Vers un atelier éditorial pour les documents structures

Jacques André, D. Decouchant, Vincent Quint, Hélène Richy

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Vers un atelier éditorial pour les documents structurés

Jacques André, Dominique Decouchant, Vincent Quint et Hélène Richy

N° 1971

Mars 1993

PROGRAMME 3

Intelligence artificielle,
systèmes cognitifs
et interaction homme-machine

ISSN 0249-6399

Vers un atelier éditorial pour les documents structurés

Jacques André [*], Dominique Decouchant [**], Vincent Quint [***] et Hélène Richy [****]

Programme 3 — Intelligence artificielle, systèmes cognitifs
et interaction homme-machine
Projet Opéra

Rapport de recherche n° 1971 — Mars 1993 — 14 pages

Résumé : Grif est un système interactif pour la production et la consultation de documents structurés professionnels, notamment ceux de la documentation technique ou de l'édition. Après avoir rappelé les principes de base de ce système, nous citons quelques-unes des tâches et des problèmes spécifiques aux milieux éditoriaux (documents multi-auteurs, documents composites ou secondaires, travail coopératif, réutilisation, gestion de versions, qualité, etc.) que nous comparons à ceux du génie logiciel.

Nous montrons alors comment Grif peut être vu comme une première étape vers la définition d'un « atelier éditorial ».

Mots-clé : Grif, atelier éditorial, travail coopératif, documents structurés, hypertexte, SGML.

(Abstract: pto)

Cette note doit paraître dans les *Actes du congrès Afcet*, Versailles, juin 1993.

∗ Irisa/Inria, Jacques.Andre@irisa.fr
∗∗ CNRS, Bull-Imag, 2 rue de Vignate, 38610 Gières, decoucha@imag.fr
∗ ∗ ∗ Inria Rhône-Alpes, Bull-Imag, quint@imag.fr
∗ ∗ ∗∗ Irisa/CNRS, Helene.Richy@irisa.fr

Unité de recherche INRIA Rhône-Alpes
46 avenue Félix Viallet,
38031 GRENOBLE Cedex 1
Téléphone : (33) 76 57 47 77
Télécopie : (33) 76 57 47 54

Unité de recherche INRIA Rennes
IRISA, Campus de Beaulieu,
35042 RENNES Cedex
Téléphone : (33) 99 84 71 00
Télécopie : (33) 99 38 38 32

Towards structured document engineering factory

Abstract: Grif is a system dedicated to interactive preparation, modification and editing of structured documents. It is specifically designed for professional documents, such as technical documentation or commercial books. In this paper, the Grif system is first described. Then, specific problems posed by the considered documents are quoted (multi-author documents, composite documents, cooperative editing, reusability, version control, quality, etc.) and the way Grif solves these problems is exhibited. With these capabilites, Grif may be considered as an environment for "document engineering" that provides documents with the same kind of services as "software engineering" environment provides to programs.

Key-words: Grif, document engineering, cooperative work, structured documents, hypertext, SGML.

1 Introduction

Grif est un système de production de documents principalement destiné à la documentation scientifique et technique et, d'une façon plus générale, aux domaines où l'on manipule des documents fortement structurés [3][4]. À la différence des logiciels de PAO, Grif ne s'intéresse pas uniquement à la forme graphique des documents. C'est un éditeur structural, dont les concepts de base se comparent plutôt à ceux d'un éditeur syntaxique.

Les premiers travaux qui ont conduit au système Grif ont commencé en 1982 [6] ; ils ont débouché sur un prototype en 1985 pour donner un outil réellement utilisable en 1988 [11] [17]. Grif utilise le système de fenêtres X sous Unix. Un transfert industriel a donné lieu à un produit commercial [1] qui est probablement le premier éditeur SGML convivial. Les recherches qui se pousuivent [20] se font sur ce prototype et sont en général transférées vers le produit commercial.

Nous décrivons ici principalement les possibilités de Grif en matière de génie éditorial. Après avoir montré quelques-uns des problèmes de ce domaine (section 2), nous entrons plus en détail dans la structure générale et les outils éditoriaux de Grif (section 3) puis dans les traitements concernant les références et index (section 4), les transformations de structures (section 5), la gestion de versions et l'édition coopérative (sections 6 et 7). En conclusion, nous essayons de montrer ce que peuvent être les rapports entre génie logiciel et génie éditorial.

2 Génie éditorial

La documentation technique et l'édition sont des secteurs d'activité très anciens, mais l'utilisation de l'informatique y est relativement nouvelle. Par ailleurs, il ne s'agit pas uniquement de composer ou mettre en page des rapports ou des romans mais aussi des documents techniques volumineux, des revues (avec de nombreux auteurs), des encyclopédies et dictionnaires (avec de nombreux renvois d'une entrée à l'autre), etc. Sans vouloir être limitatif, nous nous intéressons, ici, à la difficulté de fabrication et de mise à jour de tels documents.

1. Le nom Grif est associé à 3 choses différentes : 1) un prototype en cours de développement à l'INRIA dans le cadre du projet Opéra (à Grenoble et Rennes), 2) un produit commercial issu du précédent et 3) la société Grif SA, qui commercialise ce dernier (Grif SA, 2, bd Vauban, BP 266, 78053 St Quentin en Yvelines Cedex). Nous ne parlons ici que du prototype 1.

Le néologisme « génie éditorial » se veut le pendant de l'expression « génie logiciel » pour les tâches liées à la documentation technique, à l'édition et de façon plus générale aux activités centrées sur les documents lorsque ceux-ci sont « gros » ou/et complexes et lorsque le nombre de personnes impliquées est grand. Plutôt que de donner une définition imparfaite, prenons plutôt quelques exemples.

2.1 Documentation technique

Un premier exemple, naïvement simplifié, est celui de la documentation d'un avion.

1. C'est quelque chose de très volumineux [2] qui concerne des équipes entières de rédacteurs, lesquelles doivent pouvoir travailler sur les mêmes documents, sans conflits. Il faut donc qu'un tel système de documentation accepte le travail coopératif avec toutes les garanties de sécurité voulues (voir section 7).

2. Il ne s'agit pas de *la* documentation d'un avion, mais de *plusieurs* documentations [3] selon que l'on s'adresse à un pilote, un électricien, un employé de maintenance, un opérateur au sol, etc. Mais très souvent les contenus sont pratiquement identiques : seule change la forme. Il convient donc qu'un système de documentation technique permette de passer facilement d'une maquette à une autre, de manipuler un même texte sous diverses structures (voir section 5).

3. Ces divers documents doivent faire référence les uns aux autres. Des liens doivent donc pouvoir être établis depuis un document vers tout autre, mais ils doivent aussi être tenus à jour. En particulier, si la poignée de porte d'une soute à bagage change de modèle, donc de numéro, ceci doit apparaître dans tous les documents, y compris dans les figures (voir sections 4 et 6).

2. Il est de coutume de citer qu'un Airbus ne pourrait pas transporter toute sa documentation papier. Ceci montre aussi l'intérêt des recherches sur la documentation technique « zéro papier », qui n'a de sens que si, parallèlement, se développent des recherches sur la lisibilité des écrans mais aussi sur la communication homme-machine ou l'ergonomie de la GED (gestion électronique des documents).

3. Dans le cas des avions, la norme internationale ATA-100, définit les divers types de documents indispensables et leurs structures.

4. Il y a quatre façons de consulter un document technique : la façon séquentielle, et on conçoit que ce ne soit que rarement le cas ; une façon structurée, en se servant de la table des matières (d'où l'intérêt des « vues » de Grif) ; l'utilisation de renvois ; et enfin la recherche par mots-clés ou index. D'où l'importance d'outils de construction de renvois et tables d'index et de mécanismes hypertexte pour l'accès aux informations, comme par exemple la possibilité de cliquer sur une entrée de la table d'index pour voir apparaître le texte correspondant (voir section 4).

5. Enfin, chaque modèle d'avion évolue rapidement, à tel point qu'il n'y a probablement pas deux Airbus identiques. Chaque avion est donc une version d'un modèle plus général et a en conséquence sa propre version de documentation. Gérer les versions est donc une obligation en documentation (voir section 6).

2.2 Qualité éditoriale

Prenons maintenant un style de documents très différents : une revue de sciences humaines qui publie tous les mois une dizaine d'articles de dix pages chacun. La PAO permet aujourd'hui de faire beaucoup pour ce genre de revues et on pourrait croire qu'il n'y a plus de problèmes. Mais si un éditeur veut une qualité professionnelle (tant au niveau de la mise en page que de la correction typographique), il n'a aujourd'hui que trois solutions :

1. Faire appel, comme autrefois, aux services de correcteurs professionnels. Mais ceux-ci sont de plus en plus rares. Par ailleurs, leur tâche est souvent ingrate (lecture de centaines de pages ligne par ligne, lettre par lettre) et nécessite rigueur et compétence (notamment la connaissance du code typographique !).

2. Accepter un patchwork d'articles hétérogènes, mal corrigés et dont le secrétaire de rédaction ne pourra produire aucun index global (sauf manuel avec les risques d'erreurs bien connus) ni faire de références d'un article à l'autre. Même les sommaires risquent d'être plein d'erreurs dues à des resaisies.

3. Travailler avec un formateur « intelligent » qui permette de produire des index[4] et des références entre plusieurs documents. Où l'on puisse lancer des commandes du type « Supprimer les points parasites au bout des titres », ou « Supprimer les capitales inutiles », après avoir défini les règles maison ; etc.

Les formateurs classiques ont fait quelques progrès dans ce sens, notamment en intégrant des vérificateurs orthographiques ou en proposant diverses possibilités de micro-typographie. Mais il reste beaucoup à faire pour arriver à un niveau plus professionnel. Dans le cas de Grif, signalons qu'outre un vérificateur orthographique [22] et la construction d'index hypertexte (voir section 4.2), nous cherchons quelles sont les connaissances linguistiques et typographiques à donner à un tel système pour que l'on puisse profiter de sa structuration et de ses outils d'édition (voir section 3.2) pour diriger des vérifications typographiques.

3 Le système Grif

Grif est un éditeur structuré dont les concepts sont très proches de ceux de SGML [14].

3.1 Structures logiques et physiques des documents

Tous les traitements que Grif peut effectuer s'appuient sur un modèle de document de haut niveau. Pour Grif un document est d'abord une *structure logique* qui assemble des éléments tels que des titres, chapitres, sections, paragraphes, notes, figures, etc. Cette structure est principalement arborescente, mais pas uniquement, comme on le verra section 4. Les éléments qui la composent, ainsi que les assemblages possibles de ces éléments, sont définis dans des *schémas de structure,* chaque schéma correspondant à une classe de documents. Les schémas de structure définissent aussi les *attributs* qui peuvent être associés à certains types d'éléments, pour en préciser le rôle dans la structure du document. À titre d'exemple, le schéma de structure de la classe lettre définit les éléments adresse,

4. Typiquement, la création d'un index est quelque chose dont la place dans la chaîne éditoriale n'a jamais été très claire ; or, c'est plutôt aux auteurs de créer ces index ; par ailleurs, certains index, comme en droit ou histoire des textes, peuvent être plus volumineux que le texte lui-même. D'où l'importance de disposer d'outils appropriés.

date, référence et paragraphe et stipule qu'ils doivent apparaître dans cet ordre, une seule fois chacun, sauf les paragraphes qui peuvent être multiples. Pour la classe rapport, le schéma de structure définit d'autres types d'éléments (titre, auteur, résumé, section, sous-section, etc.) et d'autres règles d'assemblage. Un langage, appelé S, permet d'écrire les schémas de structure selon les besoins. Ces schémas ont donc le même rôle que les DTD *(Document Type Definition)* de SGML. Ajoutons qu'ils permettent de définir des documents au sens usuel, comme les lettres ou les rapports, mais aussi des objets structurés comme des dessins, des tableaux ou encore des formules mathématiques [18].

À chaque type d'élément défini dans un schéma de structure est associé un ensemble de *règles de présentation* qui définissent l'aspect graphique de ce type d'élément. C'est en appliquant ces règles aux éléments de la structure logique qu'on construit l'image du document. Un ensemble de règles de présentation définissant l'aspect graphique de tous les types d'éléments d'une classe de documents est appelé un *schéma de présentation*. Les schémas de présentation sont écrits dans un langage appelé P, qui permet aux maquettistes de spécifier leurs propres présentations. Ce langage joue le même rôle que jouera DSSSL [15] pour SGML.

Ce modèle de document a été retenu pour permettre des traitements variés et puissants, qui aillent bien au-delà de la simple restitution sur écran ou sur papier, comme on le fait en PAO. Les sections suivantes le montreront. Ces possibilités de traitement viennent d'une part de la structuration logique, mais aussi des schémas de structure. La structuration en éléments logiques permet d'identifier clairement les parties de document sur lesquelles vont s'appliquer les traitements, mais les schémas de structure permettent en plus de savoir et de prévoir les positions structurales relatives de ces éléments : ils donnent une certaine sémantique au document.

3.2 Éditeur et boîte à outils d'édition

Grif se présente comme un éditeur dirigé par les schémas de structure, qui assiste et guide l'utilisateur dans la construction d'un document conforme au schéma de structure choisi. Il est conceptuellement comparable à un éditeur syntaxique dirigé par la syntaxe d'un langage de programmation. Une différence importante vient des schémas de présentation qui offrent une grande variété de structures graphiques, autorisant l'affichage des documents complexes (comportant graphiques, tableaux, formules) avec toute la richesse (typo)graphique nécessaire.

Le noyau central de Grif manipule les structures logiques et graphiques des documents en suivant les schémas de structure et de présentation. L'éditeur est formé de ce noyau, complété d'une interface utilisateur construite sur OSF/Motif. Pour autoriser le développement de nouvelles applications liées à l'éditeur, une « interface programmatique » (API) est disponible. Elle offre au programmeur d'application l'accès à toutes les fonctions du noyau de l'éditeur et permet à des programmes d'effectuer le même type d'opérations sur les documents que celles proposées à l'utilisateur de l'éditeur.

Grâce à cette interface de programmation, le noyau de l'éditeur se présente comme une boîte à outils qui rend possible deux types d'applications :

- des applications intégrées, vues par l'utilisateur comme des outils complétant l'éditeur et accessibles depuis l'éditeur (correcteur orthographique ou typographique, outil de traitement des index, par exemple) ;
- des applications indépendantes de l'éditeur, qui n'utilisent que le noyau et effectuent des traitements automatiques sur les documents (comparaisons ou transformations de documents, par exemple).

4 Références, liens hypertexte et index

4.1 Références et liens

La structure logique hiérarchique des documents n'est pas suffisante pour représenter des références entre éléments. La notion de lien a été introduite dans Grif pour donner aux documents une dimension hypertexte [19]. Ceci permet d'établir des liens entre des éléments appartenant soit au même document soit à des documents différents. Ces liens sont orientés et sont représentés dans la structure logique soit par des élements de type référence, soit par des attributs de type référence.

Par exemple, pour représenter un renvoi à une figure tel que « voir fig. 5 », Grif utilise un élément de type référence défini dans la structure logique du document. Cet élément représente l'origine d'un lien dont la cible est une figure. Lors de la saisie sous l'éditeur, il suffit de cliquer sur la figure en question (numérotée automatiquement, bien sûr) pour mettre le numéro de la figure en référence, puis, lors de la consultation, de cliquer sur chaîne « voir fig. 5 » pour faire apparaître la figure.

Ces types de liens sont spécifiés dans le schéma de structure qui indique où peut apparaître l'élément dans la structure hiérarchique du document et quel type d'élément peut en être la cible. L'approche générique (DTD) qui a été suivie pour la définition de ces liens permet de typer fortement certains liens en contraignant les types d'éléments source et cible. Mais il est également possible de définir des liens hypertexte quelconques, entre n'importe quels types d'éléments, ou encore une liste des types autorisés comme cible.

Deux types de comportement sont proposés pour ces liens qui mettent en relation une source et une cible. Ils peuvent se comporter soit comme des *références croisées*, soit comme des *inclusions*. Une référence croisée se contente d'établir une relation entre des éléments sans les modifier ; c'est le cas d'un renvoi vers une figure, une note ou une section, par exemple. Une inclusion est en fait une copie « vivante » de l'élément cible, qui prend la place de l'élément origine : toute modification de l'élément cible est répercutée sur cette copie. Par contre, les deux copies peuvent apparaître sous une présentation différente : chaque inclusion respecte les règles de présentation locales en vigueur.

4.2 Une application des liens : les index

Des applications nouvelles, fondées sur des liens typés, ont été définies en utilisant Grif. Les index électroniques sont un exemple d'utilisation de ces liens. Cette application [21] permet de produire des index de documents analogues à ceux qui figurent à la fin de nombreux ouvrages et qui permettent au lecteur d'accéder au contenu du document à partir de termes sélectionnés. À l'aide de cette application intégrée à l'éditeur, l'auteur d'un document sélectionne les passages du document qu'il souhaitera référencer depuis un index. Il lui suffit de préciser le terme qui sera associé à chacun de ces passages. L'application se charge ensuite de fusionner tous ces termes, de les trier et de construire effectivement la table d'index, en y introduisant des liens entre index et contenu du document : l'utilisateur peut ensuite utiliser ces liens hypertexte pour se déplacer efficacement dans le document.

La consultation de tables d'index est naturelle, car ces tables sont présentées sous la même forme que dans les ouvrages imprimés. Mais l'avantage de cet index électronique est d'offrir en plus la possibilité de « naviguer » dans le document en suivant les références indiquées dans l'index : après avoir cliqué sur une entrée de l'index, l'utilisateur voit s'afficher sur son écran, dans une autre fenêtre, le passage correspondant ; le document est immédiatement positionné sur le passage désiré.

Cette application est intégrée à l'éditeur, de sorte qu'elle est disponible en permanence et pour tous les utilisateurs, quelle que soit la structure générique du document. L'auteur peut à tout moment ajouter ou modifier un terme dans l'index, indexer un nouveau passage, etc. ; lorsqu'il le souhaite, les tables d'index sont recalculées par l'application. Le lecteur peut localiser très rapidement les passages qu'il cherche, mais l'auteur, pendant qu'il écrit, peut aussi utiliser l'index existant pour se repérer dans son texte.

5 Transformations de structures

La structure logique d'un document peut évoluer au cours du temps. Ainsi, lors de la copie d'une partie d'un document dans un autre document (ou dans le même), la structure logique de cette partie peut ne pas être conforme au schéma de structure qui définit l'endroit où on veut l'insérer ; il faut alors transformer sa structure. Lorsqu'on modifie un schéma de structure, les documents qui ont été produits précédemment selon ce schéma ne sont généralement plus conformes à la nouvelle version et il faut alors les restructurer, si on veut les manipuler selon le nouveau schéma. Un autre cas est celui d'un document qui a été saisi sans structure, avec un traitement de texte par exemple, et qu'on souhaite entrer dans le système ; il faut alors le structurer selon un schéma.

Tous ces problèmes de restructuration revêtent une grande importance dans un environnement de traitement de documents structurés [12]. La structuration offre de nombreux avantages, mais elle ne doit pas constituer un frein à l'évolution des documents. C'est pour cette raison que différents outils ont été développés dans Grif, qui prennent en charge les transformations de structure.

Une étude a d'abord recensé l'ensemble des problèmes de transformation de structure qui se posent dans des documents structurés logiquement et a dégagé les questions de base [2]. Ensuite, un premier outil a été créé pour les documents traités par Grif [1] : un programme appelé *comparateur* compare deux versions d'un schéma de structure et en déduit les règles de transformation à appliquer aux documents construits selon l'ancienne version pour qu'ils deviennent conformes à la nouvelle. Un deuxième programme, le *convertisseur*, applique ces règles et met ainsi automatiquement à niveau les anciens documents. Un autre outil est en cours d'intégration dans l'éditeur, pour restructurer « au vol » les parties de documents soumises à la commande Coller. Avec ces outils, on s'affranchit d'une grande partie des contraintes imposées par les documents structurés.

6 Gestion de versions

Tout comme les logiciels, les documents évoluent et donnent lieu à des versions successives, voire parallèles, qui doivent souvent être conservées. Des outils adaptés sont nécessaires pour gérer ces versions.

Ces outils pour les documents sont très proches de ceux utilisés en génie logiciel pour les programmes. La fonction de base est la comparaison de deux versions d'un même document. Cela permet d'identifier les différences et de ne conserver intégralement qu'une des versions, les autres étant gardées sous la forme de leurs différences par rapport à cette version de référence.

Pour les documents produits par Grif, nous avons développé les fonctions de base, en nous appuyant sur la structure logique des documents [7]. Les différences entre versions sont exprimées en fonction de la structure logique. Les résultats de la comparaison de deux versions sont du genre : « ces deux paragraphes ont été permutés, une section a été ajoutée avant celle-ci, le texte de ce titre a été modifié », etc. Cela donne à l'utilisateur une bien meilleure perception de l'évolution des documents que de simple barres de révision. Cela simplifie également l'utilisation des fichiers de différences pour reconstituer une version : le fichier de différence est vu comme un ensemble de commandes d'édition structurales à appliquer à la version de référence pour obtenir une autre version.

7 Édition coopérative

Un éditeur tel que Grif ne résout qu'une partie du problème plus général du traitement des documents, surtout lorsqu'il s'agit de documents complexes, comme ceux de la documentation technique. Un problème important est celui du développement [5] d'une grosse documentation par plusieurs rédacteurs. C'est pour aborder ce problème que nous avons développé l'application Griffon [8].

Cette application permet à plusieurs utilisateurs, sur des stations de travail différentes reliées en réseau, d'éditer simultanément un ensemble de documents. C'est typiquement une application de travail coopératif [9], adaptée à l'édition des documents dans un atelier éditorial. Griffon vise l'édition *coordonnée* et interactive de documents et non la fusion a posteriori de différentes versions d'un même document. Les outils de gestion de versions de documents se sont jusqu'à

5. Le mot développement doit ici être pris dans le même sens que lorsqu'il s'agit du développement d'un logiciel.

présent intéressés à intégrer dans un même document les différentes modifications produites concurremment par plusieurs rédacteurs travaillant *indépendamment*. Ce qui nous intéresse, avec l'application Griffon, c'est d'assurer en continu l'intégration des modifications de tous les rédacteurs. En fait, les deux approches sont complémentaires aussi travaillons-nous à la coexistence des deux outils.

Un atelier éditorial doit veiller à l'intégrité de l'information contenue dans les documents et pour cela, il doit partager entre les utilisateurs l'accès aux éléments constituant les documents. Pour faciliter le partage, mais aussi pour garantir la pérennité des documents, nous utilisons la structure logique.

L'application Griffon permet d'exprimer et de modifier la façon de partager chaque document : un document peut être subdivisé en plusieurs unités de taille variable (*fragments*) sur lesquelles sont définis les rôles des différents utilisateurs autorisés (gestionnaire du partage, rédacteur, lecteur, rôle nul). Ces rôles sont exploités durant les phases d'édition et la cohérence des contributions des différents utilisateurs est assurée. Chaque utilisateur qui coopère à l'édition d'un document a une perception personnelle du document, et plus précisément des fragments qui composent le document. La présentation permet à tout utilisateur d'identifier le rôle qu'il joue dans l'édition d'un fragment de document : il peut spécifier un attribut de présentation (par exemple une couleur donnée) pour mettre en évidence les parties qu'il possède en lecture seule (rôle de lecteur).

Grâce à ces mécanismes, le responsable d'une équipe de rédaction peut distribuer dynamiquement des rôles d'intervention aux co-rédacteurs et il peut même déléguer partiellement ce pouvoir de gestion. Griffon gère automatiquement la cohérence du document, la protection et la confidentialité lors de l'accès aux parties du document. Il gère aussi la diffusion des contributions de chaque utilisateur : les modifications faites par un rédacteur sont reflétées sur les écrans des autres utilisateurs dès qu'elles sont validées.

Par rapport aux différentes expériences faites dans le domaine du « collecticiel », Griffon explore une voie originale concernant la coopération. En effet, la plupart des projets dans le domaine considèrent des architectures appartenant à l'une des classes suivantes :

– Les informations à partager sont dupliquées sur tous les sites (comme dans Grove [9]) et la cohérence des modifications est obtenue par la sérialisation des accès.

– Les informations partagées sont stockées dans une base de données (comme dans Sepia [23]) et on y accède par des primitives bien établies (transactions, verrouillages) depuis les stations distantes.

Griffon vise à fournir un service d'édition concurrente avec lequel les interactions entre utilisateurs sont fréquentes et contrôlées : tous les utilisateurs n'ont pas les mêmes prérogatives (notion de rôle). Pour cette raison, la notion de sérialisateur destinée à régler les problèmes de modifications concurrentes n'est pas adaptée. La solution fondée sur un serveur d'accès aux informations partagées n'est pas non plus adaptée à l'architecture de notre système : un réseau de stations sur lesquelles on désire traiter localement les fragments de documents. Au contraire, les fragments sont chargés sur chaque site en respectant le rôle de l'utilisateur, ils sont modifiés localement, puis sauvegardés lors de la validation des modifications.

La structuration forte des documents manipulés par Griffon renforce ce choix ; le partage peut alors s'exprimer sur des fragments de taille très variable sur lesquels les rôles d'intervention des utilisateurs peuvent être modifiés dynamiquement.

Un premier prototype de l'application Griffon développé au-dessus du système réparti à objets Guide [5] est actuellement opérationnel. Il utilise la boîte à outils d'édition de Grif. Une version Unix est prévue.

8 Conclusion

Génie éditorial et génie logiciel présentent des similitudes et un produit comme Grif s'apparente quelque peu à des logiciels comme Ada ou Eiffel qui ont le même souci d'aide à la production de gros produits par des équipes nombreuses. En particulier, de même que l'on a longtemps confondu l'écriture d'une application dans un langage de programmation spécifique (par exemple Fortran) et la conception rigoureuse de cette application (spécification, preuve, gestion des versions, etc.), de même on a tendance encore à confondre aujourd'hui le formatage d'un document (par exemple en Word) avec sa conception, comme si le formatage était la seule chose importante dans un document ; [18] et [13] introduisent la notion de « granularité » et [10] montre l'implication de ces ressemblances dans les problèmes d'enseignement.

Mais, comme nous l'avons déja montré ailleurs [16], il y a aussi de fortes dissemblances : en particulier, les éditeurs de programme ne sont pas adaptés à la manipulation de documents car on ne peut pas dissocier complètement un

document de son aspect visuel [6]. C'est d'ailleurs la grosse ambiguïté des systèmes Wysiwyg qui, eux, donnent la priorité au visuel ! Ceci explique aussi la difficulté actuelle à définir des produits comme DSSSL [15] qui restent abscons pour des graphistes et pour les fonctionalités desquels les informaticiens n'ont pas toujours la formation graphique voulue.

Il n'en reste pas moins que le génie éditorial a beaucoup à apprendre du génie logiciel lequel a quand même une bonne dizaine d'années d'antériorité. Le prototype de recherche Grif a donc encore un bel avenir devant lui.

Remerciements Nous tenons à remercier tous ceux qui ont participé à ce projet et qui n'ont pas signé ce document, en particulier E. Akpotsui, F. Burgnard, A. Denys, C. Roisin et I. Vatton à Grenoble et Ph. Louarn et É. Picheral à Rennes.

Références

[1] E. Akpotsui, V. Quint, « Type Transformation in Structured Editing Systems », *Proceedings of Electronic Publishing 1992, EP92,* C. Vanoirbeek et G. Coray, ed., 27–41, Cambridge University Press, avril 1992.

[2] E. Akpotsui, V. Quint, C. Roisin, « Type Modelling for Document Transformation in Structured Editing Systems », *First International Workshop on Principles of Document Processing,* Washington, octobre 1992.

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

[4] J. André, V. Quint, « Structures et modèles de documents », *Le document électronique,* C. Bornes, ed., 1–57, Inria, juin 1990.

[5] R. Balter, J. Bernadat, D. Decouchant, A. Duda, A. Freyssinet, S. Krakowiak, P. Ledot, M. Meysembourg, H. Nguyen Van, E. Paire, M. Riveill, C. Roisin, X. Rousset de Pina, R. Scioville, G. Vandôme, « Architecture and implementation of Guide, an object-oriented distributed system », *Computing Systems,* vol. 4, num. 1, 31–67, winter 1991.

6. Même si nous donnons l'impression d'ignorer cet aspect : rappelons que, dans cet article, nous nous sommes limités aux aspects non-visuels des documents.

[6] G. Bogo, H. Richy, I. Vatton, « Un modèle de représentation de documents généralisés », *Actes des journées sur la manipulation de documents*, J. André, ed., 221–235, IRISA, Rennes, mai 1983.

[7] F. Burgnard, A. Delfosse, « *Comparaison de Documents Structurés et Applications* », Rapport de stage 3e année ENSIMAG, Unité Mixte Bull-Imag Systèmes, 2, rue de Vignate F-38610 Gières, juin 1992.

[8] D. Decouchant, V. Quint, I. Vatton, « L'édition coopérative de documents avec Griffon », *Colloque IHM'92, quatrièmes journées sur l'ingénierie des interfaces homme-machine*, 137–142, Paris, 30 novembre – 2 décembre 1992.

[9] C. A. Ellis, S. J. Gibbs, G. L. Rein, « Groupware, some issues and experiences », *Communications of the ACM*, vol. 34, num. 1, 38–58, janvier 1991.

[10] *EPODD, Electronic Publishing – Origination, Dissemination and Design*, numéro spécial « Teaching Electronic Publishing », (J. André ed.), vol. 5(2), juin 1992, 53–102 ; voir aussi *Bigre*, n° 79, mars 1992.

[11] R. Furuta, V. Quint, J. André, « Interactively Editing Structured Documents », *Electronic Publishing – Origination, Dissemination and Design*, vol. 1, num. 1, 19–44, avril 1988.

[12] R. Furuta, P. David Stotts, « Specifying Structured Document Transformations », *Document Manipulation and Typography*, J. C. van Vliet, ed., 109–120, Cambridge University Press, 1988.

[13] Frans C. Heeman, « Granularity in structured documents », *Electronic Publishing – Origination, Dissemination and Design*, vol. 5(3), 143–155, sept. 1992.

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

[15] I.S.O., *Information technology - Text and office systems - Document Style Semantics and Specification Language (DSSSL), ISO/IEC DIS 10179*, 1991.

[16] V. Quint, M. Nanard, J. André, « Towards Document Engineering », *EP'90*, R. Furuta, ed., 17–29, Cambridge University Press, septembre 1990.

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

[18] V. Quint, I. Vatton, « Modularity in structured documents », *Woodman'89,* J. André & J. Bézivin, ed., 170–177, Bigre n° 63-64, IRISA, Rennes, mai 1989.

[19] V. Quint & I. Vatton, « Combining Hypertext and Structured Documents in Grif », *Proceeding of the ACM Conference on Hypertext*, Milano, 23–32, décembre 1992.

[20] V. Quint, I. Vatton, J. André et H. Richy, « Grif et l'édition de documents structurés : nouveaux développements », *Cahiers GUTenberg*, 9, 49–65, juillet 1991.

[21] H. Richy, « Grif et les index électroniques », *PI Irisa* n° 677 et *Rapport de recherche Inria* n° 1756, 40 pages, septembre 1992,

[22] H. Richy, P. Frison, and E. Picheral, « Multilingual String-to-String Correction in Grif, a Structured Editor », *EP92* (C. Vanoirbeek & G. Coray eds.), Cambridge University Press, 1992, 183–198. Voir aussi « Intégration d'un correcteur typographique dans l'éditeur structuré Grif », *PI Irisa* n° 609, et *Rapport de recherche Inria*, n° 1566, 1991.

[23] N. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schler, H. Schütt, M. Thüring, « SEPIA: A Cooperative Hypermedia Authoring Environment », *Proceedings of the European Conference on Hypertext and Hypermedia ECHT'92*, D. Lucarella, J. Nanard, M. Nanard, P. Paolini, ed., 11–22, ACM Press, novembre 1992.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399