolls if necessary) the element that is the origin of the link. Thus, the user does not need to know what element is a target: for instance, if he looks for all cross–references to a given section, he can select a single character in the heading of that section and will find the references required, whether these references point at the Section element or at the Heading.

As an element can be the target of several links, the Repeat option of the Search command allows all elements that are the origin of these links to be find sequentially. When an origin element is in a document that is not loaded, the editor does not load it (to avoid automatically loading too many documents), but displays a message telling the user that there are links starting from such and such documents. The user is then free to open these documents.

Inclusions are displayed as a copy of the target element.(6) But the origin of an inclusion link is not completely equivalent to the target element, as it is not the target element itself. For that reason, an inclusion is displayed with a specific colour(7), the same colour (the user can choose it) for all inclusions. As it has been said above, this copy is updated automatically by the system, and the user cannot modify directly its content: only the target element can be modified. If such a modification has to be made, the user double clicks the inclusion and the target is immediately displayed and selected. He can then edit it.

5 Applications of links

The previous sections present the basic hypertext mechanisms available in Grif. In this section, we describe some applications that have been developed on top of these mechanisms. Some applications are very simple, in the sense that they add very little to the basic mechanisms (or even nothing) while providing valuable services to users; other applications are more sophisticated and have needed some extensions of the Grif editor.

(5) This can be done either with a dialogue box or with function keys. The Search command has many options that allow a user to search within a document according to various criteria: content (character string), type of elements, attribute names, attribute values, etc.

(6) If no expansion is requested, the editor represents the inclusion by the name of the document containing the target element, so that the origin of an inclusion link is always visible.

(7) On monochrome displays, colour may be replaced either by black or by a pattern that simulates grey.

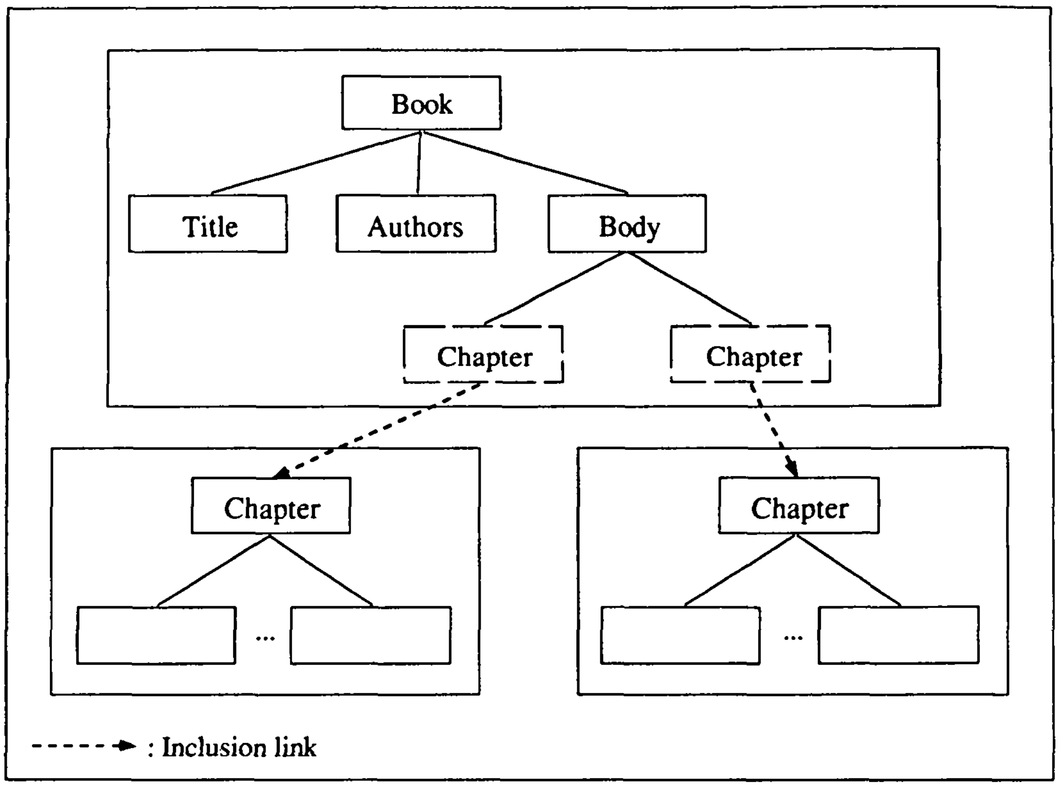

Fig. 2: A super–document

5.1 Super–documents and hyperdocuments

The basic mechanism of links can be used for building two varieties of hypertext structures, called super–documents and hyperdocuments. They can also be used for other applications, some of which are presented in sections 5.2 and 5.3.

A super–document is a document represented by a hierarchical structure that assembles basic documents. It is typically meant for representing large documents, like a technical documentation or an encyclopaedia. It allows a large document to be stored and manipulated more comfortably. The document is divided into parts, on different levels if necessary, and a DTD and at least one presentation model are written for each type of part. Each part, which is a basic document, is linked to another part at the upper level in the super–document structure. In the upper–level document this link is a reference element (a terminal element) of type inclusion, which has the whole lower–level basic document as its target.

Super–documents take advantage of the different types of inclusions (see section 3.3). Half expansion is used, for instance (see Fig. 2), for displaying a table of contents of the whole super–document: the structure of this table of contents is a sequence of inclusions of all parts, and in the DTD of these parts only the elements that the document designer wants to make visible in the table of contents are indicated as part of half–expansions. Thus, the user may have a global view of the document, although it is divided into many basic documents, and the table of contents can be used for accessing these basic documents (see section 4.2).

Each part is a separate basic document and is displayed in a different window.[4] Thus, when editing the document, the user can easily see various parts of the document simultaneously, and can freely open and close any part at any time. If the storing service provides a locking mechanism, several users can work simultaneously on the same super–document, each user loading different basic–documents in write mode (the same basic document may be loaded by different users if it is accessed in read only mode). When the document is printed, all parts are collected, and the super–document appears as one piece, but it is also possible to print each basic document separately.

We have seen in section 3.2 that, for most links, the DTD has nothing specific to say concerning their scope (internal / external); but in some cases, the document designer may need to indicate that a link must be external. Suppose that he wants to impose that a book be divided into chapters, and that each chapter must constitute a separate basic document. He writes the DTD of this book in the following way:

```
Book = BEGIN
       Title = Text;
       Authors = Text;
       Body = LIST OF (Chapter Extern);
       END;
```

The structure of a Chapter is defined in another DTD. Due to the keyword **Extern**, each chapter will be treated as a different basic document and it will be represented in the Book as an inclusion of an element Chapter with no expansion. Without that keyword, it would be possible to make a whole book with a single basic document containing all chapters, although the user would be free to choose to make any chapter an inclusion of another document (as stated above, any element may be replaced by an inclusion of an element of the same type). Notice that the fact that two DTDs (Book and Chapter) are used has nothing to do with the division of the document into several basic documents; this is a modular approach of DTDs (for more detail on the modularity in Grif, see [15]).

Although the example of a book divided into chapters may suggest that basic documents are rather large, it is also possible to define smaller pieces, such as entries in a dictionary or elementary operations in a maintenance manual.

A hyperdocument is a set of basic documents related to each other by any type of links. Its structure is a graph and is very typical of hypertexts. The links may represent cross–references between basic documents as well as inclusions of (parts of) basic documents. Contrary to the structure of super–documents, the structure of hyperdocuments

(4) In fact, several views of the same part can be displayed simultaneously in different windows, as explained in section 4.

is completely free. For this reason, there is nothing in DTDs for specifying a hyperdocument, except the definition of available links.

Although they are presented here as different structures, the notions of super–documents and hyperdocuments may be mixed. A book, presented above as a super–document, may contain references to other documents; it may also contain cross–references linking elements belonging to different basic documents of the book; it may contain elements that are targets of links starting from other documents. It can then be also considered as (a part of) a hyperdocument.

5.2 Simple applications

By simply writing DTDs and presentation models, it is possible to offer interesting hypertext services to the user. This is the way a service of cross–references has been provided to people who edit such documents as articles, reports or books. The DTDs for these classes of documents contain definitions similar to RefSect and RefFig in section 3.1. Thus, these users can let the system handle their cross–references to sections, figures, tables, equations, footnotes, bibliography, etc. It is sufficient to set the links and the system automatically computes and updates the numbers and names that represent these cross–references.

The mechanism of inclusion has been used in various ways for scientific documents. A DTD for dissertations has been written, which considers the document as a super–document and divides it into basic documents: several chapters (defined by another DTD) and a bibliography (another DTD). The bibliography itself can contain original items or inclusions of items coming from another type of document, a bibliographic file, which is shared by a group of researchers working in the same field. These bibliographic files, like those of BibTeX [12] and refer [22], are used in various types of documents, not only in dissertations.

The ability to integrate graphics (such as the drawing of Fig. 1) into documents is provided by reference attributes, as explained in section 3.4. A DTD called Draw has been written which defines a number of reference attributes in order to allow users to express the geometrical dependencies between the elements of a drawing. A presentation model computes position and size of elements from these attributes. If these attributes are correctly used, it is very easy to introduce new elements that need other elements to be moved. By moving only a few elements, space can be left for new elements and the global graphic structure is preserved.

It is also possible to use any graphics editor and to include in a Grif document the result produced by that editor, but the use of Grif itself for editing drawings has a number of advantages. It makes it possible to tailor the ''editor'' to a special type of graphics, by defining new types of graphic relationships in the DTD and by writing the corresponding presentation rules. Graphics may be edited in a much more comfortable way, at their natural place in the document, rather than in a different environment. As the drawing (its structure and contents) is really part of the document, the integration is complete: a global search or replace or a spelling check also processes the elements of the drawings; footnotes and other types of elements of the document can be associated with elements of a drawing; the drawing can inherit some caracteristics from its environment, like fonts or colours, etc.

A prototype file manager for Grif documents has been developed. A DTD defines a Folder as a Title followed by a list of inclusions (without expansion) of documents of any type (including the type Folder); in the corresponding presentation model, each inclusion generates an icon that depends on the type of the included document and the name of that document. Another presentation model displays only the name. Thus all documents of a user can be included in these Folders, and Folders can include other Folders. Folders can be presented in different ways, by changing the presentation model. Using the standard user interface of Grif, the user can open a document simply by double–clicking its icon in a Folder. A document can also be put into a Folder with the standard Insert command.

Although it is useful for accessing Grif documents, this simple application does not provide all the services one would expect from a file manager. For instance, when the representation of a document is deleted in a Folder, that document itself is not deleted, but only its link with the Folder is removed. For extending this application, specific procedures must be written, which perform specific operations in addition to (or instead of) the standard operations performed by the Grif commands. Using this approach, a more complete file manager is under development.

5.3 Complex applications

Like the file manager presented above, more complex applications can be developed which take advantage of the basic services provided by the editor. In this section we briefly present a few of those that are based on the hypertext links of Grif.

A ''table editor'' has been created in the same manner as the ''graphics editor'' described in section 5.2. In fact it is not a separate editor, but a DTD and a presentation model, plus some specific procedures that implement some treatments specific to tables. The advantages of editing tables in that way are the same as in editing graphics. The noticeable difference is that all the structural specificities of a table cannot be described in terms of a DTD, which is the reason why additional procedures are required. As the S language has been designed for representing tree structures with additional non–hierarchical links, it is impossible to completely describe, with that language, the real structure of a table, which is at least two–dimensional [23]. For instance, there is no means for specifying that all rows must have the same number of cells.

Basically, in that DTD, a table is represented as a sequence of column headings followed by a sequence of rows; a row contains an optional heading and a sequence of cells, and each cell has a reference attribute which relates the cell to the heading of its column. Due to this reference attribute, the structure of a column is represented: it is the set of cells that are connected to the same column heading by this reference attribute. This attribute is also used in the presentation model for determining the horizontal position and the width of a cell: they are the same as those of the referred column heading. Specific procedures are nevertheless needed for maintaining the consistency of the structure of a table. When a new row is created, a specific procedure is executed, that creates one cell for each column heading and associates with each cell an attribute linking the cell with the right column heading. Other procedures take care of the deletion or pasting of columns.

Another application that uses links is the ''structural diff''. This application compares the logical structure and the content of two versions of the same Grif document and annotates the second version with attributes indicating the differences with the first version. Some of these attributes are reference attributes that indicate the elements that have been moved in the second version and the previous place of these elements in the first version.

Yet another example of an application using links is the ''electronic index'' presented in [18]. This application handles the indexes that are commonly found at the end of many documents and that allow the reader to access the document by keywords. In this application, for each part of the document that the user wants to be referred by the index, he only creates a keyword element and associates it with the corresponding part of the text. Then specific procedures are executed which collect all keywords, group and sort them and constitute the real index, with a number of links from the index to the text, and from the text to the index. After this treatment, a user can take advantage of all these links for moving across the document very efficiently and in a natural way, as the index is displayed on the screen in the same form as in a paper document; an important difference being that by double–clicking the index, the document is scrolled directly to the required place.

6 Conclusion

This report has presented the hypertext features included in the system Grif, the way to use them and some typical applications. These features have been made possible by the extension of the document model which integrates the notion of link, with a variety of types. The hypertext extension is consistent with the rest of the model in that it supposes an explicit declaration, in DTDs, of the links available to end–users and allows document designers to specify their graphical aspect. With that extension, Grif offers a formal model for hierarchical and non–hierarchical links.

Due to the formal definition of links in DTDs, sophisticated handling of hypertexts is possible, thus offering very powerful commands to end–users. The electronic index application is a good example of such handling [18]. This approach to a formal definition is often opposed to the freedom offered by some hypertext systems, where any link can be set anywhere. During the first step of development of hypertext, it has been claimed that the greatest advantage of hypertext was its ability to organise information in a very flexible way. Experience has shown that it is also a strong disadvantage. In the case of Grif, it is possible to specify free links and chunks when flexibility is required, but it is also possible to define very strongly typed links and chunks, which has proven to be very useful.

The experience gained from the introduction of hypertext structures into a system that is a typical representative of the field of structured documents has shown that there is no major difference between a structured document and a hypertext, when links and chunks are considered as structured objects. The document model has proven homogeneous, and thus the system is easy to use.

Applications presented above have convinced us that merging the fields of structured documents and hypertexts is a promising path of research. Although further work is still necessary, the results we have obtained are of practical interest.

Among the new developments that we are considering, are active links and scripts. It seems that the principles proposed in [24] can be especially valuable in structured hypertexts. Another interesting extension would be to link documents with other types of objects, such as programs. Finally a recent development of the Grif system is worth to be indicated here: an open–ended version of the editor [17] is now available, that allows new applications to use some parts of the editor or to be integrated within the editor; with that new architecture and the hypertext functionalities presented in this report, it is very easy to develop new hypertext applications.

Acknowledgements

We are grateful to Hélène Richy for the early experiments she made with the links while implementing the electronic index; her results helped us to improve the basic hypertext services of Grif. We would also like to thank Cécile Roisin, A.–M. Vercoustre and Malcolm J. Bennett for their valuable comments on an early version of this report.

Bibliography

[1] J. André, R. Furuta and V. Quint, ed., *Structured Documents*, Cambridge University Press, 1989.

[2] P. J. Brown, A hypertext system for UNIX, *Computing Systems*, 2(1), pp. 37–53, 1989.

[3] P. J. Brown, Using Logical Objects to Control Hypertext Appearance, *Electronic Publishing*, 4(2), pp. 109–117, June 1991.

[4] J. Conklin, Hypertext: An Introduction and Survey, *IEEE Computer*, 20(9), pp. 17–41, September 1987.

[5] R. Furuta, V. Quint and J. André, Interactively Editing Structured Documents, *Electronic Publishing – Origination, Dissemination and Design*, 1(1), pp. 19–44, April 1988.

[6] C. F. Goldfarb, *The SGML Handbook*, Clarendon Press, Oxford, 1990.

[7] F. Halasz, Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems, *Communications of the ACM*, 31(7), pp. 836–852, July 1988.

[8] M. A. Harrison and E. V. Munson, Numbering Document Components, *Electronic Publishing*, 4(1), pp. 43–60, March 1991.

[9] I.S.O., *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*, (ISO 8879), 1986.

[10] I.S.O., *Information processing – Text and office systems – Office Document Architecture (ODA)*, (ISO 8613), 1989.

[11] ISO/IEC, *Information Technology–Text and office systems–Documents Style and Semantics and Specification Langage (DSSSL)*, ISO/IEC DIS 10179, 1991.

[12] L. Lamport, *LaTeX: A Document Preparation System*, Addison–Wesley Publishing Company, Reading, Massachusetts, 1986.

[13] V. Quint and I. Vatton, Grif: an Interactive System for Structured Document Manipulation, *Text Processing and document Manipulation, Proceedings of the International Conference*, edited by J. C. van Vliet, pp. 200–213, Cambridge University Press, 1986.

[14] V. Quint and I. Vatton, An Abstract Model for Interactive Pictures, *Human–computer interaction, Interact'87*, edited by H.–J. Bullinger and B. Shackel, pp. 643–647, North–Holland, September 1987.

[15] V. Quint and I. Vatton, Modularity in structured documents, *Woodman'89*, edited by J. André & J. Bézivin, pp. 170–177, Bigre num. 63–64, IRISA, Rennes, mai 1989.

[16] V. Quint, *Les langages de Grif*, internal report (in French), May 1992.

[17] V. Quint, *Extending a structured document editor*, (in preparation), research report, INRIA, 1992.

[18] H. Richy, Electronic Index in Grif, *submitted to ECHT'92*, 1992.

[19] A. Rizk, N. Streitz and J. André, eds., *Hypertext: concepts, systems and applications*, proceedings of the European Conference on Hypertext, Cambridge University Press, November 1990.

[20] P. D. Stotts and R. Furuta, Petrinet–based hypertext: Document structure with browsing semantics, *ACM Transactions on Information Systems*, 7(1), pp. 3–29, January 1989.

[21] P. D. Stotts and R. Furuta, Hypertext 2000: Databases or Documents?, *Electronic Publishing*, 4(2), pp. 119–121, June 1991.

[22] *refer–A Bibliograhy System*, Unix documentation.

[23] Ch. Vanoirbeek, Formatting Structured Tables, *EP 92*, edited by Ch. Vanoirbeek and G. Coray, pp. 291–309, Cambridge University Press, April 1992.

[24] P. T. Zellweger, Scripted Documents : A Hypermedia Path Mechanism, *Proceeding Hypertext '89*, special issue *SIGCHI Bulletin*, pp. 1–14, 1989.

ISSN 0249 - 6399