Chaînes éditoriales numériques : allier efficacité et variabilité grâce à des primitives documentaires

Digital publishing chains: combine efficiency and variability through document primitives

Thibaut ARRIBE (1, 2, 3), Stéphane CROZAT (2), Bruno Bachimont (1), Sylvain SPINELLI (3)

(1) HEUDIASYC UMR CNRS 7253, Université de Technologie de Compiègne
thibaut.arribe@utc.fr
bruno.bachimont@hds.utc.fr

(2) Unité ICS, Université de Technologie de Compiègne
stephane.crozat@utc.fr

(3) Société Kelis
sylvain.spinelli@kelis.fr

Résumé. Cette contribution s'inscrit dans le domaine de l'ingénierie documentaire, et plus particulièrement des chaînes éditoriales numériques, c'est à dire des systèmes assistant la production et la publication de documents structurés.
Après avoir réaffirmé l'enjeu du document structuré et l'objet de l'ingénierie documentaire, nous comparerons les notions de modèle documentaire universel et dédié. Puis nous présenterons le concept de fonction génératrice chez Cassirer et l'associerons au concept de primitive documentaire : un code informatique abstrayant des principes essentiels d'objets documentaires pour permettre la génération de code spécifique instanciant de multiples modèles documentaires. Nous montrerons que l'état de l'art se partage entre d'une part des solutions privilégiant l'efficacité au détriment de la variabilité (possibilité d'adaptation au contexte) par l'utilisation de modèles universels ; et d'autre part des solutions privilégiant la variabilité au détriment de l'efficacité par l'utilisation de modèles dédiés. Nous défendrons alors qu'un niveau d'abstraction fait défaut pour mettre en œuvre de façon réaliste des systèmes alliant efficacité et variabilité.
Notre contribution est une formalisation du système de conception de chaîne éditoriale Scenari, développé en 2004. Issu de travaux visant l'optimisation de la phase de conception de chaînes éditoriales, le système Scenari propose un niveau d'abstraction supplémentaire grâce à des primitives, et permet de concevoir des chaînes éditoriales sur mesure, avec des performances économiques et d'usage inédites.

Mots-clés. document structuré, chaîne éditoriale, abstraction, génération, modèle documentaire.

Abstract. This paper examines digital publishing chains, i.e. systems which assist the production and publication of structured documents, especially their design processes.
After reasserting the issue of structured document and the subject of document engineering, we will compare the notions of universal and dedicated document model. We will then introduce the concept of generating function from Cassirer and combine with the concept of document primitive: a computer code which abstracts the essential principles of document objects to enable the generation of specific code instantiating multiple document models. We will show that the state of the art is divided between solutions favouring efficiency over variability (ability to adapt to the context) by the use of universal document models, and solutions that promote variability at the expense of efficiency through the use of dedicated models. We will defend that a level of abstraction is missing in order to implement systems which combine efficiency and variability.
Our contribution is a formalisation of the Scenari system, a publishing chains design system developed in 2004. Stemming from works to optimize the design stage of publishing chains, the Scenari system offers a level of abstraction through primitives, and can design custom publishing chains with innovative use and economic performance.
Keywords. structured document, publishing chain, abstraction, generation, document model.

1 Introduction

Le document est un objet dont l'usage s'est considérablement démocratisé depuis l'avènement du numérique (Pédauque, 2003). Les contextes donnant lieu à l'écriture d'un document se sont démultipliés, devenant un objet d'étude à part entière. Nous citons pour l'exemple l'étude de Zacklad (Zacklad, 2007) qui répertorie ces contextes en domaines : «le domaine esthétique» (œuvres artistiques) ; «le domaine affectif-fictionnel» (œuvres fictionnelles) ; «le domaine politico-spirituel» (doctrines politiques, livres sacrés) ; «le domaine moral-idéologique» (documents militants, pratiques liturgiques) ; «le domaine scientifique» (résultats et vulgarisations scientifiques, documents pédagogiques) et «le domaine pratique-efficace» (documentation technique, juridique, administrative).

Dans cette contribution, nous nous intéressons à des contextes de forte production de documents relativement homogènes (en reprenant les domaines de Zacklad, nous nous situons principalement dans les domaines «pratique-efficace» et «scientifique»). L'ingénierie documentaire a répondu à la problématique de production de masse en faisant émerger la notion de document structuré (André *et al.*, 1988). Son enjeu est de contrôler l'homogénéité des documents par des structures qui s'articulent intimement avec les logiques applicatives d'édition, de manipulation et de publication. Cette mise en évidence de la structure permet d'instrumenter la séparation entre le fond et la forme - ou entre le fonds documentaire et ses formes (Bachimont & Crozat, 2004) - permettant ainsi une automatisation de la manipulation documentaire. Les logiciels instrumentant cette situation d'écriture sont appelés

des chaînes éditoriales XML (Crozat, 2007). Ils permettent l'écriture d'un contenu en se conformant à un modèle préalablement défini. La publication s'opère par des transformations automatiques vers des standards tels que PDF ou HTML.

Nous qualifierons les chaînes éditoriales en fonction de deux critères : leur faculté à s'adapter à un nouveau contexte, la *variabilité* ; leur apport pour la production et la maintenance, l'*efficacité*.

Un des objectifs majeurs de l'ingénierie documentaire est de maintenir la variabilité des contenus - pour respecter la spécificité de chaque contexte d'usage - tout en améliorant l'efficacité de leur gestion - pour gérer la massification.

2 Modèle universel versus modèle dédié

La notion de document structuré suppose de formaliser un modèle de représentation du document permettant d'en contrôler les opérations (Barron, 1989) (Piwowarski *et al.*, 2002).

2.1 Modèle documentaire dédié

Un modèle dédié est un modèle documentaire spécifique à un contexte d'usage métier en particulier. Le besoin documentaire est analysé puis formalisé dans un modèle, comprenant des schémas structurels, des interfaces d'éditions, des programmes de validation, de transformation... Historiquement portées par SGML ces approches sont aujourd'hui ancrées dans les technologies XML : Schema, XSLT, DOM...

L'intérêt du modèle dédié est par construction son adéquation au contexte adressé. C'est la solution juste nécessaire au problème, permettant de traiter des structures documentaires métiers (tableaux comptables, scénarios pédagogiques, plans numériques, formats dédiés...) sans scories héritées de fonctions liées à d'autres contextes d'usage.

L'utilisation d'un modèle dédié impose une forte spécificité de la chaîne éditoriale. Nous parlerons d'une approche par *création* car la chaîne éditoriale doit être développée *ex nihilo*, permettant ainsi de répondre finement à la problématique de la variabilité. Ce gain se paie sur l'efficacité du processus, notamment en raison des coûts de mise œuvre à l'initialisation, puis en maintenance. La chaîne étant fortement adhérente au contexte par construction, elle devient obsolète dès l'évolution de ce contexte et requiert par conséquent des moyens de maintenance importants. Cette barrière rend cette approche adaptée uniquement à des usages de niche et aux contextes relativement stables du point de vue des formats documentaires (presse, documentation technique des industries sensibles...).

2.2 Modèle documentaire universel

Un modèle universel est au contraire un modèle à forte valeur de généralité visant à circonscrire l'ensemble des usages pour une famille de contextes. Généralement portés par un standard (W3C, OASIS...), les modèles universels visent l'intégration d'un très large ensemble de besoins, et misent sur la mutualisation des développements autour du standard. On citera par exemple DITA, DocBook, ou la partie sémantique de HTML[1].

[1] http://dev.w3.org/html5/html-author/#understanding-semantics

La raison d'être du modèle global, une fois celui-ci standardisé et les développements associés mûris, est la possibilité de disposer de chaînes éditoriales prêtes à l'emploi. Ces chaînes prêtes à l'emploi se composent d'un code générique fourni par l'éditeur de la chaîne et d'un code spécifique permettant une relative adaptation du contexte. Nous parlerons d'approche par *déclinaison* car une chaîne éditoriale type pourra se décliner par un simple ajout de code spécifique. Cette approche est incontestablement dominante dans les faits aujourd'hui, en particulier du fait de sa forte *efficacité* lorsque l'usage reste très proche du standard. Dès qu'elle s'éloigne du standard, en revanche, elle gère mal la variabilité, il faut alors utiliser plus de code spécifique, faisant chuter fortement l'efficacité.

L'enjeu de cette contribution est de répondre aux cas nécessitant une adaptation trop spécifique pour être raisonnablement déclinée depuis un modèle universel, mais ne pouvant pas se permettre une approche dédiée pour des raisons économiques.

3 Vers un niveau d'abstraction supplémentaire

3.1 Notions de déclinaison et de génération chez Cassirer

Dans son ouvrage, Cassirer (1910) s'intéresse aux différentes théories du concept pour mettre en relief les notions de déclinaison et génération. Il y distingue deux approches : d'un coté la logique formelle forgée par Aristote, de l'autre celle des sciences modernes et contemporaines. L'objet de la logique formelle est l'étude de la métaphysique : «l'essence et l'articulation de l'être». Dit autrement, ce qui est. Du coté des sciences modernes, la notion de concept s'appuie non plus uniquement sur l'existence mais également sur la preuve, ce qui est vérifiable.

Le concept vu par la logique est « un rassemblement par similitude d'essence » c'est à dire un rassemblement d'individus par ressemblance. Par exemple, l'hirondelle, le moineau et l'aigle ont tous des plumes, des ailes, un bec, etc. Ces caractéristiques constituent l'essence du concept d'oiseau. La généralisation d'un concept vers un concept de niveau supérieur se fait en procédant au rassemblement des concepts de niveau inférieur. Un animal serait un mammifère, un oiseau, un amphibien, un poisson ou un reptile. Le concept universel serait alors une liste de toutes les essences possibles de ce qui est. L'universalité sera ici appelée abstraite car il n'existe pas de relation entre un concept et un sous-concept. Le passage de concept au sous-concept se fera alors par une déclinaison de l'ensemble des propriétés du concept.

Pour Cassirer, le concept scientifique n'est plus un rassemblement mais une *abstraction* de la liste des propriétés, permettant ainsi la réunion, dans un même concept, de sous-concepts qui ne se ressemblent pas. Une fonction génératrice attachée au concept permet de *générer* l'ensemble des sous-concepts. En partant du nombre 0 et avec la loi successeur, il est possible de générer l'ensemble des entiers naturels. La généralisation de plusieurs concepts scientifiques se fera en changeant les fonctions génératrices. L'universalité sera ici appelée concrète car les fonctions du concept universel permettent la génération de l'ensemble des individus qui le composent.

3.2 Approche par déclinaison en ingénierie documentaire

Cassirer conçoit la déclinaison en rassemblant des individus semblables. Rapportée à l'ingénierie documentaire, une approche par déclinaison consiste à construire un modèle de document générique et un système générique qui le manipule. Les éléments additionnels du modèle ou de l'application sont à définir dans un code spécifique qui, associé au code générique, construit une déclinaison. Cette approche est conforme à l'utilisation d'un modèle universel.

Conserver cette approche avec un modèle dédié revient à écrire une quantité importante de code spécifique pour adapter les composants de l'application aux spécificités du modèle. Le code spécifique de chaque composant sera dépendant du modèle. Il y aura donc une forte dépendance de chaque morceau de code spécifique. La viabilité de cette approche touchera ses limites devant la complexité du code et du contrôle des dépendances, plus la variabilité est prise en compte, plus l'efficacité chute. Cette approche n'aura plus d'intérêt quand son efficacité sera moins importante que celle de l'approche par création.

3.3 Approche par génération en ingénierie documentaire

Pour maintenir un niveau élevé de variabilité et d'efficacité, il est nécessaire de réduire les dépendances du code spécifique. Nous proposons d'isoler ces dépendances et de les gérer par des primitives couplées à une fonction de génération. Le code spécifique dépendant est alors créé automatiquement par la fonction de génération, suivant le principe de Cassirer.

Nous retrouvons des approches de ce type dans le domaine de la spécialisation logicielle (Stig Nordheim, 2004). Par exemple, Recker (Recker *et al.*, 2006) part d'un « modèle de référence », le spécialise pour un cas d'usage et génère le modèle d'entreprise qui sera utilisé par l'application. Le déploiement final est automatiquement contrôlé pour en vérifier la cohérence. Sur une même approche, Zina (Zina *et al.*, 2006) propose de construire, à partir de modèles existants, un « méta-modèle » d'application de Gestion de Cycle de vie d'un Produit (Product Lifecycle Management - PLM) permettant d'instancier de nouveaux modèles d'application.

D'un point de vue plus général, cette approche rejoint les travaux du Object Management Group's (OMG) Model Driven Architecture[2] consistant à générer tout type d'architecture logicielle à partir d'un modèle, approche généralisée à l'ensemble de l'ingénierie logicielle par Kent (Kent, 2002).

4 L'exemple de Scenari

Nous illustrons à présent le principe de génération porté par des primitives à travers le système de conception de chaînes éditoriales Scenari[3], inventé à l'UTC et édité par la société Kelis. Nous mobiliserons un exemple d'application en production depuis plusieurs années pour la gestion de la documentation métier de la société Quick.

[2] http://www.omg.org/mda

[3] http://scenari-platform.org

4.1 Le contexte Quick

La chaîne de restaurants Quick manipule trois documentations distinctes : la documentation de référence, la documentation de formation et les dossiers d'homologation.

La documentation de référence, appelée en interne, la *bible,* contient toutes les procédures nécessaires à l'exploitation des restaurants (de la réalisation des produits à l'utilisation et l'entretien des équipements). Cette documentation doit être continuellement disponible dans l'ensemble des restaurants. Elle peut être imprimée en version papier sous forme de fiches conservées dans un classeur ou consultée sur l'intranet de la société.

La documentation de formation permet à chaque restaurant de former ses nouveaux équipiers. Elle permet aux nouveaux collaborateurs d'étudier les différentes procédures à l'aide de parcours de formation appropriés. Elle permet également de sanctionner l'apprentissage par un système d'évaluation composé de séries de questions à choix multiples (QCM). La documentation de formation fait l'objet de plusieurs supports dédiés : outre les formats PDF ou HTML standard, on relève par exemple une version multimédia orientée mobiles (tablette, *smartphone)* pour un usage en situation sur le poste de travail de l'équipier.

Enfin le département innovation est en charge de l'élaboration de nouveaux produits ou de nouveaux équipements. Toute évolution dans les procédés de fabrication fait l'objet d'un document de type dossier d'homologation (DH) qui a des répercutions sur les documents de référence et pédagogiques.

L'ensemble de la documentation est par ailleurs diffusée au niveau international, nécessitant des adaptations liées aux fonctionnements locaux des restaurants (législation, adaptations culturelles...). On appelle dérivation l'adaptation d'un document à un contexte international.

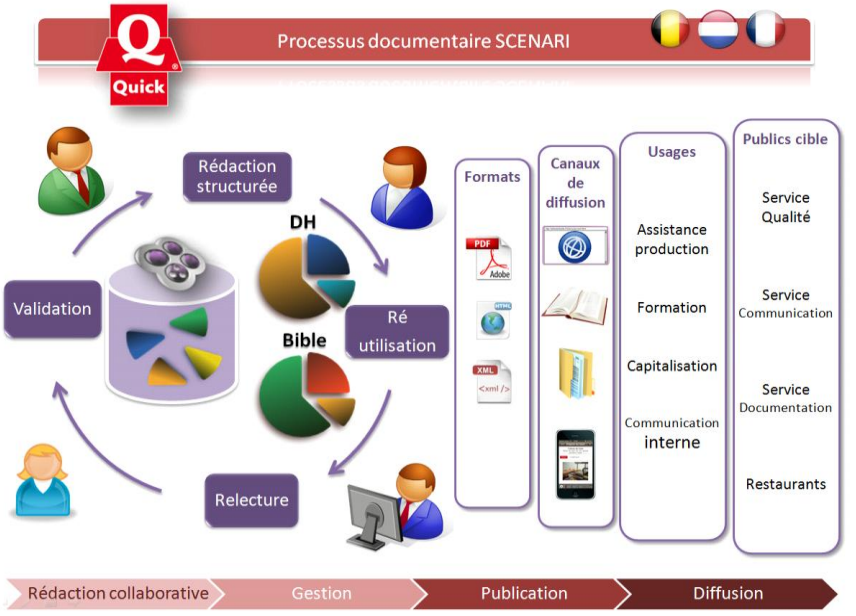

Figure 1. *Processus documentaires instrumentés par la chaîne éditoriale Quick (http://scenari.utc.fr/c2m/DOCS/L4d/html/co/quick4.html)*

Pour une société comme Quick, la documentation est un enjeu important sans toutefois être son cœur de métier et justifier un investissement trop important. Dans cette situation multi-contextes, accompagnée d'un besoin pluri-média, les approches classiques de conception de chaîne éditoriale sont mal adaptées. Le développement d'une chaîne éditoriale ex nihilo couvrant l'ensemble des contextes d'usage nécessite un investissement initial trop important. La complexité du contexte n'est pas directement adressable par un modèle universel, l'effort de déclinaison serait trop important quel que soit le standard (réutilisation inter-documents DH, bible, supports de formation ; gestion de QCM ; publications pour mobiles ; dérivations internationales...)

4.2 Instrumentation Scenari

Le système Scenari propose un principe de primitives documentaires permettant de modéliser les documents à manipuler et un système de primitives de transformation dédié à la définition des publications associées. Il existe plusieurs types de primitives (composition de primitives, méta-données associées, structuration de texte, inclusion de ressources binaires, etc.), qui une fois agencées, permettent de définir de nombreux modèles.

Définition du modèle

Les primitives documentaires et les primitives de transformation utilisées par le système Scenari s'expriment dans un formalisme XML. Les encadrés 1 et 2 donnent des exemples simplifiés de ces primitives. L'encadré 1 définit une « Fiche savoir-faire » comme la composition d'autres primitives : des métadonnées (procM.model), une première partie « Contexte » (co.model), suivie d'une « Procédure » (stepList.model). L'encadré 2 définit une publication de ce type de fiche pour XHTML, en associant les parties à des blocs titrés (WHeadingBlock) et des classes qui seront stylées en CSS.

```
<compositionPrim name="Fiche savoir-faire">
    <identification code="proc"/>
    <structure>
    <meta refUri="/qkDoss/model/content/proc/procM.model" usage="required"/>
        <part code= "context" name="Contexte" family="sub-level" usage="optional">
            <allowedModel refUri="/qkDoss/model/base/co.model"/>
        </part>
        <part code="stepList" name="Procédure : liste d'étapes" family="sub-level"
usage="required">

     <allowedModel refUri="/qkDoss/model/content/proc/stepList.model"/>
        </part>
    </structure>
</compositionPrim>
```

Encadré 1. *exemple simplifié de primitive documentaire*

```
<compositionXhtmlTransf>      <model refUri="/qkDoss/model/content/proc/proc.model"/>
    <content format="xhtml">
        <inDataOrder>
            <for codes="context">
                <WHeadingBlock widgetClass="bk_context">
                    <title>
                        <subModelTitle/>
                        <fixedTitle value="Contexte"/>
                    </title>
                    <callSubModel/>
                </WHeadingBlock>
            </for>
            <for codes="stepList">
                <WHeadingBlock widgetClass="bk_stepList">
                    <title>
                        <subModelTitle/>
                        <fixedTitle value="Procédure"/>
                    </title>
                    <callSubModel/>
                </WHeadingBlock>
            </for>
        </inDataOrder>
    </content>
</compositionXhtmlTransf>
```

Encadré 2. *exemple simplifié de primitive de transformation*

Pour simplifier l' écriture, la gestion et la maintenance des primitives, le système Scenari propose un éditeur XML dédié à travers son outil de modélisation SCENARIbuilder (voir figure 2).

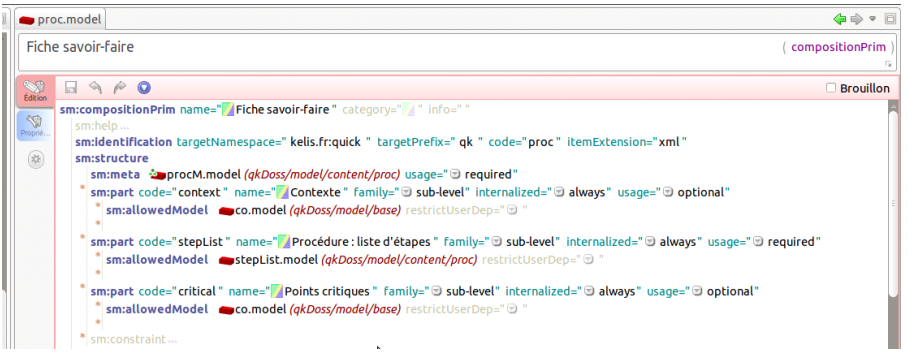

Figure 2. *Éditeur XML de primitives dans SCENARIbuilder*

SCENARIbuilder permet ensuite la compilation des primitives documentaires déclarées pour générer un code source spécifique à Quick, qui sera interprété par le code générique de Scenari à travers l'outil SCENARIchain. Le résultat de la compilation est compressé dans une archive dédiée (*wsppack*), une fois chargée dans SCENARIchain, la chaîne éditoriale est prête à l'emploi (figure 3). Elle propose alors un éditeur XML dédié au modèle (figure 4), des

logiques applicatives de gestion posées par les primitives documentaires (par exemple l'adaptation au contexte international, visible via les drapeaux dans l'éditeur) et des publications posées par les primitives de transformation (par exemple la publication XHTML, figure 5).

Figure 3. *Architecture Scenari de génération et exécution de code spécifique*

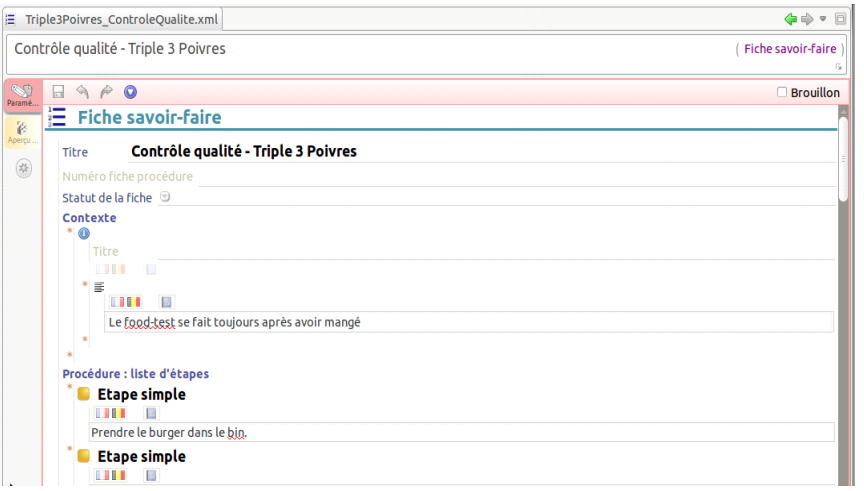

Figure 4. *Éditeur XML de fiche savoir-faire Quick*

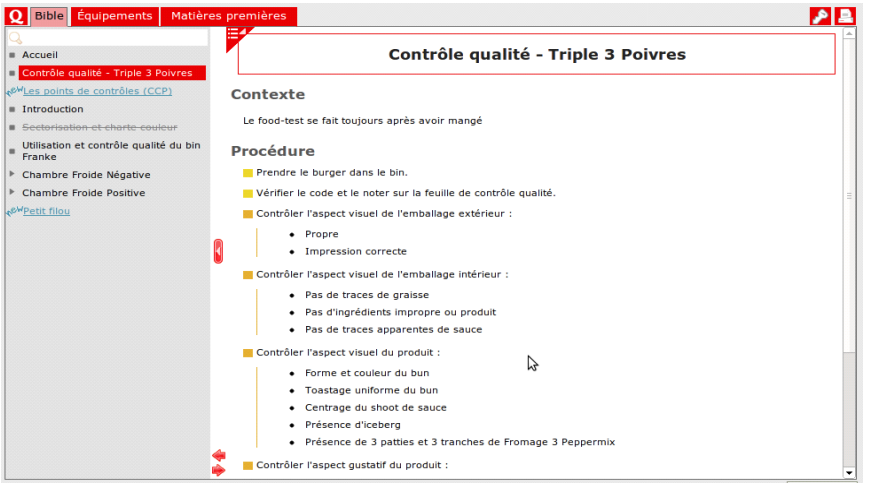

Figure 5. *Publication HTML d'une fiche savoir-faire Quick*

5 Conclusion

À travers cette contribution, nous avons souhaité montrer les limites de l'ingénierie documentaire traditionnelle qui privilégie la variabilité ou l'efficacité, mais peine à concilier les deux. L'abstraction que constitue le modèle d'un document structuré est le premier niveau traditionnellement mobilisé pour monter en efficacité tout en gérant la variabilité documentaire au sein d'une même chaîne éditoriale : le modèle permet de gérer la variabilité de documents qui se ressemblent (ils respectent un même schéma, mobilisent les mêmes transformations...).

En revanche cette solution ne permet pas de gérer efficacement la variabilité de documents qui ne ressemblent pas : à chaque nouveau modèle de document, il faut soit décliner une chaîne existante si le modèle est proche d'un cas maîtrisé, soit réinventer la chaîne *ex nihilo* lorsque la variation est trop forte. Or ces documents qui ne se ressemblent pas, présentent néanmoins des propriétés intrinsèques que l'on retrouve d'un modèle à l'autre, et qu'il est possible d'exprimer sous la forme de fonctions génératrices capables d'engendrer le code spécifique d'un modèle particulier.

C'est ce second niveau d'abstraction, celui des primitives documentaires, qui permet de gérer la variabilité des modèles documentaires (au delà de la variabilité des instances gérée par les modèles) tout en conservant un niveau d'efficacité compatible avec la plupart des contextes professionnels.

Dans le cas des restaurants Quick, le modèle est complexe mais la solution conçue s'adapte néanmoins aux contextes d'usage dans toutes leurs spécificités. L'utilisation du système Scenari a permis d'adresser la variabilité des contextes d'usage et le principe de modélisation par primitive a été mis à profit pour mutualiser de nombreux aspects du modèle avec des contextes standards, et ainsi maintenir le projet dans une économie acceptable. Le principe de conception utilisé dans Scenari permet ainsi des performances de conception et

de maintenance inédites. Depuis son développement, l'outil SCENARIbuilder dédié à l'écriture et à la génération des primitives a permis la diminution du temps nécessaire à la production du code source spécifique d'un facteur de un à dix au minimum (observations empiriques réalisées sur les projets menés par la société Kelis). Les compétences nécessaires à la conception d'une chaîne éditoriale se sont par ailleurs déplacées d'un niveau technique de type développement informatique à un niveau plus fonctionnel de type modélisation documentaire. Ce glissement renforce l'expertise documentaire des concepteurs et permet d'améliorer l'efficacité de la conception et la qualité des chaînes produites.

Nos prochains travaux seront consacrés à l'étude d'un nouveau niveau d'abstraction, complémentaire des primitives documentaires, permettant la génération de logiques applicatives d'*écriture collaborative*. Dans le cadre du projet ANR C2M[4], le concept de chaîne éditoriale collaborative a été étudié et instancié dans le logiciel Scenari4. L'enjeu est à présent de concevoir un niveau d'abstraction pour cette dimension collaborative qui soit cohérent avec celui défini pour la dimension documentaire et permette le même gain autour des enjeux de variabilité et d'efficacité.

Références

ANDRE, J., FURUTA, R., QUINT, V. (1988). *Structured Documents*. Cambridge University Press, the cambridge series on electronic publishing edition.

BACHIMONT, B., CROZAT, S. (2004). Instrumentation numérique des documents : pour une séparation fonds/forme. *Revue I3*, vol. 4, 95–104.

BARRON, D. (1989). Why use sgml? *Electronic publishing*, vol. 2, 3–24.

CASSIRER, E. (1910). *Substance et Fonction*. Berlin.

CROZAT, S. (2007). *Scenari : la chaîne éditoriale libre : Structurer et publier textes, images et son*. Eyrolles, accès libre edition.

KENT, S. (2002). Model driven engineering. *Integrated Formal Methods - Lecture Notes in Computer Science*, vol. 2335, 286–298.

PIWOWARSKI, B., DENOYER, L., GALLINARI, P., (2002). Un modèle pour la recherche d'information sur des documents structurés. In *JADT : 6es Journées internationales d'Analyse statistique des Données Textuelles*.

PEDAUQUE, R. T. (2003). Document : forme, signe et médium, les reformulations du numérique.

RECKER, J., MENDLING, J., VAN DER AALST, W., ROSEMANN, M. (2006). Model-driven enterprise systems configuration. *Advanced Information Systems Engineering - Lecture Notes in Computer Science*, vol. 4001, 369–383.

STIG NORDHEIM, T. P. (2004). Customization of enterprise content management systems : An exploratory case study. In *Proceedings of the 37th Hawaii International Conference on System Sciences*.

[4] www.utc.fr/ics/c2m

ZACKLAD, M. (2007). Réseaux et communautés d'imaginaire documédiatisées. In *A Document (Re)turn*. AM MAIN, F. (Ed.), Roswitha Skare and Andreas Varheim and Niels Windfeld Lund.

ZINA, S., LOMBARD, M., LOSSENT, L., HENRIOT, C. (2006). Generic modeling and configuration management in product lifecycle management. *International Journal of Computers, Communications & Control*, 126–138.