Cahiers GUTenberg

UNE EXPÉRIENCE DE PRODUCTION AUTOMATISÉE DE DOCUMENTS : APPROCHE, PROBLÈMES ET SOLUTIONS

Jean-Jacques GIRARDOT, Bernard AMADE

Cahiers GUTenberg, nº 21 (1995), p. 68-74.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1995___21_68_0>

Une expérience de production automatisée de documents : approche, problèmes et solutions

Jean-Jacques Girardot, Bernard Amade

École des Mines de Saint-Étienne, Diagram *Saint-Étienne*
email : girardot@emse.fr, amade@cambur.emse.fr

Cet article relate deux expériences d'automatisation de production de documents liés à des applications financières.

La principale difficulté a été de faire cohabiter une vision « logique » des documents avec des contraintes liées à la production de cette forme abstraite par les applicatifs existants, à l'addition manuelle d'informations par l'utilisateur, et à la mise en forme finale des documents.

1. Présentation

1.1. Le contexte

La société Diagram fait partie du groupe Ftlis, filiale de France-Télécom. Sa vocation est de fournir des applications financières pour la banque, l'assurance et les grands groupes industriels. Diagram développe et maintient une douzaine de produits.

Ces applications, de tailles conséquentes, sont développées dans un langage propriétaire, sdl, pour lequel existe un atelier de développement spécifique. Les plate-formes logicielles sont assez diverses : DOS, Windows, VMS et diverses moutures d'UNIX.

1.2. Le problème

Diagram s'intéresse actuellement à deux types de problèmes liés à la production de documents électroniques :

- mettre en forme, dans le cadre de la mise en qualité ISO 9000, la documentation des diverses applications existantes et à venir (qui, elles, seront développées en C++).

– fournir à ses clients une chaîne d'impression permettant d'intégrer les informations produites par ses logiciels aux documents créés par ces clients.

DIAGRAM souhaitant mettre en place une stratégie à long terme concernant la production de documents, les préoccupations ont un double aspect théorique et pratique :

– que peut-on attendre des normes et des tendances techniques concernant la structuration abstraite des documents? Un « format pivot » peut-il exister? Est-il pratique et adapté à nos problèmes? Peut-on espérer que l'industrie fournira, à terme, suffisamment d'outils adaptés?

– que peut-on réaliser comme compromis à court terme qui soient utiles, performants et qui ménagent l'avenir?

2. Mise en forme de documents techniques

Le problème est assez complexe, compte tenu des contraintes suivantes :

– Être capable de récupérer les documents préexistants, qui ont pour la plupart été créés sous Word.

– Savoir gérer, à partir d'une même source, aussi bien la documentation interne, utilisée pour la maintenance de l'application, mais non livrée au client, que la documentation utilisateur.

Le problème vient aussi du fait que chaque application est largement adaptée pour les clients : modules et fonctions diffèrent, certaines parties étant même spécifiquement réalisées pour un client particulier. Il faut donc pouvoir sélectionner, en même temps que les programmes, la documentation correspondante, tant interne qu'externe. Il est également envisagé, à plus long terme, de pouvoir générer automatiquement l'aide en ligne.

La documentation se compose de texte devant être formatté, mais aussi de figures (essentiellement de copies d'écran, selon le terme consacré) et de tables (descriptions de paramètres de commandes, de relations dans des bases de données, etc.) Il est prévu des références, permettant de se reporter d'une section à une autre, et susceptibles de devenir ultérieurement des liens dans des hypertextes.

Certains aspects de cette chaîne de production pourraient relever du *litterate programming*, technique consistant à maintenir, pour chaque application ou portion d'application, un fichier unique contenant à la fois le programme source destiné au compilateur et la documentation nécessaire à la maintenance de l'application (cf. par exemple le système CWEB de D.E. Knuth). Cette approche est à l'étude. Elle se heurte à des problèmes d'ergonomie de la saisie, et de cohérence avec le reste de notre approche.

La démarche adoptée consiste, assez classiquement, à se ramener à un format pivot, décrit par un DTD SGML [fS86] propriétaire (cf. sur SGML l'excellente introduction donnée dans [Goo95]). Toutefois, comme l'objectif premier n'était pas de promouvoir des contraintes sévères sur l'architecture de documents, l'approche a reçu le sobriquet de « S.G.M.L. Light ».

2.1. Vers un SGML « allégé »

Les documents existants sont rédigés en Word (pour la plupart), ou en `groff-MM`[1]

La transformation des documents en format pivot est fort différente selon les types de formats initiaux :

- Dans le cas de `groff`, un simple analyseur « ad hoc » devrait permettre de réaliser une traduction d'un format dans un autre (d'une certaine manière le format initial est déjà structuré). Toutefois une telle traduction n'est pas aussi évidente qu'il y parait. En effet rien ne nous assure *a priori* que les éléments d'architecture de documents trouvés dans `groff` sont équivalents à ceux du DTD cible ! On n'est même pas sûr de retrouver toujours des homomorphismes entre les deux structures sous-jacentes ! Nous reviendrons ultérieurement sur cet aspect.

- Dans le cas de Word, la situation est largement plus compliquée car rien ne garantit qu'il y ait, même implicitement, une structuration suffisante du document initial.

Pour traiter ce genre de problèmes de conversion on trouve dans le domaine public (et sur le marché) un certains nombre de produits dont les bases théoriques restent assez naïves : dans la pratique cela ne marche bien que si les textes à analyser ont déjà des propriétés structurelles très propres et si le DTD cible a lui-même une structure proche de celle du document initial.

Les meilleurs convertisseurs sont ceux basés sur un véritable langage de manipulation de structures (ex. Balise de AIS). La variété des situations possibles est telle qu'on ne peut guère créer que des mécanismes, rusés certes, mais « ad hoc ».

Dans la pratique on est donc obligé d'agir en amont : un texte ne sera raisonnablement transformable en format pivot que si l'on demande aux rédacteurs un effort de cohérence dans leur utilisation du traitement de texte.

1. Formateur de type `nroff/troff` développé au début des années 80 à l'EMSE par J.J. Girardot. Bien que non compatible avec `nroff/troff`, il dispose d'un paquetage de macros, `MM` compatible avec celui de `nroff/troff`. Il est doté d'un langage d'extension permettant une programmation assez souple des macros, et génère en sortie un format assez similaire dans l'esprit au format DVI de TEX. Notons que `groff`/EMSE n'a aucune relation avec groff (GNU roff) de la FSF.

Dans le cas de Word, une *feuille de format* est imposée au rédacteur de la documentation. Il reste maître, pour ses besoins propres, de l'apparence typographique du document, mais doit respecter une discipline bien déterminée dans le choix des éléments. Pour permettre de créer le format pivot SGML, il faut utiliser, dans le document original :

- des *noms* de styles pré-définis ("titre1", "titre2", "liste", etc.)
- des *balises* qui encadrent des objets spéciaux comme les tableaux, les graphiques. Ces objets étant à analyser sans ambiguïté, la présence de balises en facilite la reconnaissance. Ces balises peuvent être du texte caché, ou un paragraphe vide d'un style particulier.
- des *règles* typographiques qui interdisent par exemple des changements intempestifs de police au milieu d'un élément autre qu'un paragraphe courant

L'objectif à court terme est donc de disposer d'une procédure qui permette à chaque rédacteur de valider son document. Ce dernier ne peut s'intégrer à la mémoire de l'entreprise tant que le dispositif d'analyse ne sait pas retrouver les règles simples qui sont édictées.

La mise en place de cette politique est délicate : il faut définir des directives, tout en laissant une relative initiative aux rédacteurs. Il est hors de question de prétendre définir des structures de documents convenant *a priori* à tous les besoins de l'entreprise et c'est pour cela que des mécanismes SGML « lourds » visant à obtenir une conformité pointilleuse nous ont semblé irréalistes.

2.2. Quels formats de sortie utiliser ?

Il faut ensuite passer d'un document SGML à un document imprimable. La solution testée actuellement consiste à développer un logiciel spécialisé, basé sur un analyseur public, `sgmls`, associé à un système Scheme ([ASS85], [Han91]) enchassable, InTalk ([Gir93b], [Gir93a], cf. `http://kiwi.emse.fr/INTALK`).

Cette approche a été choisie car InTalk est déjà utilisé dans l'un des produit développé par Diagram (il est donc bien connu des développeurs maison), et qu'il est l'un des outils les plus performants du genre.

Le *parser* crée une structure en langage Scheme, qui reflète, et contient, l'ensemble du document original. Cette structure peut être analysée, transformée, afin de générer le document selon un autre DTD, ou encore selon un format propre à un produit spécifique d'impression. On remarquera que cette démarche est voisine de celle qui est proposée par DSSSL ([ISO94]), nous y reviendrons.

L'impression proprement dite va être réalisée par un logiciel spécialisé. Deux possibilités ont été étudiées :

- produire un format LaTeX,

– piloter un traitement de texte disposant d'une A.P.I (FrameMaker en l'occurence).

LaTeX est semble-t-il peu prisé dans le monde industriel, peut-être parce que trop peu connu. On lui reproche en général le fait de nécessiter presque toujours une correction manuelle des documents (débordements de lignes, mauvais placements des tableaux, espaces inter-paragraphes, etc.), avant d'obtenir une impression correcte. Il a typiquement les défauts de ses qualités.

Le pilotage d'une A.P.I de traitement de texte est plus complexe et représente un investissement plus important. Toutefois nous nous orientons vers cette solution pour des raisons liées à d'autres besoins qui sont exposés dans le chapitre suivant.

3. Mise en forme de données financières

Les outils vendus par Diagram fournissent comme résultats des tableaux. L'impression de ces résultats, jadis obtenue par des gestionnaires d'états, ne convient plus aujourd'hui au client. Il veut pouvoir les intégrer automatiquement à ses propres documents, les exploiter dans des *mailings*, et surtout obtenir des résultats de haute qualité typographique, car ces documents sont destinés à ses propres clients.

Dans certains cas, le processus peut être plus complexe. Ainsi, un gestionnaire d'O.P.C.V.M. (SICAV, etc) utilisant les logiciels Diagram pour gérer les avoirs de ses clients va rédiger des rapports incorporant des tableaux créés automatiquement par ces logiciels, mais il va également vouloir insérer, au sein même de ces tableaux, diverses annotations (aspects légaux, précisions sur les modes de calcul, etc.).

Ces besoins légitimes posent des difficultés techniques qui ne semblent résolues par aucun outil existant. Citons quelques problèmes concrets auxquels nous sommes confrontés :

tableaux : les tableaux à éditer sont de taille variable, parfois très grande. Le client peut désirer qu'un tableau puisse être formatté sur plusieurs pages, avec en bas de chaque page une récapitulation partielle de certaines colonnes, elle même reportée en haut de la page suivante. Ces récapitulations doivent être réalisées par des fonctions financières spécifiques. A moins de disposer d'un véritable langage de programmation, le logiciel d'impression ne peut pas réaliser ces opérations, et pourtant lui seul sait à quel moment le tableau arrive en fin de page.

fusion de documents : lorsque le client veut intégrer des notes dans un état, ou, de manière plus générale, des éléments quelconques (textes, graphiques, images), il ne peut être question de lui demander de manipuler un format « brut » tel que du LaTeX, et encore moins du PostScript. Les logiciels doivent donc générer des sorties dans un format compatible (après conversion éventuelle) avec les logiciels de traitement de texte du client.

Notre expérimentation a abordé les problèmes suivants :

- Définition d'un DTD « tableau financier ». La tâche est rude car le monde de la finance est habitué à des présentations de tableaux à deux dimensions, pour lesquels une représentation arborescente logique est difficile à construire.

- Détermination de la production du format pivot par toutes les applications financières. On imagine sans mal que le "DTD tableau financier" doit ternir compte des conditions de production de ses éléments. Une définition trop abstraite et éloignée des connaissances des logiciels de production conduirait à de grosses difficultés de génération.

  Notons toutefois que la finalité du choix de ce « format pivot » est d'harmoniser la production d'états : actuellement chaque logiciel génère des données spécifiques à un produit particulier de mise en page et la maintenance est lourde du fait de la multiplicité des savoir-faire.

- Coordination entre « producteur » et logiciel d'impression pour la prise en compte des données de report en bas de page. On notera que ce besoin n'est absolument pas pris en compte par la théorie du SGML classique.

  Plusieurs stratégies ont été envisagées :

  - Fabrication, pour toute « ligne » d'un tableau, d'un ensemble de données représentant les reports éventuels. Le logiciel de mise en page reçoit pour ordre d'ignorer ces données sauf, précisément, quand il arrive en bas de page. Cela suppose que le logiciel de composition dispose au minimum d'un langage de macros qui permette de spécifier un tel comportement (c'est le cas de `groff`, *troff* et de LaTeX)..

  - Utilisation d'un logiciel de composition doté d'un langage suffisamment évolué, pour qu'on lui puisse lui transmettre un programme chargé d'effectuer les calculs de reports.

  - Développement d'un outil spécifique de composition qui pilote l'API d'un éditeur de textes interactif. Le format pivot transfère des « annotations » (à l'intérieur du SGML) qui indiquent au pilote de composition les propriétés du tableau. On dispose alors de la description d'une action, sous une forme algorithmique. Le composeur interprète à la demande ces actions, soit dans le cadre d'une composition *batch*, soit à la demande de l'opérateur dans le cadre d'une édition interactive.
    Cette solution est complexe et coûteuse, mais nous a paru la mieux adaptée aux problèmes que l'on a évoqués.

4. Quelques remarques en guise de conclusion

Au cours de notre expérimentation, nous avons vu se dessiner deux visions de l'utilisation de SGML :

- l'une consiste à utiliser SGML pour la définition d'objets « abstraits », « portables », qui trouve son aboutissement dans l'utilisation de DTD normalisés.
- l'autre consiste à voir dans SGML un méta-langage d'une grande puissance expressive, utilisable pour des besoins spécifiques, hors de toute contrainte de normalisation ou de portabilité.

Notre approche des tableaux financiers a favorisé cette seconde vision, mais rien ne nous assure de la pérennité de ce type d'approche.

Par ailleurs, lors de cette étude, DSSSL nous avait paru être un passage obligé : c'est un outil destiné à décrire et à manipuler une représentation structurée de documents, outil qui utilise un formalisme syntaxiquement et sémantiquement très proche de Scheme. Par manque de temps et d'expérience, nous n'avons pas réussi à nous situer dans ce cadre conceptuel. Nous nous posons par exemple la question de savoir si des annotations (elles-mêmes rédigées en Scheme) seraient envisageables pour véhiculer des comportements spécifiques semblables à ceux décrits dans cet article.

Bibliographie

[ASS85] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs.* MIT Press, Cambridge, Mass., 1985.

[fS86] International Organisation for Standardization. *Language normalisé de balises généralisé (SGML) IDO 8879-1996 (F).* Geneva, 1986.

[Gir93a] Jean-Jacques Girardot. GLisp : Manuel de Référence. Technical report, 10 Rue Larionov, Saint-Etienne, Octobre 1993.

[Gir93b] Jean-Jacques Girardot. InTalk : Introduction. Technical report, 10 Rue Larionov, Saint-Etienne, Mai 1993.

[Goo95] Michel Goossens. Introduction pratique à SGML. *Cahiers GUTenberg*, 95(19):27–66, Janvier 1995.

[Han91] Chris Hanson. MIT Scheme User's Manual. Technical report, Massachusetts Institute of Technology, Boston, Mass., 1991.

[ISO94] ISO/IEC. *Document Style Semantics and Specification Language (DSSSL) IDO/IEC DIS 10179.2.* Geneva, 1994.