



Ingénierie documentaire et management des contenus



Version 10.03

Notes de cours

Publié le 9 décembre 2010

Stéphane Crozat et Bruno Bachimont
(Contributions de Erik Gebers)

Paternité - Pas d'Utilisation Commerciale - Partage des Conditions Initiales à l'Identique :
<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/>



Table des matières

I - Cours	5
A.Théorie de l'ingénierie des documents numériques.....	5
1. <i>Introduction à l'ingénierie documentaire</i>	5
2. <i>L'essence du numérique</i>	7
3. <i>Le document numérique entre signe et calcul</i>	11
4. <i>Le principe du balisage documentaire</i>	15
B.Chaînes éditoriales XML.....	17
1. <i>Introduction aux chaînes éditoriales</i>	17
2. <i>L'environnement de conception de chaînes éditoriales Scenari</i>	25
C.eXtensible Markup Language : Syntaxe et schémas.....	28
1. <i>Définition du XML par le W3C</i>	28
2. <i>Introduction à XML</i>	29
3. <i>Syntaxe XML</i>	33
4. <i>Introduction aux schémas XML</i>	39
5. <i>Définition de type de document</i>	43
6. <i>RelaxNG</i>	48
7. <i>Exemples</i>	54
8. <i>Quelques bonnes questions sur XML</i>	56
D.Transformations XML : XPath et XSL-XSLT.....	58
1. <i>Introduction à XSL-XSLT</i>	58
2. <i>Programmation XSL-XSLT</i>	60
3. <i>Exemple : Un programme XSLT pour générer du HTML</i>	65
4. <i>Approfondissement</i>	67
E.Modélisation conceptuelle de documents.....	67
1. <i>Modélisation UML</i>	67
2. <i>Méta-modélisation : séparation scenario/contenu</i>	71
3. <i>Méta-modélisation : Motifs de conception (Design Pattern) et stéréotypes</i>	76
4. <i>Passage UML - Schéma</i>	80
5. <i>Bonnes pratiques</i>	86
F.Repères méthodologiques pour un projet documentaire.....	88
1. <i>Démarche d'AMO</i>	88
2. <i>Critique des méthodes "traditionnelles"</i>	93
3. <i>Introduction aux méthodes agiles</i>	95
4. <i>Discussion : La vie en rose ?</i>	99
G.Panorama des outils documentaires.....	101
1. <i>Outils orientés production</i>	101
2. <i>Outils orientés gestion</i>	104
3. <i>Outils spécifiques Web</i>	105
4. <i>Outils périphériques</i>	113
H.Indexation et recherche.....	114
1. <i>Introduction à l'indexation</i>	114
2. <i>Introduction aux techniques d'indexation et de recherche</i>	117
I.Enterprise Content Management.....	122
1. <i>La gestion de contenu</i>	122
2. <i>Les grandes fonctions des ECM</i>	124
3. <i>Complément : Éléments commerciaux et industriels</i>	146
J.Langages de publication : SMIL et FO.....	148
1. <i>Introduction à SMIL</i>	148
2. <i>Introduction à XSL-FO</i>	152

II - Exercices	155
A.Questions théoriques ingénierie documentaire.....	155
1.Exercice rédactionnel.....	155
2.Exercice rédactionnel.....	156
B.Travaux pratiques chaînes éditoriales.....	156
1.Découvrir des chaînes éditoriales.....	156
2.Apprendre à utiliser une chaîne éditoriale.....	157
C.Exercices XML.....	158
1.Exercice.....	158
2.Exercice.....	159
3.Exercice.....	160
D.Exercices RelaxNG.....	160
1.Exercice rédactionnel.....	160
2.Exercice.....	162
E.Exercices DTD.....	164
1.Exercice rédactionnel.....	164
2.Exercice.....	166
F.Quiz DTD.....	167
1.Préambule.....	167
2.Exercice : Fichiers bien formés.....	168
3.Exercice : Fichiers valides.....	168
4.Exercice : Schéma intégrateur.....	168
G.Exercices XSLT et XPath.....	169
1.CV.....	169
2.Poème.....	170
3.Glossaire.....	172
4.Tester des expressions XPath.....	172
5.Namespaces et FO.....	174
6.Bulletins météo.....	175
H.Quiz XSLT et XPath.....	177
1.Exercice.....	177
2.Exercice.....	178
3.Exercice.....	179
I.Exercices UML, passage RelaxNG, XSLT.....	180
1.Standard W3C.....	180
2.Contraintes d'ordre.....	182
3.Cours en ligne.....	183
4.Diaporama.....	183
J.Travaux pratiques Wiki.....	186
1.Expérimenter l'usage d'un Wiki.....	186
K.Question théorique indexation.....	187
1.Exercice rédactionnel.....	187
L.Travaux pratiques ECM.....	187
1.Utiliser un ECM : Nuxéo DM.....	187
2.Documenter un ECM.....	188
M.Exercices SMIL et FO.....	190
1.Photos de vacances commentées.....	190
Glossaire	191
Signification des abréviations	193
Bibliographie	195



Cours

I

Théorie de l'ingénierie des documents numériques	7
Chaînes éditoriales XML	21
eXtensible Markup Language : Syntaxe et schémas	38
Transformations XML : XPath et XSL-XSLT	75
Modélisation conceptuelle de documents	93
Repères méthodologiques pour un projet documentaire	129
Panorama des outils documentaires	156
Indexation et recherche	180
Enterprise Content Management	191
Langages de publication : SMIL et FO	224

A.Théorie de l'ingénierie des documents numériques

1.Introduction à l'ingénierie documentaire

a)Concept de document



Fondamental

Un document est une **inscription** de **contenus** sur un **support** pérenne, établie dans un **contexte de production** et pour un **contexte de réception**.



Définition : Contenu

Un contenu est une forme d'expression pourvue d'une valeur culturelle associée à un **véhicule matériel**, il exprime une signification et suscite une réception et une interprétation.



Définition : Inscription

Une inscription est un contenu fixé sur un **support matériel**, tel qui lui apporte une permanence dans le temps.



Exemple : Exemple de contenu

Une définition donnée oralement est un contenu.

Une transmission hertzienne d'un flux audiovisuel est un contenu.



Exemple : Exemple d'inscription

Une définition écrite est une inscription.

Un enregistrement magnétique d'une transmission hertzienne d'un flux audiovisuel est une inscription.



Exemple : Exemple de document

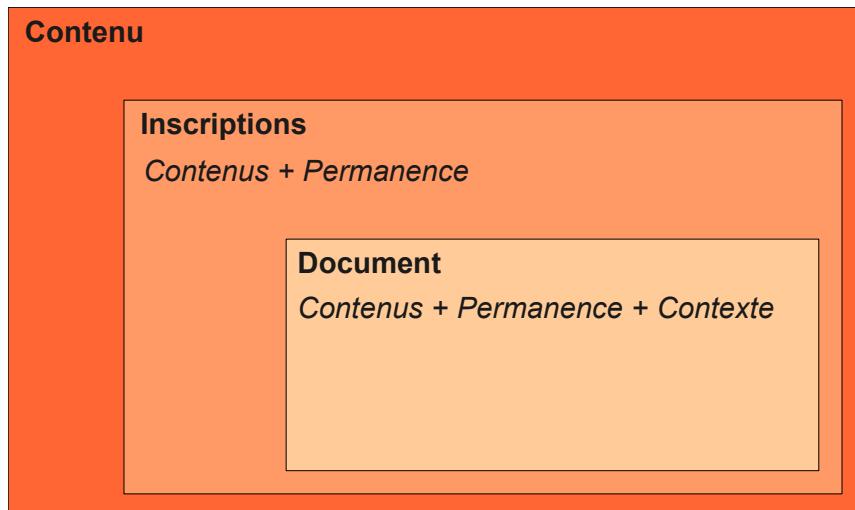
Un glossaire de définitions publié sur un livret, par un chercheur pour sa communauté est un document.

Une émission de télévision, produite par une chaîne pour ses téléspectateurs, enregistrée sur un support magnétique est un document.

Reformulations

Contenu	Forme d'expression
Inscription	Contenu + Permanence
Document	Inscription + Contexte

Tableau 1 *Document = Contenu + Permanence + Inscription*



Graphique 1 *Notion de document*



Fondamental : Caractéristiques du document

Un document est pourvu de trois propriétés fondamentales :

- **La fermeture** : Un document est délimité dans le temps et dans l'espace, il a un début et une fin, l'on sait de quoi il est constitué (parties), il est scénarisé (sa forme prescrit une ou plusieurs linéarités explicites, qui se s'imposent ou se proposent au lecteur).
- **L'intentionnalité** : Un document est intentionnel, il est destiné à un usage, il naît d'une intention auctoriale et éditoriale, il est produit dans un contexte d'écriture et publié pour un contexte de lecture.
- **La publication** : Un document est publié, il est rendu public sous une forme matérielle donnée et figée, son contexte de production et de diffusion est identifiable et objectivable.

Le document est pourvu de deux facettes :

- C'est un **objet technique** considéré pour ses propriétés physique qui relève d'une manipulation technique
- C'est un **objet culturel** considéré pour sa signification qui relève d'une interprétation



Remarque : Document "au sens large"

Notons que les définitions précédentes laissent de côté la notion de document comme objet dont l'intentionnalité est reconstruite *a posteriori*, au sens où un objet archéologique est un document pour l'archéologue par exemple.

b) Concept d'ingénierie documentaire



Fondamental : Ingénierie documentaire

L'objet de l'ingénierie documentaire est la conception de systèmes techniques permettant et optimisant l'articulation de la manipulation technique et de l'interprétation culturelle des documents.



Définition : Ingénierie des documents numériques

L'ingénierie des documents numériques est le sous-ensemble de l'ingénierie documentaire, dont l'objet est la construction de systèmes informatiques et qui ne considère que les documents dont le support est numérique.



Fondamental : Problématique de l'ingénierie des documents numériques

L'essence computationnelle du numérique fait que les documents qu'il supporte « **ont été manipulés** », ce qui interroge théoriquement les propriétés fondamentales du document (fermeture, intentionnalité, publication).

La question de la manipulation informatique est donc au cœur de l'ingénierie des documents numériques, qui s'attache d'une part à optimiser ces potentialités manipulatoires et d'autre part à protéger en pratique les propriétés fondamentales du document mises à mal par le numérique.

2.L'essence du numérique

a) Théorie du support et de la connaissance inscrite

La théorie du support, est une théorie philosophique de la connaissance, issue notamment de la phénoménologie de Husserl [husserl50], développée notamment à l'UTC à travers des philosophes comme Bernard Stiegler (1994a [Stiegler94a], 1994b [Stiegler94b]) et Bruno Bachimont (1996 [Bachimont96], 2004 [Bachimont04]).

Le concept général de la théorie du support est que toute connaissance ne peut procéder que d'une inscription sur un support matériel.

Cette idée est développée en particulier autour du cas du support numérique.



Définition : Connaissance

Une connaissance est la capacité d'exercer une action pour atteindre un but.

- Une connaissance pratique renvoie à une activité dans le monde matériel (notion savoir-faire), elle est corrélée un objet pratique (ce sur quoi porte

l'action pratique).

- Une connaissance théorique renvoie à la possibilité de produire ou reformuler des énoncés dans un code de communication (notion de savoir), elle est corrélée à un objet matériel, mais pour ce qu'il représente et non pour ce qu'il est (par exemple de l'encre sur du papier).

La connaissance pratique porte sur une modification physique du monde, la connaissance théorique porte sur une modification de notre représentation du monde.

(Bachimont, 2004 [Bachimont04], pp65,67)



Définition : La connaissance inscrite

Toute connaissance repose sur une **inscription**, dont elle est l'interprétation :

- la connaissance est l'interprétation de l'inscription
- l'inscription est la matérialisation de la connaissance

Ainsi un livre, une cassette vidéo ou un logiciel véhiculent des connaissances, sans les modéliser au sens de la logique formelle ou de l'IA★, mais en leur offrant un support de **mémorisation** et de **manifestation**.

La transmission de connaissance résulte de son inscription intentionnelle sur un support par un auteur pour un lecteur.



Complément : Le support comme prothèse

Le support technique devient alors une **prothèse** cognitive, un processus d'extériorisation, qui permet l'anticipation et la constitution de modes nouveaux de représentation : « *par pro-thèse nous entendrons toujours à la fois : posé devant, ou spatialisation (é-loignement); et posé d'avance, déjà là (passé) et anticipation (prévision), c'est à dire temporalisation.* (Stiegler, 1994 [Stiegler94a]) ».

Le support est donc un moyen de spatialiser l'information, pour la rejouer dans le temps.



Remarque

Le support est à prendre au sens le plus large, pour soutenir une telle thèse, il faut en effet élargir la notion d'inscription à tout ancrage matériel de la connaissance, et en particulier au corps :

- Une inscription est corporelle quand le support technique est le corps
- Une inscription est externe quand le support est externe au corps

(Bachimont, 2007 [Bachimont07], p255)



Fondamental : Le supplément

La signification de l'inscription est conditionnée par les **propriétés matérielles** du support :

- Le support impose un **supplément** à l'inscription car il ajoute de l'intelligibilité.
- Selon le type de support, le supplément d'intelligibilité sera différent.
- On aura donc une **rationalité spécifique par type de support**.



Complément : Les thèses de la théorie du support



La théorie du support s'articule autour de la thèse centrale suivante :

Les propriétés du substrat physique d'inscription, et du format physique de l'inscription, conditionnent l'intelligibilité de l'inscription.

Elle comprend en outre les thèses suivantes :

1. une connaissance est la capacité d'effectuer une action dans un but donné.
2. un objet technique prescrit par sa structure matérielle des actions. L'objet technique est l'inscription matérielle d'une connaissance.
3. toute connaissance procède d'une genèse technique. Seule la répétition, prescrite par les objets techniques, de l'action permet d'engendrer la connaissance comme capacité à exercer une action possible.
4. la connaissance, engendrée par la technique, prescrit une transformation dans le monde des choses (l'objet technique est alors un instrument) ou une explicitation dans le monde des représentations (l'objet technique est alors une inscription sémiotique).
5. une pensée est une reformulation effectuée par la conscience sur le support corporel qu'est le corps propre. Penser, c'est s'écrire. Toute pensée, comprise comme reformulation a pour cible de réécriture le corps propre, et comme origine, le corps propre ou une inscription externe quelconque.
6. la conscience est un pur dynamisme intentionnel, source des ré-écritures considérées comme des interprétations et non comme un mécanisme.

(Bachimont, 2004 ↗ [Bachimont04], p77)



b) La raison computationnelle



Définition : La raison graphique

Goody nous montre que l'invention de l'écriture a modifié les schèmes de représentation de la connaissance, jusque là orale, donnant naissance à une « *raison graphique* » (Goody, 79) ↗ [Goody79]. Les documents papier ont permis la représentation spatiale de l'information, en lui donnant une permanence dans le temps.

Grâce à ces possibilités nouvelles d'inscription, de nouvelles connaissances ont pu naître de **l'émergence de représentations qui ne peuvent être formulées oralement**.



Exemple : Le tableau

Goody expose l'exemple du tableau ou de la liste qui permettent de mettre en relief

des relations qui ne pourraient émerger par la description orale du tableau ou de la liste.

Soit par exemple la lecture orale suivante :

Lire le contenu du tableau ci-après à voix haute.

Soit la représentation tabulaire de la même information :

Année	Chiffre d'affaire	Bénéfice
2004	123.315	5.154
2005	115.247	7.156
2006	114.265	8.245
2007	112.250	8.300

La spatialisation permet de faire émerger des connaissances nouvelles, par exemple le fait que la baisse du chiffre d'affaire est corrélée à une hausse des bénéfices. Le changement de support a eu une influence sur la connaissance elle-même.



Exemple : Les mathématiques

Un autre exemple est celui des mathématiques, qui n'existeraient pas sans écriture.



Fondamental : Changer de support c'est changer de raisonnement

Le passage de l'oral à l'écrit n'est donc pas seulement un changement de support, c'est une révolution cognitive.



Définition : La raison computationnelle

De même que l'écrit a permis le passage du temporel au spatial par projection de la parole, le support numérique apporte de nouvelles formes de représentation des informations, basées sur le calcul. Bachimont parle de l'émergence d'une « *raison computationnelle* » (Bachimont, 2004) [Bachimont04].

En effet l'ordinateur ne traite que des séquences binaires qui, par le calcul, deviennent des signes sur un support tel que l'écran. C'est cette propriété du support numérique qui est fondamentale en tant qu'elle propose de nouvelles modalités d'inscription. Et ces nouvelles modalités induisent également la constitution de modes de représentation nouveaux, comme les tableaux pour la raison graphique en leur temps (Bachimont cite par exemple la couche, le réseau, etc.).



Fondamental : Enjeu

L'enjeu est alors de comprendre **comment** inscrire les informations sur ce support spécifique qu'est le numérique afin de repérer les structures d'inscription nouvelles pertinentes.

c)Ça a été manipulé !

Bachimont (2007) [Bachimont07], p33-34) propose de caractériser le « *noème* » du numérique (en référence à Roland Barthes à propos de la photographie : « *Ça a été* »), c'est à dire **ce qu'il faut comprendre et penser à propos du numérique**, comme : « *Ça a été manipulé* ».

C'est à dire que tout contenu numérique résulte toujours d'une construction dynamique via un calcul. Lorsque je frappe sur mon clavier un calcul transforme mon action en codage binaire et en stockage d'information dans la mémoire de l'ordinateur. Lorsque je regarde mon écran, je vois le résultat d'un calcul effectué sur

le codage binaire à partir de la mémoire.

3. Le document numérique entre signe et calcul

a) Le paradigme logique

Le positivisme logique

Le positivisme logique est un courant de pensée fondé par le Cercle de Vienne (un groupe de philosophes et logiciens animé par Moritz Schlick entre 1929 et 1936).

Il pose que toute connaissance est formelle et donc se rapporte à une expression logique (au sens de la logique mathématique) (Soulez, 1986) [Soulez85].

La calculabilité

La machine de Turing, inventé en 1936 par Allan Turing (à partir des travaux de Hilbert et Gödel), est un modèle abstrait d'une machine universelle permettant d'automatiser tout calcul symbolique (alors que l'ordinateur n'existe pas encore). Elle montre que toute formalisation logique est calculable par une machine.



Complément : Exemple de machine de Turing

http://interstices.info/jcms/c_43049/machine-de-turing

L'intelligence artificielle

L'IA★ est fondée en août 1956, au Dartmouth College, lors d'un séminaire organisé par Mac Carthy et auquel sont présents des figures telles que Simon, Newell, Minsky ou Shannon.

Elle se construit sur un concept de modélisation (représentations logiques des problèmes à traiter) et d'effectivité (les représentations sont calculables).

Le mouvement de l'IA (dite forte) a pour hypothèse qu'un ordinateur peut penser puisque la connaissance est formalisable et que le formalisé est calculable par une machine de Turing.



Complément : Exemples de réalisations célèbres de l'IA

- **Les langages de représentation des connaissances**
LISP en 1958 par John Mac Carthy
- **Les systèmes experts**
DENDRAL en 1965 par Edward Feigenbaum
- **Le traitement automatique de la langue naturelle**
ELIZA en 1966 par Joseph Weizanbaum
- **La robotique**
SHRDLU en 1970 par Terry Winograd



Fondamental : Hypothèse du paradigme logique

Le paradigme logique considère les inscriptions numériques en tant que formalisations de la connaissance.

L'ordinateur calcule sur la connaissance formalisée.



Exemple : Formalisation logique de connaissances

Est-un (Voiture, Véhicule)
Possède (Voiture, Roue, 4)
Est-un (Ferrari, Voiture)

Un ordinateur pourra calculer (déduire ?) qu'une Ferrari est un véhicule et qu'il possède quatre roues.

b) Le paradigme sémiotique

Numérisation

La numérisation d'un contenu correspond à son codage (donc un calcul) de tel façon que l'on soit capable par un décodage (un autre calcul) de le restituer **tel qu'il a été codé**.

Un tel traitement correspond à une **numérisation de la forme**, car le contenu est inscrit mais non formalisé (au sens de l'IA★).

Orthothétrie

Stiegler parle d'orthothétrie pour exprimer la correspondance exacte entre information codée et information décodée (Stiegler, 1994) [Stiegler94b].

Le numérique par essence n'est pas orthothétique, le calcul l'empêche, mais elle peut être restituée par un dispositif qui la simule (principe du WYSIWYG★).



Exemple : Exemple de systèmes numériques sémiotiques

- Scanner
- Traitement de texte
- etc.



Fondamental : Hypothèse du paradigme sémiotique

Le paradigme logique considère les inscriptions numériques en tant que simple numérisation de formes sémiotiques.

L'humain interprète ce qu'il perçoit.



Exemple : Présentation sémiotique de contenu

- Une **voiture** est un véhicule à quatre roues.
- *Exemple* : Une Ferrari est une voiture

Un humain pourra déduire qu'une Ferrari est un véhicule et qu'il possède quatre roues. Un ordinateur ne peut que présenter cette information, il n'accède pas au contenu.

c) Les limites du paradigme logique

L'aporie de l'IA

L'hypothèse du positivisme logique qui fonde l'IA, à savoir que la connaissance est formelle et n'est que formelle, se révèle impossible à tenir. Il n'est pas possible de formaliser logiquement l'ensemble des connaissances humaines et l'intelligence humaine ne peut être réduite à une manipulation de symboles sans signification.

L'IA comme projet de création de machines pensantes est remis en cause (Winograd et Flores, 1989) [Winograd89].

Des machines qui pensent aux machines qui donnent à penser

Les années 1980 voient alors la naissance d'une IA modeste (ou « *faible* »), dont l'objet n'est plus la construction de machines qui pensent, mais de « *machines qui donnent à penser* » (Bachimont, 1996) [Bachimont96].



Exemple : Traduction automatique

Parmi les programmes les plus ambitieux et les plus stratégiques de l'IA, celui de la traduction automatique, largement financé par la défense américaine en période de guerre froide, avait pour terrain d'application la traduction anglais-russe. Une illustration mythique d'échec de ce programme était le test de traduction de la phrase : « *The spirit is strong, but the flesh is weak* » (« *l'esprit est fort mais la chair est faible* ») qui donna après traduction automatique en russe, puis de nouveau en anglais : « *The vodka is good, but the meat is rotten* » (« *la vodka est forte, mais la viande est avariée* »).

Au delà de son caractère amusant cet exemple illustre très bien l'aporie d'une informatique symbolique qui bute sur des concepts de base tels que le contexte, la polysémie, etc.



Complément : Théorème d'incomplétude de Gödel

Gödel montre en 1931 que les mathématiques ne peuvent être réduite à la logique. Un corollaire direct est qu'il existe des connaissances qui ne sont pas représentables par la logique.

Notons que Turing a également démontré que certains problèmes sont indécidables par une machine de Turing (sans pour autant que cela remette en cause sa vision positiviste).



Complément : La chambre chinoise de Searle

L'ordinateur ne pense pas car il n'accède pas au sens, ainsi que l'illustre Searle avec la métaphore de la "chambre chinoise" : Un opérateur qui recevrait des idéogrammes chinois et disposerait de procédures de traitement adapté à ces signes pourrait exécuter des opérations correctes, par application stricte des procédures (algorithmes). Pour autant cet opérateur ne comprendrait pas le chinois.

Ainsi, réaliser automatiquement des actions au sens d'une machine de Turing n'est pas penser.



Complément : Représentations locales versus représentation globale

La représentation formelle des connaissances a donné des résultats opérationnels tout à fait satisfaisant (par exemple certains systèmes experts), tant que cette représentation et que les raisonnements qui la concerne restent local à un domaine, en particulier technique (médecine par exemple).

C'est la formalisation globale du monde qui est aporétique.

d) Les limites de l'approche sémiotique

L'enregistrement orthothétique numérique est impossible

Parce que le contenu numérique est toujours calculé (« *il a été manipulé* »), ce que l'on lit n'est jamais ce qui a été écrit.

La simulation de l'enregistrement orthothétique est une falsification

L'approche consistant à simuler une orthothétilité masque le fonctionnement réel de la machine. Si les algorithmes sont suffisamment robustes cela n'aura pas de conséquence pratique, mais si au contraire une faille s'immisce cela entraîne l'incompréhension de l'utilisateur.

C'est une sous-utilisation de l'informatique

Si l'usage du support numérique reste limité au codage et décodage orthothétique de l'information, les potentialités du calcul, le propre du numérique, sont sous-utilisées.



Exemple : Photocopieuse numérique

Une approche exclusivement sémiotique équivaudrait au champ fonctionnel d'une photocopieuse, sans la certitude de la conformité de la copie à l'original fournie par la photocopieuse.



Complément : De la numérisation à l'informatisation

La question se pose donc de l'opérationnalisation, au sens du calcul, de connaissances en langue naturelle : Comment manipuler par des algorithmes des informations codées orthothétiquement ?

« [...] le formalisme comme principe de modélisation n'est plus tenable. En revanche, le formalisme comme principe d'effectivité reste indispensable (Bachimont 1996) » [Bachimont96] »

L'enjeu est donc d'**enrichir** le codage orthothétique pour lui ajouter des propriétés qui seront exploitables par le calcul.

Bachimont parle d'informatisation, comme état supérieur de la numérisation, du point de vue de la manipulation calculatoire (Bachimont, 2004) [Bachimont04].



Complément : La désorientation

L'impossibilité de l'enregistrement orthothétique et le calcul comme principe de manipulation conduisent à une perte de **référence** (in-vérifiabilité théorique de ce que suis je en train de lire) avec comme conséquence potentielle la désorientation du lecteur. Une informatique documentaire se doit donc de prendre acte de cet état de fait pour l'intégrer dans sa réflexion et ses instrumentations.

e) Le paradigme documentaire

Coupler enregistrement sémiotique et modélisation logique

Le support numérique est un **outil sémiotique**, permettant de mémoriser et de restituer de l'information moyennant un **calcul**.

Une approche uniquement calculatoire butte sur les limites de la formalisation en tant que principe de représentation de la connaissance et une approche uniquement sémiotique butte sur la nature même du support qui est le calcul. D'où la nécessité de coupler les deux.

Une informatique documentaire consiste à associer une dimension spatio-temporelle, l'inscription sémiotique de contenus, et une dimension calculatoire, le modèle de manipulation de ces inscriptions. Le support numérique n'a plus vocation à modéliser la connaissance, il a vocation à la véhiculer, et l'objectif d'une informatique documentaire est de concevoir des modèles qui optimisent ce véhicule.



Fondamental : Hypothèse du paradigme documentaire

Le paradigme documentaire propose une description calculable d'inscriptions sémiotiques. Il s'agit d'un compromis entre la formalisation logique du sens et l'enregistrement sémiotique de la forme.

L'ordinateur calcule sur les descriptions formelles pour décider des modalités de présentation des inscriptions, et l'humain interprète le résultat présenté qu'il perçoit.



Exemple

```
<definition>
    Une <notion>voiture</notion> est un véhicule à quatre roues.
</definition>
<exemple>
    Une Ferrari est une voiture.
</exemple>
```

L'ordinateur peut répondre à la requête telle que "Je voudrais une définition de la notion voiture" en présentant l'information :

Une **voiture** est un véhicule à quatre roues.

Il n'a aucune idée de ce qu'est une voiture, mais il permet à l'humain de le savoir en calculant une présentation adéquat du contenu qui véhicule cette connaissance.

4.Le principe du balisage documentaire

a)Balises et poignées de calcul

L'ingénierie documentaire met à profit deux thèses complémentaires :

- le contenu est numérisé dans sa forme signifiante : il est manipulable par la machine mais indépendamment de sa signification qui lui reste inaccessible ;
- le contenu est enrichi par des balises qui sont connues syntaxiquement et sémantiquement par la machine ; elle sait quoi en faire.



Fondamental

Le principe du balisage consiste à enrichir un contenu numérisé (dans sa forme sémiotique), sans l'altérer, pour lui ajouter des poignées qui vont être manipulables par l'ordinateur (logiquement).



Remarque

Le contenu est donc interprétable par l'homme et la machine, chacun via ce qui lui est destiné :

- l'humain interprète le contenu signifiant numérisé ;
- la machine interprète les balises ;



Exemple : XML

XML est une illustration de ce principe, puisque l'on va coupler une information sémiotique (texte, image, etc.) destinée à l'interprétation humaine, avec un ensemble de balises qui permettent de décrire formellement une structure documentaire qui sera alors manipulable par le calcul.

b)La structuration logique



Définition : Structuration logique

On appelle structuration logique d'un contenu une inscription explicitant la structure de ce contenu en fonction des attributs intrinsèques qui le caractérisent et non en fonction de propriétés de présentation sur un support.



Définition : Mise en forme

On appelle structuration physique ou mise en forme d'un contenu une inscription décrivant la façon dont ce contenu doit être présenté sur un support donné.



Remarque

Il est possible de calculer une ou plusieurs structurations physiques pour une même structuration logique. Il est possible de calculer d'autant plus de structurations physiques que la structuration logique est indépendante de ses supports de présentation.



Remarque

La structuration logique est associée au fond du contenu, tandis que la structuration physique est associée à sa forme sur un support.

c) Exemple de structuration logique



Exemple : Un exercice structuré logiquement

Soit la structuration logique d'un exercice :

```

Exercice = {Enonce, Question, Indice, Solution}
avec
Enonce = Soit un triangle rectangle disposant d'un angle de 30
degrés.
Question = Donner la valeur des autres angles du triangle.
Indice = La somme des angles d'un triangle est égale à 180
degrés.
Solution = 90 et 60 degrés.

```

Il est possible à partir de cette représentation de calculer différentes présentations. Pour l'écran on peut générer une présentation HTML, en laissant la solution en hyperlien cliquable. Pour le papier on peut générer une présentation PDF, en affichant la solution sur une page séparée de l'énoncé. Pour un usage multimédia on pourra générer une présentation SMIL, avec affichage de l'énoncé, lecture de la question, et affichage de la solution après un temps de pause.

Notons que si l'on avait choisi une des représentations physiques, plutôt que la représentation logique, il n'aurait pas été possible de générer les autres représentations.



Exemple : Un exercice mis en forme

Soit la mise en forme en HTML du même exercice :

```

<HTML>
<BODY>
Soit un triangle rectangle disposant d'un angle de 30 degrés.
</BR>
<B> Donner la valeur des autres angles du triangle. </B> </BR>
<A HREF="ex001i01.html"> Vous avez besoin d'aide ? </A> </HR>
<A HREF="ex001s01.html"> Vérifier votre réponse ! </A>
</BODY>

```

</HTML>

On voit que dans ce format la structure logique n'apparaît plus explicitement et qu'il n'est plus possible d'identifier l'énoncé, la question et la solution sans comprendre le contenu.



Remarque

L'exemple montre que l'on peut calculer la mise en forme à partir de la structure logique, **mais non l'inverse**.

B.Chaînes éditoriales XML

1.Introduction aux chaînes éditoriales

Une chaîne éditoriale numérique est un procédé de création de contenu multimédia qui, à la différence des outils bureautiques typiquement, ne fixe pas la mise en forme graphique du contenu a priori. La chaîne éditoriale permet en effet d'éditer une forme neutre par rapport à la mise en forme, et d'ajouter au contenu des descripteurs de structure (la partie B est incluse dans la partie A) et des descripteurs sémantiques (la partie A décrit une procédure, la partie B est une définition, ...).

La chaîne éditoriale est alors capable de traiter automatiquement cette forme neutre pour associer à chaque partie de contenu une mise en forme en fonction des descripteurs associés et du contexte de publication visé : support, public, modalité, ... Cette approche permet en particulier d'associer automatiquement plusieurs organisations et mises en forme différentes pour un même contenu.

Ce procédé d'industrialisation permet des gains substantiels en terme de productivité, de qualité et de diffusion de l'information.

a)Définition des chaînes éditoriales XML



Définition : Chaîne éditoriale XML

Une chaîne éditoriale XML est un système informatique permettant la production de documents structurés en XML, grâce à :

- un éditeur **WYSIWYM** ("What You See Is What You Mean" ou "ce que vous voyez est ce que vous voulez dire"),
- la publication **polymorphe** (Web, papier, diaporama, etc.),
- et la **ré-éditorialisation** sans recopie (dés-agrégation / ré-agrégation de documents).



Complément : Implementations

On peut distinguer trois grandes modalités pour implémenter une chaîne éditoriale :

- L'implémentation « sur mesure » consiste à mobiliser différentes briques logicielles et à programmer un logiciel spécifique pour un besoin particulier. Par exemple en couplant un éditeur du marché avec un outil de stockage et en réalisant des moteurs de publication sous la forme de feuilles XSL-XSLT.
- L'implémentation « modèle dédié » consiste à implémenter en dur un modèle de chaîne éditoriale pour un besoin assez transversal, tel que la publication de livres ou de modules de cours universitaires.

- L'implémentation « générique » consiste à réaliser un système paramétrable, indépendant d'un modèle en particulier, et à le configurer en fonction des besoins. Il s'agit alors de ce que l'on peut appeler un Système de Gestion de Chaînes Éditoriale par analogie aux Systèmes de Gestion de Bases de Donnée

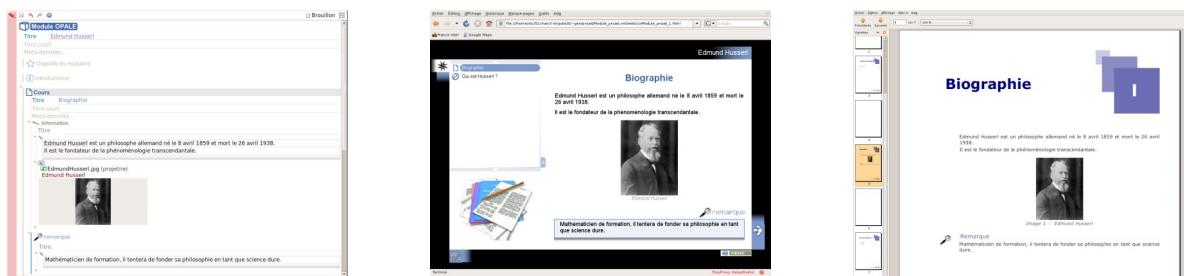


Complément : Historique : de la recherche au développement économique

Le concept de chaîne éditoriale trouve son origine dans l'édition et dans la presse : il consiste à organiser les tâches de production et de publication, en séparant les métiers intervenant dans le processus. Une chaîne éditoriale est donc un processus avant d'être un outillage, qui est né d'un besoin d'industrialisation. Historiquement, les deux technologies ayant permis l'implémentation de chaînes éditoriales numériques sont LaTeX (1982) et SGML (norme ISO depuis 1986). XML (standard W3C depuis 1998) est aujourd'hui la technologie de référence pour la réalisation de chaînes éditoriales. C'est sa maturité qui a permis de sortir ce procédé des domaines auxquels il était confiné jusque là (documentation technique stratégiques dans l'aviation ou publication scientifique typiquement).

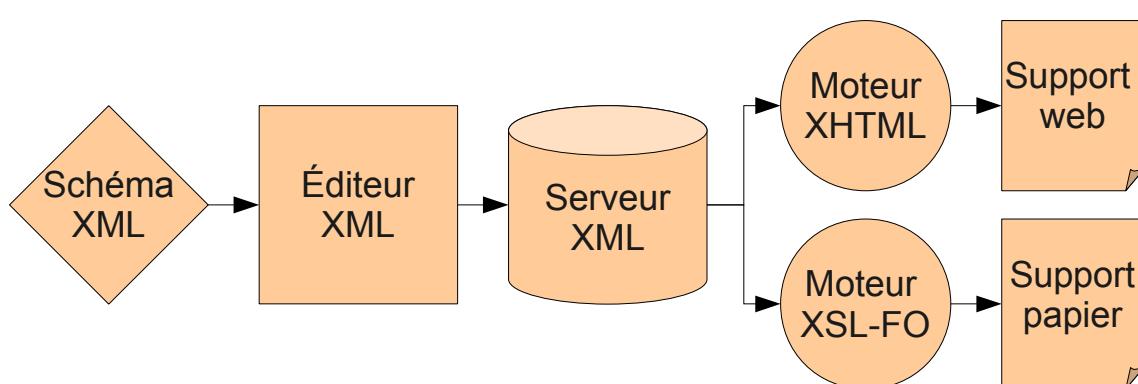
En 1999, l'Université de Technologie de Compiègne réalise Scenari, premier environnement intégré de conception de chaînes éditoriales XML proposant un langage déclaratif de modélisation et de publication de haut niveau (Scenari, la chaîne éditoriale libre [Crozat07], Ingénierie des connaissances et des contenus [Bachimont07]).

b) Illustration d'une chaîne éditoriale



c) Architecture des chaînes éditoriales XML

Architecture classique



Graphique 2 Architecture classique d'une chaîne éditoriale

L'approche classique pour réaliser une chaîne éditoriale XML est la suivante :

- Formalisation du schéma documentaire, avec un langage de modélisation de schéma XML (XML Schema, Relax NG, etc.)
- Utilisation d'un éditeur XML standard et stylage de cet éditeur (il existe de très nombreux éditeurs XML, plus ou moins graphiques, qui se paramètrent

automatiquement lorsqu'on leur fournit un schéma : Oxygen, XMetal, Epic Arbortext, etc.).

3. Utilisation de serveurs de fichiers XML pour la gestion centralisée des contenus (pour les projets les plus importants surtout).
4. Réalisation de moteurs de transformation avec les technologies XSL-XSLT, combinées avec des langages de rendu (XSL-FO pour le papier, XHTML pour l'écran, etc.)

L'ensemble est en général intégré avec un langage applicatif tiers (Java, PHP, etc.).



Exemple : Exemple de solutions

Comme solution existantes aujourd'hui l'on peut mentionner à titre d'exemple :

- **Arbortext** est une suite logicielle de création de documents, organisée autour d'un éditeur XML, de notoriété mondiale, développée par la société PTC Arbortext. Arbortext permet de développer un environnement XML selon une DTD donnée puis de développer les programmes XSLT de publication du contenu. Leur approche est orienté WYSIWYG, en mettant à disposition un éditeur XML le plus proche possible d'un traitement de texte tel que Word. Une des forces d'Arbortext est de s'intégrer avec des bases de données et des logiciels puissants de CAO pour étendre les fonctionnalités aux dessins techniques, particulièrement adaptées aux industries mécaniques. La solution - réservée aux grands groupes - est lourde et coûteuse (licences) et peu souple (pour s'orienter vers le multimédia typiquement).
- **Publimap** est un outil qui a été entièrement re-développé sur le socle de l'ECM Nuxeo, l'un des principaux éditeurs ECM français (www.nuxeo.fr¹). Publimap permet de rédiger dans un éditeur XML un contenu selon la DTD MIS et de le publier sous différents formats. Grâce à l'intégration de Nuxeo, Publimap propose une gestion des processus d'écriture et un archivage des fragments XML. La solution se fonde sur le développement « en dur » (et non paramétrable) autour d'une DTD (MIS), qui se veut universelle (comme peut être considérée celle de Word ou d'Open Office) qui permette d'écrire n'importe quel type de documents. Cette approche renoue plus avec la bureautique et n'est donc pas orientée métiers.
- **Adobe Technical Communication Suite 2** est une suite Adobe construite autour de FrameMaker qui s'ouvre sur le multimédia et le multi-supports à travers Captivate et la publication AIR. La solution est surtout orientée vers la documentation technique à grande échelle (avec l'intégration du standard DITA par exemple).
- **Scenari** est une solution libre issue de l'UTC qui propose une instrumentation complète et intégrée de la modélisation à la publication. Elle vise notamment à réduire les coûts de création et de maintenance d'une nouvelle chaîne, grâce à l'outil de paramétrage de haut niveau SCENARIbuider. Des projets récents portent sur l'intégration avec des ECM (Documentum et Alfresco), afin de renforcer la dimension collaborative.

d) Édition WYSIWYM

La plupart des systèmes d'édition numériques se fondent sur le paradigme du WYSIWYG, « What you see is what you get », ou littéralement en français : « Ce que vous voyez est ce que vous obtenez ». Cette approche vise, en résumé, à permettre à l'utilisateur de créer un document tout en composant le rendu final, comme dans sur un support traditionnel non numérique.

La chaîne éditoriale se fonde, elle, sur le concept du WYSIWYM, « What you see is

1 - <http://www.nuxeo.fr>

what you mean », c'est à dire : « Ce que vous voyez est ce que vous voulez dire ».



Définition : WYSIWYM

« Le WYSIWYM oppose sa démarche à celle du WYSIWYG en partant non pas du résultat graphique, mais de l'information à véhiculer, de sa signification et de l'intention auctoriale (Spinelli, 2006² [w_scenari-platform.org02]). »

« Le principe est pour un logiciel, de représenter les informations en fonction de leur sens, de l'information à véhiculer (par opposition à "représenter les informations sous leur forme finale, pour impression, avec mise en forme à l'identique...") (Wikipédia²). »

L'édition WYSIWYM est l'instrument technique de la séparation entre format de création et format de publication et du formatage sémantique orienté métier.



Exemple : Exemple WYSIWYM

```
<exemple>
<texte> Un texte <important>structuré</important></texte>
</exemple>
```



Exemple : Contre-exemple WYSIWYG

Contre exemple : contenu et style sont **mélangés**.

Format de création orienté métier

Le format de création d'une chaîne éditoriale s'appuie sur une description métier, c'est à dire qui véhicule une sémantique liée au domaine documentaire et à l'**intention** de l'auteur, et non au traitement informatique à réaliser. Ainsi l'on n'inscrira pas dans le format de création qu'un paragraphe est en rouge et encadré (ce qui est une indication de mise en forme liée à la publication), mais l'on dira que c'est une définition (ce qui est une information liée au contexte d'usage, compréhensible par un être humain connaissant le domaine). Un algorithme informatique fixera par ailleurs les règles de transformation du langage métier vers des langages de mise en forme.

L'intérêt de cette approche est :

- de bénéficier d'un plus haut niveau de **sémantique** (il y a plus d'informations dans « définition » que dans « rouge », car on peut appliquer la règle « définition=>rouge », mais on ne peut pas déduire « rouge=>définition », rouge pouvant renvoyer à une autre intention).
- d'utiliser ce niveau de description pour réaliser des **calculs** (traitements informatiques) plus puissants sur le contenu
- d'assurer la **pérennité** de l'information en rendant indépendant le format de création par rapport aux évolutions technologiques (le format reste humainement interprétable)

e) Polymorphisme et rééditorialisation



Définition : Séparation des formats de création et de publication

La séparation entre les formats de création et de publication, également appelée séparation fond/forme ou fonds/formes consiste à utiliser un format informatique pour la **production** du contenu et un ou plusieurs autres pour sa **publication** (Bachimont, Crozat, 2004² [Bachimont04b]).

2 - http://fr.wikipedia.org/wiki/What_you_see_is_what_you_mean

Dans le cadre d'une chaîne éditoriale numérique, le passage du format de création aux formats de publication est assuré automatiquement par un programme informatique.

L'intérêt de cette séparation est :

- de pouvoir utiliser des formats adaptés en fonction des objectifs opérationnels (créer, archiver, réutiliser, diffuser, etc.),
- de pouvoir disposer de plusieurs formats de publication à partir d'un seul format de création.



Définition : Polymorphisme

Le polymorphisme (ou séparation fonds/formes) consiste en la possibilité technique de disposer d'une **source unique** (*single sourcing* en anglais, voir Wikipédia [\[wikipediaA\]](#)) de contenu et de la transformer à volonté selon les supports et mises en formes désirés.

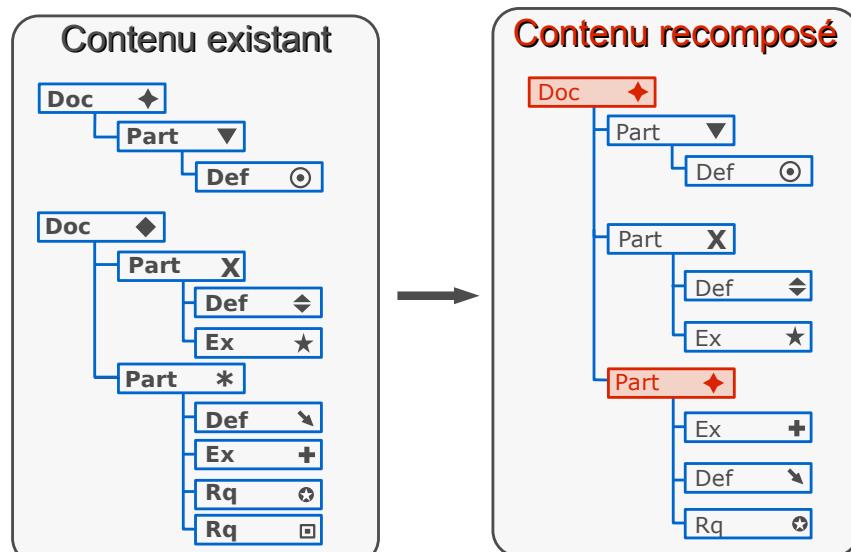
Le polymorphisme est un possible technologique qui reste limité dans la pratique : en effet il est rare que l'on souhaite présenter exactement la même information sous deux supports différents pour deux usages différents. Une nouvelle publication implique généralement la sélection du contenu (telle partie en plus, telle partie en moins), sa réorganisation (telle partie avant telle autre), sa remise en contexte (introduction, conclusions, transitions), etc.



Définition : Réutilisation

La réutilisation (ou séparation scénario/contenu) consiste à profiter du découpage logique du contenu balisé avec un langage XML métier, pour appliquer des césures physiques (découpage de fichiers XML et utilisation de liens par référence).

Il devient alors possible de partager de mêmes fragments documentaires entre plusieurs documents, ce qui permet la réutilisation sans recopie.



Graphique 3 Principe de la réutilisation : lors de la création de nouveaux documents des fragments de contenus existants peuvent être utilisés.



Définition : Ré-éditionnalisation

On appelle ré-éditionnalisation (le terme anglais de *repurposing* étant encore plus adéquat) la remise en contexte de **fragments** issus d'un fonds documentaire, par

leur **ré-agencement** au sein d'un nouveau document, leur **augmentation** par une création de contenus spécifiques et leur **publication** sur un nouveau support et/ou pour un nouveau public.

C'est une combinaison de la réutilisation et du polymorphisme.

f) Les 6 "M"

Modèle

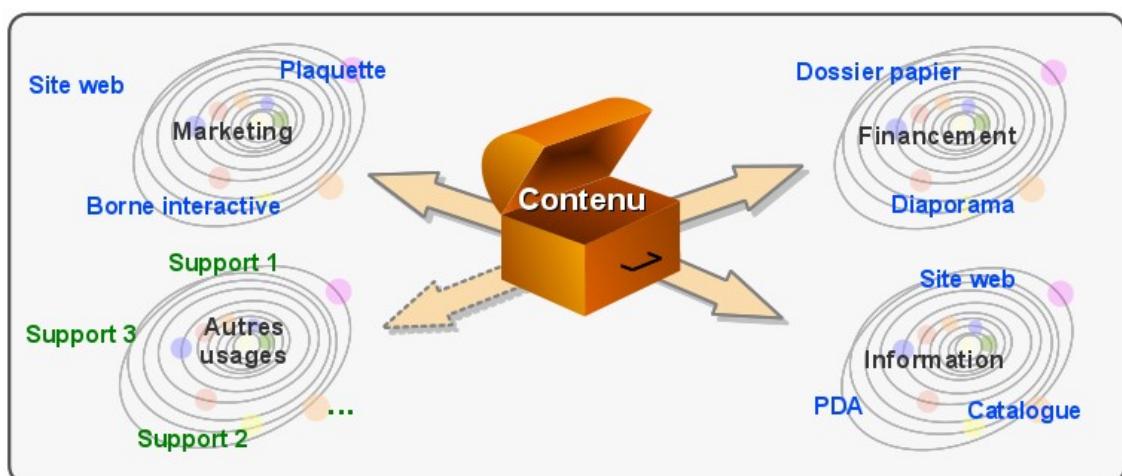
Le modèle sert de guide d'écriture en proposant à l'auteur les éléments de contenu pertinents dans le contexte de sa rédaction. Il assure une qualité standard de structuration, quelque soient les compétences éditoriales des auteurs. Il permet d'homogénéiser les publications, dans le respect des chartes graphiques et malgré les disparités d'auteurs.

Multisupports

Le multisupports permet de n'écrire qu'une seule fois ce qui est publié plusieurs fois. Il permet un gain de temps à la production du contenu, mais surtout il permet de ne maintenir qu'un seul fonds documentaire, sans redondance d'information.

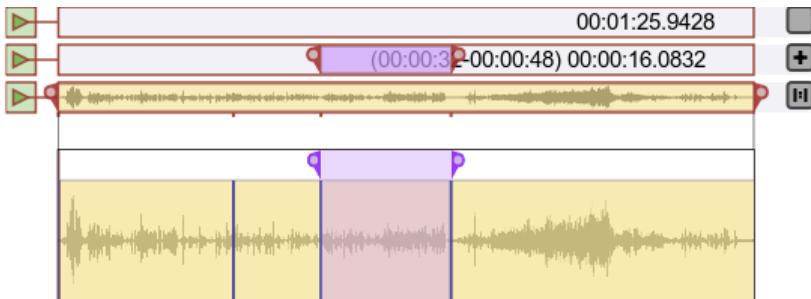
Multiusages

Le multiusages consiste à découper le contenu en unités documentaires autonomes et recombinables. Il permet de réutiliser ces unités documentaires sans recopie (par référence), afin d'adapter un discours à différents contextes.



Multimédia

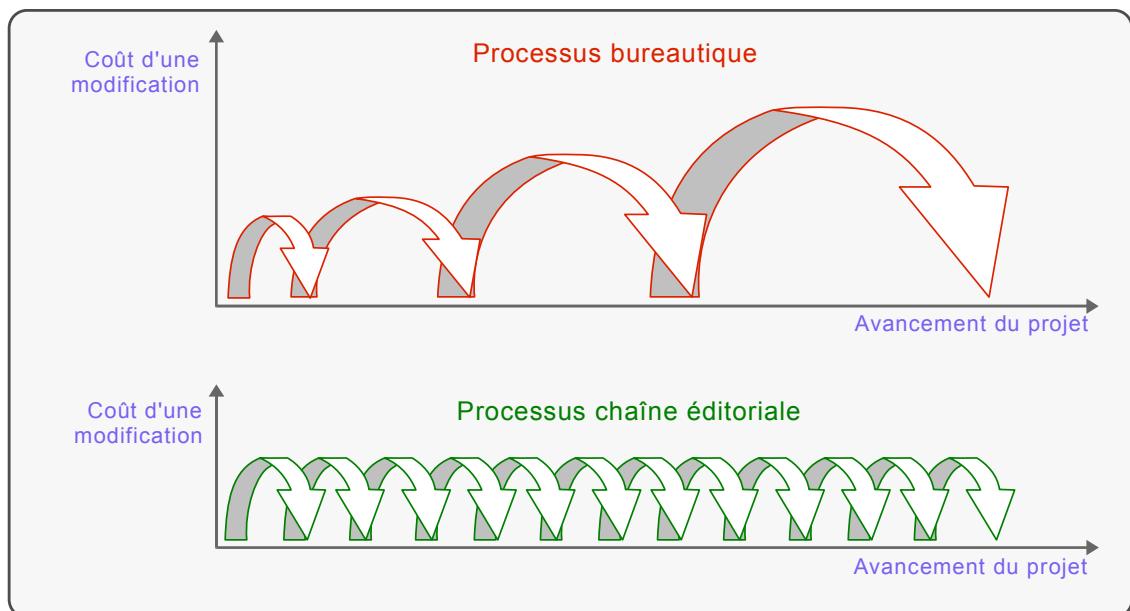
Le multimédia signifie la capacité à intégrer tous types de contenus (du texte à l'audiovisuel) et à structurer aussi bien les contenus textuels, mais aussi spatiaux (par exemple en définissant des zones dans des images) ou temporels (par exemple en définissant des segments dans un flux audio).



Exemple d'interface de structuration d'un document sonore (Scenari)

Maintenance

La maintenance des contenus est assurée dans le temps par l'indépendance technologique des formats de création (ils ne sont pas liés aux évolutions techniques comme les formats de publication, ils ne sont liés qu'aux évolutions du métier). La maintenance des publications est facilitée par le principe de génération automatique qui permet d'intervenir systématiquement sur la forme de la totalité du fonds (changement de charte graphique par exemple) sans retoucher aux contenus (la quantité de travail pour maintenir la publication est indépendante du volume de contenu).



Graphique 4 Processus d'amélioration continue des contenus

Métier

Le modèle de document est contextuel et s'appuie sur le langage du domaine (et non sur un langage technique de publication), il est donc facilement appropriable par les acteurs. Cela permet également de configurer les interfaces aux justes besoins

(plutôt que de disposer de fonctions génériques jamais toutes utilisées) et d'archiver les contenus dans un langage non technique et donc d'assurer leur accessibilité dans le temps.

g) Problématique : Motivations et contexte d'usage

Industrialisation : réduction des coûts et contrôle qualité

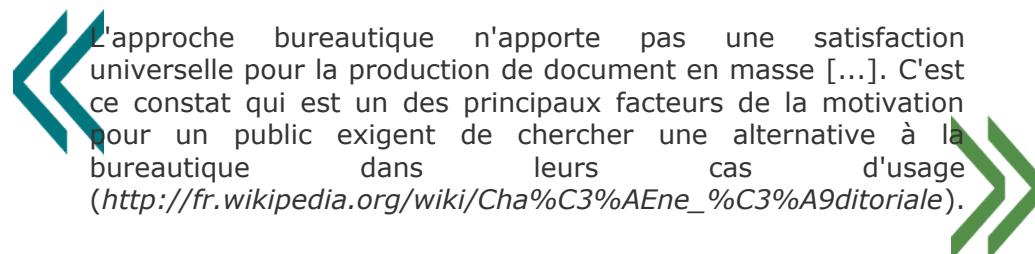
L'objectif d'une chaîne éditoriale numérique est d'instrumenter l'industrialisation d'une production documentaire. On retiendra dans le concept d'industrialisation les notions de :

- **Massification** : être capable de produire de grands volumes (plusieurs milliers de pages), malgré la rareté de compétences techniques (comme la capacité à mettre correctement en forme un document selon les canons du contexte et du support).
- **Économie d'échelle** : être capable de réduire les coûts de production et de maintenance
- **Contrôle qualité** : être capable d'assurer a priori un niveau de qualité requis (homogénéité, respect de règles éditoriales, graphiques, métiers, accessibilité, etc.)

On peut faire le parallèle de ce point de vue entre une chaîne éditoriale est une chaîne de production de produits manufacturés : l'objectif est de rationaliser pour massifier. L'approche s'oppose à une approche artisanale ou artistique (principe de l'œuvre unique).

Les limites du paradigme de la bureautique

La bureautique a permis la **démocratisation** de l'usage du numérique pour les pratiques documentaires, en rendant accessible les outils à tous. Mais dans son instrumentation, elle s'est majoritairement limitée à calquer les pratiques antérieures au numérique (comme la machine à écrire), en les améliorant, mais sans les repenser. Les raisons étant essentiellement opérationnelles, cette approche promettant de toucher le plus grand nombre le plus rapidement.



L'approche bureautique n'apporte pas une satisfaction universelle pour la production de document en masse [...]. C'est ce constat qui est un des principaux facteurs de la motivation pour un public exigeant de chercher une alternative à la bureautique dans leurs cas d'usage (http://fr.wikipedia.org/wiki/Ch%C3%A9ne_%C3%A9ditoriale).

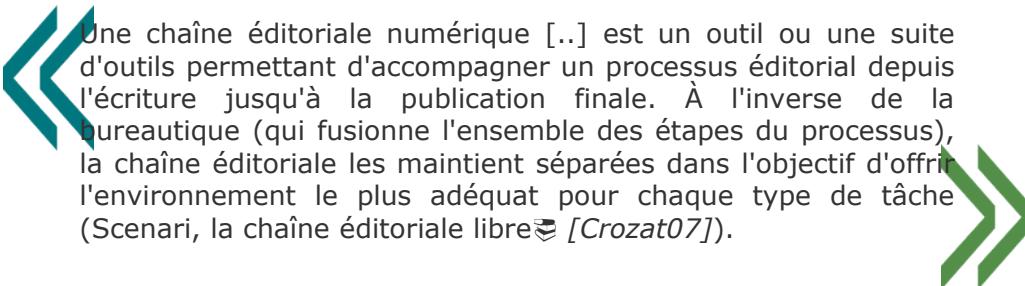
La chaîne éditoriale propose de changer le paradigme fondateur, et plutôt que de copier les pratiques antérieures, elle propose une approche originale en symbiose avec les principes du numérique. Après l'objectif quantitatif atteint par la bureautique, l'enjeu est de remettre en avant des considérations qualitatives qui commencent à faire défaut dans les usages.

La séparation des métiers

Traditionnellement la production documentaire fait appel à plusieurs métiers (auteur, rédacteur, correcteur, éditeur, diffuseur, etc.). L'outil informatique, en facilitant certaines tâches (correcteurs orthographiques, outils simplifiés de mise en page, etc.), a tendu à fusionner tous les métiers en un seul « auteur-rédacteur-éditeur ». Mais, au delà de l'aspect technique, ces métiers sous-tendent des compétences qui font en général défaut à l'auteur (savoir écrire n'est pas savoir éditer), et la

conséquence en est une dégradation importante des publications réalisées par les éditeurs « amateurs » que nous sommes (presque) tous.

L'objectif poursuivi par la chaîne éditoriale est de réintroduire ces métiers, en réorganisant une chaîne de production où chaque compétence est mise à profit pour ce qu'elle est.



Une chaîne éditoriale numérique [...] est un outil ou une suite d'outils permettant d'accompagner un processus éditorial depuis l'écriture jusqu'à la publication finale. À l'inverse de la bureautique (qui fusionne l'ensemble des étapes du processus), la chaîne éditoriale les maintient séparées dans l'objectif d'offrir l'environnement le plus adéquat pour chaque type de tâche (Scenari, la chaîne éditoriale libre [Crozat07]).



Fondamental

Ainsi la chaîne éditoriale vise d'une part à **rompre avec les techniques traditionnelles** de production prolongées par la bureautique, et d'autre part à **réhabiliter les processus professionnels** de production éditoriaux, mis à mal par la bureautique.

2.L'environnement de conception de chaînes éditoriales Scenari

a) Définition de Scenari



Définition : Scenari

Scenari est un logiciel libre permettant de créer des contenus multimédia structurés selon une approche innovante : celle de la chaîne éditoriale XML.

Il permet de répondre aux enjeux actuels de la création et la gestion des documents numériques :

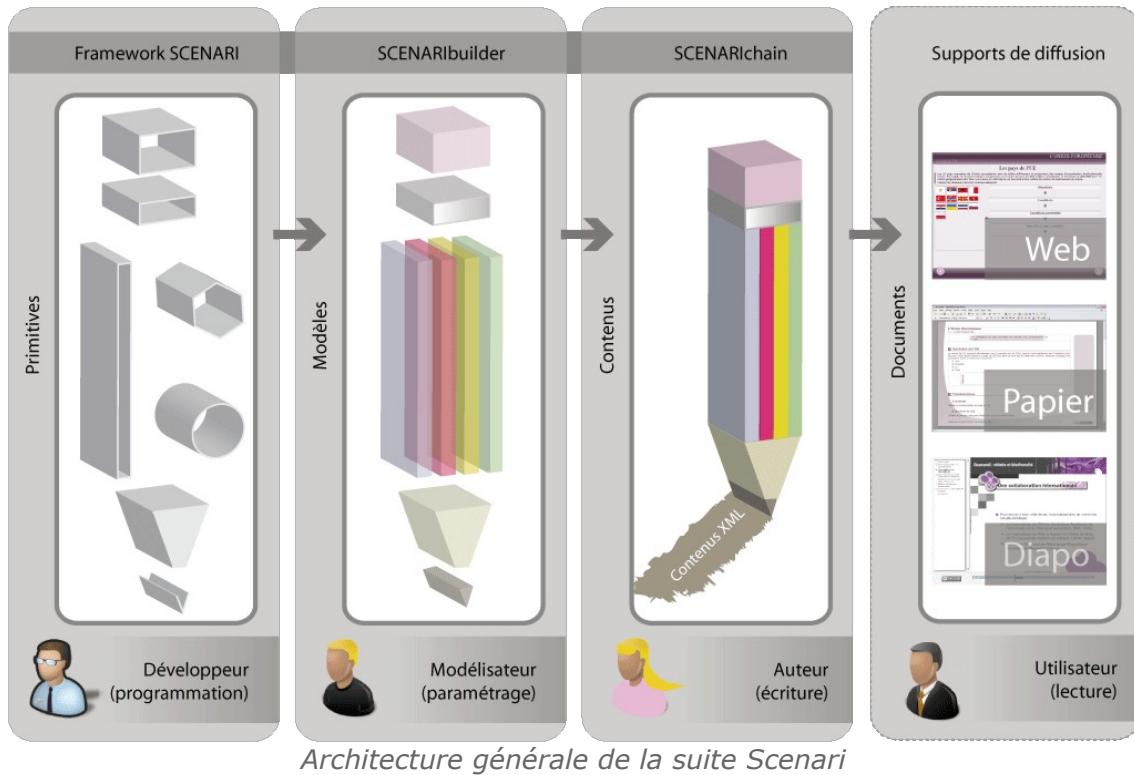
- réduction des coûts de production et de maintenance
- maîtrise de la qualité des publications
- multiplication des canaux de diffusion
- diversification des modalités de communication
- pérennisation de l'information



Complément

<http://scenari-platform.org>

b) La suite logicielle Scenari



- SCENARIconain permet à des auteurs de créer des contenus et de publier des documents dans le cadre d'une chaîne éditoriale créée avec SCENARIbuilder.
- SCENARIbuilder permet à des modélistes de concevoir des chaînes éditoriales spécifiques à chaque métier, par l'agencement de composants paramétrables (primitives).
- Le framework Scenari est le système technique générique qui permet aux développeurs de mettre à disposition des modélistes les composants paramétrables pour concevoir une chaîne éditoriale dans SCENARIbuilder.

c) La communauté scenari-platform.org

Site principal du projet

scenari-platform.org³

- Téléchargement
- Documentation
- Wiki
- Mailing-list
- etc.

Constitution de la communauté

- Unité ICS⁴ de l'UTC : Un porteur de projets de recherche et communautaires
- Kelis⁵ : Un éditeur professionnel
- *scenari-sup*⁶ : Un réseau d'utilisateurs et de points relais dans l'enseignement supérieur (universités, grandes écoles, UNT et UNR)

3 - <http://scenari-platform.org>

4 - <http://www.utc.fr/ics>

5 - <http://www.kelis.fr>

6 - <http://www.utc.fr/ics/scenarisup>

- *scenari-territorial*⁷ : Un réseau d'intégrateurs territoriaux
- Une offre commerciale via des consultants et des intégrateurs (Odigi, Amexio, Dixap, Syfadis, Nuxéo, ...)
- Des entreprises et administrations utilisatrices (Axa, Sncf, Ina, Crédit Agricole, Société Générale, AMUE, MINEFI, MEDAD, CFSSI, ...)
- Des utilisateurs autonomes (enseignants du secondaires, indépendants, ...)

Indicateurs

- 500 inscrits à la lettre d'information hebdomadaire
- 15.000 visiteurs (en IP unique par mois)
- 3.000 téléchargements (par mois)
- 7.000 messages sur les forums
- 200 organismes utilisateurs et projets estimés

d) Exemples d'usages



Exemple : Documents de formation

- Contenus pédagogiques académiques (scenari-sup, universités, secondaire, professeurs)
- Contenus pédagogiques professionnels (centres de formation, grandes entreprises, gendarmerie, ministères)
- e-learning (Axa, Crédit Agricole, Comptalia)
- Formation-action orientée méthode (universités, SNCF, PSA, Oséo-Anvar)
- Formation à base de cas (CCMP, milieu médical, Écoles des Mines)



Exemple : Documentation technique

- Documentation logicielle (AMUE, Medad, Ircam, PMB Services, éditeurs Open Source, cybercafés)
- Documentation métier (Vidal, SGIB)



Exemple : Documentation institutionnelle et administrative

- Présentation d'entreprise (PME, laboratoires)
- Catalogue (centres de formation, MINEFI, ADEME)
- Arrêtés municipaux (mairies)
- scenari-territorial (EPN, valorisation, patrimoine, etc.)



Exemple : Gestion des connaissances

- Documentation qualité (université d'Angers, SNCF, PSA)
- Knowledge management (Crédit Agricole, Société Générale, Total, Quick)



Exemple : Audiovisuel et multimédia

- Webradio (Ina)
- WebTV (Ina)



⁷ - <http://scenari-territorial.net>

Complément : Voir aussi

- Vitrine de projets
<http://scenari-platform.org/projects/scenari/fr/showroom>
 - Exemple de contenus en ligne
<http://scenari-platform.org/trac/scenari/wiki/PoweredByScenari>
- e) Chaînes éditoriales collaboratives multimédia

Objectif

Répondre aux besoins de la création multimédia collaborative, en couplant les chaînes éditoriales XML, les outils collaboratifs de gestion de contenu (Enterprise Content Management) et les solutions d'édition multimédia afin de concevoir un système de création, gestion et publication numérique complet, profitant du meilleur des trois mondes :

1. le pouvoir d'expression du multimédia ;
2. la puissance de l'écriture structurée et de la publication multi-supports automatisée ;
3. la fiabilité, l'organisation et la dynamique collaborative.

www.utc.fr/ics/c2m⁸

Plan de travail

- Existant : SCENARIserver3.7
- En cours : Développement de Scenari 4 et de projets de couplages Scenari/ECM : Alfresco, Documentum, Nuxéo (Crédit Agricole, Société Générale, École de l'Armée de l'Air)
- En R&D : Projet "Mémoire Scientifique" (UTC, École des Mines d'Albi), projet 2IE, projet Quick
- Projet ANR CONTINT (2009-2011) : application INA et Radio France (démarré le 18 septembre 2009)

Autres axes d'évolution

- Mobilité (publications iPhone/Android, eBook, podcasting, ...)
- Accessibilité (handicap, ...)
- Développement à l'international
- ...

C.eXtensible Markup Language : Syntaxe et schémas

1.Définition du XML par le W3C

XML★ est un standard du W3C★ : Extensible Markup Language (XML) 1.0 (Fifth Edition) [w_w3c.org/TR/2008/REC-xml].

Les définitions suivantes sont directement copiées du site du W3C [w_w3c.org/XML] (novembre 2009).



Définition : What is XML?

The Extensible Markup Language (XML) is a **simple text-based format** for representing **structured information**: documents, data, configuration, books,

8 - <http://www.utc.fr/ics/c2m>

transactions, invoices, and much more. It was derived from an older standard format called **SGML** (ISO 8879), in order to be more suitable for Web use.



Définition : What is XML Used For?

XML is one of the most widely-used formats **for sharing structured information** today: between programs, between people, between computers and people, both locally and across networks.



Exemple : A short example:

```
<part number="1976">
    <name>Windscreen Wiper</name>
    <description>The Windscreen wiper automatically removes rain
from your windscreen, if it should happen to splash there. It
has a rubber <ref part="1977">blade</ref> which can be ordered
separately if you need to replace it.
    </description>
</part>
```

2. Introduction à XML

a) Définition du XML



Définition : XML

L'eXtensible Markup Language XML [w3c.org/XML] est un **méta-langage** permettant de définir des langages à **balises**.

Il standardisé comme une recommandation par le W3C★ depuis **1998**.



Définition : Méta-langage

Un méta-langage est un langage permettant de définir d'autres langages, les langages ainsi définis permettent à leur tour la description d'informations respectant ces langages.



Définition : Langage à balises

Un langage à balises et un langage permettant d'associer à un contenu (généralement du texte) des balises explicites rendant compte de la structure.



Exemple

- HTML★ est un langage à balise.
- XHTML est un langage XML

b) Exemple : Un document XML



Exemple : Un mail

```
<?xml version='1.0' encoding='iso-8859-1'?>
<mail>
    <de>stephane.crozat@utc.fr</de>
    <a>fabrice.issac@utc.fr</a>
    <objet>Demande d'information</objet>
```

```

<date>01-03-2001</date>
<corps>
    <paragraphe>Bonjour Fabrice,</paragraphe>
    <paragraphe>Peux tu m'expliquer ce qu'est le langage XML?
</paragraphe>
    <paragraphe>Il me faudrait en effet ton avis éclairé sur le sujet</paragraphe>
    <paragraphe>J'attends ta réponse, à bientôt</paragraphe>
</corps>
</mail>

```

c) Historique : de SGML à XML

SGML

XML hérite historiquement (et fonctionnellement) du méta-langage SGML, développé au début des années 1980 et largement utilisé dans le cadre des systèmes documentaires, notamment dans les industries de hautes technologies pour lesquelles la fiabilité de la documentation est cruciale. SGML restait un formalisme insuffisamment explicite et trop complexe, ce qui rendait difficile la programmation d'applications SGML dans des cadres plus larges. SGML laissait notamment des ambiguïtés en terme d'interprétation du balisage. XML est un langage respectant la syntaxe SGML, mais en y ajoutant des contraintes supplémentaires, afin d'en lever les ambiguïtés.

- **A l'origine SGML :**
 - norme ISO (1986) créé pour la représentation de texte structuré
 - bien adapté aux systèmes documentaires massifs
 - exemple de langage SGML : HTML
- **Un formalisme pas assez explicite :**
 - rend difficile la programmation d'applications SGML (navigateurs, etc.)
 - XML (98) : hérite de SGML en contraignant la syntaxe
 - NB : Tout document XML est un document SGML



Exemple : Ambiguïté en SGML

En SGML une balise ouvrante n'est pas obligatoirement fermée par une balise fermante (sa fermeture pouvant rester implicite), tandis qu'en XML toute balise ouvrante est obligatoirement fermée.

Le code SGML `<A>` est ambigu car on ne sait pas s'il faut l'interpréter comme :

- `<A>` ou
- `<A>`.

En XML seule l'une des deux formulations non ambiguës est autorisée.



Exemple : HTML est un langage SGML

En SGML une balise ouvrante n'est pas obligatoirement fermée par une balise fermante (sa fermeture pouvant rester implicite), tandis qu'en XML toute balise ouvrante est obligatoirement fermée.

Ainsi le code SGML `<A>` est ambigu car on ne sait pas s'il faut l'interpréter comme :

- `<A>` ou
- `<A>`.

En XML seule l'une des deux formulations non ambiguës est autorisée.



Fondamental

XML hérite de SGML en contraignant la syntaxe, tout document XML est donc un document SGML.

d)Discussion : HTML à XML



Fondamental : XML et HTML ne sont pas au même niveau

XML est un méta-langage et HTML est un langage, XML et HTML ne sont donc pas directement comparables. En particulier XHTML est un langage XML, c'est donc du XML !

XML et le Web

Une question pertinente est en revanche de se demander pourquoi HTML ou XHTML ne suffisent pas pour les usages Web, et donc pourquoi XML est utile en dehors de son instantiation via XHTML.

La réponse est que si HTML, et *a fortiori* XHTML, fournissent une solution de publication efficiente, ils n'offrent pas le même potentiel en terme **ingénierie documentaire**. XML possède donc un potentiel de **manipulation** plus important (par exemple pour le multi-supports, la rééditorialisation, ...).



Exemple : Exemple de limites du HTML

Le HTML★, langage de référence du web qui a su s'imposer par sa simplicité d'écriture, possède également des inconvénients qui ont motivé les recherches du W3C★ sur le XML★ :

- **Jeu de balises restreint** : HTML est un langage, les balises sont définies *a priori*, il est impossible de définir ses propres balises et de créer un vocabulaire propre à un domaine spécifique. Ce manque d'**extensibilité** le rend mal approprié pour l'indexation de documents variés, le multimédia, ...
- **Mélange des descripteurs physiques et logiques** : Les balises HTML fournissent des informations relatives à la typographie ou la mise en page d'un texte (CENTER, B, ...) mais également relatives à son rôle dans le document (H1, H2, ...). Cette non séparation stricte entre contenu et présentation nuit à la publication multi-supports et la rééditorialisation par exemple.
- **Liens** : En HTML, les liens sont généralement des pointeurs directs vers une autre ressource (A). Cette mise en œuvre limite fortement les possibilités de réutilisation, notamment par le risque de lien cassé.

e)Caractéristiques recherchées pour un document XML



Remarque : Non ambiguïté

Les règles syntaxiques strictes d'XML rendent son interprétation informatique univoque.



Remarque : Lisibilité

Un document XML a pour objectif d'être lisible par un être humain, afin de plus facilement en apprêhender le format.



Remarque : Passivité

Un document XML est passif, il dépend de programmes informatiques qui le transforment.

f) Exemples de langages XML



Exemple : Exemples de langages XML pour la mise en forme de documents

- SMIL
- XHTML
- XSL-FO
- etc.



Exemple : Exemples de langages XML comme formats numériques

- SVG
- etc.



Exemple : Exemples de langages XML de structuration logique de document

- DocBook
- DITA
- etc.



Exemple : Exemples de langages XML de description et d'indexation de document

- MPEG-7
- etc.



Exemple : Exemples de langages XML d'échange de données

- SOAP
- ebXML
- etc.



Exemple : Exemples de langages XML de programmation

- XSL-XSLT
- ANT
- etc.



Attention : Langages locaux

Tous les langages spécifiés localement !

g) Remarques concernant XML



Remarque : XML orienté document et XML orienté données

Il existe quatre grandes classes d'usage du XML :

• **Le langages XML orientés données**

Ils permettent d'enregistrer et de transporter des données informatiques structurées (comme par exemple des données gérées par des bases de données) selon des formats ouvert (c'est à dire dont on connaît la syntaxe) et facile à manipuler (les structures arborescentes XML étant plus riches que des fichiers à plat par exemple).

- **Les langages XML orientés structuration logique des documents**

Ils permettent d'enregistrer et de manipuler des fonds documentaires.

- **Les langages XML orientés publication de documents**

Ils définissent des formats de fichier pour une mise en forme, en général pour un support donné. HTML en est un exemple pour l'écran et XSL-FO un autre pour le papier.

- **Les langages XML de programmation**

Ils utilisent le formalisme XML pour définir des langages de programmation, à l'instar du C ou du Java. XSL-XSLT est un exemple de langage de programmation écrit en XML. Les langages de programmation écrits en XML sont généralement à vocation déclarative.



Remarque : XML orienté document

XML permet la structuration logique des documents, mais que XML est tout à fait indépendant de ce principe, puisqu'il ne définit que des principes syntaxiques.

XML permet la structuration logique du texte :

- Par repérage des éléments constitutifs du texte (chapitre, paragraphe, mots importants, etc.) et des relations de hiérarchie entre ces éléments (un paragraphe appartient à un chapitre).
- Sans se préoccuper de la mise en forme du texte (mise en page, polices, etc.). On ne voit pas à l'écran ce que donnera le document publié.



Complément : Versions de XML

La version historique de XML est la version 1.0, qui reste aujourd'hui la plus largement utilisée.

Il existe également une version 1.1 de XML, qui propose des différences mineures et nécessaire uniquement dans des contextes particuliers. Une nouvelle version a été nécessaire pour des raisons de compatibilité des outils existants. Les évolutions principales de XML 1.1 sont de nouveaux caractères permis pour les noms d'éléments (pour suivre l'évolution d'Unicode depuis 1998), de nouvelles conventions pour les caractères de fin de ligne (pour la compatibilité avec des ordinateurs *main frame*) et de nouveaux caractères de contrôle de contrôle dans le contenu.

3.Syntaxe XML

a) Notion de document bien formé

Un document XML★ permet d'utiliser un ensemble de balises fournissant des informations supplémentaires sur le contenu. Cet ensemble doit être établi d'après les différents constituants (logiques et sémantiques) du document (ou du type de document traité).



Définition : Document bien formé

Un document est dit bien formé lorsqu'il respecte les règles syntaxiques du XML★.

XML★ n'impose pas de sémantique à priori mais uniquement des règles syntaxiques. Les règles syntaxiques à respecter sont :

- Il n'existe qu'un seul élément père par document, cet élément contenant tous les autres. Il est également appelé **racine**.

- Tout élément fils est inclus complètement dans son père (pas d'éléments croisés).



Remarque

XML ne définit pas le comportement ni la manière dont doit être traité un document.

b) Balises



Définition : Balise

Les balises sont les composants fondamentaux permettant l'écriture de documents XML★.

Elles se distinguent du contenu en utilisant les caractères <, > et /. Elles n'ont pas de présentation ou de signification définie par le langage mais elles peuvent avoir un sens pour les applications et/ou le lecteur.

Il existe trois types de balises (les attributs sont optionnels) :

- balise d'ouverture : <nom_élément [attributs]>
- balise de fermeture : </nom_élément>
- balise élément vide : <nom_élément [attributs]/>



Exemple : Exemples de balises XML

```
<adresse>12, rue de Paris</adresse>
<date jour='15' mois='02' annee='2003' />
```

c) Éléments



Définition : Élément

Un élément XML★ est de la forme :

- <nom [attributs]> contenu </nom>
- ou <nom [attributs]/> pour les éléments vides.

Le nom d'un élément peut être composé de tout caractère alphanumérique plus "_", "-" et ". ". De plus, ils doivent commencer par un caractère alphabétique ou "_" et ceux commençant par la chaîne "xml" sont réservés (que ce soit en minuscules, majuscules ou un mélange des deux).



Attention

XML★ différencie les majuscules des minuscules, il faut donc respecter la casse.

Le contenu d'un élément peut être une combinaison de **texte**, d'autres **éléments**, ou être **vide**. Il peut également contenir des commentaires et des instructions de traitement .



Attention

Deux éléments **ne doivent pas** avoir un contenu croisé :

- <nom1> ... <nom2> ... </nom1> ... </nom2>



Remarque : Attributs

Les attributs d'un élément sont formés d'une suite d'affectations séparées par des espaces :

- attribut1='do' attribut2='ré' ...

d) Attributs



Définition : Attribut

Un attribut permet de fournir une valeur caractérisant l'élément pour lequel il est défini. Il se présente sous la forme d'une affectation où la valeur est indiquée entre apostrophes ou guillemets (au choix, mais pas de mélange des deux) :

- Nom_attribut='valeur' ou Nom_attribut= "valeur"



Remarque

Utilisez des apostrophes si la valeur de l'attribut inclut des guillemets et vice et versa.

Le nom d'un attribut est soumis aux mêmes contraintes que les noms d'éléments. Il peut être composé de tout caractère alphanumérique plus "_", "-" et ".". De plus, il doit commencer par un caractère alphabétique ou "_" et ceux commençant par la chaîne "xml" sont réservés (que ce soit en minuscules, majuscules ou un mélange des deux).

La valeur de l'attribut quant à elle peut contenir tout caractère à l'exception de "^", "%" et "&".

e) Structure

Un document XML★ est constitué de :

- un prologue
Il est facultatif et comprend :
 - une déclaration XML★, indiquant la version du langage XML★ utilisé, le codage des caractères dans le document et si le document contient des déclarations extérieures (cf. exemple 1). Chacune de ces informations est optionnelle mais leur ordre est obligatoire.
 - une déclaration de type de document (optionnelle), précisant la structure à laquelle le document doit être conforme (cf. exemple 2).
- un arbre d'éléments contenant au moins un élément.
- des commentaires et instructions de traitement
Ils sont facultatifs :
 - Les commentaires se présentent de la manière suivante :
<!-- Ceci est un commentaire -->
 - Quant aux instructions de traitement, ce sont des informations destinées à des applications pour faciliter le traitement du document.



Exemple : Prologue

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

Indique que le document est codé en utilisant un langage XML★ de version 1, avec des caractères codés selon la norme ISO-8859-1 et que le document ne fait pas référence à des documents externes.



Exemple : Déclaration de type de document

```
<!DOCTYPE mon_document SYSTEM "mon_document.dtd">
```

Indique que les règles de structure définies dans la DTD★ contenue dans le fichier mon_document.dtd seront à respecter dans ce document XML★.



Exemple : Instance

```
<lettre>
  <expediteur>moi</expediteur>
</lettre>
```

f) Exemple de fichier XML - Le courriel



Exemple : Un courriel imprimé

Date: Mar, 28 Oct 2003 14:01:01 +0100 (CET)

De : Marcel <marcel@ici.fr>

A : Robert <robert@labas.fr>

Sujet: Hirondelle

Salut,

Pourrais-tu m'indiquer quelle est la vitesse de vol d'une hirondelle transportant une noix de coco ?

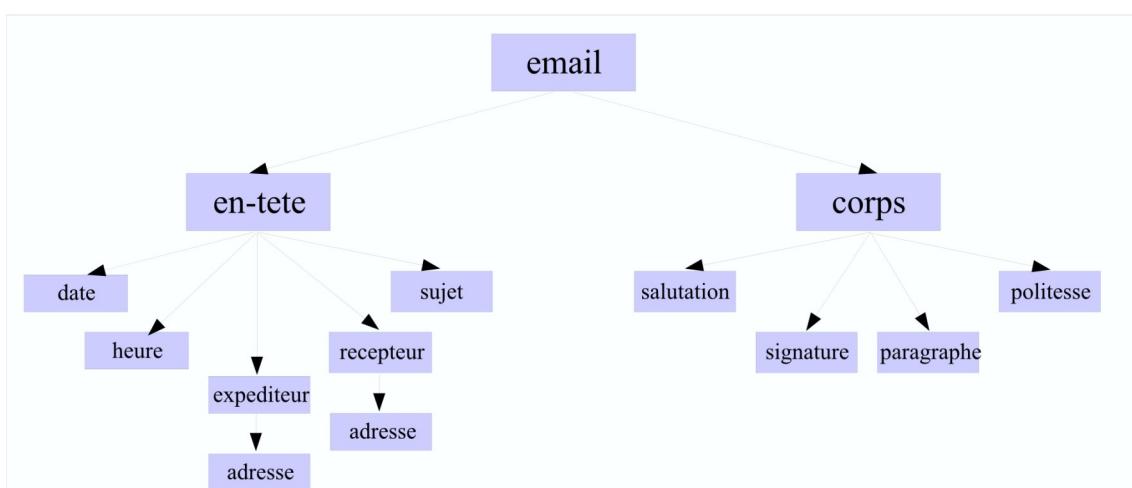
A très bientôt,

Marcel

Représentation possible de ce message en XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<email>
  <en-tete>
    <date type='JJMMAAAA'>28102003</date>
    <heure type='24' local='(GMT+01 :00)'>14:01:01</heure>
    <expediteur><adresse
      valeur='marcel@ici.fr'>Marcel</adresse></expediteur>
    <recepteur><adresse
      valeur='robert@labas.fr'>Robert</adresse></recepteur>
    <sujet>Hirondelle</sujet>
  </en-tete>
  <corps>
    <salutation>Salut,</salutation>
    <paragraphe>Pourrais-tu m'indiquer quelle est la vitesse de
      vol d'une hirondelle transportant une noix de coco ?
    </paragraphe>
    <politesse>A très bientôt,</politesse>
    <signature>Marcel</signature>
  </corps>
</email>
```

Arbre d'éléments du document email correspondant



Représentation d'un document XML sous forme d'arbre

g) La syntaxe XML en résumé

Les règles de syntaxe d'XML sont très simples : un document XML comporte un en-tête permettant de l'identifier en tant que document XML et un élément racine contenant tous les autres éléments. Un élément est caractérisé par une balise ouvrante et une balise fermante et contient d'autres éléments et/ou du contenu.

Aspect	Syntaxe	Explication
Entête	<?xml version='1.0' encoding='iso-8859-1'?>	Tout document XML contient une ligne d'entête qui sert à identifier le document en tant que document XML, et généralement à spécifier le système d'encodage de caractères utilisé dans le document.
Elément racine	<NomElementRacine> ... </NomElementRacine>	Tout document XML contient un et un seul élément, dit contenant tous les autres.
Elément	<NomElement> ... </NomElement>	Un élément est caractérisé par une balise ouvrante et une balise fermante obligatoire, les éléments sont donc inclus les uns dans les autres et ne peuvent se croiser. Les balises tiennent compte de la casse des caractères.
Attribut	<NomElement NomAttribut='valeur'>	Un attribut associé à une valeur peut être spécifié dans la balise ouvrante d'un élément.
Elément vide	<NomElementVide/>	Un élément peut être vide, il est alors noté par une unique balise à la syntaxe particulière. Un élément vide contient en général des attributs.
Contenu	<NomElement> contenu ici </NomElement>	Le contenu est placé à l'intérieur des éléments, c'est à dire entre les balises.
Commentaires	<!-- commentaires -->	Un document peut également contenir des commentaires.

Tableau 2 Aspects principaux de la syntaxe XML

h) Namespace

Principe

Un *namespace* (ou espace de noms en français) est une mécanique qui permet d'assurer l'unicité des noms des éléments utilisés au sein des fichiers XML, dans l'objectif de pouvoir « mélanger » différents schémas.

- Soit un extrait de schéma S1 qui définit un élément de syntaxe en informatique comme contenant du code : ... **{element syntax {code {...}}}***
- Soit un extrait de schéma S2 qui définit un élément de syntaxe en mathématique comme contenant une équation : ... **{element syntax {equation {...}}}***

Si je souhaite, dans un schéma S3 réutiliser les éléments syntaxe issus de S1 et S2, je rencontre un conflit de noms : en effet, deux éléments portant le même nom, « syntaxe », définissent en fait des éléments différents.

Le *namespace* va me permettre de différencier ces deux éléments, en associant un nom unique aux schémas S1 et S2, par exemple une adresse web.

Si j'associe **www.utc.fr/S1** au premier schéma et **www.utc.fr/S2** au second, j'obtiens alors deux noms de balises différents :

- **www.utc.fr/S1:syntaxe**
- **www.utc.fr/S2:syntaxe**

Préfixe

Cette écriture étant quelque peu fastidieuse, il est également possible d'associer un préfixe au namespace. La correspondance entre le namespace et le préfixe est déclarée dans le fichier XML, ce qui permet à un programme informatique de

remplacer les préfixes par les namespaces.

Finalement, on obtient :

- Préfixe s1 associé au namespace www.utc.fr/S1 et écriture XML **s1:syntaxe**
- Préfixe s2 associé au namespace www.utc.fr/S2 et écriture XML **s2:syntaxe**



Remarque : Namespace par défaut

Il est possible de définir un namespace par défaut ce qui permet d'avoir un namespace pour chaque balise, sans avoir à utiliser de préfixe.



Syntaxe

```
<elementRacine
    xmlns:prefixe1="namespace1"
    xmlns:prefixe2="namespace2" ...
    xmlns:prefixeN="namespaceN"
    xmlns="namespaceDesÉlémentsNonPréfixés"
>
```

i) Syntaxe XML et espaces

Soit les trois syntaxes XML suivantes (les espaces sont symbolisés par des ~ et les retours chariot par des §) :

1. <a>bonjour~~~~~aurevoir
2. <a>bonjour§
aurevoir~~~~~
3. <a>bonjour~aurevoir§

Ces trois syntaxes sont strictement équivalentes.

Donc :

- À l'intérieur d'un élément : 1 espace = N espaces = 1 retour chariot = N retours chariot
- Entre deux éléments : 1 espace = N espaces = 1 retour chariot = N retours chariot = Rien (absence de caractère)

4. Introduction aux schémas XML

a) Notion de document valide



Définition : Schéma

Un schéma est une description de la structure que doit respecter un document lui faisant référence, c'est à dire qu'il établit la liste des éléments XML autorisés (avec leurs attributs), ainsi que l'agencement possible de ces éléments.

On parle aussi de **grammaire**, au sens où le schéma définit l'enchaînement autorisé des balises et vient en **complément de la syntaxe XML** (qui elle est indépendante d'un schéma particulier).



Définition : Document valide

Un document XML★ bien formé est dit valide pour un certain schéma s'il respecte les règles structurelles imposées par ce schéma.

Ce contrôle de la structure permet :

- De s'assurer l'homogénéité structurelle des documents de même type.
- Le traitement automatique d'un ensemble de documents (mise en forme, stockage, extraction d'informations...) de même type.
- La création de standards et leur respect.



Exemple : Exemples de langages de schéma

Il existe plusieurs langages de définition schéma, mais les trois principaux sont :

- Document Type Définition (W3C) : Un langage hérité de SGML qui fait partie du standard XML
- W3C XML Schema (W3C) : Une alternative aux DTD destiné à moderniser et compléter ce langage historique
- Relax NG (OASIS, ISO) : Une autre alternative, compromis entre W3C XML Schema et DTD

b) Document Type Definition

Le formalisme de définition de schéma DTD est le premier qui a été introduit dès la première version du standard XML. Il est en fait intégré au standard W3C de XML.

Il est directement hérité de la norme SGML.

Les DTDs utilisent un langage spécifique (non XML) pour définir les règles structurelles. Un fichier de DTD peut contenir principalement deux types de déclarations :

- **des déclarations d'éléments,**
indiquent les éléments pouvant être inclus dans un document et l'organisation du contenu de chaque élément (éléments fils ou texte).
- **des déclarations d'attributs,**
définissent les attributs pouvant être associés à un élément ainsi que leur type.



Exemple : Exemple de DTD

```
<!ELEMENT document (paragraphe+)>
<!ATTLIST document type CDATA #REQUIRED>
<!ELEMENT paragraphe (#PCDATA)>
```



Exemple : Exemple de document XML valide

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE document SYSTEM "document.dtd">
<document type='memo'>
  <paragraphe>XXXXXXXXXX</paragraphe>
  <paragraphe>XXXXXXXXXX</paragraphe>
  <paragraphe>XXXXXXXXXX</paragraphe>
</document>
```

c) W3C XML Schema

Les XML Schema ont été proposés par le W3C pour permettre de dépasser les limites des DTD.

On notera en particulier :

- Extension de l'expression des règles d'organisation structurelle (héritage, réutilisation, etc.)
- Ajout d'un langage de typage des éléments (particulièrement utile pour les applications orientées données de XML)

Exemple : Exemple de DTD

```
<!ELEMENT document (paragraphe+)>
<!ATTLIST document type CDATA #REQUIRED>
<!ELEMENT paragraphe (#PCDATA)>
```

Exemple : Exemple de W3C XML Schema correspondant

```
<?xml version="1.0" encoding="UTF-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
  <xss:element name="document">
    <xss:complexType>
      <xss:sequence>
        <xss:element maxOccurs="unbounded" ref="paragraphe"/>
      </xss:sequence>
      <xss:attribute name="type" use="required"/>
    </xss:complexType>
  </xss:element>
  <xss:element name="paragraphe" type="xs:string"/>
</xss:schema>
```

Complément

- Tutoriel  [Vlist, 2000b]
- Guide  [Vlist, 2000]
- Standard  [W3C, 2000]
- Syntaxe de déclaration d'un élément : <http://www.w3.org/TR/2000/CR-xmlschema-1-20001024/#element-element>

d) Regular Language for XML Next Generation

RelaxNG (REgular LAnguage for XML Next Generation) est un langage de schéma XML.

- RelaxNG est une alternative aux DTD et à W3C XML Schema, qui combine les avantages de ces deux autres langages.
- RelaxNG est un standard OASIS et une norme ISO/CEI.
- Deux syntaxes : une syntaxe XML (alternative à W3C Schema) et une syntaxe compacte (alternative aux DTD).
- RelaxNG ne définit que la structure (comme les DTD) et utilise W3C XML Schema pour le typage des données.

<http://relaxng.org/>

Complément

Le standard est porté par James Clark depuis ses travaux sur Trex (il est issu de la fusion de Trex et Relax de Murata Makoto).

Exemple : Exemple de schémas publics définis en Relax NG

OpenDocument (format bureautique)

DocBook (format documentaire)

Atom (syndication)

Exemple : Exemple de DTD

```
<!ELEMENT document (paragraphe+)>
<!ATTLIST document type CDATA #REQUIRED>
<!ELEMENT paragraphe (#PCDATA)>
```

Exemple : Exemple de schéma RelaxNG correspondant (syntaxe compacte)

```
start = element document {
    attribute type {text},
    element paragraphe {text}+
}
```

Exemple : Exemple de schéma RelaxNG correspondant (syntaxe XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <start>
        <element name="document">
            <attribute name="type"/>
            <oneOrMore>
                <element name="paragraphe">
                    <text/>
                </element>
            </oneOrMore>
        </element>
    </start>
</grammar>
```

Exemple : Autre exemple de schéma RelaxNG correspondant (syntaxe compacte, patterns nommés)

```
start = document
document = element document {attribute type {text},
paragraphe+ }
paragraphe = element paragraphe {text}
```

Exemple : Autre exemple de schéma RelaxNG correspondant (syntaxe XML, patterns nommés)

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <start>
        <ref name="document"/>
```

```

</start>
<define name="document">
    <element name="document">
        <attribute name="type"/>
        <oneOrMore>
            <ref name="paragraphe"/>
        </oneOrMore>
    </element>
</define>
<define name="paragraphe">
    <element name="paragraphe">
        <text/>
    </element>
</define>
</grammar>

```

5. Définition de type de document

a) Déclaration d'éléments



Syntaxe : Déclaration d'éléments

Les déclarations d'éléments sont de la forme :

```
<!ELEMENT nom (modèle)>
```

où :

- Le `nom` obéit aux mêmes règles que les noms dans les balises, c'est à dire commençant par un caractère alphabétique (ou un tiret-souligné) suivi des caractères alphanumériques, point ou tiret-souligné.
- Le modèle décrit la combinaison d'éléments et de texte qui pourra être contenue dans cet élément.

Il se présente sous la forme d'une liste pouvant être constituée de :

- (nom d'un ou plusieurs élément fils)
Indique que l'élément peut contenir un ou des fils ayant le nom indiqué.
- (#PCDATA)
Indique la possibilité de contenir un flot de caractères (*Parsed Character Data*).



Remarque : Élément vide EMPTY

Il est possible de déclarer un élément vide en utilisant la syntaxe EMPTY :

```
<!ELEMENT element_vide EMPTY>
```



Exemple : Exemple de déclaration d'éléments

```
<!ELEMENT texte (paragraphe)>
<!ELEMENT paragraphe (#PCDATA)>
```

Un texte contient un paragraphe qui contient un flux de caractères.

```
<texte>
    <paragraphe>Ceci est un flux de caractères</paragraphe>
```

```
</texte>
```



Syntaxe : Séparateur de structuration

Les éléments fils déclarés peuvent être combinés pour décrire en détail la structure du contenu de l'élément en les séparant par :

- ,
Indique une relation d'ordre (connecteur logique AND avec une notion d'ordre) : si par exemple si le modèle est (x, y) alors l'élément x devra être présent, et ce avant l'élément y .
- |
Indique une alternative (connecteur logique "ou exclusif" XOR). Le modèle $(x | y)$ indique "soit x soit y ".

L'utilisation de parenthèses permet de créer des sous-listes dans la liste principale pour lesquelles les suffixes de cardinalité sont également applicables.



Exemple : Exemple de déclaration de deux éléments fils ordonnés

```
<!ELEMENT texte (titre, paragraphe)>
<!ELEMENT paragraphe (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
```

```
<texte>
  <titre>Ceci est le flux de caractères du titre</titre>
  <paragraphe>Ceci est le flux de caractères du
  paragraphe</paragraphe>
</texte>
```



Exemple : Exemple de déclaration d'alternative

```
<!ELEMENT texte ((titre | accroche), paragraphe)>
<!ELEMENT paragraphe (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT accroche (#PCDATA)>
```



Syntaxe : Suffixes de cardinalité

Les éléments fils peuvent être déclarés avec des suffixes permettant d'exprimer leur cardinalité :

- ? : l'élément devra être présent de 0 à 1 fois.
- * : l'élément devra être présent de 0 à n fois.
- + : l'élément devra être présent de 1 à n fois.

L'absence de suffixe indique que l'élément doit apparaître une et une seule fois.



Exemple : Exemple général

Les éléments x , y , $z1$ et $z2$ sont représentés vides pour alléger l'exemple.

```
<!ELEMENT mon_elem (x?, y*, (z1, z2)+)>
```

permet :

```
<mon_elem> <x/><y/><z1/><z2/> </mon_elem>
<mon_elem> <z1/><z2/> </mon_elem>
<mon_elem> <z1/><z2/><z1/><z2/> </mon_elem>
<mon_elem> <x/><y/><y/><z1/><z2/><z1/><z2/> </mon_elem>
<mon_elem> <y/><y/><y/><y/><z1/><z2/> </mon_elem>
...

```

b) Déclaration d'éléments EMPTY et ANY



Syntaxe : Element de type ANY

```
<!ELEMENT nom_element ANY>
```

L'élément pourra contenir des flots de caractères ainsi que n'importe quels éléments déclarés par ailleurs.



Syntaxe : Element de type EMPTY

```
<!ELEMENT nom_element EMPTY>
```

L'élément doit être vide. Il contiendra néanmoins généralement des attributs.



Remarque

ANY et EMPTY ne sont pas combinables avec d'autres définitions d'éléments fils, mais sont compatibles avec des définitions d'attributs.



Exemple : Exemple 1 : Le paramètre ANY

```
<!ELEMENT monElement1 ANY>
<!ELEMENT monElement2 EMPTY>
permet :
<monElement1><monElement2 />bonjour</monElement1>
<monElement1>au revoir<monElement2 /></monElement1>
<monElement1></monElement1>
...

```



Exemple : Exemple 2 : Le paramètre EMPTY

```
<!ELEMENT monElement1 EMPTY>
permet :
<monElement1/> ou <monElement1></monElement1>
```

c) Déclarations de listes d'attributs



Syntaxe : Déclaration d'attribut

Les déclarations d'attributs correspondent à la forme générale :

```
<!ATTLIST nom-élément nom-attribut type-attribut déclaration-
de-constrainte>
```

où :

- **nom-élément :**

Nom de l'élément XML★ pour lequel les attributs déclarés seront applicables.

- **nom-attribut :**

Nom de l'attribut, des éléments différents peuvent avoir des attributs de même nom sans qu'il y ait de confusion possible car un attribut est toujours déclaré en même temps que l'élément auquel il est attaché.

- **type-attribut :**

Les deux types principaux sont :

- **CDATA**

L'attribut aura pour valeur une chaîne de caractères.

- **Liste de choix**

Une liste de noms symboliques correspondant aux valeurs possibles pour l'attribut et se présentant sous la forme : (choix1 | choix2 | ... | choixN).

- **déclaration-de-contrainte :**

Les deux formes principales de contrainte sont :

- **#REQUIRED**

L'attribut est obligatoire.

- **#IMPLIED**

L'attribut est facultatif.



Remarque : Déclaration de liste d'attributs

La syntaxe ATTLIST permet de déclarer des listes d'attributs, car il est possible de répéter le *pattern* : nom-attribut type-attribut déclaration-de-défaut.



Exemple : Déclaration de liste d'attributs

```
<!ELEMENT x EMPTY>
<!ATTLIST x
    att1 CDATA #REQUIRED
    att2 (a | b | c) #IMPLIED
>
```



Complément : Autres types d'attributs

En plus des types CDATA et "liste de choix", les attributs peuvent avoir les types suivants :

- **ID ou IDREF**

Sont utilisés pour définir des renvois à l'intérieur d'un document, ID identifiant de manière unique tous les éléments pouvant être référencés et IDREF indiquant la référence à cet identifiant.

- **NMTOKEN ou NMTOKENS**

Permet à l'attribut de prendre pour valeur un ou des noms symboliques quelconques, formés de caractères alphanumériques.

- **ENTITY ou ENTITIES**

L'attribut prendra pour valeur le nom d'une ou plusieurs entités externes non XML (images...).

- **NOTATION**

Est utilisé pour des éléments ayant un contenu non XML, l'attribut aura alors pour valeur le nom de l'application qui a été associée à l'application externe traitant le type de contenu concerné lors d'une déclaration de type NOTATION.



Complément : Autres déclarations de contraintes

En plus des déclarations #IMPLIED et #REQUIRED, les attributs peuvent avoir les déclaration de contrainte :

- **Valeur par défaut**

Indique la valeur par défaut prise par l'attribut (en accord avec son type) s'il n'est pas renseigné. L'attribut peut donc être renseigné ou non (comme un #IMPLIED), mais s'il ne l'est pas il prend la valeur par défaut spécifiée. Se note 'valeur'.

- **#FIXED 'valeur'**

L'attribut prend toujours la valeur indiquée, il est "constant". Il doit donc toujours être renseigné (comme un #REQUIRED), mais toujours avec la même valeur.

d) La syntaxe DTD en résumé

La syntaxe des DTD repose sur un langage déclaratif, permettant de spécifier les éléments autorisés et pour chaque élément autorisé les fils qu'il peut contenir. Les aspects essentiels de la syntaxe sont présentés dans le tableau ci-dessous.

Aspects	Syntaxe	Explications
Elément racine	<!ELEMENT a (...)>	L'élément racine est déclaré en premier par soucis de lisibilité (ce n'est pas obligatoire), c'est le seul élément à n'être inclus dans aucun autre.
Elément contenant d'autres éléments	<!ELEMENT a (b)>	Tout élément est déclaré en indiquant les fils qui lui sont autorisés (ici on déclare l'élément a qui contient un fils b)
Le 'ET' (,)	<!ELEMENT a (b,c)>	La virgule permet d'indiquer la relation ET entre les éléments fils (ici a contient un b ET un c).
Le 'OU' ()	<!ELEMENT a (b c)>	La barre permet d'indiquer la relation OU entre les éléments fils (ici a contient un b OU un c).
Le '1' (par défaut)	<!ELEMENT a (b)>	Par défaut tout élément fils déclaré doit être présent une et une seule fois dans le père (ici a contient obligatoirement un et un seul b)
Le '0 ou 1' (?)	<!ELEMENT a (b?)>	Le point d'interrogation permet de spécifier que le fils est optionnel (ici a contient zéro ou un b).
Le '0 à N' (*)	<!ELEMENT a (b*)>	L'étoile permet de spécifier que le fils est optionnel et que le père peut contenir plusieurs fois le fils (ici a contient zéro, un ou plusieurs b).
Le '1 à N' (+)	<!ELEMENT a (b+)>	Le plus permet de spécifier que le père contient au moins une fois le fils, mais peut le contenir plusieurs fois (ici a contient un ou plusieurs b).
Attribut d'un élément	<!ATTLIST NomElement NomAttribut Type Contrainte>	Plusieurs attributs peuvent être déclarés pour un élément. Chaque attribut est déclaré en spécifiant son nom, son type (libre, énuméré, identifiant unique, etc.) et la contrainte qui lui est associée (obligatoire, optionnel ou fixé).
Elément vide	<!ELEMENT a EMPTY>	Un élément vide est spécifié à l'aide du mot clé EMPTY à la place de ses fils.
Elément contenant du contenu	<!ELEMENT a (#PCDATA)>	Pour spécifier qu'un élément contient du contenu, on utilise le mot clé #PCDATA à la place du nom d'un élément fils (#PCDATA équivaut à chaîne de caractères).
Commentaires	<!-- commentaires -->	On peut ajouter des commentaires en respectant la même syntaxe qu'en XML.

Tableau 3 Aspects principaux de la syntaxe des DTD

6.RelaxNG

Spécifications : <http://www.oasis-open.org/committees/relax-ng/spec.html>

a) Relax NG : Syntaxe XML



Exemple : syntaxe XML

<http://www.oasis-open.org/committees/relax-ng/tutorial.html>

<http://xmlfr.org/oasis/committees/relax-ng/tutorial-20011203-fr.html>



Syntaxe : Element

```
<element name="">
  <element name="">
    ...
  ...
</element>
```



Syntaxe : Attributs

```
<element name="">
  <attribute name="">
    ...
  ...
</element>
```



Syntaxe : Nœuds texte

```
<element name=""><text/></element>
```



Syntaxe : Cardinalité

```
<element name="">
  <zeroOrMore>
    <element name="">
  /<zeroOrMore>
  <oneOrMore>
    <element name="">
  /<oneOrMore>
  ...
</element>
```



Syntaxe : Optionalité

```
<element name="">
  <optional>
    <element name="">
  /<optional>
  ...
</element>
```



Syntaxe : Alternative

```
<element name="">
  <choice>
    <element name="">
      <element name="">
        <element name="">
    /<choice>
```



Syntaxe : Énumération

```
<element name="">
```

```
<choice>
  <value></value>
  <value></value>
  <value></value>
</choice>
```

```
<attribute name="">
  <choice>
    <value></value>
    <value></value>
    <value></value>
  </choice>
```

Syntaxe : Pattern nommés

```
<define name="">
  ...
</define>
  ...
<ref name="" />
```

Syntaxe : Syntaxe générale

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start>
    <element name="">
      <ref name="" />
    </element>
  </start>
  <define name="">
  ...
  <define name="">
  ...
</grammar>
```

Syntaxe : Types de données

Spécifier datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"

```
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-
  datatypes">
  ...
<element name="">
  <data type="string">
</element>
```

Complément

Guidelines for using W3C XML Schema Datatypes with RELAX NG :
<http://relaxng.org/xsd.html>

Syntaxe : Namespace cible (unique pour tout le schéma)

```
<grammar ns="">
```

Syntaxe : Namespace cible (pour un élément particulier)

```
<element name="" ns="">
```

Complément: Tutoriel

<http://xmlfr.org/oasis/committees/relax-ng/tutorial-20011203-fr.html>

b) Relax NG : Syntaxe Compacte

Exemple

Le code RelaxNG version compacte suivant permet de spécifier qu'un élément "document" contient des éléments "page" qui contiennent du texte :

```
element document {element page {text}}*
```

Exemple : Syntaxe compacte

Tutoriel [OASIS, 2003]

Syntaxe : Element

```
element e1 { element e2 { ... }}
```

Syntaxe : Attributs

```
element e2 { attribute a1 { ... }}
```

Syntaxe : Nœuds texte

```
element e1 {text}
```

Syntaxe : Cardinalité

```
element e1 { element e2 {...}*}, element e3 {...}+, ... }
```

Syntaxe : Optionalité

```
element e1 { element e2 {...}?, attribut a1 {...}? }
```

Syntaxe : Alternative

```
element e1 { element e2 {...} | element e3 {...} }
```

Remarque : Composition des cardinalité et alternative

Utiliser les parenthèses

**Exemple : Ensemble non ordonné (set)**

```
element setElement { (element e2 {...} | element e3 {...})* }
```

**Exemple : Contenu mixte (mixed-content)**

```
element paraTag { (text | element inlineTag {text})* }
```

**Syntaxe : Énumération**

```
attribute a1 { "v1" | "v2" }
```

```
element e1 { "v1" | "v2" }
```

**Syntaxe : Pattern nommés**

```
Pattern1 = ...
...
element {Pattern1}
```

**Syntaxe : Syntaxe générale**

```
start = element e1 {...}
Pattern1 = ...
Pattern2 = ...
```

**Syntaxe : Types de données**

Spécifier datatypes xsd="http://www.w3.org/2001/XMLSchema-datatypes" avant le start

```
datatypes xsd="http://www.w3.org/2001/XMLSchema-datatypes"
start =
element e1 {xsd:decimal {maxInclusive="100"}}
```

Guidelines for using W3C XML Schema Datatypes with RELAX NG :
<http://relaxng.org/xsd.html>**Syntaxe : Namespace**

```
namespace rng = "http://relaxng.org/ns/structure/1.0"
default namespace = "http://foo.bar.org/xml-schemas/mtoSchema"
start = ...
```

c) Inclusion de schémas (Relax NG compact)



Syntaxe : Inclusion simple

```
include "fichier.rnc"
```



Syntaxe : Inclusion avec redéfinition

```
include "fichier.rnc" {
    ...
}
```



Exemple : Inclusion avec redéfinition du start

```
include "fichier.rnc" {
    start = ...
}
```

d) Types de données

Spécification des datatypes

<http://www.w3.org/TR/xmlschema-2/>



Définition : Primitive datatypes

- string
- boolean
- decimal
- float, double
- date, dateTime, duration, time, gYearMonth, gYear, gMonthDay, gDay, gMonth
- hexBinary, base64Binary
- anyURI
- QName, NOTATION

Facette

Paramètre de spécialisation des types primitifs.



Exemple : Type string

Facettes :

- length : longueur de la chaîne
- pattern : expression régulière permettant de valider la chaîne par rapport au patron (définition en intention)
- enumeration : liste de valeurs autorisées (définition en extension)
- ...



Définition : Built-in datatypes

Dérivés des types primitifs.

Par exemple :

- integer, dérivé de decimal

- normalizedString, dérivé de string
- language, dérivé de string
- ID, IDREF

Exemple : RelaxNG XML

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start>
    <element name="mail"
datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
      <data type="string">
        <param name="pattern">([^\s]+@[^\s]+\.)+([^\s]+)</param>
      </data>
    </element>
  </start>
</grammar>
```

Exemple : RelaxNG compact

```
datatypes xsd="http://www.w3.org/2001/XMLSchema-datatypes"
start = element age {xsd:decimal {maxInclusive="100"}}
```

Complément

<http://www.xml.dvint.com/docs/SchemaDataTypesQR-2.pdf>

7.Exemples

a) Exemple : Carnet d'adresse (XML, DTD, Relax NG)

Exemple : Instance XML

```
<?xml version="1.0" encoding="UTF-8"?><?oxygen
RNGSchema="adresse1.rnc" type="compact"?>
<addressBook>
  <card>
    <name>John Smith</name>
    <email>js@example.com</email>
  </card>
  <card>
    <name>Fred Bloggs</name>
    <email>fb@example.net</email>
  </card>
</addressBook>
```

Exemple : DTD

```
<!ELEMENT addressBook (card*)>
<!ELEMENT card (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

Exemple : RelaxNG (version 1 : simple)

```
element addressBook {
    element card {
        element name { text },
        element email { text }
    }+
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="addressBook"
xmlns="http://relaxng.org/ns/structure/1.0">
    <oneOrMore>
        <element name="card">
            <element name="name"><text/></element>
            <element name="email"><text/></element>
        </element>
    </oneOrMore>
</element>
```

Exemple : RelaxNG (version 2 : avec pattern)

```
namespace rng = "http://relaxng.org/ns/structure/1.0"
start = AddressBook
AddressBook = element addressBook { Card* }
Card = element card { Name, Email }
Name = element name { text }
Email = element email { text }
```

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <start>
        <element name="addressBook">
            <zeroOrMore>
                <element name="card"><ref
name="cardContent"/></element>
            </zeroOrMore>
        </element>
    </start>
    <define name="cardContent">
        <element name="name"><text/></element>
        <element name="email"><text/></element>
    </define>
</grammar>
```

Exemple : RelaxNG (version 3 : avec datatype)

```
namespace rng = "http://relaxng.org/ns/structure/1.0"
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"
start = AddressBook
AddressBook = element addressBook { Card* }
Card = element card { Name, Email }
Name = element name { xsd:string }
```

```
Email = element email { text }
```

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-
datatypes">
  <start>
    <element name="addressBook">
      <zeroOrMore>
        <element name="card"><ref
name="cardContent"/></element>
      </zeroOrMore>
    </element>
  </start>
  <define name="cardContent">
    <element name="name"><data type="string"/></element>
    <element name="email"><data type="anyURI"/></element>
  </define>
</grammar>
```



Complément

b) Exemple : Bulletins météo (XML, DTD, RNC, XSD)

8. Quelques bonnes questions sur XML

Réalisé à partir d'une interview de Michaël Tartar dans un *chat* de LeJournalDuNet.
http://www.journaldunet.com/chat/retrans/011206_tartar.shtml

Quel est l'avantage de ce langage par rapport au HTML ?

HTML est un langage de mise en forme. XML est beaucoup plus générique. Il permet de définir des langages alors que HTML définit un jeu de balises limitées et donc non-extensibles.

Pourquoi le XML n'a-t-il pas encore remplacé le HTML ?

Le XML n'a pas vocation à remplacer le HTML. XML permet aux architectes de sites de se concentrer sur l'information à fournir à l'internaute, et dans un deuxième temps de la mettre en forme pour des besoins spécifiques : PDF pour des documents à imprimer, HTML pour les navigateurs, ...

Quel est l'intérêt de l'utilisation d'XML sur le web en opposition aux fichiers CSS ?

CSS★ ne s'applique qu'à la définition de règles graphiques de restitution d'une information, alors que XML permet de structurer les informations. A ce titre, sur le Web, ils sont complémentaires.

Qui est le mieux placé pour monter en puissance sur XML ? Un webmaster ou un développeur ?

Les deux : Le développeur devra mettre à disposition du webmaster une information XML prête à être intégrée au site. Le webmaster sélectionnera les morceaux d'informations qu'il souhaite afficher en fonction du contexte.

Pour le stockage du contenu XML, dans quels cas faut-il s'appuyer sur une base de données et dans quels cas peut-on se satisfaire de conserver les fichiers XML à proprement dit ?

C'est le même problème avec n'importe quel type d'information. Qu'elle soit structurée en XML ne change rien : les bases de données apportent des services de confidentialité, d'intégrité, de montée en charge, ... que n'apportent pas les simples systèmes de fichiers.

Comment les outils de recherche vont-ils s'adapter à ce langage ?

XML permet de définir un vocabulaire et donc une sémantique particulière à chaque information. Les outils de recherche devront s'adapter et proposer des méthodes d'accès aux informations complémentaires à la recherche plain-texte (microformats, Web sémantique, ...).

Y a-t-il un lien entre XML et le Web sémantique ?

Le Web sémantique est un Web sur lequel les informations disponibles ont un sens pour les humains **et** pour les machines. XML a alors une place de choix pour sa mise en œuvre dans la mesure où le principe du balisage mêle justement information pour l'homme et pour la machine. Mais ce n'est pas le seul domaine de développement de XML, dont les usages dépassent largement le domaine du Web.

Les jeux de balises extensibles, c'est bien, mais trop de jeux différents ne risque-t-il pas de nuire à l'universalité que veut véhiculer XML ?

Cette question est centrale concernant XML. Il faut d'abord distinguer les langages XML **spécifiques** à chaque organisation, ceux que chacun se crée sur mesure pour ses propres besoins, des langages XML **standard**.

Pour les seconds, il faut bien comprendre que chaque vocabulaire sera spécialisé pour une application particulière. Par exemple SVG pour modéliser les documents vectoriels en 2D, XSL-FO pour modéliser des mises en pages à destination de l'impression papier, etc. Donc, sous réserve que deux langages distincts ne répondent pas aux mêmes besoins, ces langages resteront complémentaires.

Concernant les langages "maison" de chacun, il est évident qu'ils se recoupent les uns les autres et participent d'une certaine « **babélistation** ». Et c'est une des raisons pour lesquelles les solutions d'EAI_✉ sont amenées à se développer, ne serait-ce que pour assurer la transformation des informations à priori identiques. Sur le Web, des services de traduction assurent déjà la réconciliation sémantique de vocabulaires frères. Les technologies comme XSL-XSLT de transformation automatique permettent de concilier une approche spécifique à chaque contexte et des échanges dans via langages standards ou des traductions de dialecte à dialecte.

Via quels outils XML va-t-il se généraliser dans les échanges entre entreprises ?

Clairement, les outils d'EAI★ prendront une part importante, ne serait-ce que pour traiter les transformations des différents langages frères, ou encore assurer la gestion transactionnelle des échanges.

Quelles relations y a-t-il entre XML et EDI ?

L'EDI★ propose une infrastructure d'échange d'informations entre entreprises, ce qui comprend le format des données (par exemple EDIFACT) et le réseau à valeur ajoutée, qui assure la sécurité de la communication entre les entreprises. XML ne propose que la structuration. Un support transactionnel reste nécessaire, avec, par exemple, Internet pour le transport et ebXML★ pour la modélisation des processus d'échange d'informations.

Est-ce que l'évolution de l'Internet mobile va favoriser l'utilisation du XML dans le développement d'applications ?

Certainement : dès que l'information doit être restituée sur plusieurs supports, XML est un choix d'excellence.

Est-ce que les XML Schema sont d'un usage répandu ? Peut-on se concentrer dessus et oublier DTD, ou l'usage de DTD est-il encore trop répandu ?

Clairement, les DTD★ sont très répandues et sont amenées à le rester pour encore un bon moment, ne serait-ce que par leur adaptation à l'aspect documentaire de XML. Concernant l'aspect données, les limites des DTD sont telles que le passage par les schémas est nécessaire. Le fait que le W3C ait passé les schémas à l'état de recommandation permettra d'assurer un support de cette technique de définition des vocabulaires XML dans les différents produits du marché.

Si XML a l'air de résoudre autant de problèmes, pourquoi ne l'a-t-on pas inventé plus tôt ?

Pouvait-on l'inventer plus tôt ? Les premiers problèmes à résoudre étaient surtout liés à l'interopérabilité des systèmes. XML se place un cran au dessus en permettant de structurer l'information de manière standardisée, et surtout acceptée par tous.

XML est-il LA réponse au stockage structuré de n'importe quel document ?

Les documents graphiques binaires, tels que les photos ou les films, resteront dans des formats binaires, bien que certainement décrits par des fichiers XML (MPEG-7 pour la vidéo par exemple). Quant aux informations amenées à être réutilisées, recherchées, partagées, XML est effectivement **la** réponse.

D.Transformations XML : XPath et XSL-XSLT

1.Introduction à XSL-XSLT

a)Un langage pour publier les documents XML

- XML lorsqu'il est utilisé pour définir des formats documentaire métier est un format de représentation de l'information, et non un format de publication (de présentation) de cette information (*i.e.* un tel fichier XML n'est pas utilisable tel que par un lecteur).
- HTML ou PDF sont des exemples de formats de présentation de l'information.

- XML ne peut donc être utilisé pour des langages abstraits que si l'on est capable de transformer les documents sources en document publiés lisibles grâce à un format de présentation (HTML par exemple dans le cas de publication Web).

b) Définition de XSL-XSLT



Définition : XSL-XSLT

XSL-XSLT est une partie du standard W3C XSL qui a trait à la transformation des documents XML (l'autre partie étant XSL-FO).

XSL-XSLT est un langage de programmation déclaratif écrit en XML (un programme XSL-XSLT est un document XML).

- XSL-XSLT est langage de manipulation de document XML (fondé sur Xpath et sur le modèle arborescent de représentation des documents XML)
- XSL-XSLT est utilisé pour transformer un document XML source dans un autre format, typiquement HTML, mais aussi tout autre format codé sur des caractères dont la syntaxe est connue.
- XSL-XSLT est aussi utilisé pour faire des changements de schéma XML (export d'un XML vers un autre XML structuré différemment) par exemple pour échanger des données selon un standard.



Remarque : XSL-XSLT, XSL-FO, XSLT, XSL, FO

On parle souvent (par simplification) de XSL ou de XSLT pour désigner XSL-XSLT et de FO pour désigner XSL-FO.

XSL utilisé seul désigne donc par convention XSL-XST (et non XSL-FO).

c) Principe de XSL-XSLT

XSL-XSLT fonctionne selon le principe suivant :

1. Il prend en entrée un fichier XML bien formé
2. Il livre en sortie un fichier texte (XML, HTML ou texte sans balise)

Algorithme

L'algorithme général de XSL-XSLT est :

1. Il sélectionne (*match*) les éléments XML du fichier source.
2. Pour chaque élément reconnu il génère une sortie sur le fichier cible.

Notion de règle

Un programme XSL-XSLT est composé d'une succession de règles.

Chaque règle est indépendante des autres et à charge de sélectionner un élément dans la source et d'effectuer une écriture dans la cible.



Exemple : Application d'une règle XSL-XSLT

Sources :

```
<a>
  <b/><b/><c/>
</a>
```

Règle XSL-XSLT :

```
<xsl:template match="/a/b">
    BONJOUR
</xsl:template>
```

Résultat :

```
BONJOUR BONJOUR
```

2. Programmation XSL-XSLT

a) L'arbre du document XML

Il est possible de représenter un document XML sous forme d'arbre, tel que :

- L'arbre possède une racine / qui a comme fils l'élément racine
- l'élément racine est l'élément du document XML qui contient tous les autres
- chaque nœud a comme fils les éléments et le texte qu'il contient, ainsi que ses attributs.



Exemple : Fichier XML

```
<?xml version="1.0" encoding="UTF-8"?>
<document modele="ULCoursGeneral" code="BP-
Incendie3_S1_E2_UL1">
    <entete>
        <identification>
            <titre>L'assurance de la responsabilité de
voisinage</titre>
            <date>21/02/01</date>
            <auteur>AEA</auteur>
            <version>1.00</version>
        </identification>
    </entete>
    <corps>
        <contenu>
            <paragraphe>Cette garantie est appelée : recours des
voisins et des tiers.</paragraphe>
            <remarque>
                <paragraphe>L'image suivante <ressource URIsrc="img07.jpg"
titre="Recours des voisins et des tiers" type="image"/> montre
la garantie.</paragraphe>
            </remarque>
        </contenu>
    </corps>
</document>
```



Exemple : Arbre XML

```
/
|document
| @modele = "ULCoursGeneral"
| @code = "BP-Incendie3_S1_E2_UL1"
| entete
|     | identification
```

```

|titre
|text() = "L'assurance de ..."
|date
|text() = "21/02/01"
|auteur
|text() = "AEA"
|version
|text() = "1.00"
|corps
|contenu
|paragraphe
|text() = "Cette garantie ..."
|remarque
|paragraphe
|text() = "L'image suivante"
|ressource
|@URIsrc = "img07.jpg"
|@titre = "Recours des voisins..."
|@type = "image"
|text() = "montre la garantie."

```



Remarque : Ordre des nœuds

L'ensemble des nœuds de l'arbre d'un document est muni d'un ordre, qui est celui de l'ordre dans le document XML sérialisé.

b) Introduction à XPath



Définition : Expression XPath

XPath est un langage d'expressions permettant de pointer sur n'importe quel élément d'un arbre XML depuis n'importe quel autre élément de l'arbre.

- Une expression XPath peut-être **absolue** (sa résolution est **indépendante** d'un contexte ou nœud courant : elle commence dans ce cas par /).
- Une expression XPath peut-être **relative** (sa résolution est **dépendante** d'un contexte ou nœud courant : elle ne commence dans ce cas pas par /, elle peut commencer par ./ (syntaxe développée)).



Fondamental

Une expression XPath renvoie

- un *node-set*, ou ensemble de nœuds, c'est à dire un sous-arbre de l'arbre du document
- une chaîne de caractères
- un booléen
- un réel



Exemple : Exemples d'expressions XPath

```

/document/entete/identification/titre
/document/@modele
corps//contenu
contenu/*
contenu/remarque[1]
../paragraphe

```

@type

c) Syntaxe Xpath



Définition : Pas de localisation

Une expression XPath est composé de plusieurs **pas de localisation** successifs séparés par des /.

Un pas de localisation est caractérisé par :

- un axe,
- un test de nœud,
- un prédictat (éventuellement aucun ou plusieurs).



Syntaxe : Expression XPath

pas-de-localisation-1/.../pas-de-localisation-N



Syntaxe : Pas de localisation

axe:::test-de-nœud [prédictat]



Syntaxe : Parcours de l'arbre

- / : sélectionne la racine (expression absolue)
- . : sélectionne le nœud courant (expression relative)
- child:::x ou x ou ./x : sélectionne les éléments fils "x"
- child:::x/child:::y ou ./x/y ou x/y : sélectionne les éléments y fils de l'élément fils x (petits fils)
- attribute:::a ou ./@a ou @a : sélectionne l'attribut "a" du nœud courant
- child::* ou /* ou * : sélectionne tous les fils
- attribute::* ou .//*@ ou *@: sélectionne tous les attributs
- child:::text() ou ./text() ou text() : sélectionne les nœuds de type texte
- parent:::x ou ../x : sélectionne le père "x" (ancestor::x sélectionne les ancêtres "x")
- descendant:::x ou .//x : sélectionne tous les descendants "x" (enfants, petits enfants, etc.)
- preceding:::x, following:::x, preceding-sibling:::x, following-sibling:::x : sélectionne les nœuds précédents ou suivants dans le document (ordre), de même niveau uniquement ("fratrie") pour les "sibling".



Exemple : Prédictats

x[1], x[2], etc. : sélectionne le premier x, le second x, etc.

x[last()] : sélectionne le dernier x

x[@a='valeur'] : sélectionne les x tels que leur attribut a est égal à "valeur"

x[y=1] : sélectionne les x tels qu'ils ont un élément fils y égal à 1

x[@a='v1' and @b='v2'] : sélectionne tous les x tels que leurs attributs a et b sont respectivement égaux à v1 et v2

x[y or z] : sélectionne tous les x tels qu'ils possède un élément fils y ou z



Complément : current()

La fonction `current()` permet de renvoyer le nœud courant dans le contexte d'une exécution XSLT.

Cela permet de différencier :

- `elem1[@att1=@att2]` : Les elem1 qui ont deux attributs att1 et att2 égaux
- `et elem1[@att1=current()/@att2]` : Les elem1 qui ont un attribut att1 égal à l'attribut att2 du nœud courant

d) Syntaxe générale XSL-XSLT



Syntaxe : Structure générale d'un programme XSI-XSLT

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:output method="html" indent="yes" encoding="iso-8859-1"/>
  <xsl:template match="XPATH">
  ...
  </xsl:template>
  <xsl:template match="XPATH">
  ...
  </xsl:template>
  ...
</xsl:stylesheet>
```



Syntaxe : Règle (template)

```
<xsl:template match="XPATH">
  Instruction XSLT et/ou génération de texte sur la sortie
</xsl:template>
```

e) Principales instructions XSLT



Fondamental : Les deux instructions de base

- `<xsl:apply-templates select="XPATH"/>` : Relance les règles du programme sur le sous-arbre pointé par le `select` (fondement de la récursivité)
- `<xsl:value-of select="XPATH"/>` : Génère le **texte** contenu dans le nœud ou attribut pointé par le `select`
`{XPATH}` : Génère le texte contenu dans le nœud ou attribut pointé par le XPath entre accolades (alternative à `xsl:value-of` à utiliser dans la génération d'attributs exclusivement, par exemple : ``)

Autres instructions courantes

- `<xsl:copy-of select="XPATH"/>` : Génère le **sous-arbre** pointé par le `select`
- `<xsl:if test="XPATH">...</xsl:if>` : Ne s'exécute que si `test` est vrai
- `<xsl:for-each select="XPATH">...</xsl:for-each>` : Exécute pour

chaque sous-arbre renvoyé par le select



Complément : Extension XPath "document()"

document(chemin-accès) ou chemin d'accès permet d'accéder à un fichier sur le disque.

Par exemple :

- document("c:\monfichier.xml")//x : Tous les éléments x présents dans le fichier monfichier.xml.
- document(child::source)/* : La racine du document XML pointé par l'expression XPath child::source.



Complément : Déclaration explicite sur la cible

- <xsl:element name="">contenu</xsl:element>
- <xsl:attribute name="">valeur</xsl:attribute>
- <xsl:text>chaîne de caractère</xsl:text>



Complément : Références rapides

<https://developer.mozilla.org/fr/XSLT/Éléments>⁹

<http://www.daniel-lemire.com/inf6450/mod3/xsltintro.xhtml>

<http://personnel.univ-reunion.fr/fred/Enseignement/XML/intro-xslt.html>

f) Fonctionnement des programmes XSL-XSLT

La récursivité

```
<xsl:template match="XPATH-1">
  ...
  <xsl:apply-templates select="XPATH-2"/>
  ...
</xsl:template>
<xsl:template match="XPATH-2">
  ...
  <xsl:apply-templates select="XPATH-3"/>
  ...
</xsl:template>
<xsl:template match="XPATH-3">
  ...
  <xsl:value-of select="XPATH-4"/>
  ...
</xsl:template>
```

Règles par défaut

XSLT comporte trois règles par défaut qui sont appliquées quand aucune règle du programme n'est sélectionnée.

1. La première règle s'applique à la racine et à tous les éléments et déclenche un appel récursif sur tous les fils du nœud courant :

```
<xsl:template match="* | /"> <xsl:apply-templates/> </xsl:template>
```

9 - <https://developer.mozilla.org/fr/XSLT/%C3%89%C3%A9ments>

2. La deuxième règle s'applique aux nœuds texte et aux attributs et insère le résultat textuel de ces nœuds dans le document résultat :

```
<xsl:template match="text() | @*"> <xsl:value-of select=". "/>
</xsl:template>
```

3. La troisième règle s'applique aux commentaires et aux instructions de traitement et les ignore :

```
<xsl:template match="processing-instruction() | comment()" />
```



Exemple : Le programme XSLT minimal

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"/>
```



Attention

Il faut donc prendre garde aux règles par défaut, sources d'effets de bord.



Conseil

Veiller à bien traiter tous les cas.



Méthode

Court-circuiter les règles par défaut, en générant des messages d'erreurs sur la sortie.



Complément: Mode, priorité, règles nommées

- priority : permet de surcharger les règles de priorité par défaut
- mode : permet de définir plusieurs règles pour une même prémissse et de choisir explicitement celle que l'on veut au moment de l'appel
- name : permet de définir des *templates* nommés, qui sont appelés explicitement (comme des fonctions) par l'instruction `<call-template>` et non plus seulement par le moteur de récursivité.

Priorité entre les règles

- Si un attribut priority est défini sur la règle, la règle est prioritaire, l'attribut le plus élevé est prioritaire sur le moins élevé.
- Sinon, c'est celle dont le XPath est le plus **spécifique** d'abord (expression d'un axe, d'un prédicat)

3.Exemple : Un programme XSLT pour générer du HTML



Exemple : Fichier XML source

```
<?xml version="1.0" encoding="iso-8859-1"?>
<document titre="XSLT">
  <!--Première division-->
  <div titre="XSLT : Un besoin">
```

```

<paragraphe>XML est un format de
<important>représentation</important> de
l'information.</paragraphe>
<paragraphe>XML n'est pas un format de
présentation.</paragraphe>
</div>
<!--Seconde division-->
<div titre="XSLT : Un langage">
<paragraphe>XSLT est un langage de
<important>manipulation</important> de documents
XML.</paragraphe>
<paragraphe>XSLT est utilisé pour exporter une source XML sous
un autre format, par exemple HTML.</paragraphe>
</div>
</document>

```

Exemple : Fichier HTML cible souhaité

```

<HTML>
<!--Head-->
<HEAD>
<TITLE>XSLT</TITLE>
<META content="text/html" charset="iso-8859-1"/>
</HEAD>
<!--Body-->
<BODY>
<H1>XSLT : Un besoin</H1>
<P>XML est un format de <B>représentation</B> de
l'information.</P>
<P>XML n'est pas un format de présentation.</P>
<H1>XSLT : Un langage</H1>
<P>XSLT est un langage de <B>manipulation</B> de documents
XML.</P>
<P>XSLT est utilisé pour exporter une source XML sous un autre
format, par exemple HTML</P>
</BODY>
</HTML>

```

Exemple : Programme XSLT permettant la transformation

```

<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="html" indent="yes" encoding="iso-8859-
1"/>
<!--1ère règle-->
<xsl:template match="document">
  <HTML>
    <HEAD>
      <TITLE><xsl:value-of select="@titre"/></TITLE>
      <META content="text/html" charset="iso-8859-1"/>
    </HEAD>
    <BODY>
      <xsl:apply-templates/>
    </BODY>
  </HTML>
</xsl:template>

```

```

</HTML>
</xsl:template>
<!--2nde règle-->
<xsl:template match="div">
<H1><xsl:value-of select="@titre"/></H1>
<xsl:apply-templates/>
</xsl:template>
<!--3ème règle-->
<xsl:template match="paragraphe">
<P><xsl:apply-templates/></P>
</xsl:template>
<!--4ème règle-->
<xsl:template match="important">
<B><xsl:value-of select=".."/></B>
</xsl:template>
</xsl:stylesheet>

```

4.Approfondissement



Complément: Standards

<http://www.w3.org/TR/xslt>

<http://www.w3.org/TR/xpath/>



Complément: Cours détaillés



Complément: Références rapides

http://www.mulberrytech.com/quickref/XSLT_1quickref-v2.pdf

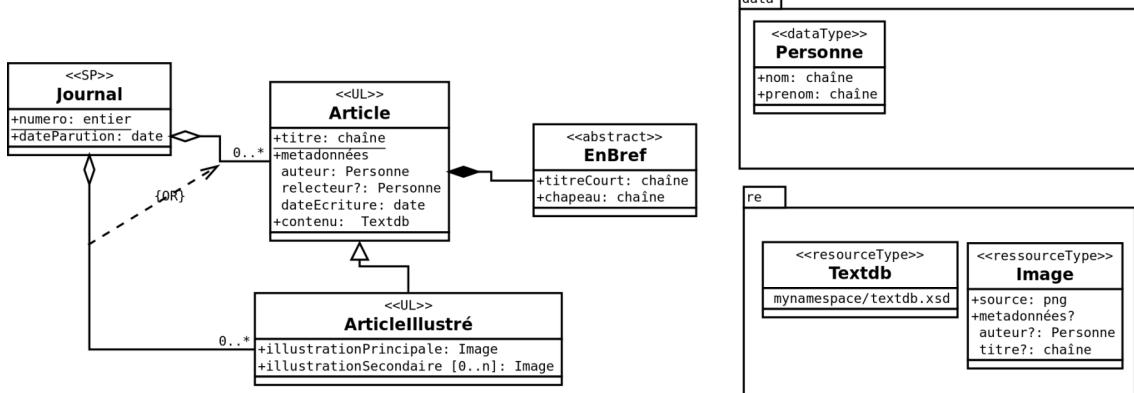
E.Modélisation conceptuelle de documents

1.Modélisation UML

a)Exemple



Exemple



Exemple de modélisation documentaire UML

b)Classes



Définition : Classe

Une classe est un type abstrait caractérisé par des propriétés (attributs et méthodes) communes à un ensemble d'objets et permettant de créer des instances de ces objets, ayant ces propriétés.



Syntaxe

Nom de la classe
Attributs
Méthodes()

Représentation UML d'une classe



Méthode

Créer une nouvelle classe pour chaque type d'entité documentaire identifiable et significatif.



Exemple

On créera une classe pour les parties principales d'un document : chapitres, section, etc.



Exemple : Contre-exemple

On ne créera pas de classe pour un paragraphe par exemple ou pour un titre, qui sont des éléments trop fins et qui pourront être représentés comme des attributs par exemple.

c) Associations



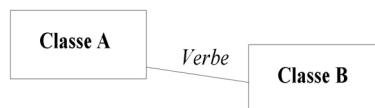
Définition : Association

Une association est une relation logique entre deux classes (association binaire) ou plus (association n-aire) qui définit un ensemble de liens entre les objets de ces classes.

Une association est nommée, généralement par un verbe. Une association peut avoir des propriétés (à l'instar d'une classe). Une association définit le nombre minimum et maximum d'instances autorisée dans la relation (on parle de cardinalité).



Syntaxe



Notation de l'association en UML



Remarque

Une association est généralement bidirectionnelle (c'est à dire qu'elle peut se lire dans les deux sens). Les associations qui ne respectent pas cette propriété sont dites unidirectionnelles ou à navigation restreinte.



Méthode

Créer des associations pour exprimer les liens entre les entités documentaires significatives identifiées.

Privilégier les associations de type Agrégation ou Composition lorsque ces liens sont de nature hiérarchique (ce qui est courant dans un cadre documentaire).



Définition : Association d'agrégation

On appelle agrégation une association particulière qui possède la propriété suivante : L'agrégation associe une classe composite (ou agrégat) et des classes parties, tel que tout objet partie appartient à au moins un objet composite (c'est une association binaire de cardinalité libre).

L'agrégation garde néanmoins les possibilités d'une association classique en terme de cardinalité, cycle de vie, etc. Une partie peut appartenir à plusieurs agrégats.



Définition : Association de composition

On appelle composition une association particulière qui possède les propriétés suivantes :

- La composition associe une classe composite et des classes parties, tel que tout objet partie appartient à un et un seul objet composite. C'est donc une association 1:N (voire 1:1).
- La composition n'est pas partageable, donc un objet partie ne peut appartenir qu'à un seul objet composite à la fois.
- Le cycle de vie des objets parties est lié à celui de l'objet composite, donc un objet partie disparaît quand l'objet composite auquel il est associé disparaît.

Une composition est donc une agrégation avec des contraintes supplémentaires (non partageabilité et cycle de vie lié).



Remarque : Non symétrie

Compositions et agrégations ne sont plus **symétriques**, une classe joue le rôle de conteneur pour les classes liées, elle prend donc un rôle particulier a priori.



Méthode

On utilisera les compositions lorsque les entités documentaires parties ne sont pas réutilisables et les agrégations lorsqu'elles sont réutilisables.



Exemple

L'agrégation ou la composition sont typiques des relations qui lient un document avec un chapitre, un chapitre avec un sous-chapitre, etc.

d) Héritage



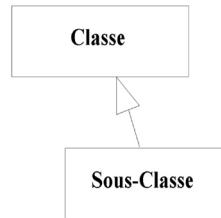
Définition : Héritage

L'héritage est association entre deux classes permettant d'exprimer que l'une est plus générale que l'autre. L'héritage implique une transmission automatique des propriétés (attributs et méthodes) d'une classe A à une classe A'.

Dire que A' hérite de A équivaut à dire que A' est une sous-classe de A. On peut également dire que A est une généralisation de A' et que A' est une spécialisation de A.



Syntaxe



Notation de l'héritage en UML



Méthode

Utiliser l'héritage de façon classique pour factoriser les propriétés des entités documentaire et les regrouper en grandes catégories.



Exemple

Les parties d'un document (chapitre, section, sous-section, ...) pourront hériter d'une même classe "partie" qui définira des propriétés générales (titre, ...).

e) Paquetages



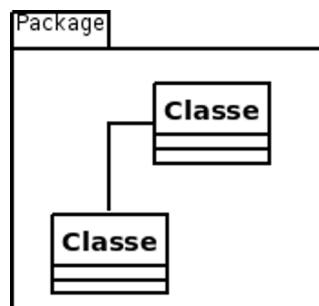
Définition : Package

Les paquetages (plus communément appelés *package*) sont des éléments servant à organiser un modèle.

Ils sont particulièrement utile dès que le modèle comporte de nombreuses classes et que celles-ci peuvent être triées selon plusieurs aspects structurants.



Syntaxe



Syntaxe des paquetages



Méthode

On utilisera les paquetages pour distinguer les grandes catégories de classes, comme par exemple pour les différents niveau de granularité du document.



Exemple

On pourra faire un paquetage pour les éléments structurants du document (types de parties) et un autre pour les éléments informatifs (types de contenus).

f) Stéréotype



Définition : Stéréotype UML

Un stéréotype UML est une syntaxe permettant d'ajouter de la sémantique à la modélisation des classes. Il permet de définir des **types de classe**, afin de regrouper conceptuellement un ensemble de classes (à l'instar d'une classe qui permet de regrouper conceptuellement un ensemble d'objets).

C'est une mécanique de métamodélisation : elle permet d'étendre le métamodèle UML, c'est à dire le modèle conceptuel du modèle conceptuel.



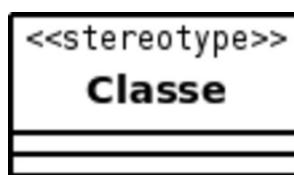
Définition : Méta-modèle

Un métamodèle est le modèle d'un modèle. Par exemple le métamodèle UML comprend les concepts de classe, attribut, association, cardinalité, composition, agrégation, contraintes, annotations, ... On mobilise ces concepts (on les instancie) pour exprimer un modèle particulier suivant le formalisme UML.

Les stéréotypes permettent donc d'ajouter au métamodèle UML standard, celui que tout le monde utilise, des concepts locaux pour enrichir le langage de modélisation que l'on utilise pour réaliser des modèles.



Syntaxe



Syntaxe d'un stéréotype en UML



Conseil : Stéréotypes spécifiques et stéréotypes standard

Un stéréotype spécifique enrichit le métamodèle UML, mais selon une sémantique qui est propre à celui qui l'a posé, non standard donc. La conséquence est que pour un tiers, l'interprétation du stéréotype n'est plus normalisée, et sera potentiellement plus facilement erronée. Il convient donc de ne pas abuser de cette mécanique.

Deux ou trois stéréotypes spécifiques, correctement définis, sont faciles à transmettre, plusieurs dizaines représenteraient un nouveau langage complet à apprendre pour le lecteur du modèle.

Il existe des stéréotypes fournis en standard par UML, ou communément utilisés par les modélisateurs. L'avantage est qu'ils seront compris plus largement, au même titre que le reste du métamodèle (ils ont une valeur de standard).



Exemple : Exemples de stéréotypes pour l'ingénierie documentaire

- <<scenario>> : Pour les classes d'organisation ou de scénarisation du contenu (sections, chapitres, parties, ...)
- <<contenu>> : Pour les classes d'agrégation de contenu (articles, exercices, ...)
- <<resource>> : Pour les classes de modélisation de formes sémiotique (texte, image, son, ...)
- <<data>> : Pour les classes de modélisation de données (vCard, ...)

2. Méta-modélisation : séparation scenario/contenu

Le principe de séparation entre scenario et contenu est une technique de modélisation documentaire destinée à favoriser la réutilisation de fragments documentaires.

Elle postule qu'un document peut être décomposé en unités de contenus portant l'information à transmettre d'une part, et en unités de scénario organisant ces unités de contenus en vue de la lecture d'autre part.

a) Réutilisation de fragments XML

Notion de réutilisation

Grâce au numérique, des mêmes **fragments** documentaire peuvent être réutilisés plusieurs fois au sein de plusieurs documents différents.

- Par copie (copier/coller)
- Par référence (pointeur)

Autonomie

Un fragment est réutilisable s'il est **autonome**, c'est à dire qu'il peut prendre du sens dans plusieurs contextes.

Il n'est pas (en général) indépendant, c'est à dire que le sens qu'il prend dépend de son contexte d'exploitation (il n'a pas un sens absolu).

Il n'est pas (en général) générique, c'est à dire qu'il ne peut pas prendre un sens dans tous les contextes imaginables, mais seulement dans un sous-ensemble.

Un fragment autonome peut être réutilisé, même s'il n'est pas indépendant, ni générique (Bachimont 98 [Bachimont98]).



Exemple

Par exemple pour le cours de bases de données NF17, la notion de requête permettant la "projection" suppose le SQL et l'algèbre relationnel comme contexte. Elle peut être utilisée pour différents usages : formations aux bases de données à l'université, formation MySQL dans une entreprise, exemple d'un cours théorique sur l'algèbre relationnel, etc.



Remarque

La réutilisation par copie ou par référence induisent des différences de **gestion** :

- Par copie, je réutilise un fragment que je fais miens, éventuellement que je modifie, je le dissocie de son auteur d'origine et de la vie documentaire qui lui est associé dans ce cas, je ne peux pas profiter des modifications ultérieures faites par ce dernier
- Par référence, je maintiens la paternité et l'authenticité du fragment d'origine, je pourrai intégrer les modifications ultérieures qui seront apportées, je ne peux pas modifier le fragment réutilisé dans le cadre spécifique d'un usage (le fragment est le même dans tous ses usages)

b) Document XML multi-fichiers



Méthode : Réutilisation et XML

La réutilisation par copie se fait en copiant un fragment XML depuis un fichier dans un autre fichier.

Mais pour la réutilisation par référence, il faut séparer le contenu au sein de plusieurs fichiers, pouvant alors se référencer par des mécanismes d'URI (la réutilisation étant induite du fait que plusieurs fichiers peuvent référencer le même fichier).



Définition : Internalisation

Fragment XML inclu dans le fichier l'utilisant. Le fragment n'est réutilisable que par copie.

```
<document>
  <fragment>
    ...
  </fragment>
</document>
```



Définition : Externalisation

Fragment XML référencé depuis le fichier l'utilisant ou le réutilisant. Le fragment est réutilisable par référence (et toujours pas copie).

```
<document>
  <fragment refUri="/chemin/fragment.xml"/>
</document>
```

```
<fragment>
  ...
</fragment>
```



Définition : Modèle mixte (user dependant)

Il est possible d'autoriser un modèle mixte permettant alternativement les deux logiques (internalisée ou externalisée), au choix du producteur du fichier XML (on pourra parler d'externalisation *user dependant*).



Exemple : Chapitres internalisés

```
<!--document.xml-->
<document>
  <chap>
    <texte>Ceci est mon premier chapitre</texte>
  </chap>
  <chap>
    <texte>Ceci est mon second chapitre</texte>
  </chap>
</document>
```



Exemple : Chapitres externalisés

```
<!--document.xml-->
<document>
  <chapitre refUri="chap01.xml"/>
  <chapitre refUri="chap02.xml"/>
</document>
```

```
<!--chap01.xml-->
<chap>
```

```
<texte>Ceci est mon premier chapitre</texte>
</chap>
```

```
<!--chap02.xml-->
<chap>
    <texte>Ceci est mon second chapitre</texte>
</chap>
```

Exemple : Chapitres "user dependant" (modèle mixte)

```
<!--document.xml-->
<document>
    <chapitre refUri="chap01.xml"/>
    <chap>
        <texte>Ceci est mon second chapitre</texte>
    </chap>
</document>
```

```
<!--chap01.xml-->
<chap>
    <texte>Ceci est mon premier chapitre</texte>
</chap>
```

Fondamental : Équivalence sémantique

L'internalisation ou l'externalisation ne sont que des représentations de gestion liées à la réutilisation, **elles n'influencent pas l'interprétation de la structure XML**.

Ainsi les trois exemples précédents sont strictement équivalents et devront être traités de façon identique, lors d'une publication typiquement (on peut considérer que tous les fragments externalisés sont réinternalisés avant traitement, par exemple).

Remarque : 1 document = N fichiers

L'externalisation rompt la correspondance "traditionnelle" 1 document = 1 fichier.

Notons néanmoins que dans le cas de XML, le référencement de fichiers binaires, aura déjà, le cas échéant, rompu cette unité.

Remarque

- Les schémas XML ne sont pas faits pour la validation d'un document XML multi-fichier, il faut donc une mécanique complémentaire pour valider la structure d'un tel document.
- L'externalisation multiplie les fichiers et complique la gestion informatique (gestion des renommages et déplacements des fragments par exemple) et humaine (complexité induite par la difficulté à appréhender un document dans sa totalité via ses fragments)
- Le modèle mixte peut compliquer le travail du rédacteur (choix de gestion supplémentaire, externaliser ou pas lors de la rédaction).



Complément : Réutilisation par référence sans fragmentation (pointeur intra-fichier)

Il est possible de définir des pointeurs pointant à l'intérieur d'un document XML, en utilisant des mécaniques telle que XPath par exemple. Ainsi la syntaxe `<fragment refUri="/chemin/fragment.xml" xPath="//element[1]" />` pointe le premier élément element du fichier fragment.xml.

Cette solution pose néanmoins de nombreux problèmes, qui la rende peu opérationnalisable :

- La logique de réutilisation n'est pas formalisée a priori, tout élément peut pointer tout élément de tout fragment (tandis que la fragmentation spécifie les points possibles de réutilisation, par construction)
- Les logiques de gestion sont beaucoup plus complexes (changement de structure du document XML avec pour conséquence l'invalidation d'un XPath par exemple)
- ...

c) Liens forts versus liens faibles



Définition : Lien fort

Un lien fort signifie qu'un fragment a nécessairement besoin du fragment lié pour avoir du sens.



Définition : Lien fort hiérarchique

Un lien fort est souvent un lien hiérarchique, il s'agira donc de **composition** ou **d'agrégation**.

- La composition exprime le fait que le lien est fort **dans les deux sens** : le composite a toujours besoin du composant et le composant n'existe que pour le composite.
- L'agrégation exprime le fait que le lien n'est fort que du composite vers le composant : le composite a toujours besoin du composant, mais le composant peut exister dans d'autres contextes.



Exemple

Les liens forts de type composition sont utilisés pour la structuration interne d'un contenu (parties, sous-parties).

Les liens forts de type agrégation sont utilisés pour les relations hiérarchiques entre scénarios et contenus (un scénario perd son sens sans contenu).



Définition

Un lien faible signifie que deux fragments peuvent être mis en relation, ils peuvent donc être lus à la suite, mais que la continuité de lecture n'est pas nécessaire, un fragment peut être interprété indépendamment de l'autre.

Le lien faible est généralement matérialisé par une **association classique**.



Exemple

Les liens de type "renvoi" sont des liens faibles : voir aussi, complément, note, articles connexes, etc.



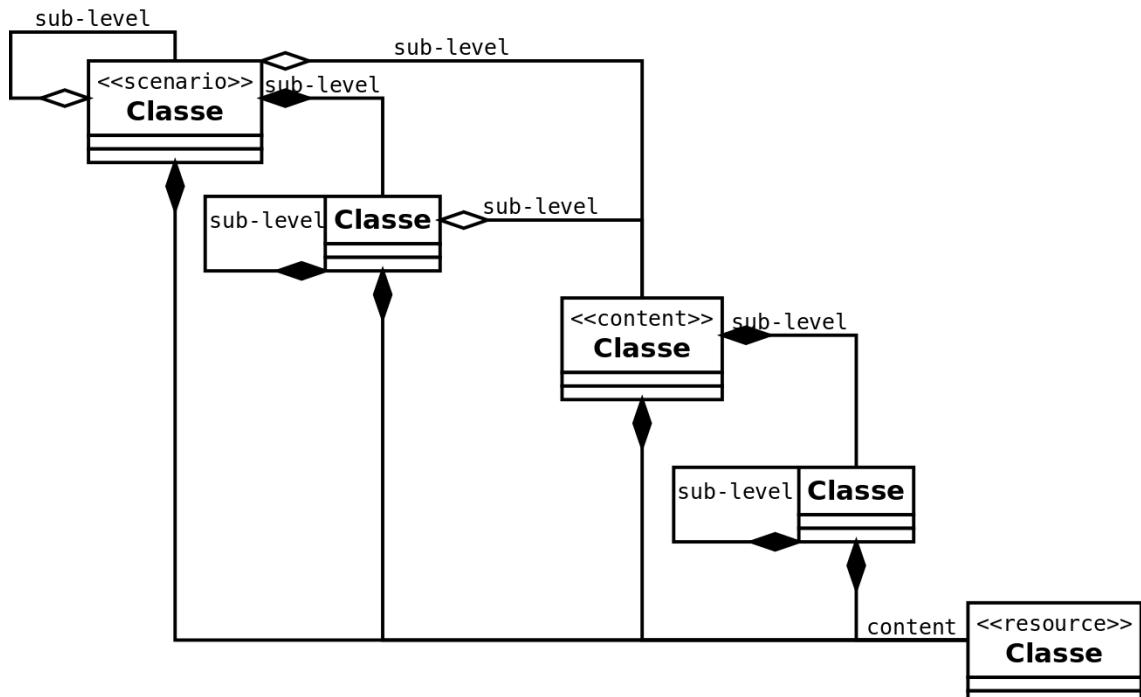
Conseil

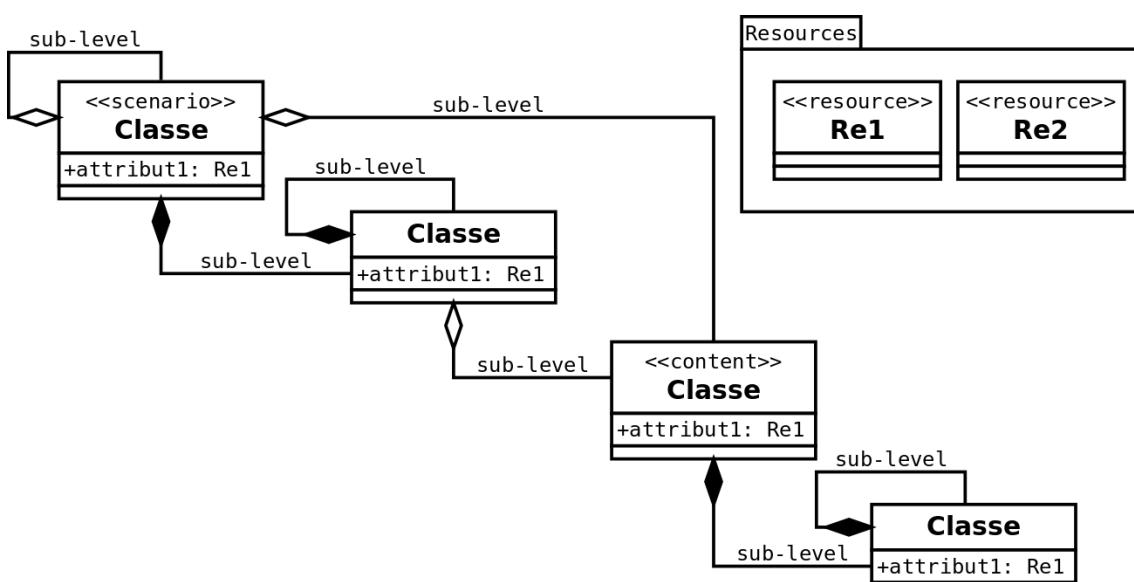
- Un lien fort est facteur de **cohérence** (il maintient le sens) et d'**adhérence** (il limite la réutilisation).
- Un lien faible est facteur de **désorientation** (il accroît le risque de perte de sens) et d'**indépendance** (il favorise la réutilisation).

3. Méta-modélisation : Motifs de conception (Design Pattern) et stéréotypes

Un *design pattern* est une façon récurrente de traiter un problème, c'est à dire une solution typique (principe issu de la modélisation Orientée Objet).

a) Scénario / Contenu / Ressource

**Syntaxe****Syntaxe**



Modélisation du motif scenario/content/resource (variante)



Remarque : Agrégation vers les scénarios et contenu

Les scénarios et contenus sont référencés par des agrégations, les scénarios et contenus étant réutilisables. Les autres associations d'organisation en sous-niveau sont des compositions.

Les liens vers les ressources peuvent être également des agrégations, si l'on souhaite la réutilisation d'images par exemple.



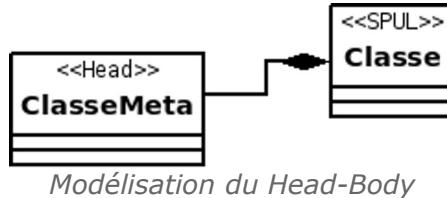
Remarque : Autres liens

Des liens faibles, de type association, peuvent exister de façon libre en surimpression de ces principes.

b)Entête / Corps



Syntaxe : Syntaxe



Document : head, body

**Remarque**

Différencier :

- les métadonnées liées au contenus (à gérer), par exemple l'auteur
- des métadonnées "système" (a priori pas géré dans le fichier XML, ni même par la même couche logicielle), par exemple la date de dernière modification

**Remarque : Standard**

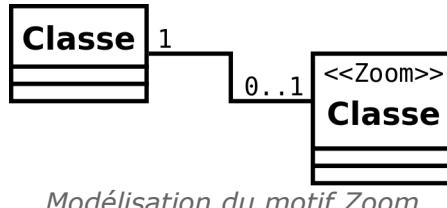
Standards (DC, LOM, etc.)

Classifications (Dewey)

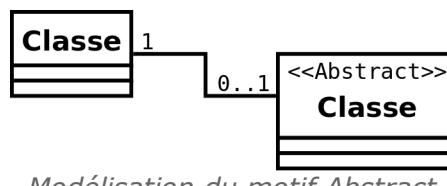
**Remarque : Classifications locales**

Thésaurus, ontologies

c) Contenu à profondeur variable (Abstract / Zoom)

**Syntaxe**

Modélisation du motif Zoom

**Syntaxe**

Modélisation du motif Abstract

**Exemple : Chapeau**

Dans un journal le "chapeau" est un résumé en une ou deux phrases de l'article.

**Attention**

L'écriture "à profondeur variable" est coûteuse : elle est difficile en soi et "frustrante" dans la mesure où le lecteur n'exploitera qu'une partie du contenu produit.

d) Renvoi (Confer)



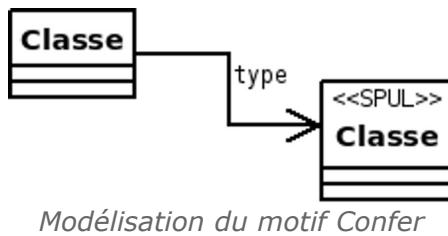
Définition

Lien faible entre une fragment et un autre de type SP ou UL.

Le renvoi doit être typé sémantiquement (pour donner du sens à la navigation proposée) :

- Renforcement ("Voir aussi")
- Approfondissement ("Pour aller plus loin")
- Antécédent ("Pour comprendre")
- etc.

Syntaxe



Modélisation du motif Confer

e) Références

Définition

Objets ayant une propriété d'indépendance (le sens n'est pas relatif au contexte).

Exemple

- Glossaire
- Bibliographie
- Acronymes
- etc.

f) Modèle encyclopédique

Définition

Un modèle encyclopédique correspond à un ensemble d'unités logiques de contenu (UL) reliées entre elles par des renvois (liens faibles).

Chaque unité peut être lue indépendamment des autres (lecture autonome) et indépendamment d'un parcours donné (lecture libre dirigée par le lecteur).

g) Modèle de parcours contrôlé

Définition

Un modèle de parcours contrôlé est un modèle fondé sur le numérique, qui propose une linéarisation dynamique du contenu. Le lecteur ne perçoit qu'une linéarité (un seul parcours de lecture), mais celle-ci est calculée en fonction de son activité et/ou de son profil (choix, réponses à des questions, profil d'utilisateur, comportement, etc.). Il existe donc plusieurs parcours possibles, chaque lecteur perçoit donc potentiellement une linéarité différente, mais c'est le programme informatique qui se charge de la linéarisation pour le lecteur (à différencier d'un modèle encyclopédique où c'est le lecteur seul qui fixe sa linéarité et d'un modèle livresque classique où c'est l'auteur qui fixe l'unique linéarité du document).



Exemple : En pédagogie

Les standards SCORM (Simple Sequencing) ou IMS (Learnind Design) proposent des branchements automatiques en fonction de réponses à des évaluations par exemple.



Exemple : Questionnaire guidé

Un questionnaire guidé, proposant de nouvelles questions en fonction des réponses précédentes est un modèle de parcours contrôlé (de questions en l'occurrence).



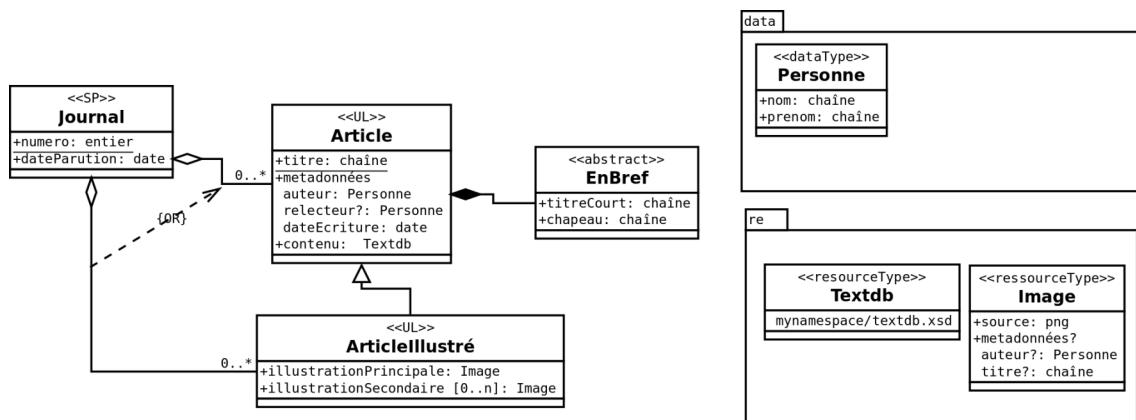
Attention

Un tel contenu est difficile à produire (il faut anticiper l'activité des lecteurs et multiplier les variantes de contenus) et à qualifier (la validité des parcours calculés restent difficile à valider).

4.Passage UML - Schéma

a) Exemple

Modèle UML d'un journal



Exemple de modélisation documentaire UML

Schémas Relax NG compact

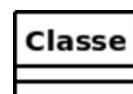
Télécharger le schéma sur le support en ligne.

Exemples d'instances

Télécharger des exemples d'instance sur le support en ligne.

b) Classe

Modélisation UML



Créer un schéma.



Syntaxe : Relax NG compacte

```
start = element classe {Classe}
Classe =
    ...
```

c)Attributs

Classe
+cle: text +attribut: text +attributMultiValué[1..n]: text +attributComposé sous-attribut1: text sous-attribut2: text

Syntaxe : Relax NG compacte

```
Classe =
    element attributCle {text},
    element attribut {text},
    element attributOptionnel {text}?,
    element attributMultivalué {text}+,
    element attributComposé {element sous-attribut1 {text},
    element sous-attribut2 {text} }
```

d)Références entre fichiers XML

Modalité

- URI (utilisation du nom du fichier)
- code unique (abstraction du stockage physique, mais gestion de code à implémenter)

Technique

Référence dans le fichier incluant vers le fichier inclus



Remarque : Liens vers des objets hétérogènes

Déconseillé sauf même catégorie (sinon le lien devient a-sémantique)

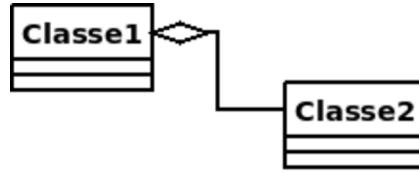


Remarque : Contrôle des fichiers liés

Ce n'est pas exprimable en schémas XML (contraintes inter-fichiers), le système doit implémenter ces contraintes complémentaires.

e)Agrégation

Modélisation UML



Créer un schéma pour chaque classe (externalisation, modèle mixte possible).



Syntaxe : Relax NG compacte

```

start = element classe1 {Classe1}
Classe1 =
  ...
  element refClasse2 {xsd:anyURI}
  ...
  
```

```

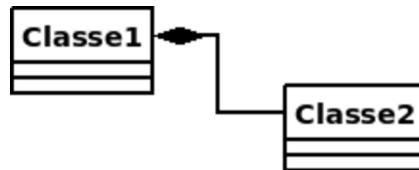
start = element classe2 {Classe2}
Classe2 =
  ...
  
```



Remarque : Cardinalités possibles

- 0..N:0..N
 - 0..N:1..N
 - 0..N:1..1
 - 0..N:0..1
- f)Composition

Modélisation UML



Créer un seul schéma pour les deux classes (internalisation).



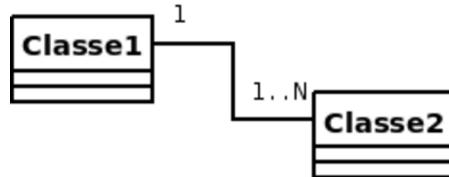
Syntaxe : Relax NG compacte

```

start = element classe1 {Classe1}
Classe1 =
  ...
  element classe2 {Classe2}*
  ...
  Classe2 =
  ...
  
```

g)Association

Modélisation UML



Créer un schéma pour chaque classe (externalisation).



Syntaxe : Relax NG compacte

```

start = element classe1 {Classe1}
Classe1 =
  ...
  element refClasse2 {xsd:anyURI}+
  ...
  
```

```

start = element classe2 {Classe2}
Classe2 =
  ...
  
```



Rappel: Cardinalités

La cardinalité côté fichier inclus est exprimée par la cardinalité de la clause "element".

La cardinalité côté fichier incluant est par défaut de 0..N. Tout autre cardinalité demande un contrôle du système hors schéma XML (contrôle inter fichier).



Attention : Ordre

UML ne permet pas d'ordonner les attributs et les associations, conserver l'ordre logique ou probable lors de la traduction en schéma (qui instaure un ordre).

h) Annotation des associations

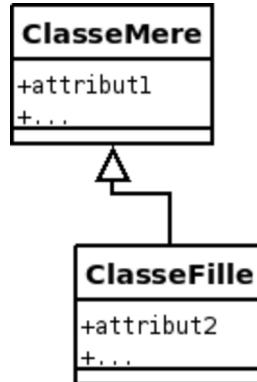
OR (indépendance)

AND (dépendance)

XOR (exclusion mutuelle)

i) Héritage

Modélisation UML



Créer un schéma pour chaque classe.

Rappeler le schéma de la classe mère dans la classe fille.



Syntaxe : Relax NG compacte

```

start = element classeMere {ClasseMere}
ClasseMere =
  element attribut1 {text},
  ...

include "classeMere.rnc" {
  start = element classeFille {ClasseFille}
}
ClasseFille = ClasseMere,
  element attribut2 {text},
  ...
  
```



Remarque : Classe abstraite

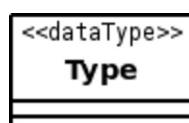
Si la classe est abstraite, ne pas définir de `start`.

j) Package

Utiliser un namespace différent pour chaque package.

k) Type de données

Modélisation UML



Créer un schéma.



Syntaxe : Relax NG compacte

```

Type =
  ...
  
```



Remarque : Définitions sous forme de schéma

Les types de données complexe peuvent être directement spécifiés sous forme de schéma, sans passer par la définition UML.

C'est également le cas lorsque l'on réutilise un type standard.



Exemple : Type standard



Méthode : Utilisation dans les diagramme UML

Types personnels et standard complexe commencent par une majuscule

Types de base (entier, chaîne, etc.) par une minuscule

I) Type de ressource



Exemple

- texte
- image
- schéma
- tableaux
- vidéo
- etc.



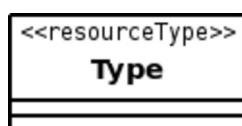
Méthode

Associer un format à chaque ressource.

Privilégier les formats logiques lorsque c'est possible (texte), standard sinon (format graphiques).

Attention à l'existant : récupération automatique ?, manuelle ? Changement de format ? Transformation systématique ? ; automatique ? manuelle ?

Modélisation UML



Créer un schéma.



Syntaxe : Relax NG compacte

```

Ressource =
...
  
```

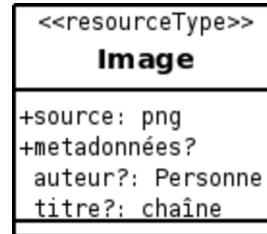


Remarque

Les ressources externalisées sont typées en Relax NG sous la forme d'URI.



Exemple : Type standard



Remarque : Définitions sous forme de schéma

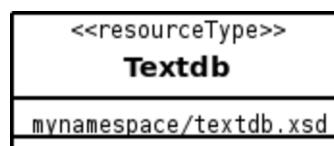
Les types de ressources complexe peuvent être directement spécifiés sous forme de schéma, sans passer par la définition UML.

C'est également le cas lorsque l'on réutilise un type standard.



Exemple : Exemple de ressource Texte standard

Réutiliser un sous ensemble de la docBook.



5.Bonnes pratiques

a)Modèles souples versus modèles contraints

- Modèle contraint simple = formulaire (facile, ex : CV, Webradio)
- Modèle couple simple = Wiki-like (facile, ex : OpaleStarter)
- Modèle constraint complexe = Rédaction contrôlée (rédaction difficile, résultat très qualitatif pour les auteurs, ex : Linio)
- Modèle couple complexe = Bureautique (puissant, mais difficile car ressemble à la bureautique sans en être, ex : Opale)

b)Format des ressources



Méthode

- Utiliser des formats logiques plutôt que physique lorsque c'est possible (texte, tableaux logiques, mathématiques, etc.)
- Utiliser des formats standards sinon (OD, MPEG, etc.)
- Utiliser des formats métiers ou déjà connus sinon (formats déjà utilisés dans le contexte)



Exemple

- Texte (XML sous ensemble de la DocBook)
- Tableau graphique (ODS, CALS, HTML)
- Tableau logique (XML sur mesure de type colonne / ligne)
- Photo (JPG)

- Dessins bitmap (PNG)
- Dessins vectoriels (ODG, SVG)
- Audio (MP3, OGG)
- Vidéo (MPEG-4, FLV, OGG)
- Animations (SWF, SMIL, HTML)
- Interactions (ZIP)
- Documents "publiés" (PDF)



Conseil : Alternatives et accessibilité

Prévoir dans le modèle des "alternatives" permettant d'associer à une ressource qui ne pourra pas être systématiquement publier des ressource, qui bien que moins pertinentes, permettront une diffusion plus large de l'information.

Les deux usages caractéristiques de l'alternative sont le **multi-supports** (une animation ne peut être publiée sur papier, il faut prévoir par exemple une ou plusieurs image fixes représentative de l'animation) et l'**accessibilité** (par exemple un moteur de lecture automatique pour non-voyant ne pourra pas décrire une image, il faut donc associer un texte à cette dernière).

c) Utilisation de standard



Conseil

Lorsque c'est pertinent il convient de s'inspirer ou de réutiliser des standards existants ou des sous-parties de ceux-ci.



Exemple

Par exemple des sous-ensemble de MPEG-7 peuvent être mobilisés pour la description multimédia, ou de DocBook pour le texte.

d) Récursivité versus contrôle de profondeur



Définition

Un modèle récursif est un modèle qui permet un arbre d'éléments de profondeur infinie, à l'inverse d'un modèle contrôlant la profondeur.



Conseil

Le modèle récursif est plus souple à l'écriture et facilite donc la phase d'écriture, l'auteur retrouve des procédures récurrentes. Par contre il est plus difficile à gérer lors de la publication, en effet l'on est alors tenu de prendre en compte des cas dont la probabilité est très faible et de décider d'un cas limite.



Exemple

Pour une structure de liste, elle peut être contrôlé et limitée à deux niveau (liste1 => item | liste2 ; liste2 => item) ou récursive (liste => item | liste).

e) Récupération de l'existant

Types de situations

- Production-Récupération : Désigne l'usage de la chaîne éditoriale pour produire des contenus structurés en XML à partir de contenus existants sous des formats quelconques.
- Production de Nouveaux Contenus : Désigne l'usage de la chaîne éditoriale

pour produire des contenus structurés en XML qui n'existent pas encore. Une PNC peut-être rapportée à une PR avec une phase de production par l'auteur en plus.

- Production-Amélioration Continue : Désigne l'usage de la chaîne éditoriale pour améliorer des contenus déjà récupérés ou produits selon les modèles documentaires et pédagogiques

Organisation de la production

Organiser la production :

- pour permettre la mise en œuvre de la production
- identifier les métiers et acteurs nécessaires à la production



Méthode

- Dimensionner les besoins en moyens humains
- Obtenir l'implication de l'institution
- Organiser l'implication des auteurs
- Identifier les acteurs de la chaîne et préparer les fiches de poste
- Identifier les risques et les points durs
- Préparer les outils de gestion de la chaîne au quotidien
- Prévoir les formations de l'ensemble des acteurs

F. Repères méthodologiques pour un projet documentaire

1. Démarche d'AMO

a) Préambule



Définition : Maîtrise d'œuvre (MOE)

Celui qui fait le travail, mais qui n'en possède pas le résultat ni n'en décide la finalité ou la nature.



Définition : Maîtrise d'ouvrage (MOA)

Celui à qui est destiné l'ouvrage réalisé. Il maîtrise le résultat et sa finalité, mais non le processus pour y parvenir.



Définition : Assistance à maîtrise d'ouvrage (AMO)

Celui qui fait le lien entre la MOE et la MOA; il analyse le métier de la MOA pour reformuler le problème de manière pertinente pour la MOE. Il fait le lien entre ce qu'il faut faire et ce qu'on peut faire. Il assure le montage du projet, son suivi et sa recette.

b) Analyse de la demande



Définition : Analyse de la demande

Il s'agit de reformuler et préciser la demande client, en la situant dans son contexte, et en dégageant la valeur visée par le client.

L'enjeu est surtout de bien identifier **qui** est le client et **pourquoi** il commande le projet.



Attention : Difficulté

La demande est souvent mal formulée, et l'objet réel de la demande est rarement celui mis en avant par le client.

Par exemple : demander des supports de cours en ligne, alors que la demande porte sur l'accès aux contenus et le suivi pédagogique après les cours.



Méthode : Approche

- Cerner le contexte : les acteurs, les objets, les processus et les échanges.
- Situer la demande, et son insertion dans le contexte.
- Reformuler la demande en précisant l'objectif poursuivi par le client.



Méthode : Plan type

- présentation générale du client,
 - présentation du secteur à l'origine de la demande (organisation et activité),
 - motivation du problème,
 - formulation de l'objectif et du besoin,
 - formulation des critères de succès (indicateurs) .
- c) Expression de besoin



Définition : Expression de besoin

Mise en exergue, dans le détail, de ce que les utilisateurs veulent faire avec le système.

L'enjeu est ici de comprendre de **quoi** on parle.



Attention : Difficulté

- ne pas réduire la demande à ce que les utilisateurs et outils savent faire
- ne pas traiter les problèmes posés par les outils mais non pertinents pour le besoin
- ne pas trop focaliser sur les problèmes des utilisateurs sans rapport avec le projet



Méthode : Scénarios d'usage

Il s'agit de raconter en petites histoires de 10 lignes environ les principaux scénarios d'utilisation illustrant l'usage du système visé.

Ces scénarios se construisent souvent autour des rôles des utilisateurs : administrateur, usager standard, usager expert, etc.

Ces histoires permettent d'avoir un fil conducteur mental pour visualiser les fonctions pertinentes. Elles doivent être rédigées (en français !) et ne pas se réduire à un simple schéma. C'est le fait d'écrire qui permet de penser au scénario dans son déroulement temporel, de penser la cohérence des actions et processus dans leur agencement et succession.

Si un schéma appuie l'histoire, il ne doit pas la remplacer. Un scénario n'est pas un use case, au sens UML, justement parce que c'est rédigé et non schématique.



Exemple : Transcription d'une vidéo

 Valérie est en charge de la transcription et de la synchronisation de la collection des 8 entretiens de metteurs en scène. Cette opération préalable à la mise en ligne d'un entretien est longue et fastidieuse. Elle compte ainsi profiter des nouveaux outils pour accélérer cette étape de traitement. Elle est averti de la mise à disposition de l'entretien de Roger Planchon pour l'étape de transcription/synchronisation. Elle consulte alors le document et peut commencer à taper au kilomètre le texte brut. Valérie peut ajouter quelques éléments de typage (important) dans le corps de son texte. A chaque frappe, le player vidéo se met en pause et reprend, une fois la frappe terminée, quelques secondes auparavant. Cette méthode de saisie est très appréciée par Valérie qui peut ainsi écouter un fragment puis taper le texte sans avoir à mettre constamment en play/pause le curseur vidéo, ni à revenir avec la barre de navigation.

(par Ludovic Gaillard, INA, projet C2M,
<http://www.utc.fr/ics/c2m>) 



Méthode : Glossaire

Mettre en place un glossaire avec les concepts clés du projet. Cela permet de poser un langage commun et vérifier que l'on se comprend bien, si l'on partage effectivement une définition et pas seulement des mots.

d) Cahier des charges fonctionnel



Définition : CDCF

Un système technique est envisagé pour répondre à la demande et traiter les besoins associés. Le CDCF a pour but de spécifier les fonctions et services que ce système remplira.

Il n'assume pas à lui tout seul le besoin : l'organisation et le management environnants font également partie de la réponse au besoin.



Attention : Difficulté

- bien traduire les besoins
- bien se positionner par rapport à la technique : ne pas trop orienter les choix techniques, tout en restant réaliste par rapport au contexte de réalisation (déjà là technologique, budgets, compétences, règles DSI, ...)



Conseil : Conseil

Être systématique



Méthode : Analyse fonctionnelle

A partir des scénarios d'usage racontant les différentes utilisations visées, situées et scénarisées dans leur contexte de fonctionnement, en situation « réelle » :

- **Analyser les situations** : définir les différentes situations d'usage en précisant :
 - les acteurs interagissant avec le système ;
 - Les objets et systèmes de l'environnement interagissant avec le système ;
 - Les objets et processus produits par le système ;
- **Analyser les fonctions** : lister chaque fonction, pour chaque situation, selon le format suivant :
 - le système permet à <acteur> de <action> sur <objet>
 - on précise ensuite la fonction lorsque c'est nécessaire : exemple, reformulation, spectre de valeurs possibles, ...

Exemple : Fonctions d'indexation manuelle



1. Le système permet aux auteurs d'ajouter des métadonnées documentaires aux items
 - Typiquement : auteur, description, etc.
2. Le système permet aux bibliothécaires d'ajouter des métadonnées documentaires complémentaires, ou de modifier les métadonnées fournies par l'auteur
 - après la publication du document-dossier (rôle de documentaliste)
 - sans modifier le contenu du document-dossier (accès en écriture aux métadonnées seulement, ou à un item de métadonnées spécifique)
3. Le système propose des aides à la saisie des métadonnées
 - profil de remplissage
 - valeurs par défaut
 - extraction du système (date, user courant, etc.)

(par Stéphane Crozat, UTC, projet C2M,
<http://www.utc.fr/ics/c2m>)



Méthode : Évaluation des fonctions

- **Le but de la fonction** : À quoi est-elle utile ? A quoi sert-elle ? Quel service permet-elle de remplir ?
- **La nécessité de la fonction** : Pourquoi faut-il assurer cette fonction ? Que se passerait-il si cette fonction n'était pas remplie ? Le service serait-il assuré par ailleurs ? Ou il ne serait pas réclamé ?

Alors que le but répond à la question « A quoi ça sert ? », la nécessité répond à la question « Pourquoi il faut le faire ? ».



Méthode : Analyse fonctionnelle détaillée : Caractérisation des fonctions

- Les termes utilisés : quel acteur, quelle action, quel objet.
- Les principes de la fonction : on liste les différentes manières de réaliser techniquement la fonction.



Complément : Paradigme technique

Pour mener à bien l'analyse fonctionnelle, il faut avoir en tête une architecture générale permettant d'optimiser la valeur recherchée à travers le service.

Les **paradigmes techniques** déterminent un type de valeur, car leur élaboration s'est effectuée en fonction de certaines options sur les traitements à effectuer et les bénéfices attendus. Les exemples de paradigmes sont :

- Application (JAVA) ou Contenu (XML) ;
- Serveur centralisé ou Contenus distribués ;
- Traitement chez le client ou traitement centralisé
- etc.

Le problème traité introduit des contraintes ; on les formule en qualifiant en menace ou opportunité les changements qui peuvent survenir dans la solution technique.

Par exemple, l'éditeur de la base de donnée fait faillite :

- c'est très grave, car nos applications sont directement interfacées avec la base, et il faut tout refaire ;
- c'est très bien, car on attaquait la base à travers des couches normalisées (SQL) : on peut changer de base en en prenant une meilleure et garder tout le reste.

L'objectif est de choisir un paradigme technique qui maximise les changements en opportunités et non en menaces.

e) Cahier des charges technique



Définition : Principe

Le système technique se construit à partir des outils et méthodes de l'état de l'art. Le CDCT propose des solutions qu'il justifie d'un point de vue de faisabilité technique, rationalité économique, pertinence organisationnelle et ergonomique.

Le CDCT permet de préciser l'architecture choisie et les principaux outils qui seront utilisés.



Attention : Contenu

Le CDCT donne les spécifications générales, non les spécifications détaillées qui sont en général du ressort du programmeur.



Méthode : Méthode

Les méthodes proposées par le génie logiciel. On pourra s'appuyer sur UML, les modèles conceptuels de données, etc.



Conseil

Attention de ne pas se perdre dans les détails : il ne s'agit pas de faire le travail du prestataire ou de la MOE, mais de l'encadrer.

f) Étude industrielle



Définition : Étude industrielle

Associée au cahier des charges techniques, on aura une analyse des offres industrielles, de leur segmentation et de leur complémentarité pour concevoir une architecture générale.

L'objectif est de repérer les principaux acteurs du domaine : que ce soit pour les contacter comme prestataires ou pour savoir ce que les prestataires apporteront dans leurs réponses.

L'étude doit par conséquent :

- délimiter les différents segments et périmètres du marché ;
- déterminer ceux sur lesquels il faut se concentrer ;
- identifier les principales offres du marché : cible fonctionnelle et économique, coûts, principes techniques, jugement.

2. Critique des méthodes "traditionnelles"

a) Rappel des méthodes traditionnelles

Définition : Méthode en cascade

Méthode issue des projets "brick & mortar" : Il faut **planifier** puis réaliser chaque phase l'une après l'autre.

1. Recueil de besoins
2. Analyse fonctionnelle
3. Conception technique
4. Développement
5. Tests
6. Intégration

Définition : Méthode en V et en Spirale

Méthodes "standard" de l'industrie du développement logiciel, inspirées de la méthode en cascade, avec des anticipations et des boucles de rétro-actions.

b) Le problème de la planification

Fondamental : Les histoires de projet finissent mal, en général...

Ces méthodes traditionnelles sont fondées sur la **planification** de l'ensemble du projet :

1. On dit ce que l'on veut
2. On fait ce qu'on a dit

Mais en général :

- On ne sait ce qu'on veut !
- On n'arrive pas à faire ce qu'on a dit !



Définition : « La certitude de l'incertitude »



- Nous ne savons pas précisément ce que nous allons développer [...] le client ne sait pas toujours ce qu'il veut
- Nous ne connaissons pas toujours les individus qui seront amenés à collaborer dans l'équipe [...]
- Nous n'avons, souvent, qu'une idée de la solution [...]
- Nous ne maîtrisons pas toujours les technologies [...]

(Gestion de projet : Vers les méthodes agiles [Messenger09], p12)



Attention : L'assurance "Doc"



Comme l'on sait que ça marche mal (en général), le client et le fournisseur passent beaucoup de temps à planifier et à formaliser.

Mais l'objectif n'est plus la gestion du projet, mais la **gestion des responsabilités** en face des problèmes qui vont inévitablement surgir (dépassement de planning, réduction de spectre fonctionnel, incohérence des demandes, etc.).

Une part importante de l'énergie est alors passée à rédiger des documents, que personne ne lit, et dont la seule fonction est de servir de garantie en cas de conflit.

c) Autres problèmes des méthodes traditionnelles

- Effet tunnel : Utilisateurs et concepteurs se voient au début et à la fin, ils ne **partagent** pas le projet
- Mauvaise communication
- Levée tardive des problèmes
- ...

(Gestion de projet : Vers les méthodes agiles [Messenger09], p40-41)

Tout ce que vous écrirez pourra être retenu contre vous

Le formalisme à outrance tue l'information dont a réellement besoin le projet.

L'objectif n'est plus de dire ce qui est bon pour le projet, mais ce qui est bon pour sa défense.



Attention : Coût du changement d'avis

Plus on passe de temps à concevoir une solution a priori, plus il sera coûteux de changer d'avis après, même si la **solution n'est pas bonne** !

- Coût effectif de la conception passée
- Coût psychologique (théorie de l'engagement)

Parler du loup...

L'on passe la plupart du temps à parler de quelque chose qui n'existe pas (le futur système), que personne n'a vu, voire que personne ne verra jamais !

3. Introduction aux méthodes agiles

a) Les méthodes agiles



Fondamental : De la planification au pilotage

Le fondement des méthodes agiles est qu'il n'est pas possible de tout prévoir correctement, et qu'il vaut mieux planifier au niveau macro uniquement et **piloter** au niveau micro.



Définition : Méthode agile

Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif, avec juste ce qu'il faut de formalisme. (Gestion de projet : Vers les méthodes agiles [Messager09], p42)



Fondamental : Manifeste pour le développement logiciel agile



- Les individus et leurs interactions avant les processus et les outils.
- Des fonctionnalités opérationnelles avant la documentation.
- Collaboration avec le client plutôt que contractualisation des relations.
- Acceptation du changement plutôt que conformité aux plans

(Gestion de projet : Vers les méthodes agiles [Messager09], p49)



<http://www.agilemanifesto.org/>



Exemple : Exemple de méthodes agiles

- Rapid Application Development (RAD)
- Adaptative Software Development (ASD)
- Crystal
- Scrum
- eXtreme Programming (XP)

b) Organisations agiles



Définition : Caractéristiques

- Communication informelle
- Pilotée par les besoins clients

- Réactive

Travail en équipe souple

- Organisation horizontale : co-décisions, le chef de projet tranche quand c'est nécessaire, il ne commande pas
- Binôme : les travaux en binômes sont favorisés (lorsque les tailles d'équipe le permettent) pour éviter les compétences uniques risquées dans tout projet, a fortiori peu formalisé
- Partage d'information et du travail : serveurs de code source, de documentation, forums de discussion, de veille, ...
- Polyvalence : les membres de l'équipe sont intéressés largement aux travaux (plusieurs modules, participation à la gestion, aux relations client, ...)

La méthode de Monsieur Jourdain, comme souvent...

Une méthode agile est avant tout une méthode de bons sens, dans laquelle on travaille en bonne intelligence.

Si on est réactif, prêt à changer d'avis quand cela est sensé, que l'on suit de près son projet, qu'on ne perd pas de temps avec le superflu, etc... on fait de l'agile sans le savoir !

c) Outils agiles



Définition : Timeboxing

Principe du *timeboxing* : Les itérations ont une durée fixe, sont à court terme (< 1 mois), on évalue chaque itération à la fin et on réajuste le plan de travail en fonction des résultats.



Définition : Product backlog

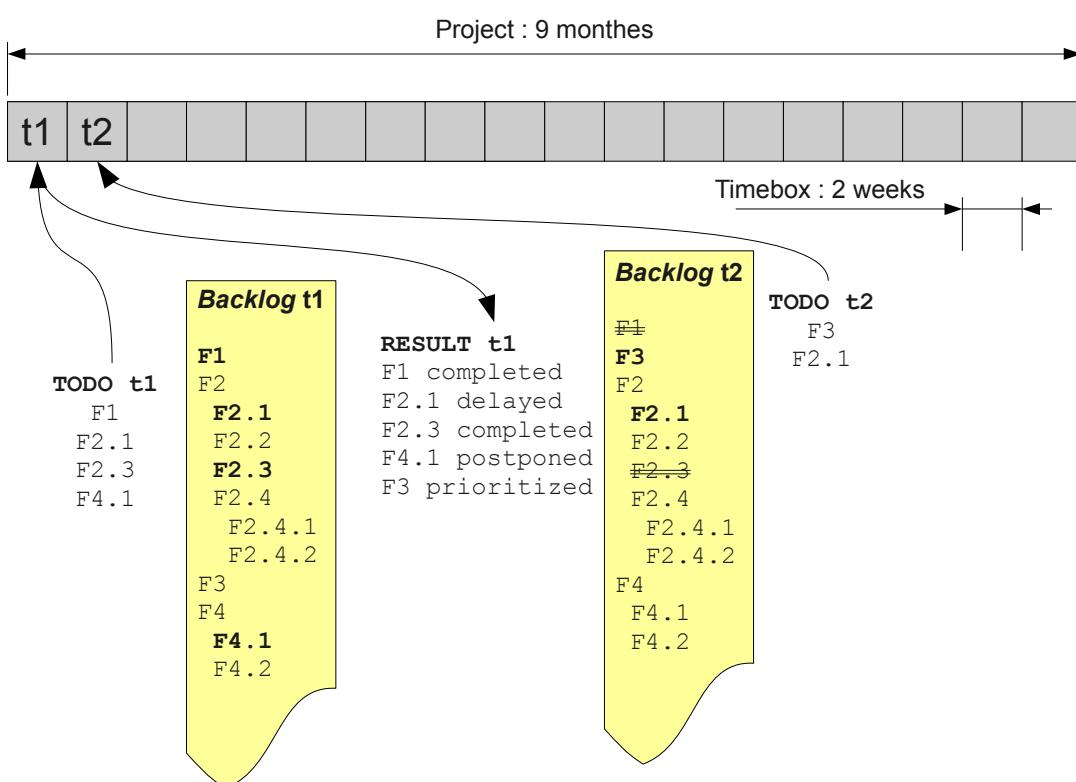
On établit en début de projet la liste des besoins exprimés, par exemple sous forme fonctionnelle. Ces fonctions sont hiérarchisées par facteur d'importance en terme d'usage et par délais, difficulté et/ou coût de réalisation. C'est le *product backlog*.

Avant chaque *timebox* le *product backlog* sert d'outil d'arbitrage, il permet de commencer par le plus important, le moins coûteux, le plus urgent, le plus simple, ...

Après chaque itération le *product backlog* est mis à jour : fonctions réalisées, évolution de l'importance relative d'un besoin, évolution du coût de réalisation étant donné le chemin parcouru ou l'apparition d'une nouvelle technologie, ...



Exemple



Graphique 5 Timeboxing & product backlog



Remarque : Moins de specs, moins de proto

Chaque itération produit une application fonctionnelle, et non un prototype ou une spécification.

On travaille moins sur les représentations du logiciel (spécifications, modèles, etc.) et

plus sur le logiciel lui même.

d)Avantages des méthodes agiles



Fondamental

Meilleure adéquation au besoin, puisque l'on renégocie le champ fonctionnel du système **au fur et à mesure** de la mise en place des usages réels (et non en fonction d'usages ou d'outils fantasmés).

- Meilleure communication (moins de formalisme et plus d'information utile)
- Meilleure visibilité (réelle et non fantasmée)
- Qualité gérée en continue (et non tâche de documentation déconnectée et à la fin, par exemple)
- Détection des problèmes au fur et à mesure (et non bilan a posteriori)
- Équipe plus solide (démarche de progression et non de conflit, collaboration, relations humaines)



Fondamental

Valeur ajoutée et visibilité apportées au fur et à mesure et non en fin de projet.

e)Budgets et délais



Définition : Budget à la régie

Les hommes.jours sont payés à la consommation a posteriori (en général dans le cadre d'une enveloppe a priori permettant de gérer les disponibilités des RH★ et compatible avec le budget global du client).



Définition : Budget au forfait

Les hommes.jours sont évalués a priori en fonction d'un spectre fonctionnel donné et payés quelque soit la consommation réelle (en théorie, car en pratique le projet dérive, les besoins évoluent et les forfaits sont renégociés, mais souvent dans la douleur).

Intermédiaire forfait/régie

Les méthodes agiles s'accordent bien :

- de budgets à la régie, dans le cadre d'une enveloppe générale plus ou moins stricte correspondant à une évaluation a priori de la MOE➡ en accord avec les moyens de la MOA➡.
- de budgets au forfait, à condition qu'ils soient re-négociables dans une certaine mesure (et dans les deux sens), et qu'ils ne soient pas (trop) liés contractuellement à un spectre fonctionnel (trop) précis.

Finalement, les deux approches convergent vers un entre-deux...

Délais

Le principe est globalement le même pour les délais : le projet pose a priori une date cible de livraison, avec plus ou moins de marge de manœuvre et l'on joue sur le spectre fonctionnel pour ajuster, voire sur le budget (accroissement des ressources en parallèle).



Remarque : Hyperstatisme et degrés de liberté

Les méthodes traditionnelles ont tendance à l'hyperstatisme, rien ne peut être bougé, car tout est fixé : délais, coûts, fonctions.

Dans une méthode agile, en revanche, l'on va maintenir un ou plusieurs degrés de liberté :

- Le budget est fixé, voire les délais, mais pas le champ fonctionnel
- Le budget est fixé plus ou moins 30%, les délais sont fixés (livraison obligatoire), une partie du champ fonctionnel est optionnelle
- ...

Exemple

L'enveloppe initiale fixée a priori est de 100k€, le délai de 6 mois, l'équipe de 4 personnes, pour 200 h.j de travail. À la fin de la première itération, il apparaît qu'une fonction F1 est beaucoup plus complexe que prévue, elle est simplifiée en F1' et un complément de 20 h.j est estimé, un supplément budgétaire de 10k€ est alloué. À la seconde itération, la fonction F1' est terminée, finalement sans surcoût, grâce à une nouvelle technologie identifiée ayant permis de gagner du temps. Les 10k€ supplémentaires sont maintenus, mais alloués à une nouvelle fonction F2 non prévue initialement, que le client a identifié entre temps dans un produit concurrent.

To be continued...

Projets agiles = risque de dérive ?

Au contraire car on inverse le raisonnement habituel : On ajuste le contenu aux moyens plutôt que le contraire.

Si l'on souhaite faire quelque chose d'imprévu, ou que l'on rencontre un problème, l'on va modifier le contenu de ce que l'on souhaitait faire globalement pour absorber le coût de la modification.

Alors que dans un projet traditionnel, l'on est soumis à la contrainte globale qui empêche les ajustements locaux sans négociation (avenant, etc.).

Fondamental

L'objectif des méthodes agiles est de payer plus de réalisation et moins de régulation.

Exemple : Contre-exemple (officiel !)

Les prestataires de services ont coutume d'appliquer un facteur d'ajustement une fois les jours de développement évalués pour payer les coûts de régulation d'une part et tenter de palier par avance aux incertitudes d'autre part.

Ce facteur est en général bien supérieur à 2... Donc pour une journée productive en terme de réalisation, plusieurs sont passées à en parler ou à se protéger de ses éventuelles conséquences néfastes !

4. Discussion : La vie en rose ?

Vite fait, mal fait !

La rapidité des cycles ou la proximité du code ne doit pas être un prétexte à bâcler ou à ne pas réfléchir.

Une bonne conception, un algorithme intelligent ou un programme efficace demandent le même temps de réalisation en méthode agile ou traditionnelle.

La valse des parapluies

Il faut des hommes qui s'engagent, or...

...souvent, tout le monde est chef de quelque chose, mais personne n'est

responsable de rien !

Les salariés sont souvent dans une démarche personnelle plutôt que collective d'entreprise (qui le leur rend bien)...

Le turn-over

Moins formelles les approches agiles sont plus ancrées sur les personnes, le *turn-over* chez les prestataires, mais surtout chez le responsable de projet côté client est catastrophique :

Il faut tout renégocier avec le nouveau responsable du projet, tout peut être remis en cause, et la confiance acquise devient parfois au contraire source de suspicion (« *c'est pas normal qu'ils s'entendent bien ! C'est pas sérieux... »*).

C'est qui qui commande ?

Ces approches fonctionnent lorsque le rapport de force est équilibré :

- une dépendance trop forte au client par exemple risque de devenir source de domination sur le prestataire ;
- autre exemple, un client trop absent ne pourra pas réguler son prestataire

Les systèmes qualité ou de régulation extérieure

Les projets agiles sont évaluables au résultat mais pas (ou peu) aux processus, donc pour celui qui n'accède pas au résultat (n'en perçoit pas la valeur) ou qui a en charge l'évaluation des processus (qualité), ils sont peu visibles.

Les déploiements fréquents et massifs

Pour pouvoir livrer des applications tous les mois et avoir des retours, il faut des terrains et des utilisateurs eux aussi agiles, disponibles et adaptables.

On ne peut pas déployer plusieurs milliers de poste ou changer l'interface d'un site Internet chaque mois.

NB : L'alternative est un terrain de test qui n'est pas le terrain réel.

Il faut bien faire tourner les usines à gaz...

Toutes les technologies ne s'adaptent pas bien aux méthodes agiles, les applications informatiques lourdes (grosses bases de données par exemple) coûtent cher à défaire et à refaire et sont moins modularisables (surtout quand il y a une ancienneté à gérer).

Construire une maison en méthode agile semble peu aisée, car même si c'est le toit le plus utile, il vaut mieux s'appliquer un peu sur les fondations avant...

Les AO et cadres réglementaires

S'articulent mal en général avec des approches moins formelles (mais on trouve des solutions...)

...

G.Panorama des outils documentaires

1.Outils orientés production

a)La bureautique



Définition : La bureautique traditionnelle

La bureautique individuelle constitue la modalité la plus répandue de création de contenu numérique, typiquement les outils de traitement de texte et de création de diaporama. Cette technique a permis la démocratisation de l'accès à l'écriture et à la publication numérique.

Ses principes fondateurs sont le **WYSIWYG** d'une part et la **page blanche** d'autre part. Ils s'inscrivent dans une forme de transposition des pratiques de l'écriture traditionnelle, la page ou la machine à écrire.



Remarque : Point de vue

La bureautique est facilement utilisée car elle s'inscrit dans la continuité des pratiques traditionnelles. Le corollaire est que cette technologie est peu novatrice en terme de technique d'écriture, au regard de ce que permet théoriquement le numérique.

De fait aujourd'hui, la bureautique traditionnelle est de moins en moins adaptée aux logiques de publication multimédia ou aux nouveaux médias de communication.



Remarque : Bureautique en ligne

Notons l'émergence de la bureautique en ligne (suite bureautique Google Apps notamment). Elle permet de faciliter l'usage des outils bureautiques dans un cadre collectif en instrumentant la collaboration entre auteurs.



Complément: Bureautique et WYSIWYG

« *La bureautique se fonde sur le concept du WYSIWYG, « What you see is what you get », ou littéralement en français : « ce que vous voyez est ce que vous obtenez ».* Cette approche vise, en résumé, à permettre à l'utilisateur de créer un document tout en en composant le rendu final. On visualise à l'écran, à mesure que l'on écrit, ce que l'on obtiendra à l'impression pour un document papier, à la vidéo-projection pour un diaporama, ou à la mise en ligne pour un site web. Cette démarche est une transposition directe de l'approche traditionnelle de l'édition non numérique. [...] Elle fait l'hypothèse que les rôles d'auteurs (celui qui écrit le contenu) et d'éditeur (celui qui compose le rendu final) sont fusionnés. (Scenari, la chaîne éditoriale libre [Crozat07], pV-VI) »



Complément: Historique

Le mot bureautique a été créé en 1976 par Louis Naugès, pour désigner l'informatique relative au travail de bureau. On en retiendra principalement la mise en forme électronique d'informations afin de la communiquer en tant que documents. La bureautique désigne aujourd'hui une classe d'outils assez large qui permet de créer facilement sur un ordinateur personnel des documents écrits, sonores et visuels. On parle d'ailleurs de « suite bureautique ». La plus célèbre est celle de Microsoft, on retiendra aussi la suite libre OpenOffice.org. L'outil bureautique le plus

utilisé au monde est certainement le traitement de texte, dont Microsoft Word est le représentant historique et OpenOffice Writer la principale alternative dans le monde du logiciel libre.

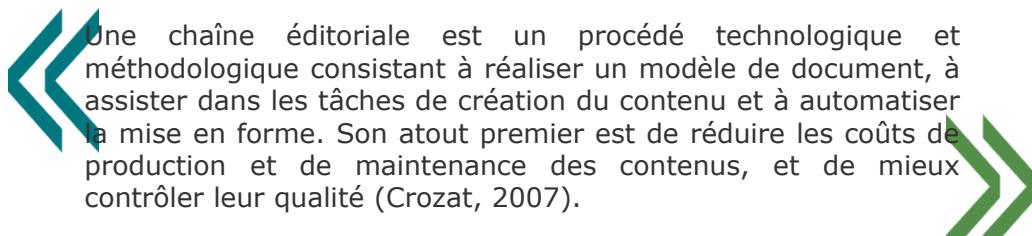
Microsoft Word est né en 1983, en tant que Word 1.0. Il est considéré comme le premier traitement de texte WYSIWYG grand public, même si cette première version était encore très loin de celles que nous connaissons aujourd'hui, et même si le vrai précurseur est le logiciel Bravo sous Xerox en 1975. Le succès de la bureautique est intrinsèquement lié au développement des interfaces graphiques sur la micro-informatique, initié par le projet Lisa d'Apple en 1979, qui donnera le Macintosh en 1984. Les années 90 ont vu la démocratisation des outils bureautiques. Ces derniers ont investi progressivement les entreprises, puis les foyers, en poisons pilotes de la micro-informatique. Les années 2000 commencent à voir l'évolution de ces outils, qui ont fait quelques pas en direction des formats et fonctions jusque là réservés aux chaînes éditoriales.

(Scenari, la chaîne éditoriale libre [Crozat07], p.20)

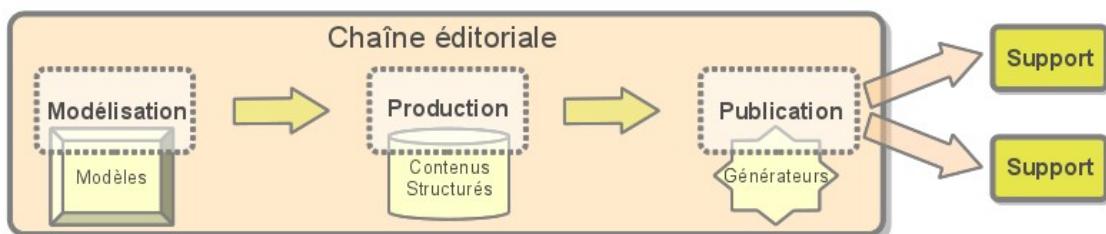
b) Les chaînes éditoriales XML



Définition : Chaîne éditoriale

Une chaîne éditoriale est un procédé technologique et méthodologique consistant à réaliser un modèle de document, à assister dans les tâches de création du contenu et à automatiser la mise en forme. Son atout premier est de réduire les coûts de production et de maintenance des contenus, et de mieux contrôler leur qualité (Crozat, 2007).

Architecture générale



Architecture générale d'une chaîne éditoriale : Modèle-Édition-Publication

Une chaîne éditoriale se compose classiquement :

- D'un ensemble de modèles formels (appelés « schémas documentaires ») permettant de contrôler automatiquement la structure des documents (exemple de technologie : DTD en XML), les références aux ressources binaires (images, etc.) et les références entre documents (liens entre fichiers) ;
- D'un ensemble d'interfaces d'édition WYSIWYM permettant de créer et modifier des documents dans un format conforme aux modèles, de référencer des ressources binaires depuis ces documents et de créer des liens entre ces documents.
- D'un ensemble d'espaces permettant de stocker les fichiers (documents structurés et ressources binaires)
- D'un ensemble de moteurs de publications permettant de transformer les documents structurés et ressources binaires pour rendre des publications lisibles (HTML, PDF, etc.)



Complément : Historique

Le concept de chaîne éditoriale trouve son origine dans l'édition et dans la presse : il consiste à organiser les tâches de production et de publication, en séparant les métiers intervenant dans le processus. A l'origine une chaîne éditoriale est donc un processus avant d'être un outillage, qui trouve son origine dans un besoin d'industrialisation. Historiquement, les deux technologies ayant permis l'implémentation de chaînes éditoriales numériques sont LaTeX (créé en 1982 sur la base du langage d'édition TeX, lui-même créé en 1978) et SGML (norme ISO depuis 1986, issue des travaux d'IBM initiés en 1979 avec GML). XML (standard W3C depuis 1998) est aujourd'hui la technologie de référence pour la réalisation de chaînes éditoriales. C'est sa maturité qui a permis de sortir ce procédé des domaines auxquels il était confiné pour en élargir les usages. Les chaînes éditoriales existent donc depuis le début des années 80, mais sont restées confinées, jusqu'à la fin des années 90 à usages de niche (documentation technique stratégiques dans l'aviation avec SGML, publication scientifique avec LaTeX, etc.). On peut citer Arbortext Epic (1982), Adobe Framemaker (1986), ou Grif (1983). Les premières démocratisations de l'approche se manifestent sur la base d'applications LaTeX avec Lyx, commencé en 1995 (sous le nom de Lyrix) ou PolyTeX, également débuté en 1995 (Bachimont et Charlet, 1998).

En 1999, l'Université de Technologie de Compiègne réalise Scenari, premier environnement de conception de chaînes éditoriales XML proposant un langage déclaratif de modélisation et de publication de haut niveau.

c) Les éditeurs multimédia structurés



Définition : Éditeur multimédia

On appelle éditeur multimédia un outil informatique permettant de produire des compositions spatiales de contenus multimédia (texte, image et son), ainsi que leur scénarisation temporelle.



Remarque

L'écriture de documents multimédia reste le plus souvent une activité réservée à des professionnels. Ceci est dû à la nécessité de maîtriser techniquement à la fois l'acquisition, le traitement des données multimédias (audio, vidéo, animations) et les outils pour en permettre la scénarisation (table de montage, script).

Éditeurs SMIL

Des langages standard comme SMIL ont favorisé la création de contenus à l'aide d'environnements d'édition structurée XML.

Des éditeurs SMIL (par exemple et parmi les plus aboutis : LimSee2¹⁰) existent, sans pour autant s'imposer auprès des utilisateurs, en partie parce que la généricité du langage limite la réalisation d'éditeurs faciles d'accès.

Ajoutons également qu'une des plus importantes limites de SMIL reste l'absence de player robuste.

Outils "auteurs"

Des outils d'édition plus simples, avec une interface utilisateur performante, offrent des services ciblés en s'appuyant sur des structures prédéfinies de documents (comme par exemple pour la scénarisation de présentations incluant des

10 - <http://limsee2.gforge.inria.fr/>

transparents et des vidéos, ou les contenus e-learning). Les rendus sont souvent proposés en Flash.

Si la première approche permet l'obtention d'un grand pouvoir d'expression au détriment de la facilité d'utilisation, la seconde présente des caractéristiques opposées : simplicité mais fonctionnalités limitées.

Outils à base de "template"

Des approches intermédiaires (comme celle adoptée pour *LimSee3*¹¹) intègrent des mécanismes permettant une structuration à la fois logique, spatiale et temporelle. Elles favorisent la modularité et la réutilisation, ainsi que la conception de gabarits (*templates*).

De tels outils peuvent ainsi être adaptés à chaque contexte applicatif et rester indépendant des formats de restitution (comme SMIL, XHTML+Time, XHTML+javascript, etc). Ces outils se rapprochent en cela des chaînes éditoriales.

2.Outils orientés gestion

a) La Gestion Électronique de Documents



Définition : GED

La GED (Gestion Électronique de Documents) est introduite dans les milieux professionnels, en particulier ceux manipulant de larges masses documentaires, par exemple l'industrie pharmaceutique.

L'approche de la GED consiste généralement à greffer une couche de gestion documentaire sur des pratiques existantes. Ces solutions ne font pas (ou peu) d'hypothèse sur la nature des objets documentaires gérés, adressant en général les documents bureautiques traditionnels, c'est notamment ce qui a permis leur dissémination aisée (mais c'est aussi un frein important dans les usages, le document bureautique étant peu manipulable automatiquement car ancré dans une forme physique, c'est une boîte noire).

Les fonctions de gestion concernent typiquement les droits des utilisateurs, la formalisation de processus (**workflow**) éditoriaux (création, relecture, validation, diffusion, etc.), la centralisation et la sécurisation des documents, la gestion de la concurrence. Initialement ces systèmes misent massivement sur la bureautique pour la création et la génération PDF pour la diffusion. Mais la généralisation du Web et la convergence des systèmes, conduit de plus en plus les GED à se rapprocher des Web CMS, et par la même des Wiki.



Exemple

Le domaine est historiquement dominé par de gros éditeurs :

- Documentum (ECM)
- FileNet (IBM)
- SharePoint (Microsoft)
- Open Text
- Stellent (Oracle)

Mais de nouveaux acteurs émergent, notamment issus du monde Open Source :

- Alfresco : éditeur Open Source le plus actif actuellement au niveau international (plus de 10 millions de dollars en 2008, dont 50% aux États Unis, 100 employés)
- Nuxeo : éditeur Open Source français en croissance (environ 6 millions

11 - <http://limsee3.gforge.inria.fr/>

d'euros en 2008, dont 60% en France, 40 employés) [01informatique du 19/02/2009]



Complément : Voir aussi

- APROGED (Association des Professionnels de la Gestion Electronique de Documents) : <http://www.aproged.org/>
 - http://fr.wikipedia.org/wiki/Gestion_%C3%A9lectronique_de_documents
- b) Le Digital Media Asset Management



Définition : DMAM

Le DMAM★ (ou DAM★, ou gestion des médias numériques) est spécifiquement orientés sur l'archivage et l'exploitation (stockage, classement, indexation, recherche, extraction) de larges dépôts de ressources numériques (graphiques, photos, vidéo, musique, etc.). Le développement de ces outils est lié à l'accroissement des ressources numériques.

Ils sont de plus en plus associés à des fonctions collaboratives d'annotations, avec des fonctions telles que la discussion et les commentaires (forums), le classement (*rating*), la mise en relation des médias les uns avec les autres (*linking*), etc.



Exemple

On pourra citer par exemple Cumulus (Canto) qui allie gestion de fichiers multimédia, édition de métadonnées, lien avec la PAO et workflow.

3.Outils spécifiques Web

a) Les Web Content Management System



Définition : WCMS

Un Web CMS (WCMS★ ou CMS★ ou SGC★) est un outil de production de pages HTML, généralement adossé à une base de données relationnelle, à des logiques de gabarits de pages (*templates*) et des feuilles de style CSS.

Cette classe d'outils née au tout début des années 2000 a permis de démocratiser la réalisation de sites Web et d'Intranets d'entreprise, en délestant les auteurs de contenus des tâches techniques d'encodage HTML et de programmation client (JavaScript) ou serveur (PHP généralement).

Ils ont également permis de réintroduire des **processus éditoriaux** : ce ne sont pas seulement des éditeurs HTML (comme Nvu par exemple), mais des systèmes de gestion d'une publication Web, qui intègrent des rôles tels que :

- administration
- gestion éditoriale globale
- contribution rédactionnelle
- validation d'articles
- composition graphique
- etc.



Exemple : Exemple de CMS

Parmi la pléthore de systèmes existants, la plupart Open Source, l'on pourra citer les systèmes les plus couramment rencontrés tels que SPIP, Joomla ou Plone/Zope.

http://fr.wikipedia.org/wiki/Liste_de_syst%C3%A8mes_de_gestion_de_contenu



Exemple : Exemple sous Spip

RACINE DU SITE [?]

Rubrique 2
 Rubrique 1

Retour

Modifier l'article :
Nouvel article

Sur-titre [?]
[Text input field]

Titre [Obligatoire] [?]
Nouvel article

Sous-titre [?]
[Text input field]

À l'intérieur de la rubrique : [?]
Rubrique 1
[N'oubliez pas de sélectionner correctement ce champ.]

Descriptif rapide [?]
(Contenu de l'article en quelques mots.)
[Text input field]

Lien hypertexte (référence, site à visiter...)
Titre :
URL :
[Text input fields]

Créer un article sous Spip

Chapeau [?](#)
(Texte introductif de l'article.)

Texte [?](#)

You pouvez enrichir la mise en page de votre texte en utilisant des « raccourcis typographiques ». [?](#)



Créer du contenu sous Spip



Remarque : Workflow éditorial

Ces outils intègrent aujourd'hui des logiques de gestion de processus éditoriaux (workflow éditorial) et de collaboration (agenda, forum, etc.).



Remarque : Paradigme XHTML

S'ils s'appuient sur un premier niveau de séparation entre les données et leur présentation, et s'ils savent sérialiser tout ou partie de leurs données sous format XML, ils restent dans une logique mono-support (XHTML) et dans le paradigme du WYSIWYG.

b) Les Wiki



Définition : Wiki

Un Wiki est site Web permettant à ses utilisateurs de modifier facilement et dynamiquement le contenu du site.

Les Wiki sont nés dans le milieu des années 2000, pour répondre à un besoin que ne couvraient pas les CMS, à savoir la production "ouverte" de contenu : tout utilisateur, référencé ou non, doit pouvoir librement compléter/modifier/critiquer le contenu d'un site Web, avec effet immédiat sur celui-ci.

Effaçant la traditionnelle barrière entre auteur et lecteur, ces systèmes permettent par exemple la capitalisation de connaissances dans une logique "émergente", non directive, tout un chacun étant à son tour lecteur et auteur. Ils refusent en outre la fonction d'éditeur, chacun étant l'éditeur de l'autre.



Exemple : Exemples de moteurs de Wiki

Il existe de très nombreux Wiki, parmi lesquels MediaWiki (utilisé pour Wikipédia) ou XWiki (logiciel libre français).

http://fr.wikipedia.org/wiki/Liste_de_logiciels_Wiki



Exemple



Exemple : Syntaxe Wiki

```
== Titre ==
* Mot en **gras**
* Mot en //italique//
* Mot __souligné__
* Lien Web : [[http://www.utc.fr]]
```

Gestion du collaboratif dans les Wiki

- Gestion de commentaires
- Gestion de versions

- Édition directe du contenu (après identification ou non)



Remarque : Wiki "purs" et Wiki "éditoriaux"

On observe aujourd'hui que les CMS et Wiki tendent à converger. Les premiers s'ouvrent sur des usages de moins en moins contraints. Les seconds se scindent en deux catégories :

- Les Wiki "purs" qui restent dans la logique de l'informel et de l'imparfait. En conséquence, le lecteur doit assumer une importante activité de critique de ce qui est lu. On peut citer par exemple les Wiki des documentations de projets Open Source.
- Les Wiki qui ré-intègrent une fonction éditoriale et conséquemment ajoutent du contrôle en amont (identification des utilisateurs après inscription libre par exemple) et en aval (publication soumise à validation). C'est typiquement le cas de Wikipédia. Ces systèmes sont une forme de compromis entre qualité et dynamicité.

Un problème des Wiki "éditoriaux", comme Wikipédia, est que la syntaxe devient aussi compliquée du HTML !



Complément : Wiki et lecture critique

Un Wiki est un outil collaboratif au sens où il mise plus sur le réseau d'acteurs et la convergence d'intérêt (lien N:N, tout le monde a quelque chose à dire à tout le monde), que sur une politique éditoriale classique (lien 1:N, une personne souhaite dire quelque chose aux autres). Il apporte dans la relation à l'écrit numérique la possibilité d'inscrire facilement du contenu sans cadre formel préalable. Il est alors un levier permettant de centraliser une importante quantité d'information (enjeu quantitatif).

Mais la question des modalités de l'exploitation de cette information "brute" reste posée, et avec elle celle de l'efficience : les gains apparents en amont à la production, ne sont-ils pas perdus en aval à la lecture ?

Ces solutions, extrêmement légères technologiquement, ont l'avantage d'une grande dynamicité, mais restent assez pauvres d'un point de vue documentaire. Elles sont adaptées aux productions spontanées et peu structurées (drafting) mais conviennent mal dès que des besoins d'organisation ou de publication avancées s'imposent.

c) Exemple de Wiki : MediaWiki



Exemple : Wikipédia (moteur MediaWiki)

Cours

The screenshot shows a web browser window displaying the French Wikipedia article on "Scenari". The page title is "Scenari". A yellow box at the top right of the article content area states: "Cet article est une ébauche concernant les logiciels. Vous pouvez partager vos connaissances en l'améliorant. (Comment ?)". The main content discusses Scenari as a suite of logic software used for document creation, mentioning XUL/Javascript and Java. A sidebar on the right provides technical details about the software, including its latest version (SCENARIchain 3.4.1.02), environments (Windows, Mac OS X, GNU/Linux), language (Français), type (Chain editorial), licenses (quadruple license, including MPL, GPL, LGPL, CeCILL), and the website (scenari-platform.org). The bottom of the page includes a note about the chain editorial being a technological and methodological process, and a link to SCENARIdiscovery 1.

Visualiser un article (Wikipédia)

The screenshot shows a web browser window displaying the discussion page for the "Scenari" article on Wikipedia. The page title is "Discuter:Scenari". It shows a single edit history entry by user "Alchemical" from October 19, 2007, at 13:02 (CEST). The edit summary was: "Par contre, pascal a raison sur les majuscules / minuscules, depuis 2 semaines nous avons adopté cette nouvelle écriture moins austère pour le mot Scenari. Je ne connais pas trop la procédure ? suppression de celle ci + renomage de la page d'origine ?". The bottom of the page includes a note about the last modification date (October 19, 2007) and the Creative Commons Attribution-ShareAlike license information.

Discuter un article (Wikipédia)

Nous n'êtes pas actuellement connecté comme utilisateur enregistré. Si vous modifiez cette page, votre **adresse IP sera archivée publiquement dans l'historique**. Créer un [compte utilisateur](#) permettra de masquer votre adresse IP et de vous procurer d'autres avantages.

Vous devez prévisualiser la page avant de publier votre modification.

...Scenari... est une suite logicielle (logiciel libre/libre) (avec des bibliothèques propriétaires) de conception et d'utilisation de chaînes éditoriales pour la création de documents multimédia.

Les langages utilisés sont [[XUL]] / [[Javascript]] pour l'interface graphique et [[Java (langage)|Java]] pour la partie interne. Les documents de l'auteur sont enregistrés en [[Extensible Markup Language|XML]] et traités par des transformations [[XSLT]].

== Principes ==
Une [[chaîne éditoriale]] est un procédé technologique et méthodologique, issu de la recherche en ingénierie documentaire. L'approche consiste à réaliser un modèle de document, à assister les tâches de création du contenu et à automatiser leur mise en forme.

Scenari est destiné aux personnes confrontées aux problématiques suivantes :

- * Marquer l'identité d'une organisation : l'auteur est contraint de publier des documents avec une structure et une charte graphique homogène, ces informations étant prédefinies dans un modèle documentaire.
- * Diffuser des contenus sous différents formats : le contenu est saisi une seule fois dans l'interface auteur, mais les formats de publications sont multiples.
- * Maintenir les documents à jour : lorsque le contenu doit être changé, il n'y a qu'une seule source de données à mettre à jour.

L'approche est différente de la méthode [[bureaucratique]] dite "artisanale".

== Découvrir le fonctionnement de Scenari par des exemples ==
[SCENARI discovery](http://scenari-platform.org/projects/scenari/fr/demo/co/) est un programme permettant de découvrir le monde des chaînes éditoriales.

Choisissez un modèle de document parmi les 6 proposés en exemple :

* CV

→ Ne copiez pas de texte d'une page Web : respectez le droit d'auteur (aide) ;
→ Fondez vos informations sur des sources vérifiables, et citez ces sources (aide)

En publiant, vous placez votre contribution sous la [GNU Free Documentation License](#) : vos écrits pourront être modifiés et distribués sans votre accord.

Résumé :

Publier (après prévisualisation) | Prévisualiser | Changements en cours | Annuler | Aide (ouvre une nouvelle fenêtre)

Caractères spéciaux : æ Å à Ä å Ä ä Ä ç Ç € é É è È è È ï ï ö Ö ù û ù û ù ù ÿ ÿ « » {{}} {{subst:}} {{|}} {{|}} {{|}} {{|}} {{|}} {{|}}

Terminé

FoxyProxy: Désactivé(e)

Éditer un article (Wikipédia)

Cours

<http://fr.wikipedia.org/w/index.php?title=Scenari&action=history>

Historique des versions de « Scenari » (?)

Voir les opérations sur cette page

(Dernières contributions | Premières contributions) Voir (50 plus récentes) (50 plus anciennes) (20 | 50 | 100 | 250 | 500).

Outils : Obtenir la liste des auteurs - Détails des contributions

Discussions liées :

Légende : (actu) = différence avec la version actuelle, (diff) = différence avec la version précédente, m = modification mineure
Comparer les versions sélectionnées

- (actu) (diff) 4 avril 2008 à 21:29 Badmood (Discuter | Contributions) m (5 024 octets) (nouveau style de bandeau de portail, voir Wikipedia:Prise de décision/Bandeaux de portail) (défaire)
- (actu) (diff) 22 février 2008 à 13:03 Stephane.poinsart (Discuter | Contributions) m (5 046 octets) (sortie d'une nouvelle version mineur, mise à jour du numéro) (défaire)
- (actu) (diff) 8 février 2008 à 17:29 Stephane.poinsart (Discuter | Contributions) (5 046 octets) (sorties de nouvelles versions) (défaire)
- (actu) (diff) 8 février 2008 à 00:17 Traumrune (Discuter | Contributions) m (5 034 octets) (cat) (défaire)
- (actu) (diff) 5 février 2008 à 21:35 DumZiBoT (Discuter | Contributions) m (5 003 octets) (Robot : mise en application de Wikipedia:Prise de décision/Changement de style des messages d'avertissement (sortie des modèles d'avertissement de {{multi bandeau}}), et ajout du modèle {{ébauche}} pour les bandeaux d'ébauches multiples) (défaire)
- (actu) (diff) 19 janvier 2008 à 04:14 PieRRoBoT (Discuter | Contributions) m (5 003 octets) (Bot : Remplacement de texte automatisé (-XML +XML)) (défaire)
- (actu) (diff) 2 novembre 2007 à 23:33 Stephane.poinsart (Discuter | Contributions) m (4 949 octets) (mise à jours du numéro de version : 3.3.1 -> 3.4.0) (défaire)
- (actu) (diff) 27 octobre 2007 à 11:57 Piglop (Discuter | Contributions) m (4 949 octets) (suppression commentaire html) (défaire)
- (actu) (diff) 27 octobre 2007 à 11:55 Piglobot (Discuter | Contributions) (5 325 octets) (Correction automatique de l'infobox Logiciel) (défaire)
- (actu) (diff) 19 octobre 2007 à 14:54 Pascal Boulerie (Discuter | Contributions) (5 523 octets) (défaire)
- (actu) (diff) 19 octobre 2007 à 14:53 Pascal Boulerie (Discuter | Contributions) m (5 527 octets) (défaire)
- (actu) (diff) 19 octobre 2007 à 13:16 Stephane.poinsart (Discuter | Contributions) m (5 507 octets) (Changement d'écriture du mot SCENARI -> Scenari (sauf pour noms de programmes)) (défaire)
- (actu) (diff) 19 octobre 2007 à 13:02 Alchemica (Discuter | Contributions) m (a renommé SCENARI en Scenari: selon la nouvelle orthographe du nom) (défaire)
- (actu) (diff) 21 septembre 2007 à 04:15 MMBot (Discuter | Contributions) m (5 441 octets) (Bot : Remplacement de texte automatisé (-\r\n|\r\n|\r\n|n==+\r\n|\r\n|\r\n|n==)) (défaire)
- (actu) (diff) 19 septembre 2007 à 12:30 Badmood (Discuter | Contributions) m (5 442 octets) (normalisation des liens) (défaire)

<http://fr.wikipedia.org/w/index.php?title=Scenari&action=history> FoxyProxy Désactivé(e)

Historique des modifications (Wikipédia)

<http://fr.wikipedia.org/w/index.php?title=Scenari&diff=28318815&oldid=26572322>

Scenari

(Différences entre les versions)

Version du 22 février 2008 à 13:03 (modifier) (défaire)
Stephane.poinsart (Discuter | Contributions)
m (sortie d'une nouvelle version mineur, mise à jour du numéro)
→ Différence précédente

Version actuelle (4 avril 2008 à 21:29) (modifier) (défaire)
Badmood (Discuter | Contributions)
m (nouveau style de bandeau de portail, voir Wikipedia:Prise de décision/Bandeaux de portail)

Ligne 78 :

* {{fr}}	* {{fr}}
[http://www.journaldunet.com/solutions/0703/070307-gestion-contenu-documentaire-sc article journal du net]	[http://www.journaldunet.com/solutions/0703/070307-gestion-contenu-documentaire-si article journal du net]
- {{Multi bandeau Portail logiciels libres Portail informatique}}	+ {{Portail logiciels libres informatique}}
[[Catégorie:Logiciel libre]]	[[Catégorie:Logiciel libre]]

Version actuelle

Cet article est une ébauche concernant les logiciels.
Vous pouvez partager vos connaissances en l'améliorant. ([Comment ?](#))

Scenari est une suite logicielle libre (avec des bibliothèques propriétaires) de conception et d'utilisation de chaînes éditoriales pour la création de documents multimédia.

Les langages utilisés sont Xul / Javascript pour l'interface graphique et Java pour la partie interne. Les documents de l'auteur sont enregistrés en XML et traités par des transformations XSLT.

Sommaire [masquer]

- 1 Principes
- 2 Découvrir le fonctionnement de Scenari par des exemples
- 3 Autres modèles documentaires
- 4 Construire ses propres modèles
- 5 La communauté scenari-platform
- 6 Notes et références de l'article
- 7 Voir aussi
 - 7.1 Articles connexes
 - 7.2 Liens et documents externes

Dernière version

SCENARIchain 3.4.1.02
SCENARibuilder 3.4.1.02 SCENARiserver 3.4.1.02 SCENARiclient 3.4.1.02 SCENARIdiscovery 1.1

Environnements Windows, Mac OS X, GNU/Linux

Langues Français

Type Chaîne éditoriale

Licences quadri-licence, au choix : MPL, GPL, LGPL,

[Terminé](#) FoxyProxy Désactivé(e)

Comparer des versions (Wikipédia)

4.Outils périphériques

a)Les moteurs de recherche et d'indexation



Définition : Moteur de recherche

Un moteur de recherche est un logiciel permettant de retrouver des informations numériques stockées dans des fichiers sur un ordinateur ou un réseau, quelque soit les formes de fichier (document, mail, base de données, etc.).



Exemple : Exemple de moteurs de recherche

- Google
- Lucene (Apache, Open Source, Java)
- Exalead (Français)



Exemple : Exemple d'interfaces de recherche

- Google Desktop
- Beagle (IHM pour la recherche Desktop sur Lucene)



Complément : Stratégies d'indexation

La démarche dominante ("Google-like") est surtout fondée sur une approche **yntaxique** (mots clés, lemmatisation, anti-dictionnaire, etc.) et statistique (occurrence de mots, occurrence de requêtes, etc.).

À noter, deux approches alternatives et complémentaires à cette approche :

- L'indexation sémantique (ontologies, taxonomie)
- L'indexation collaborative (folksonomie)



Définition : Ontologie

Une ontologie est une **représentation formelle d'une conceptualisation partagée** (<http://websemantique.org/Ontologie>).

Cette définition hérite de la définition initiale proposée par Gruber : "an explicit specification of a conceptualization".



Définition : Folksonomy

« Système de classification collaborative et spontanée de contenus Internet, basé sur l'attribution de mots-clés librement choisis par des utilisateurs non spécialistes, qui favorise le partage de ressources et permet d'améliorer la recherche d'information. »(Le Grand Dictionnaire Terminologique [Le Grand Dictionnaire Terminologique]).

b)Les logiciels de groupe de travail



Définition : Logiciel de groupe de travail

Un logiciel de groupe de travail (ou logiciel de groupe, ou plus communément sous sa forme anglaise : *groupware*) est un logiciel qui propose, pour un groupe de personnes, des outils de partage de documents et de travail collaboratif (mails, planning, contacts, etc.).



Exemple : Exemple de groupware

- Lotus Notes (acteur historique)
- MS Exchange
- Zimbra (logiciel libre, racheté par Yahoo! en 2007)

Exemple



Recueil de fonctions établies à partir du logiciel eGroupware (source Wikipédia, GNU General Public License, <http://fr.wikipedia.org/wiki/EGroupWare>)

H.Indexation et recherche

1. Introduction à l'indexation



L'indexation a pour principal objectif de rendre accessibles des informations, que l'on repère pour cela au moyen d'index. L'indexation est le processus selon lequel le contenu d'un document est analysé pour être ensuite reformulé dans une forme permettant d'accéder au contenu et de le manipuler. Le terme d'indexation qualifie à la fois le processus et son résultat. Une indexation est par conséquent la description d'un document effectuée dans la perspective d'une utilisation et exploitation données.

L'indexation repose traditionnellement sur deux étapes clairement distinguées :

- une étape d'analyse conceptuelle : le contenu est analysé et interprété par un documentaliste pour définir les principaux concepts permettant de le caractériser ;
- une étape de reformulation documentaire : l'analyse conceptuelle permet au documentaliste de reformuler le contenu dans une forme permettant sa manipulation.

(Bachimont, 2007) [Bachimont07]



a) Définition de l'indexation



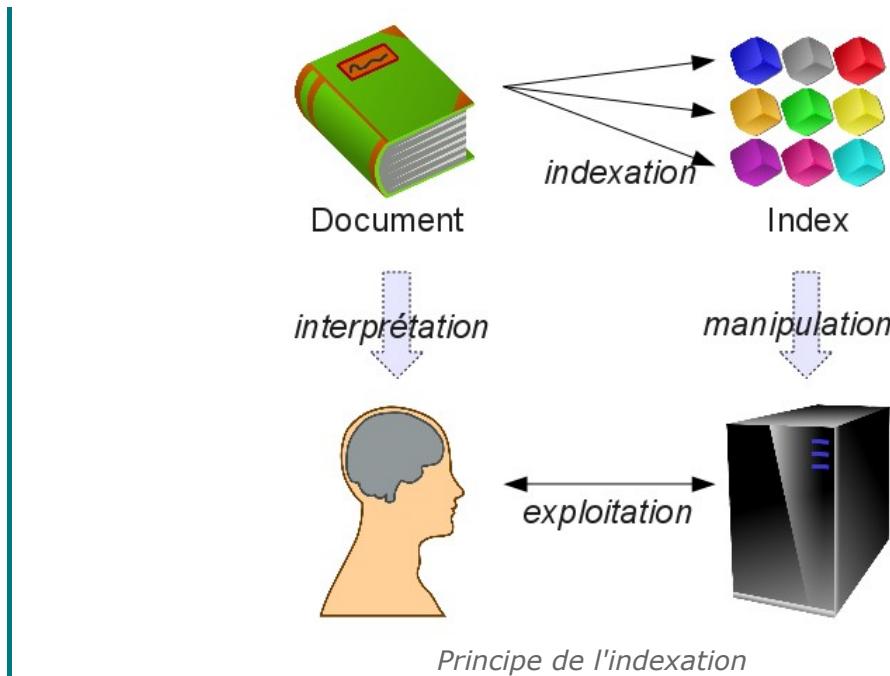
Définition

L'indexation est un processus consistant à reformuler le contenu d'un document sous une forme plus adaptée à son exploitation dans une application donnée (par exemple la recherche).

Dans le contexte du numérique :

- Les documents sont destinés à être interprétés par des êtres humains.
- Les index sont des représentations des documents destinés à être manipulés par des ordinateurs.





Exemple

- Exemple d'exploitation de l'indexation : chercher, réutiliser, recomposer, etc.
- Exemple de formes d'index : liste de termes, langue naturelle, formalisme logique, etc.

b) Indexation de fragments documentaires numériques

De l'analogique au numérique

En contexte analogique l'unité d'indexation (ce que l'on référence) correspond exactement au document physique complet :

- on ne sait pas manipuler de sous-ensemble plus fin
- on doit gérer chaque support physique séparément

Le numérique permet :

- d'**intégrer** l'indexation (une indexation unique pour des contenus multiples)
- d'indexer l'**intérieur** du document (tout fragment de contenu devient adressable).

Intégration

Les contenus étant unifiés par la numérisation, il devient possible de gérer une indexation également unifiée y compris pour des contenus hétérogènes.

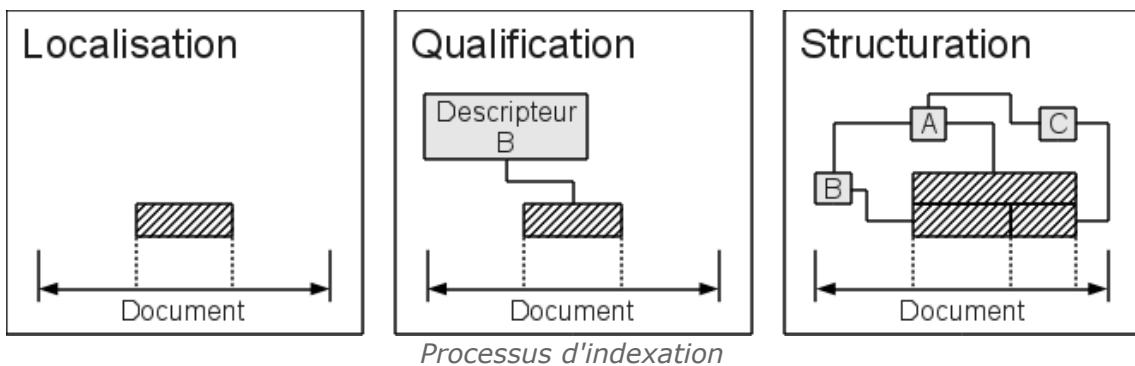
Indexation fine du contenu

Toute information possédant une adresse exploitable par le processus d'indexation devient accessible, donc tout segment arbitraire du contenu (fragment) est potentiellement indexable.

c) Principes de l'indexation

L'indexation de documents numériques repose sur 3 principes :

- La localisation : qu'est ce ce qui est indexé ?
- La qualification : par quelle valeur ?
- La structuration : dans quelle organisation ?



La localisation

Le problème est de repérer les unités significantes : Il n'existe pas **a priori** d'unités significantes, les unités significantes résultent d'une interprétation.

La qualification

La qualification est la condition sous laquelle on est capable d'attribuer une valeur, une utilité à une unité d'indexation, elle justifie son exploitation.

Structuration

Il s'agit de rassembler les différentes qualifications effectuées en un système organisé.

2. Introduction aux techniques d'indexation et de recherche

a) Techniques d'indexation

- Indexation plein-texte : des mots clés sont extraits automatiquement du contenu
- Méta-données documentaires : des propriétés documentaires sont renseignées manuellement
- Référentiels : les documents sont associés à des entrées de référentiel (taxinomie, thésaurus)
- Ontologies : Le contenu des documents est représenté logiquement
- Folksonomies : Les documents sont associés à des tags (mots clés) posés par les utilisateurs finaux
- ...



Exemple : Méta-données Dublin Core

Le Dublin Core est une norme ISO proposant un jeu de méta-données très générales pour la description des ressources numériques. Il comporte 15 descripteurs de base (titre, créateur, mots-clés, description, date, langue, ...).

Propriétés

* Nom:	nf29cas.pdf
Type de contenu:	Adobe PDF Document
Codage:	UTF-8
Titre:	Etude de cas : base de contenus pédagogiques
Description:	
Auteur:	Stéphane Crozat
* Éditeur:	UTC
* Contributeur:	Manuel Majada
* Type:	Document Scenari/Opale
* Identifiant:	nf29cas.pdf
* Source:	http://www4.utc.fr/~nf29/2008p/src
* Couverture:	France
* Droits d'accès:	CC BY-NC-SA (http://creativecommons.org/licenses/by-nc-sa/2.0/)
* Sujet:	Ingénierie documentaire

Interface d'indexation Dublin Core (Alfresco)



Exemple : Référentiels

Les valeurs des méta-données peuvent être contrôlées par des référentiels, ici mots-clés et couverture.

Titre *	La théorie de la relativité		
Description			
Mots-clés	Vie quotidienne Sciences Humaines Société Technologie	Sélectionnez une valeur. Astronautique Électronique Energie	Ajouter
	Sélection: <input checked="" type="checkbox"/> Technologie/Energie		
Droits			
Source			
Couverture	Europe	Sélectionnez une valeur.	
Créé le	24/02/10 15:03		

Indexation selon des référentiels géographique et thématique (Nuxéo-DM)
b)Thésaurus



Définition : Glossaire

Un glossaire est une liste de termes associés à leurs définitions, en relation avec un **domaine** particulier.

Les termes sont généralement présentés par ordre alphabétique, c'est une sorte de dictionnaire contextuel, et limité à des mots spécifiques ou complexes.



Définition : Taxinomie

Une taxinomie (parfois taxonomie, *taxonomy* en anglais) est un système de classification **hiérarchique** de l'information (le terme est issu à l'origine de la classification des espèces vivantes.)

La relation de structuration est de type **est-un** (*is-a*) : épagneul est-un (chien est-un (canin est-un (mammifère))).



Définition : Thésaurus

Un thésaurus est un **vocabulaire normalisé** concernant un domaine particulier, en général pour un objectif particulier, par exemple l'indexation.

Il se présente sous la forme d'un **réseau** de termes reliés entre eux par des associations sémantiques : synonymie, hyper/hyponymie, ...

Les termes peuvent également être décrits par une définition en texte libre, permettant d'en faciliter la compréhension, il a alors également valeur de glossaire.



Méthode : Élaboration d'un thésaurus

1. Déterminer les termes qui peuvent être pris en compte :
 - Descripteurs utilisables
 - Non descripteurs ne pouvant pas l'être
2. Associer des relations sémantiques entre ces groupes :
 - Hyperonymie ou hyponymie
 - Synonymie
 - Association



Méthode : Synonymie

Choisir un terme préféré parmi les termes synonymes, et déclarer les autres comme non-descripteurs :

- Logiciel, descripteur, employé-pour (Software, Application)
- Software, non-descripteur, employer (Logiciel)
- Application, non-descripteur, employer (Logiciel)

Utiliser la relation de synonymie pour gérer les polyséries :

- Bateau, non descripteur, employer (Bateau pavé, Navire)



Méthode : Hyper/hyponymie

Ces relations permettent de déclarer les relations de généricté/spécifité entre les descripteurs.

- Europe, descripteur, hyperonyme-de (France, Allemagne)
- France, descripteur, hyponyme-de (Europe)
- Allemagne, descripteur, hyponyme-de (Europe)
- Oiseau, descripteur, hyperonyme-de (Canard)
- Canard, descripteur, hyponyme-de (Oiseau)

Ces relations peuvent être ontologiques (sorte-de) ou méréologiques (partie-de). Ces deux types peuvent être distingués pour affiner la description :

- France, descripteur, partie-de (Europe)
- Canard, descripteur, sorte-de (Oiseau)



Méthode : Association

L'association est mal définie dans les thésaurus, c'est en fait toute relation qui n'est pas l'une des précédentes :

- Relation entre agent et action
- Relation entre action et objet
- Relation entre des termes co-occurrents
- ...

c)Ontologies

Définition : Ontologie

Une ontologie (en informatique, ne pas confondre avec le concept philosophique) est :

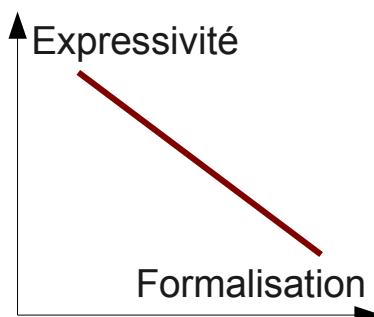
- une taxinomie ou un thésaurus
- **plus** une représentation dans un langage formalisé doté d'une sémantique formelle, permettant de faire des inférences sur les termes

Une ontologie est donc une **conceptualisation calculable** d'un domaine, qui poursuit deux objectifs :

- Expliciter une **compréhension** commune de notions ou concepts d'un domaine par les hommes (expression d'un consensus dans une communauté, ce sur quoi tout le monde est d'accord)
- Permettre une **opérationnalisation** des concepts par les machines (formalisation permettant des inférences conformes à la sémantique des concepts)

Remarque

Plus on peut exprimer de choses, moins on peut les formaliser et calculer sur elles ; plus c'est formalisé et calculable, moins c'est expressif.



Graphique 6 Dualité expressivité/formalisation

Fonction des ontologies

L'objectif des ontologies est d'améliorer l'indexation :

- en augmentant les taux de réponse, typiquement pour trouver un document qui ne contient pas un terme recherché, mais un synonyme ou un hyperonyme.
- en diminuant les taux de bruit, typiquement pour éliminer un document qui contient un homonyme.



Complément : Schémas de classification : thésaurus, taxonomie, ontologie...

<http://www.dia-logos.net/ressources/2009-11/schemas-de-classification-thesaurus-taxonomie-ontologie>

d) Techniques de recherche

- Requêtes par la formulation de mots clés libres qui sont appariés avec tous les index
 - Extension avec utilisation de connecteurs logiques (AND / OR)
 - Extension avec des modèles probabilistes (indice de pertinence)
- Requêtes par la formulation de mots clés libres ou contrôlés index par index
 - auteur = X AND/OR titre=Y ...
- Recherche par la navigation
 - dans des référentiels (ontologies, thésaurus, etc.)
 - dans des "nuages" de mots-clés ayant une proximité statistique
- ...

I. Enterprise Content Management

1. La gestion de contenu



Définition : ECM

L'ECM (Enterprise Content Management ou gestion de contenu) regroupe les systèmes informatiques permettant de gérer les processus documentaires au sein d'une organisation.



Complément : Autres définitions de l'ECM

- « *La gestion de contenu (en anglais Enterprise Content Management, ECM) vise à gérer l'ensemble des contenus d'une entreprise. Il s'agit de prendre en compte les informations sous forme électronique, qui ne sont pas structurées, comme les documents électroniques, par opposition à celles déjà structurées dans les bases de données* » (http://fr.wikipedia.org/wiki/Gestion_de_contenu).
- « *Enterprise Content Management (ECM) is the technologies used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM tools and strategies allow the management of an organization's unstructured information, wherever that information exists* » (AIIM® [AIIM]).
- « *Gestion associée à l'archivage, la réutilisation et la distribution des contenus numériques et multimédias (textes, images, animations, photos, fichiers audio ou vidéo, etc.) d'une organisation* » (Le Grand Dictionnaire Terminologique® [Le Grand Dictionnaire Terminologique]).



Exemple

On trouve notamment dans cette catégorie les outils de gestion de contenu collaboratifs, tels que les Web CMS, les Wiki, les systèmes de GED et de gestion de workflow et les DAM (Digital Asset Management). L'ECM est également à rapprocher du domaine de l'indexation et des moteurs de recherche, et de plus en plus des logiciels de travail collaboratif (*groupware*).



Complément : Historique

Le terme de GED★ apparaît dans les années 80. Son objectif initial est la gestion de l'archivage de documentations papiers, puis électronique, par la gestion de **métadonnées**.

Le terme de GEIDE★ est promu en 1994 pour ancrer la gestion des formats électroniques et le passage à la numérisation/dématerrialisation (images scannées, PDF, bureautique via RTF notamment).

Le terme ECM★ réactualise et généralisent le domaine du "contenu numérique" adressé par la GEIDE.



Complément : Mots clés de l'AIIM

L'AIIM★ (AIIM - The Enterprise Content Management Association, www.aiim.org¹²) propose une revue très large des processus documentaire à l'œuvre dans une organisation. On a reproduit ci-dessous la liste de ces processus (détaillés sur AIIM [AIIM]).

- BUSINESS PROCESS MANAGEMENT/WORKFLOW
- CONTENT AND DOCUMENTS
- SCANNING
- DOCUMENT IMAGING
- FORMS PROCESSING
- RECOGNITION
- CATEGORIZATION/TAXONOMY
- INDEXING
- DOCUMENT MANAGEMENT
- RECORDS MANAGEMENT
- EMAIL MANAGEMENT
- WEB CONTENT MANAGEMENT
- DIGITAL ASSET MANAGEMENT
- REPOSITORIES
- STORAGE
- CONTENT INTEGRATION
- MIGRATION
- BACKUP/RECOVERY
- SEARCH/RETRIEVAL
- SYNDICATION
- LOCALIZATION
- PERSONALIZATION
- PUBLISH
- PAPER ELECTRONIC
- SECURITY
- COLLABORATION
- LONG-TERM ARCHIVAL



Complément : Information structurée ou non structurée ?

On lit et entend souvent que la gestion de contenu adresse les informations dites "non structurées", par opposition aux bases de données ou aux PGI★ qui adressent les informations dites "structurée".

L'information est généralement dite "**non structurée**" lorsqu'elle est portée par un support numérique qui ne peut être interprétée que par l'homme. On parle de **document**.

L'information est dite "**structurée**" lorsqu'elle est représentée de telle façon qu'elle est calculable par un algorithme : c'est le cas dans les bases de données grâce au modèle relationnel, ou en Intelligence Artificielle. On parle plutôt de **donnée**.

Et l'on parle parfois d'information "**semi-structurée**" lorsque sa structure est calculable, mais pas son contenu, comme c'est le cas avec langages XML métiers (ou l'on peut manipuler algorithmiquement la structure, mais pas le contenu).

Il faut donc entendre ici **structuré** du point de vue de la machine, au sens de logique, calculable. **Mais un document est bien une information structurée, du point de vue humain !**

2.Les grandes fonctions des ECM

a)Indexation



Définition : Indexation

L'indexation consiste à ajouter des méta-données au contenu, en particulier à des fins de recherche.

Il existe plusieurs types d'indexation :

- Indexation plein-texte : des mots clés sont extraits du contenu
- Méta-données documentaires : des propriétés documentaires sont renseignées
- Référentiels : les documents sont associés à des entrées de référentiel (théâtre arborescent)
- Ontologies : Le contenu des documents est représenté logiquement
- Folksonomies : Les documents sont associés à des *tags* (mots clés) posés par les utilisateurs finaux
- ...



Exemple : Ajout de propriétés Dublin Core (Alfresco)

Cours

Propriétés

* Nom:	nf29cas.pdf
Type de contenu:	Adobe PDF Document
Codage:	UTF-8
Titre:	Étude de cas : base de contenus pédagogiques 
Description:	 
Auteur:	Stéphane Crozat
* Éditeur:	UTC
* Contributeur:	Manuel Majada
* Type:	Document Scenari/Opale
* Identifiant:	nf29cas.pdf
* Source:	http://www4.utc.fr/~nf29/2008p/src
* Couverture:	France
* Droits d'accès:	CC BY-NC-SA (http://creativecommons.org/licenses/by-nc-sa/3.0/)
* Sujet:	Ingénierie documentaire

Interface d'indexation Dublin Core (Alfresco)



Exemple : Indexation selon des référentiels

Un document peut être indexé selon un ou plusieurs référentiels, qui peuvent ensuite être navigués pour retrouver les documents associés. Un référentiel se comporte comme une organisation virtuelle de documents.

Screenshot of a Nuxeo-DM document editing interface showing the indexing of a document.

The interface includes the following fields:

- Titre***: La théorie de la relativité
- Description**: (Empty text area)
- Contenu**:
 - Aucun
 - Charger
 - Parcourir...
- Mots-clés**:
 - Vie quotidienne
 - Sciences Humaines
 - Société
 - Technologie
 - Astronautique
 - Électronique
 - Energie

Sélection: Technologie/Energie

Ajouter
- Droits**: (Empty text area)
- Source**: (Empty text area)
- Couverture**: Europe / Sélectionnez une valeur.
- Créé le**: 24/02/10 15:03

Indexation selon des référentiels géographique et thématique (Nuxéo-DM)

Cours

The screenshot shows the Nuxeo-DM interface. The left sidebar has a 'Navigation par sujet' section with categories like Art, Sciences Humaines, Société, Vie quotidienne, Technologie, and others. The 'Technologie' category is expanded, and 'Energie' is selected. The main panel shows a search bar and a breadcrumb trail: 'Navigation virtuelle par thème'. A table lists documents: 'Albert Einstein' (modified 25/02/10 22:58) and 'La théorie de la relativité' (modified 25/02/10 22:59). Buttons at the bottom include Coller, Supprimer, Copier, and Ajouter à mon lot de documents.

Recherche dans un référentiel thématique (Nuxéo-DM)

The screenshot shows the Nuxeo-DM interface. The left sidebar has a 'Navigation par couverture' section with 'Europe' selected. The main panel shows a search bar and a breadcrumb trail: 'Navigation virtuelle par couverture'. A table lists documents: 'La théorie de la relativité' (modified 25/02/10 22:59). Buttons at the bottom include Coller, Supprimer, Copier, and Ajouter à mon lot de documents.

Recherche dans un référentiel géographique (Nuxéo-DM)

b) Versioning

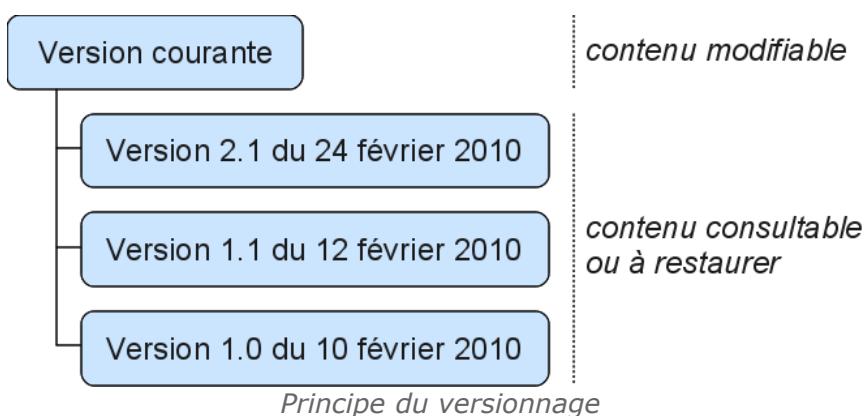


Définition : Version

Une version est une copie d'un fichier permettant de figer son état à un instant donné et de le retrouver par la suite. Le fichier et ses versions forment une liste qui permet de **remonter dans le temps**.

Le fichier modifiable est dit en version courante, on parle aussi parfois de *live document*.

Une version est identifiée, en général par un numéro et/ou une date.



i Versionnage



Définition : Versionnage manuel

Fonction permettant aux utilisateurs de créer une version manuellement, correspondant en général à une phase de la production. Il s'agit d'une fonction classique des GED.

Synonyme : *versioning*



Exemple : Nuxéo

La version est associée à un numéro (souvent de la forme `versionMajeure.versionMineure`) et à une description qui explicite ce à quoi correspond la version.

Les versions antérieures peuvent être restaurées, c'est à dire qu'elles peuvent remplacer la version courante (qui est généralement préalablement versionnée elle-même)

Cours

Votre identité : ICI55-01 | Tableau de bord | Retour à la base documentaire | Gestion des utilisateurs | Me déconnecter

ICI55-01

Contenu Modification Mes notifications Historique Administration

Nouveau document Importer un fichier

Titre	Dernière modification	Auteur	Cycle de vie
La théorie de la relativité	24/02/10 15:03	ICI55-01	En projet

Coller Supprimer Copier Ajouter à mon lot de documents

RSS | ATOM

Copyright© 2006-2009 Nuxeo. Visit [nuxeo.com](#) | Get support | Join the community | QuickStart

Choisissez votre langue : [français](#) Changer

Terminé

Dépôt d'un document initial (Nuxéo-DM)

Edition Affichage Historique Marque-pages Outils Aide

Votre identité : ICI55-01 | Tableau de bord | Retour à la base documentaire | Gestion des utilisateurs | Me déconnecter

La théorie de la relativité

Résumé Modification Fichiers Publication Relations Workflow Mes notifications Commentaires Historique

Titre* La théorie de la relativité générale

Mise à jour des versions

- Pas de montée de version
- Montée de version mineure
- Montée de version majeure

Commentaire Réécriture du chapitre 1.

Enregistrer

Copyright© 2006-2009 Nuxeo. Visit [nuxeo.com](#) | Get support | Join the community | QuickStart

Choisissez votre langue : [français](#) Changer

Terminé

Modification d'un document et modification manuelle de la version (Nuxéo)

Cours

The screenshot shows the Nuxeo Document Management System (DM) interface. At the top, there is a navigation bar with links for 'Fichier', 'Edition', 'Affichage', 'Historique', 'Marque-pages', 'Outils', and 'Aide'. Below the navigation bar, the URL is http://172.21.130.224:8080/nuxeo/nxdoc/default/a8bacad13-ad1f-46f1-90a1-6d3aa8e965017/view_documents?tabId=&conversationId=ONXMAIN. The main header says 'nuxeo' and 'ICI55-01'. The page title is 'ICI55-01'. There are tabs for 'Contenu' (which is selected), 'Modification', 'Mes notifications', 'Historique', and 'Administration'. A search bar is present at the top right. The main content area displays a table of documents. The first document listed is 'La théorie de la relativité générale' with a modification date of '24/02/10 15:05', author 'ICI55-01', and status 'En projet'. Action buttons for 'Coller', 'Supprimer', 'Copier', and 'Ajouter à mon lot de documents' are shown next to each document entry. On the left sidebar, there are links for 'Mes documents' (which is selected) and 'Presse-papier'. A message 'Aucun document dans la sélection.' is displayed. The bottom right corner features a decorative graphic of yellow circles and the text 'RSS | ATOM'. The footer contains copyright information: 'Copyright© 2006-2009 Nuxeo. Visit [nuxeo.com](#) | Get support | Join the [community](#) | QuickStart'. It also includes a language selection dropdown set to 'français' and a 'Changer' button.

Visualisation de la nouvelle version courante (Nuxéo-DM)

The screenshot shows the Nuxeo DMS interface. At the top, there's a navigation bar with links like 'Eichier', 'Edition', 'Affichage', 'Historique', 'Marque-pages', 'Outils', and 'Aide'. Below the navigation bar, the URL is http://172.21.130.224:8080/nuxeo/nxdoc/default/67c0227b-25ab-4587-b383-9e792a5043c1/view_documents?tabId=TAB_CONTENT_HISTORY&conversationId=ONXMAIN. The main content area displays a document titled 'La théorie de la relativité générale'. The document has a blue header bar with tabs: 'Résumé' (selected), 'Modification', 'Fichiers', 'Publication', 'Relations', 'Workflow', 'Mes notifications', 'Commentaires', and 'Historique'. Below the tabs, there are two sub-tabs: 'Prévisualisation' and 'Administration'. A sidebar on the left shows a tree structure with 'ICI55-01' expanded, and a 'Mes documents' section with 'Presse-papier' selected. The 'Presse-papier' section contains a message: 'Aucun document dans la sélection.' Below the document title, there's a section titled 'Journal des événements' with a table showing two entries:

Action enregistrée	Date	Heure	Utilisateur	Catégorie	Commentaire	État dans le cycle de vie
Modification	24/02/10	15:05	IC155-01	Document	Réécriture du chapitre 1.	En projet
Création	24/02/10	15:03	IC155-01	Document		En projet

Below the journal, there's a section titled 'Versions archivées' with a table showing one entry:

Version	Action
1.0	<button>Restaurer</button> <button>Consulter la version archivée</button> <button>Supprimer</button>

At the bottom of the page, there are copyright notices: 'Copyright© 2006-2009 Nuxeo. Visit [nuxeo.com](#) | Get support | Join the community | QuickStart'. There's also a language selection dropdown: 'Choisissez votre langue :

Historique, consultation et restauration (Nuxéo-DM)

ii Historisation



Définition : Historisation automatique

Fonction créant une nouvelle version automatiquement suivant un événement prédéfini du système, par exemple à chaque enregistrement du fichier.



Exemple : MediaWiki

Les Wiki fonctionnent en général en historisation automatique, chaque fois qu'un utilisateur enregistre la page, la précédente est préalablement historisée. Les versions sont identifiées par leur date d'enregistrement.

Les pages historisées peuvent être consultées en lecture seule. Le système permet également de défaire les versions de la plus récente à la plus ancienne, pour ainsi retrouver un état antérieur.

Crozat page de discussion préférences liste de suivi contributions déconnexion

Modification de MaPage

You have followed a link to a page that does not exist or has been deleted.

To create this page, enter your text in the box below (you can consult the [page d'aide](#) for more information).

If you arrived here by mistake, click the **return** button in your browser.

=Chapitre 1=

Ut accumsan pede quis mauris. Quisque eleifend convallis urna. Mauris mauris arcu, cursus sed, viverra at, pellentesque sit amet, mi. Nullam dapibus neque vel ante. Morbi libero pede, aliquet et, hendrerit et, suscipit molestie, elit. Vestibulum nulla sapien, convallis eu, varius eget, blandit eget, tortor Etiam consecteturt accumsan urna. Sed orci. Donec consequat mattis nisi. Aenean ultricies neque sed odio. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam urna metus, commodo non, tincidunt vitae, lobortis nec, est.

=Chapitre 2=

Quisque ullamcorper sollicitudin quam. Maecenas dignissim, sapien a ornare varius, felis lorem lobortis dolor, ac varius turpis lorem in lorem. Donec auctor semper enim. Praesent vestibulum.

Création d'une page (MediaWiki)

Cours

The screenshot shows a MediaWiki interface. At the top, there is a navigation bar with links for 'Édition', 'Affichage', 'Historique', 'Marque-pages', 'Outils', and 'Aide'. Below the bar, the URL is http://scenari-platform.org/ici55/wiki/MaPage. The user 'Crozat' is logged in. The main content area has a header 'MaPage'. Below it, there are two sections: 'Chapitre 1' and 'Chapitre 2'. The 'Chapitre 1' section contains the text: 'Ut accumsan pede quis mauris. Quisque eleifend convallis urna. Mauris mauris arcu, cursus sed, viverra at, pellentesque sit amet, mi. Nullam dapibus neque vel ante. Morbi libero pede, aliquet et, hendrerit et, suscipit molestie, elit. Vestibulum nulla sapien, convallis eu, varius eget, blandit eget, tortor Etiam consecteturtuer accumsan urna. Sed orci. Donec consequat mattis nisi. Aenean ultricies neque sed odio. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam urna metus, commodo non, tincidunt vitae, lobortis nec, est.' The 'Chapitre 2' section contains the text: 'Quisque ullamcorper sollicitudin quam. Maecenas dignissim, sapien a ornare varius, felis lorem lobortis dolor, ac varius turpis lorem in lorem. Donec auctor semper enim. Praesent vestibulum.' On the left side, there is a sidebar with a logo for 'utc Université de Technologie Compiegne', a 'navigation' menu with links like 'Accueil', 'Communauté', 'Actualités', etc., a 'rechercher' search bar, and a 'boîte à outils' toolbox with icons for 'Pages liées', 'Suivi des pages liées', 'Téléverser un fichier', and 'Pages spéciales'.

Visualisation de la version courante (MediaWiki)

The screenshot shows the same MediaWiki interface as above, but in edit mode. The URL is now http://scenari-platform.org/ici55/index.php?title=MaPage&action=edit. The user 'Dupont' is logged in. The main content area has a header 'Modification de MaPage'. Below it, there is a rich text editor toolbar with buttons for bold, italic, underline, etc. The edit text area contains the same text as the previous screenshot: 'Chapitre 1', 'Chapitre 2', and 'Chapitre 3'. The text is identical to the visualized version. The sidebar on the left is identical to the previous screenshot.

Modification d'une page (MediaWiki)

The screenshot shows a MediaWiki interface with the following details:

- Header:** Fichier, Edition, Affichage, Historique, Marque-pages, Outils, Aide. The URL is http://scenari-platform.org/utcSS/wiki/MaPage.
- User Information:** Dupont (User), page de discussion, préférences, liste de suivi, contributions, déconnexion.
- Breadcrumbs:** page (highlighted), discussion, modifier, historique, renommer, suivre.
- Main Content:** The page title is MaPage. It contains three sections: **Chapitre 1**, **Chapitre 2**, and **Chapitre 3**. Each section has a [modifier] link at the end.
- Text Content:**
 - Chapitre 1:** Ut accumsan pede quis mauris. Quisque eleifend convallis urna. Mauris mauris arcu, cursus sed, viverra at, pellentesque sit amet, mi. Nullam dapibus neque vel ante. Morbi libero pede, aliquet et, hendrerit et, suscipit molestie, elit. Vestibulum nulla sapien, convallis eu, varius eget, blandit eget, tortor Etiam consecetur accumsan urna. Sed orci. Donec consequat mattis nisi. Aenean ultricies neque sed odio. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam urna metus, commodo non, tincidunt vitae, lobortis nec, est.
 - Chapitre 2:** Quisque ullamcorper sollicitudin quam. Maecenas dignissim, sapien a ornare varius, felis lorem lobortis dolor, ac varius turpis lorem in lorem. Donec auctor semper enim. Praesent vestibulum.
 - Chapitre 3:** Ut accumsan pede quis mauris. Quisque eleifend convallis urna. Mauris mauris arcu, cursus sed, viverra at, pellentesque sit amet, mi. Nullam dapibus neque vel ante. Morbi libero pede, aliquet et, hendrerit et, suscipit molestie, elit. Vestibulum nulla sapien, convallis eu, varius eget, blandit eget, tortor Etiam consecetur accumsan urna. Sed orci.
- Sidebar:** navigation (Accueil, Communauté, Actualités, Modifications récentes, Page au hasard, Aide), rechercher (Lire, Rechercher), boîte à outils (Pages liées, Suivi des pages liées, Téléverser un fichier, Pages spéciales).

Visualisation de la nouvelle version courante (MediaWiki)

Cours

The screenshot shows a MediaWiki interface for the page "MaPage". At the top, there is a navigation bar with links for "Dupont", "page de discussion", "préférences", "liste de suivi", "contributions", and "déconnexion". Below the navigation bar, the title "Historique des versions de « MaPage »" is displayed. A link "Voir les opérations sur cette page" is present. A search bar and a "navigation dans l'historique" section are also visible. The main content area displays two recent changes made by users Dupont and Crozat on February 24, 2010, at 11:04 and 11:02 respectively. The changes involve discussions and contributions. A legend indicates that "(actu)" means difference from the current version and "(diff)" means difference from the previous version. A note mentions that the page was created with the text "Chapitre 1 = Ut accumsan pede quis mauris. Quisque eleifend convallis urna. Mauris mauris arcu, cursus sed, viverra at, pellentesque sit amet, mi. Nullam dapibus neque vel a...". A "Comparer les versions sélectionnées" button is shown.

Consultation de l'historique (MediaWiki)

The screenshot shows a MediaWiki interface for the page "MaPage". At the top, there is a navigation bar with links for "Dupont", "page de discussion", "préférences", "liste de suivi", "contributions", and "déconnexion". Below the navigation bar, the title "MaPage" is displayed. A link "Version du 24 février 2010 à 11:02 par Crozat (discuter | contributions)" is present. Below the title, the content of the page is shown: "Chapitre 1" followed by a large block of Latin text. A link "Chapitre 2" is also present. The sidebar on the left contains a "navigation" section with links to "Accueil", "Communauté", "Actualités", "Modifications récentes", "Page au hasard", and "Aide". It also includes a "rechercher" section with a search bar and buttons for "Lire" and "Rechercher". A "boîte à outils" section contains links for "Pages liées", "Suivi des pages liées", "RSS Atom", "Téléverser un fichier", and "Pages spéciales".

Visualisation d'une version antérieure (MediaWiki)

Historique des versions de « Isaac Asimov » (?)
Voir les opérations sur cette page

Navigation dans l'historique
A partir de l'année (et précédentes) : [] A partir du mois (et précédentes) : tous [] Lister

Outils externes : Détails des contributions - Rechercher l'auteur d'un passage de l'article - Statistiques : modifications - consultations - qui suit cette page ?
Discussions liées :

Légende : (actu) = différence avec la version actuelle, (diff) = différence avec la version précédente, m = modification mineure
(dernière page | première page) Voir (50 plus récentes | 50 plus anciennes) (20 | 50 | 100 | 250 | 500)

Comparer les versions sélectionnées

- # (actu) (diff) ○ 16 février 2010 à 11:20 Palamède (discuter | contributions) m (36 102 octets) (-La psychohistoire et le Cycle de Fondation) (défaire)
- # (actu) (diff) ○ 16 février 2010 à 11:15 62.100.133.105 (discuter | contributions) m (36 102 octets) (-Les robots) (défaire)
- # (actu) (diff) ○ 10 février 2010 à 19:31 Patangel (discuter | contributions) m (36 103 octets) (Wiki) (défaire)
- # (actu) (diff) ○ 10 février 2010 à 02:02 ArthurBot (discuter | contributions) m (36 059 octets) (robot Ajoute: war:Isaac Asimov) (défaire)
- # (actu) (diff) ○ 9 février 2010 à 22:35 StephenTitusG (discuter | contributions) m (36 038 octets) (Redondance) (défaire)
- # (actu) (diff) ○ 8 février 2010 à 18:08 195.75.187.6 (discuter) m (36 723 octets) (-Autres romans : liste redondante avec les romans de l'Empire) (défaire)
- # (actu) (diff) ○ 7 février 2010 à 04:40 Xobot (discuter | contributions) m (36 997 octets) (robot Ajoute: be:Айзек Азимов) (défaire)
- # (actu) (diff) ○ 7 février 2010 à 01:43 ArthurBot (discuter | contributions) m (36 966 octets) (robot Ajoute: ug:آیزاک آسیموو) (défaire)
- # (actu) (diff) ○ 2 février 2010 à 22:13 StephenTitusG (discuter | contributions) m (36 931 octets) (-Curiosités de ce cycle : Redondance avec la page du Cycle de David Starr) (défaire)
- # (actu) (diff) ○ 23 janvier 2010 à 21:43 Loveless (discuter | contributions) m (37 641 octets) (robot Modifie: sr:Ајзак Асимов; changement de type cosmétique) (défaire)
- # (actu) (diff) ○ 15 janvier 2010 à 16:50 83.206.226.2 (discuter) m (37 597 octets) (défaire)
- # (actu) (diff) ○ 15 janvier 2010 à 16:48 83.206.226.2 (discuter) m (37 596 octets) (défaire)
- # (actu) (diff) ○ 12 janvier 2010 à 16:21 Udufruhdu (discuter | contributions) m (37 597 octets) (enlève Catégorie:Polymathe supprimée en P&S) (défaire)
- # (actu) (diff) ○ 27 décembre 2009 à 20:59 Alexandre.jaborska (discuter | contributions) m (37 622 octets) (-Bibliographie : => changé en "biographies" car le contenu du chapitre donne des ouvrages sur la vie d'Asimov et non la liste de ses livres.) (défaire)
- # (actu) (diff) ○ 27 décembre 2009 à 01:16 Patangel (discuter | contributions) m (37 624 octets) (Ajout des œuvres principales dans l'Infobox) (défaire)
- # (actu) (diff) ○ 19 décembre 2009 à 23:07 Arkanos (discuter | contributions) m (37 701 octets) (LiveRC : Révocation des modifications de 92.133.86.99; retour à la version de GrouchoBot) (défaire)
- # (actu) (diff) ○ 19 décembre 2009 à 23:06 92.133.86.99 (discuter) m (37 704 octets) (défaire)
- # (actu) (diff) ○ 16 décembre 2009 à 18:59 GrouchoBot (discuter | contributions) m (37 701 octets) (robot Ajoute: fy:Isaac Asimov) (défaire)
- # (actu) (diff) ○ 16 décembre 2009 à 16:01 Patangel (discuter | contributions) m (37 681 octets) (Correction de liens pour L'homme bientenaire (nouvelle vs. recueil)) (défaire)
- # (actu) (diff) ○ 14 décembre 2009 à 16:36 Poulos (discuter | contributions) m (37 605 octets) (-Éducation et carrière : refec sur "apprit seul à lire" et "refusa de participer aux essais de la bombe A") (défaire)
- # (actu) (diff) ○ 5 décembre 2009 à 22:12 Cymbella (discuter | contributions) m (37 583 octets) (cosmétique) (défaire)
- # (actu) (diff) ○ 27 novembre 2009 à 15:45 79.86.32.120 (discuter) m (37 577 octets) (-Biographie) (défaire)
- # (actu) (diff) ○ 27 novembre 2009 à 15:45 79.86.32.120 (discuter) m (37 576 octets) (-Biographie) (défaire)

Exemple d'historique (Wikipédia)

iii Comparaison et fusion de versions



Définition : Comparaison de version

Fonction permettant de aux utilisateurs de visualiser simultanément deux versions et les différences entre ces deux versions.

Il est ainsi plus facile de savoir ce qui a changé entre deux versions.



Exemple : MediaWiki

Comparaison des versions (MediaWiki)



Définition : Fusion

Certains systèmes proposent également des fonctions de fusion permettant de choisir dans chaque version ce que l'on souhaite garder ou non, pour créer une nouvelle version résultant de cette fusion.

Synonyme : *merge*

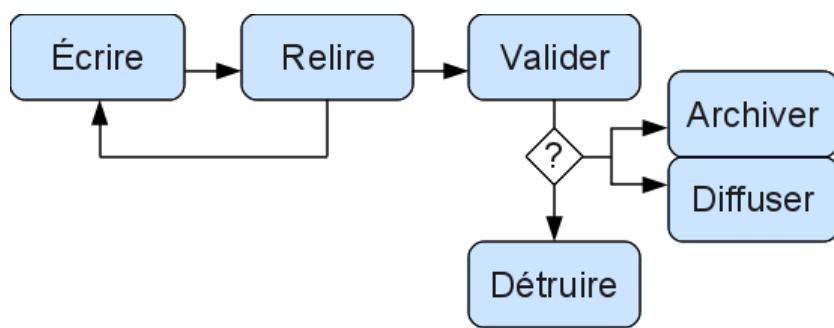
c) Workflow



Définition : Workflow documentaire

Un workflow documentaire est une gestion informatique d'un processus métier lié à un document.

La plupart des systèmes de GED intègrent des moteurs de workflow qui permettent de modéliser des processus complexes.



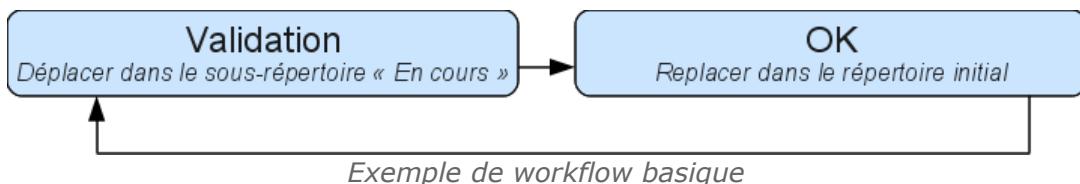
Principe du workflow



Exemple : Définition d'un workflow simple (Alfresco)

Soit un workflow d'approbation très simple :

1. Les documents peuvent être envoyés en validation, ils sont alors déplacés dans un sous-répertoire spécifique
2. Une fois relus et validés, éventuellement modifiés, il peuvent être replacés dans leur répertoire d'origine



Workflow simple (Alfresco) : Les répertoires



Workflow simple (Alfresco) : Définition de la première action



Workflow simple (Alfresco) : Définition de la seconde action

Soit un espace (répertoire) principal nf29 contenant un sous-espace En cours que l'on destine à recevoir les contenus en cours de validation.

L'action Validation est définie pour l'espace nf29 comme l'action de déplacer un document vers l'espace En cours. D'autres actions pourraient être faites concomitamment : comme modifier des métadonnées, envoyer une alerte à l'utilisateur en charge de la validation, etc.

De la même façon l'action OK est définie pour l'espace En cours comme l'action de déplacer un document vers nf29.



Exemple : Utilisation d'un workflow simple (Alfresco)

Quand un document est dans l'espace nf29 il dispose de l'action supplémentaire Validation qui l'envoie dans l'espace En cours.

Workflow simple (Alfresco) : Exécution de la première action

Quand il est dans l'espace En cours il dispose de l'action supplémentaire OK qui le renvoie dans l'espace d'origine.

Workflow simple (Alfresco) : Exécution de la seconde action

d) Transactions



Définition : Transaction

Une transaction est un mécanisme qui permet de gérer la concurrence des accès au contenu.

Elle permet de gérer le cas où deux personnes (ou plus) cherchent à modifier le même fichier en même temps.



Définition : Verrouillage

Le verrouillage est une technique de gestion de transaction, qui consiste à interdire l'accès en écriture à deux utilisateurs en même temps. Le premier utilisateur qui souhaite accéder au contenu en écriture verrouille le fichier, et les suivants doivent attendre le déverrouillage pour faire également des modifications.

	Demande d'écriture	Demande de lecture
Verrou absent	Acceptée (pose d'un verrou)	Acceptée

		Cours
	Demande d'écriture	Demande de lecture
Verrou déjà posé	Rejetée (attente)	Acceptée

Tableau 4 Autorisation d'accès à un document



Exemple : Principe du verrouillage manuel

Le verrouillage est fait directement par un utilisateur. Il a le choix de verrouiller ou non, s'il ne le fait pas la concurrence n'est pas gérée. L'utilisateur doit également penser à déverrouiller lorsque le verrou n'est plus utile.

The screenshot shows the Nuxeo Document Management (DM) interface. At the top, there's a navigation bar with links like 'Fichier', 'Edition', 'Affichage', 'Historique', 'Marque-pages', 'Outils', and 'Aide'. Below the bar, the URL 'http://172.21.130.224:8080/nuxeo/view_documents.faces' is visible. The main content area shows a document titled 'Albert Einstein'. The document details include:

- Résumé**: Document verrouillé !
- Fichiers**: Albert Einstein.txt (1 kB)
- Méta-données**:

Auteur	IC155-01	Format	
Cycle de vie	En projet	Langue	
Version	1.0	Source	
Dernière modification	24/02/10	Droits	

A message box on the right states: "Ce document est verrouillé | Déverrouiller" followed by "par IC155-01, le Feb 24, 2010".

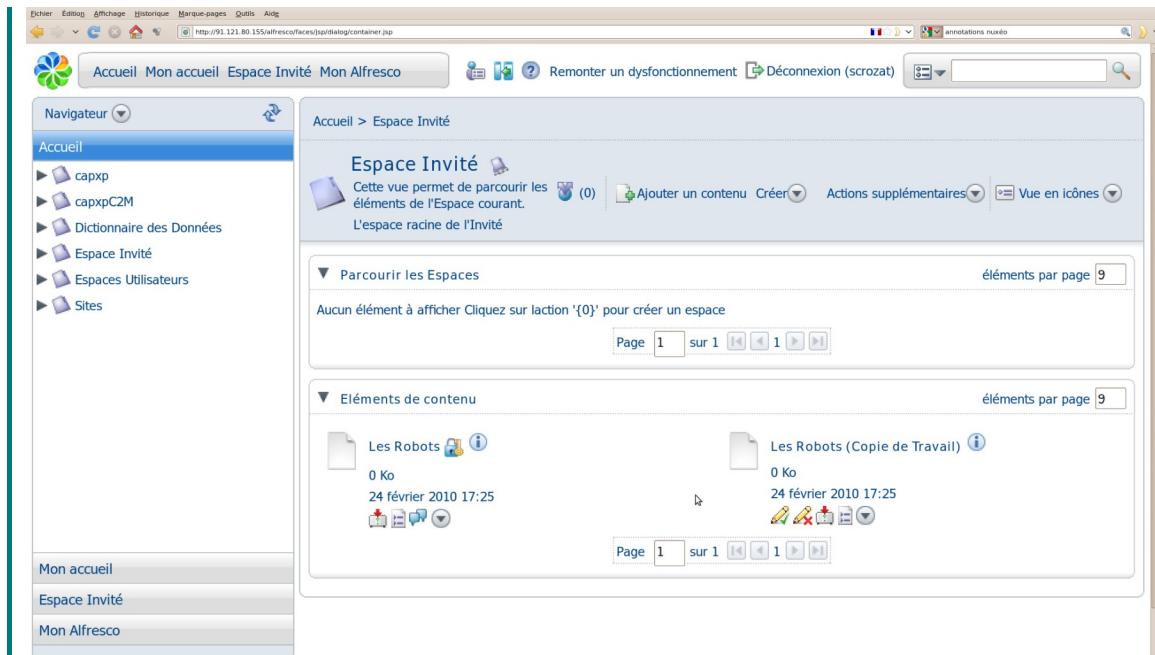
Exemple de gestion de transactions par verrouillage manuel (Nuxéo-DM)



Exemple : Principe du check-in / check-out

La plupart des systèmes de GED, proposent le mécanisme suivant, un peu plus automatisé :

- Lorsqu'un utilisateur souhaite modifier un document, il doit effectuer un *check-out*, qui verrouille le document et crée une **copie de travail**.
- Cette copie de travail est privée (seul lui peut la voir), les autres utilisateurs ne voit que le document original verrouillé
- Il peut ainsi modifier cette copie sans risque de conflit
- Lorsqu'il souhaite mettre à jour le document, il effectue un *check-in*, qui libère le document après l'avoir remplacé par la copie de travail (qui est supprimée)



Exemple de gestion de transactions par check-in / check-out (Alfresco)



Exemple : Principe du verrouillage automatique

Le verrouillage automatique pose un verrou dès qu'un utilisateur souhaite effectuer une modification (ouverture en édition) et le libère dès que le document est relâché (fermeture). C'est le principe général des outils bureautique traditionnel utilisés en réseau (pas des outils bureautique en ligne qui en général ne proposent pas de verrouillage).



Remarque : Le problème des transactions longues

Si un utilisateur garde un fichier verrouillé pendant longtemps, les autres utilisateurs peuvent être bloqués dans leur processus de travail.



Remarque : L'historisation dans les wikis

L'historisation dans les wikis est une façon de gérer les transactions, en reportant la gestion des conflits *a posteriori* :

1. Si deux utilisateurs A et B éditent le même contenu en même temps, c'est le dernier qui enregistre qui "à raison", puisque son enregistrement écrase le précédent. Si c'est B qui enregistre le dernier, il annule donc les modifications de A.
2. Mais comme le contenu de A a été historisé, les modifications de A restent enregistrées dans une version.
3. Elles peuvent donc être réinjectées dans le contenu *a posteriori*, une fois le conflit détecté.



Remarque : La simultanéité dans les outils bureautiques en ligne

Les outils bureautique en ligne (comme les Google Apps) se basent sur la simultanéité de l'écriture. À chaque enregistrement, retour chariot, voire à chaque frappe sur le clavier, le document commun est mis à jour sur le serveur. Ainsi chaque co-auteur peut voir en temps réel ce que les autres écrivent, et en conséquence s'ajuster.

e) Commentaires



Définition : Paratexte

Le paratexte désigne l'ensemble des informations qui traite d'un document, sans faire partie du document lui même. On peut citer par exemple les annotations ou les discussions que les utilisateurs peuvent faire autour d'un document.

Les fonctions de commentaires permettent de gérer le paratexte afin de le contextualiser et de le mémoriser.

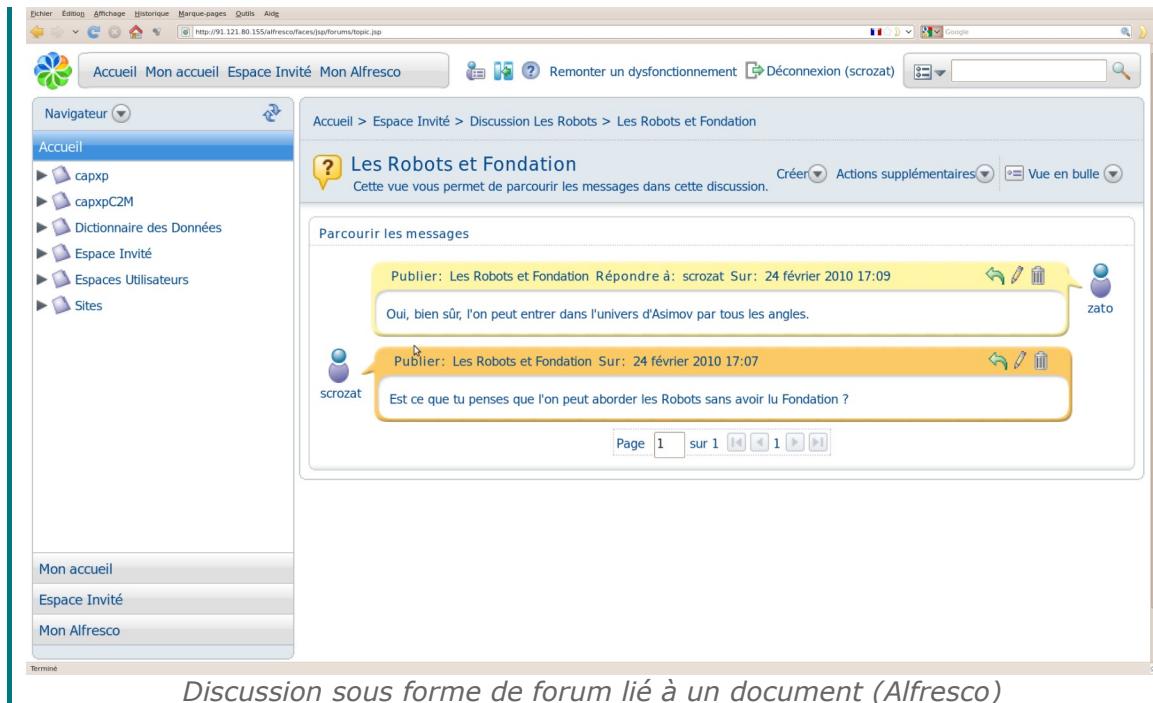


Exemple : GED

Dans les systèmes de GED, il est généralement possible d'associer un forum de discussion à chaque document. Cela permet aux utilisateurs d'échanger dans le contexte de ce document.

Discussion sous forme de forum lié à un document (Nuxéo-DM)

Cours



The screenshot shows a web browser window with the URL <http://91.121.80.155/alfresco/faces/jsp/forum/topic.jsp>. The page title is "Accueil > Espace Invité > Discussion Les Robots > Les Robots et Fondation". The main content area displays a discussion titled "Les Robots et Fondation". Two messages are visible:

- Message 1: "Publier: Les Robots et Fondation Répondre à: scrozat Sur: 24 février 2010 17:09" - "Oui, bien sûr, l'on peut entrer dans l'univers d'Asimov par tous les angles."
- Message 2: "Publier: Les Robots et Fondation Sur: 24 février 2010 17:07" - "Est ce que tu penses que l'on peut aborder les Robots sans avoir lu la Fondation ?"

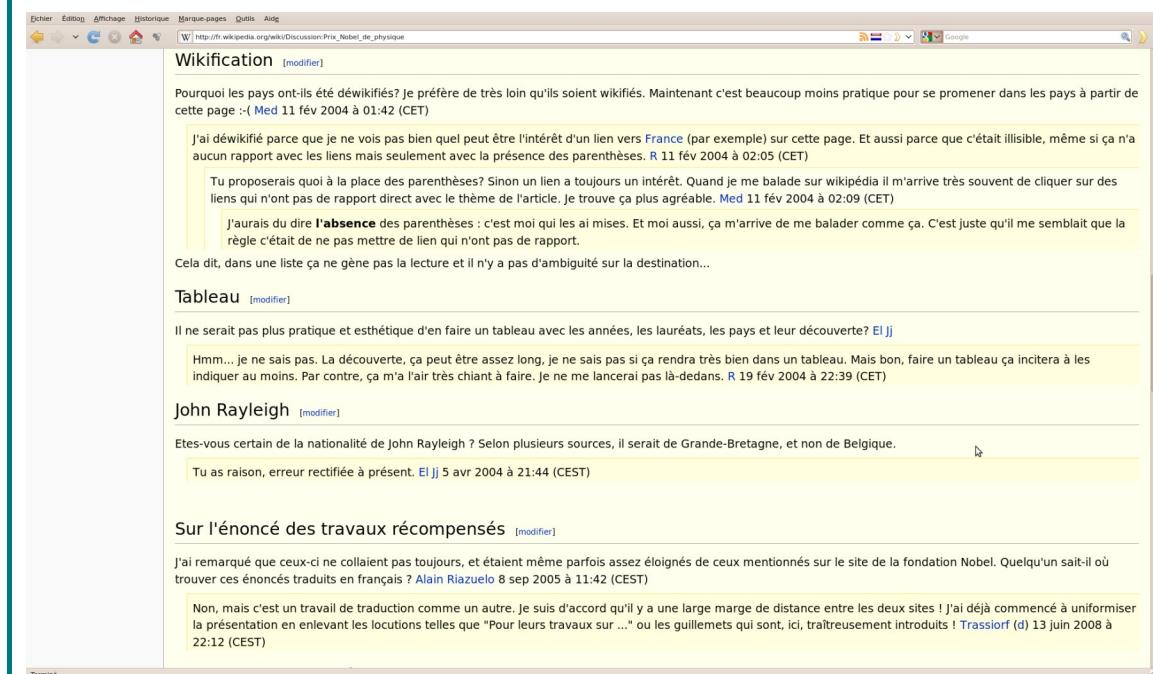
The sidebar on the left contains a "Navigateur" section with links to various Alfresco components like capxp, capxpC2M, Dictionnaire des Données, Espace Invité, Espaces Utilisateurs, and Sites.

Discussion sous forme de forum lié à un document (Alfresco)



Exemple : Wiki

Sur les Wiki les discussions sont parfois de simples pages Wiki comme les autres, dont le contenu porte sur une autre page.



The screenshot shows a web browser window with the URL http://fr.wikipedia.org/wiki/Discussion:Prix_Nobel_de_physique. The page title is "Wikification [modifier]". A discussion is taking place about the use of parentheses in links:

"Pourquoi les pays ont-ils été déwikifiés? Je préfère de très loin qu'ils soient wikifiés. Maintenant c'est beaucoup moins pratique pour se promener dans les pays à partir de cette page :-(Med 11 fév 2004 à 01:42 (CET)"

"J'ai déwikifié parce que je ne vois pas bien quel peut être l'intérêt d'un lien vers France (par exemple) sur cette page. Et aussi parce que c'était illisible, même si ça n'a aucun rapport avec les liens mais seulement avec la présence des parenthèses. R 11 fév 2004 à 02:05 (CET)"

"Tu proposerais quoi à la place des parenthèses? Sinon un lien a toujours un intérêt. Quand je me balade sur wikipédia il m'arrive très souvent de cliquer sur des liens qui n'ont pas de rapport direct avec le thème de l'article. Je trouve ça plus agréable. Med 11 fév 2004 à 02:09 (CET)"

"J'aurais du dire l'**absence** des parenthèses : c'est moi qui les ai mises. Et moi aussi, ça m'arrive de me balader comme ça. C'est juste qu'il me semblait que la règle c'était de ne pas mettre de lien qui n'ont pas de rapport.

Cela dit, dans une liste ça ne gène pas la lecture et il n'y a pas d'ambiguité sur la destination..."

The page also includes sections for "Tableau" and "John Rayleigh", each with its own discussion history.

Discussion sous forme de page Wiki (Wikipédia)



Exemple : Annotation

The screenshot shows the Nuxeo Document Management (DM) interface. At the top, there's a navigation bar with links like 'Fichier', 'Edition', 'Affichage', 'Historique', 'Marque-pages', 'Outils', and 'Aide'. Below the navigation bar, the URL is visible: http://172.21.30.224:8080/nuxeo/nxdoc/default/1963819f-caef-4f72-b769-c40ac7ca3746/view_documents?tabId=TAB_PREVIEW&conversationId=ONXMAIN2. The main content area displays a document titled 'Albert Einstein'. The document content is: 'Lorem ipsum dolor sit amet, consectetur adipisic elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.' There are several annotations overlaid on this text. One annotation from '24/02/10 ICI55-01' highlights the first sentence with yellow text and a blue border. Another annotation from '24/02/10 ICI55-01' highlights the last sentence with yellow text and a blue border. A third annotation from '24/02/10 • ICI55-01' highlights the entire paragraph with yellow text and a blue border. The annotations contain text such as 'Peut-on vraiment parler de cela ?' and 'n ntre'. The bottom of the interface shows a toolbar with icons for search, advanced search, and other functions.

Annotation des documents (Nuxéo-DM)

f) Information collaborative

Dans les systèmes d'écriture collaborative, il devient possible d'informer chaque utilisateur de l'activité des autres utilisateurs.

Cela permet à chacun de savoir ce que les autres font en relation avec son propre travail, d'éviter de refaire le même travail plusieurs fois, de mieux planifier son travail, etc.



Exemple : Exemple d'information

- Le document dont le titre est ... a été créé par l'utilisateur ...
- Le document ... a été modifié le ...
- Le document ... est en cours de modification par ... et ...
- Le document ... a été commenté par ...

- Une nouvelle version du document ... a été ajoutée le ... par ...
- etc.



Exemple

Exemple de tableau de bord (projet CAP-XP, UTC)

3.Complément : Éléments commerciaux et industriels

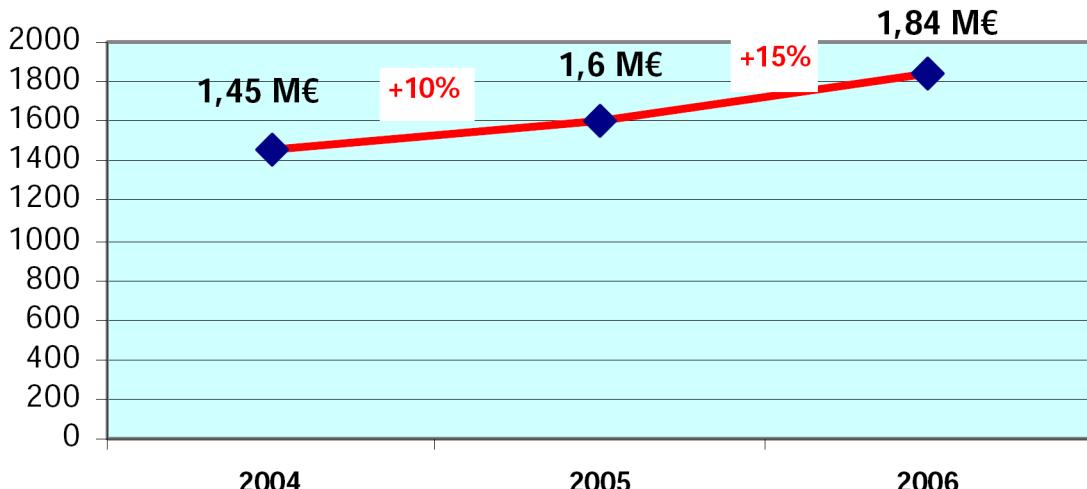
Marché de l'ECM

Le marché mondial de l'édition de logiciels ECM représente environ 3 milliards de dollars (3,2M\$ en 2005 selon IDC et 2,6 M\$ en 2006 selon Gartner). Le marché global intégrant les prestations de service est de l'ordre de 10 milliards de dollars. La progression annuelle est de l'ordre de 10% (Serda Lab, 2007 [Serda Lab 2007]).

« Globalement, le marché de l'information non-structurée représente selon les principaux analystes actuellement un chiffre d'affaires mondial de près de 3 Milliards de dollars en 2006, et uniquement pour le marché des éditeurs de logiciels spécialisés dans l'ECM. Il devrait croître de plus de 10% chaque année jusqu'en 2010, ce qui en fait un des marchés les plus dynamiques du secteur des Technologies de l'Information. Par ailleurs et pour estimer complètement la valeur globale du marché, ce chiffre doit être au moins triplé pour couvrir l'ensemble des prestations de service et de conseil » (Couillault & Le Foll, 2007 [Couillault et Le Foll 2007]).

Le marché global français (édition et prestations) représente 1,84 milliards d'euros en 2006 en croissance de 15% (Serda Lab, 2007). Il se repartit ainsi :

- Intégrateurs (38%) : 700 millions d'euros (+17% par rapport à 2005)
- Éditeurs (30%) : 535 millions d'euros (+9%)
- Prestataires (26%) : 480 millions d'euros
- Conseil (2%) : 41 millions d'euros (+16%)



Évolution du marché Documents et Flux Numériques en France (Serda Lab, 2007)

Exemple d'ECM Open Source : Alfresco

Leader des ECM Open Source :

- Créé en 2005 par des anciens de Documentum
- 10 M€ de CA
- 100 personnes

Références :

- Tutoriel [End-User :
http://freefr.dl.sourceforge.net/sourceforge/alfresco/Alfresco-Tutorial-1.2.pdf](http://freefr.dl.sourceforge.net/sourceforge/alfresco/Alfresco-Tutorial-1.2.pdf)
- Wiki (communauté) : http://wiki.alfresco.com/wiki/Main_Page

Exemple d'ECM Open Source Français : Nuxéo

Leader français des ECM Open Source (solution numéro 2 au plan international derrière Alfresco) :

- Créé en 2000 :
 - Nuxéo CPS basé sur Zope/Python
 - En 2007 : Nuxéo 5, réécrit en Java
- 4 M€ de CA
- 40 personnes

« Founded in 2000, Nuxeo SAS is part of the "second wave" of open source companies, and focuses on developing and supporting applications, instead of system software or development tools. By entering the ECM field early in 2002, Nuxeo has established itself as the leader of open source ECM, with customers for critical projects in the Government, Energy and Finance sectors. Nuxeo currently has 40 employees, about half of them developers. <http://doc.nuxeo.org/5.1/books/nuxeo-book/html/preface.html#d1801e18> »

Références :

- User Guide : <http://doc.nuxeo.org/5.2/books/nuxeo-user-guide/html/>
- Wiki (communauté) : <http://doc.nuxeo.org/xwiki/bin/view/Main/>
- Interfaces de développement de Nuxéo :
<http://www.nuxeo.org/static/book-draft/> (tutoriel)
<http://doc.nuxeo.org/5.1/books/nuxeo-book/html/> (Référence HTML)
<http://doc.nuxeo.org/5.1/books/nuxeo-book/pdf/nuxeo-book.pdf> (Référence PDF)

J.Langages de publication : SMIL et FO

1.Introduction à SMIL

a)Qu'est ce que SMIL ?



Définition

SMIL (pour *Synchronised Multimedia Integration Language*, prononcé « smile » à l'anglaise, pour "sourire") est un langage XML de présentation multimédia fondé sur la synchronisation de contenus spatiaux et temporels.

SMIL est un standard du W3C.



Remarque : SMIL est un langage de publication

SMIL permet de spécifier des présentations multimédias : c'est un langage de de mise en forme spatiale et temporelle.

En tant quel, les présentations SMIL sont engendrées la plupart du temps par une transformation XSLT à partir de contenus XML et de ressources binaires (vidéos, sons, photos).



Remarque : Pour comprendre plus que pour faire...

SMIL est un concurrent standard et déclaratif de Flash. Mais SMIL reste peu utilisé dans les faits, peu d'outils le lisent, aucun ne l'exécute parfaitement, et le succès de Flash n'a pas favorisé son développement.

SMIL a probablement peu d'avenir, mais reste intéressant comme approche pédagogique de la synchronisation de contenus multimédias.



Complément : Players SMIL : Real et Ambulant

- *Real Player*¹³, qui ne couvre pas l'ensemble du standard
- *AMBULANT Open SMIL Player*¹⁴, une alternative open source et plus standard



Complément : LimSee2 : Un éditeur WYSIWYG pour SMIL

<http://limsee2.gforge.inria.fr>



Complément : Une alternative en construction...

HTML Timing, développé par l'INRIA dans le cadre du projet de recherche C2M, a

13 - <http://www.real.com/>

14 - <http://www.ambulantplayer.org/>

pour objectif d'étendre HTML5 avec des balises SMIL pour intégrer la synchronisation, notamment de ressources audiovisuelles. Il permet également de définir des *Timesheets* (feuilles de temps), qui dans l'esprit des feuilles de styles, permettent de mutualiser les scénarialisations récurrentes.

<http://labs.kompozer.net/timesheets/>

b) Structure de base d'un contenu SMIL



Attention : Simplification volontaire

Les structures présentées sont simplifiées à des fins pédagogiques afin de permettre une appréhension rapide des principes du langages.

Il existe de nombreux autres éléments dans le standard, en conséquence les schémas proposés ne sont pas les schémas standards, mais ils permettent de créer des documents valides par rapport aux schémas standards (les schémas proposés sont inclus dans les schémas standards).



Syntaxe : smil

Un document SMIL comprend une entête, `<head>` et un corps `<body>`.

```
<!ELEMENT smil (head?,body)>
```

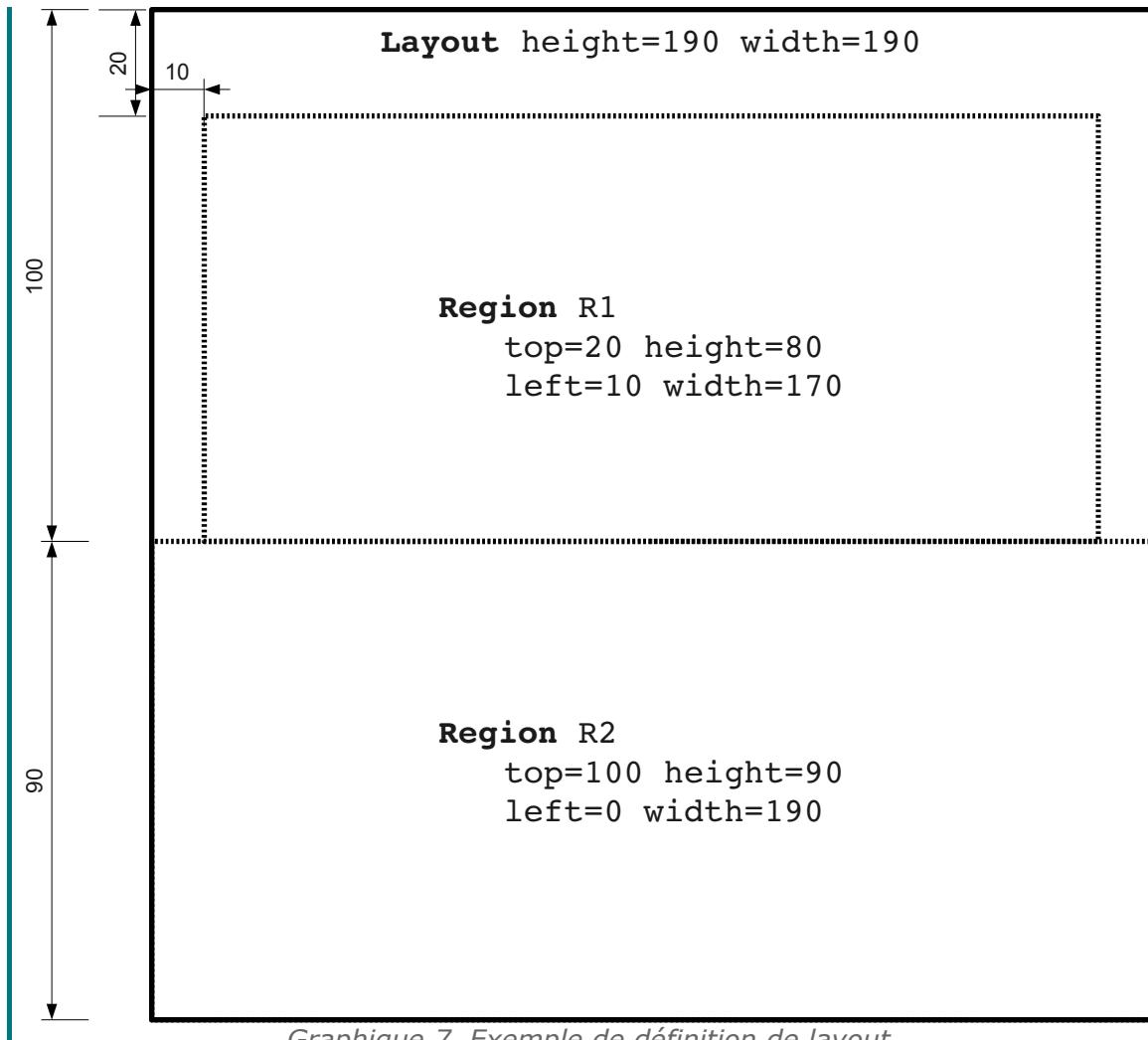


Syntaxe : head

L'entête permet (notamment) de déclarer :

- des méta-données `<meta>`
- une structure générale de l'écran de présentation découpé en zones `<layout>`

```
<!ELEMENT head (meta*,layout?)>
<!ELEMENT meta EMPTY>
<!ATTLIST meta
  name (title | author) #REQUIRED
  content CDATA #REQUIRED>
<!ELEMENT layout (root-layout, region*)>
<!ELEMENT root-layout EMPTY>
<!ATTLIST root-layout
  width CDATA #REQUIRED
  height CDATA #REQUIRED>
<!ELEMENT region EMPTY>
<!ATTLIST region
  id ID #REQUIRED
  width CDATA #REQUIRED
  height CDATA #REQUIRED
  left CDATA #IMPLIED
  top CDATA #IMPLIED>
```



Graphique 7 Exemple de définition de layout



Syntaxe : body

Le corps permet de déclarer une séquence `<seq>` d'éléments parallèles `<par>` : Chaque `<par>` est un ensemble de ressources synchronisées (textes, images, sons, vidéos, etc.).

Chaque ressource est associée à :

- une région `region` du *layout*
- une durée `dur` en secondes
- éventuellement une date de début `begin` en seconde

```

<!ELEMENT body (seq)>
<!ELEMENT seq (par+)>
<!ELEMENT par (text | img | audio | video)*>
<!ELEMENT text EMPTY>
<!ATTLIST text
src CDATA #REQUIRED
region CDATA #REQUIRED
dur CDATA #REQUIRED>
<!ELEMENT img EMPTY>
<!ATTLIST img
src CDATA #REQUIRED
region CDATA #REQUIRED
dur CDATA #REQUIRED>

```

```
<!ELEMENT audio EMPTY>
<!ATTLIST audio
src CDATA #REQUIRED
region CDATA #REQUIRED
dur CDATA #REQUIRED>
<!ELEMENT video EMPTY>
<!ATTLIST video
src CDATA #REQUIRED
region CDATA #REQUIRED
dur CDATA #REQUIRED>
```



Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<smil>
  <head>
    <meta name="title" content="Titre de mon projet"/>
    <meta name="author" content="Stéphane Crozat"/>
  <layout>
    <root-layout width="600" height="600"/>
    <region id="image" width="550" height="400" left="25"
top="0"/>
    <region id="texte" width="600" height="200" left="0"
top="400"/>
  </layout>
  </head>
  <body >
    <seq>
      <par>
        
        <text src="texte01.rt" region="texte" dur="2"/>
      </par>
      <par>
        
        <text src="texte02.rt" region="texte" dur="2"/>
      </par>
    </seq>
  </body>
</smil>
```



Complément : Pour aller plus loin...

"Publier à partir de XML : SMIL" (par Bruno Bachimont) (cf. "Publier à partir de XML : SMIL" (par Bruno Bachimont))



Complément : Tutoriels en ligne

<http://real-and-smil.com/tutorialsmil.php>

c) Texte dans SMIL (RealText)

Les textes utilisables dans SMIL et lisibles dans Real PLayer sont écrits selon le format Real Text (rt).

Ce format est un "sous" XML, c'est à dire un langage XML qui doit néanmoins respecter des contraintes de syntaxe supplémentaires par rapport à XML.



Attention : Un "sous" XML

1. Les texte RT ne doivent pas avoir de prologue <?xml?>, ils commencent donc directement par l'élément racine.
2. Les espaces sont pris en compte, l'indentation n'est donc pas neutre pour le mise en forme



Exemple : Texte RT

```
<window height="200" width="600" bgcolor="yellow" >
<font face="arial" color="blue">
<br/>
Première ligne<br/>
Seconde ligne
</font>
</window>
```



Syntaxe : Texte temporisé

Il est possible de temporiser l'affichage du texte en encadrant ce texte dans un élément <time begin="x">.



Exemple : Texte RT temporisé

```
<window height="200" width="600" bgcolor="yellow" >
<!-- NB : Durée du texte = 2--&gt;
&lt;font face="arial" color="blue"&gt;
&lt;br/&gt;
Première ligne&lt;br/&gt;
Début de la seconde ligne &lt;time begin="1"&gt;suite de la seconde
ligne &lt;br/&gt;&lt;/time&gt;
&lt;time begin="2"&gt;Troisième ligne&lt;/time&gt;
&lt;/font&gt;
&lt;/window&gt;</pre>
```



Complément: Pour aller plus loin

<http://www17.real.com/help/library/guides/realttext/realttext.htm>

2. Introduction à XSL-FO

a) Qu'est ce que XSL-FO ?



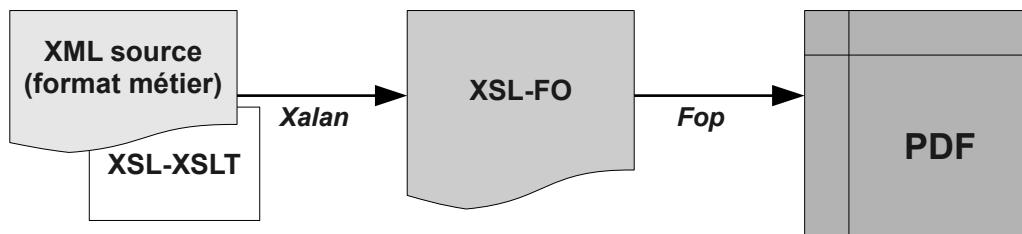
Définition : XSL-FO

XSL-FO (on parle généralement plus simplement de FO) est la seconde partie du langage XSL, qui définit un langage de présentation en vue de la publication de documents imprimables. C'est un langage XML permettant d'obtenir un fichier PDF une fois traité par un moteur de rendu FO.



Remarque : FO est un langage de publication

Le langage FO n'est pas destiné à être écrit manuellement, mais plutôt à être obtenu automatiquement après une transformation XSL-XSLT.



Graphique 8 Chaîne de publication FO (Technologies Apache)



Complément : Moteurs FO

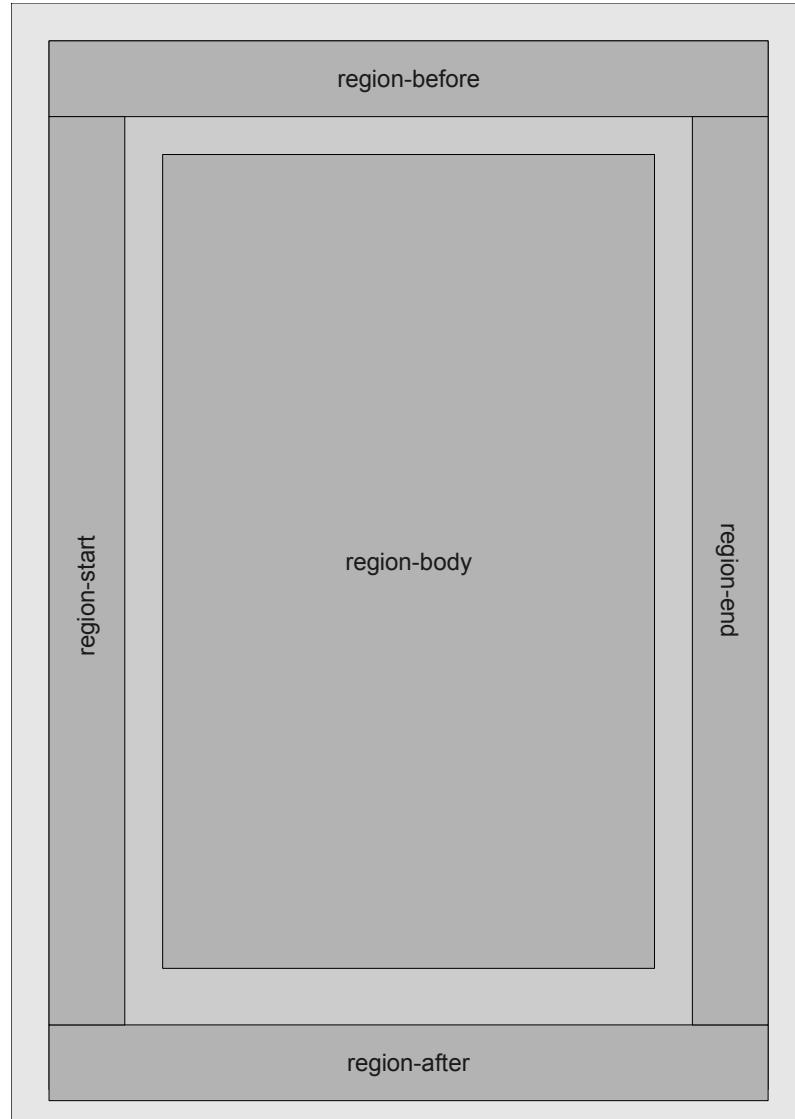
- Apache FOP¹⁵, moteur *open source*, ne supportant pas la totalité du standard
- XEP¹⁶, moteur commercial, plus complet

b) Exemple XSL-FO

```

<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4" page-height="297mm"
page-width="210mm" margin-top="1cm" margin-bottom="1cm"
margin-left="1cm" margin-right="1cm">
      <fo:region-body margin="3cm"/>
      <fo:region-before extent="2cm"/>
      <fo:region-after extent="2cm"/>
      <fo:region-start extent="2cm"/>
      <fo:region-end extent="2cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        <fo:inline font-weight="bold">Hello world!</fo:inline>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
  
```

15 - <http://xmlgraphics.apache.org/fop/>
 16 - <http://www.renderx.com/>



Graphique 9 Exemple de structure de page FO



Complément : Pour aller plus loin

"Publier à partir de XML : Formatting Objects" (par Bruno Bachimont) (cf. "Publier à partir de XML : Formatting Objects" (par Bruno Bachimont))



Exercices

II

Questions théoriques ingénierie documentaire	233
Travaux pratiques chaînes éditoriales	235
Exercices XML	236
Exercices RelaxNG	238
Exercices DTD	242
Quiz DTD	245
Exercices XSLT et XPath	247
Quiz XSLT et XPath	255
Exercices UML, passage RelaxNG, XSLT	258
Travaux pratiques Wiki	264
Question théorique indexation	265
Travaux pratiques ECM	265
Exercices SMIL et FO	268

A. Questions théoriques ingénierie documentaire

1. Exercice rédactionnel

[60 minutes]

Chaque question sera traitée sur une demi-page environ. Vous mobiliserez vos connaissances théoriques (concepts clés du cours) ainsi que pratiques (exemples concrets). Vous veillerez particulièrement à la précision de vos énoncés et à ne pas effectuer de contre-sens ou d'erreur majeure. La qualité générale de la rédaction (et de l'orthographe) sera prise en compte.

Question 1

Pourquoi peut-on dire que la séparation entre le fond et la forme est un principe théoriquement faux, bien que mobilisé dans les chaînes éditoriales XML ?

Indice :

- *La connaissance n'existe qu'inscrite, donc il n'existe pas de fond indépendant d'une forme*
- *Les chaînes éditoriales ne font que simuler ce principe, en favoriser une forme qui serait plus proche du "fond" - en fait de l'intention auctoriale - et plus éloignée de la mise en forme.*
- *C'est un principe de rationalité, cela permet d'accéder à certaines propriétés intéressantes pour l'ingénierie du contenu (réutilisation, multi-*

- supports, etc.)*
- *Exemple : en pratique il est difficile d'écrire tout indépendamment d'une forme pour un auteur par exemple, l'aller-retour entre forme d'écriture et forme de publication est souvent nécessaire.*

Question 2

Expliquer le principe d'une chaîne éditoriale XML basée sur la séparation entre un format de stockage orienté métier (fond) et des formats de mise en forme orientés présentation (forme).

Question 3

Pourquoi doit-on se poser la question de savoir si un document numérique est encore un document ?

Question 4

En quoi le principe du balisage est-il un élément de réponse à la problématique de l'ingénierie des documents numériques ?

2.Exercice rédactionnel

[90 minutes]

Pour chaque question, vous produirez un argumentaire en deux ou trois pages maximum. Vous mobiliserez les concepts vus en cours, dans les ouvrages de référence, et dans vos propres lectures complémentaires. Vous veillerez à un usage précis des termes employés, et à la rigueur de l'argumentaire : Vous avez un espace d'expression limité, il convient de le "rentabiliser" au mieux en ne produisant que des énoncés dignes d'intérêt pour le lecteur - ou, en clair, évitez le remplissage et l'utilisation approximative de notions mal maîtrisées !

Question 1

Quelles différences peut-on relever entre un document et une base de données ? Peut-on les exploiter de la même manière ? Contiennent-ils le même type d'information ? On reviendra pour cela sur les types de contenus, les formats et les exploitations que l'on peut en faire.

Question 2

Est-ce qu'un auteur a pleinement sa place au sein des chaînes éditoriales XML ? Peut-on anticiper des difficultés à leur mise en œuvre et donc déduire des recommandations pratiques pour l'organisation du travail et l'accompagnement des usages ? Quels seraient les métiers proches, dans le monde classique de l'édition et de la presse par exemple, du rôle attendu de l'auteur dans la chaîne éditoriale ?

B.Travaux pratiques chaînes éditoriales

1.Découvrir des chaînes éditoriales

Opale : une chaîne métier

La chaîne éditoriale Opale est dédiée à la production de documents pédagogiques :

- <http://scenari-platform.org/opale>
- Exemple de contenus universitaires :
http://www4.utc.fr/~nf17/co/nf17_FR_2.html

Webradio : une chaîne multimédia

La chaîne éditoriale Webradio est dédiée à la production de documents sonores enrichis :

- <http://scenari-platform.org/projects/webradio/fr/pres/co/>
- Chapitre "Créer interviews, podcasts et webradios"¹⁷ (Scenari, la chaîne éditoriale libre [Crozat07])
- Exemple de contenus à l'Ina : <http://www.ina-entreprise.com/entreprise/activites/recherches-musicales/webradio.html>

L'on peut aussi expérimenter WebMédia, le successeur de WebRadio (encore en développement) :

- <http://scenari-platform.org/projects/webmedia/fr/pres/co/>

2.Apprendre à utiliser une chaîne éditoriale

Découverte de Scenari via OptimOffice

OptimOffice est une chaîne éditoriale Scenari généraliste assez simple d'accès et idéale pour se familiariser avec l'outil et les concepts.

<http://scenari-platform.org/optimoffice>

Télécharger le document en ligne pour réaliser l'exercice.

1. Installer l'application Scenari OptimOffice
2. Créer un atelier
3. Créer un espace de travail dans l'atelier
4. Créer un item racine (papier)
 - saisir le titre : « Chaînes éditoriales numériques »
 - saisir les métadonnées (paternité, date)
 - NB : notion de partie optionnelle (partie préalable)
5. Saisir le premier chapitre « Historique : De TeX à XML »
 - titre et contenu
 - balisage dans le paragraphe (mise en relief)
 - NB : notion de croix rouge (informations manquantes)
 - NB : prendre l'habitude d'enregistrer (CTRL+S)
6. Publier
 - générer, puis consulter le support papier
 - générer et consulter les autres supports
7. Saisir le second chapitre « Fondements : La raison computationnelle »
 - insérer un bloc "mise en relief"
 - NB : fonctionnent des étoiles d'insertion
8. Saisir les autres chapitres
 - citations
 - sections
 - listes
 - insérer une image
9. Récupérer un résultat de génération
 - télécharger
 - révéler
 - déployer (ftp)

Utilisation avancée des chaînes éditoriales

Utiliser le manuel :
<http://scenari-platform.org/projects/optim/fr/pres/co/documentation.html>

[http://scenari-](http://scenari-platform.org/projects/optim/fr/pres/co/documentation.html)

17 - http://www.eyrolles.com/Chapitres/9782212121506/Chap4_Crozat.pdf?xd=72e89ce317ac45cc0cdeebe11ec9a15a

1. Rééditorialisation
 - externaliser un chapitre
 - créer un second papier avec uniquement le premier et le dernier chapitre (réutilisation par référence)
 - utiliser le bloc de type "liste d'événements"
 - créer un diaporama avec des sections originales et des sections réutilisées
 - créer un site web avec des sections originales et réutilisées
2. Copier/coller
 - copier / coller avant
3. Multi-support avancé
 - exclusion de blocs
4. Utiliser les références
 - ajouter des définitions
5. Tester d'autres éléments du modèle
 - utiliser d'autres types de blocs (complément)
 - insérer un tableau
 - insérer des caractères spéciaux
 - insérer un schéma ODG, le modifier via Scenari
 - créer un tableau ODS directement via Scenari
 - insérer des liste (à puce, ordonnées), notion d'imbrication des listes
 - mettre un titre aux images
6. Tester d'autres fonctions de SCENARIchain
 - affichage du plan
 - revenir en arrière après une action
 - rechercher un mot dans l'item (CTRL+F), remplacer
 - afficher / masquer les balises et espaces insécables
 - apprendre à utiliser plusieurs onglets
 - passer l'éditeur en plein écran
 - rechercher un item ans l'atelier
 - afficher le brouillon

C.Exercices XML

1.Exercice

Soit le fichier XML ci-après, quelles sont les assertions vraies ?

```
<?xml version="1.0" encoding="UTF-8"?>
<papier type="scientifique">
  <titre>Réinterroger les structures documentaires</titre>
  <sousTitre>De la numérisation à l'informatisation</sousTitre>
  <auteur>Stéphane Crozat</auteur>
  <auteur>Bruno Bachimont</auteur>
  <resume>Nous proposons dans cet article d'aborder
  ...</resume>
  <abstract>In this paper we define...</abstract>
  <motsCles>
    <terme>Ingénierie des connaissances</terme>
    <terme>XML</terme>
    <terme>Document</terme>
  </motsCles>
  <keywords>
    <word>Knowledge engineering</word>
    <word>XML</word>
```

```

<word>Document</word>
</keywords>
<publication date="2004-07-05"/>
<version maj='1' min='0'/>
<ressource
uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/si
c_00001015.pdf"/>
</papier>
```

- Ce fichier n'est pas un fichier XML
- Ce fichier est un fichier XML, il possède un élément racine unique qui contient tous les autres.
- Ce fichier est un fichier XML, tous ses éléments sont inclus dans les uns dans les autres.
- Ce fichier est un fichier XML, toutes ses balises ouvertes sont fermées.
- Ce fichier est un fichier SGML.

2.Exercice

Compléter les trous avec les balises **fermant**es adéquates.

```

<?xml version="1.0"?>
<document>
    <entete>
        <titre>Document à corriger</titre>
        <auteur>Stéphane Crozat</auteur>
        <date>01-03-01</date>
        <version numero="1"/>
    </entete>
    <corps>
        <division titre="Première partie">
            <paragraphe>Ce texte doit être
            <important>corrigé </important> </paragraphe> <paragraphe>Le contenu est
            sans importance ici</paragraphe>
        </division>
        <division titre="Seconde partie">
            <paragraphe>Ai-je le droit de mettre du texte ici ?</paragraphe>
        </division>
    </corps>

```

3.Exercice

Fermer les balises du fichier suivant afin qu'il soit un document XML bien formé.

```
<docAbstrait>
<contenu>
<paragraphe>Lorem ipsum dolor sit amet, consectetur adipiscing
elit.</>
<Texte>Nulla erat tellus, molestie a ultrices sed, gravida eget ipsum.
<para>Phasellus lacinia, ipsum vitae interdum tincidunt.
<P>Nullam pulvinar diam et tellus ullamcorper
eleifend.</></></></></>
<ligne>Nunc eu lectus in diam tempus adipiscing in rhoncus
elit.</></>
```

D.Exercices RelaxNG

1.Exercice rédactionnel

Représenter la lettre officielle suivante en utilisant le formalisme XML.

Exercices



Stéphane Crozat
Département Génie Informatique
Université de Technologie de Compiègne
60 200 Compiègne

Tel: 06 81 06 10 70

Compiègne, le 1 mars 2001

Objet : Démonstration XML
A l'attention de Monsieur Dupont

Monsieur,

Xxx
xx
xx

Xxx
xx
xx

Xxx
xx
xx
xxxxxxxxxxxxxxx

Veuillez accepter, Monsieur Dupont, l'expression de ma considération distinguée,

Stéphane Crozat

Une lettre

Question 1

Repérer la structure physique du document.

Question 2

Déduire une structure logique depuis la structure physique et écrire le document XML correspondant

Question 3

Écrire le schéma RelaxNG des lettres respectant ce modèle.

Indice :

Utiliser :

- Des patterns, par exemple `Texte = paragraph+*`
- Des datatype, par exemple pour la date
- Proposer votre propre namespace

2.Exercice

Compléter le schéma RelaxNG schema.rng afin que les fichiers XML file1.xml, file2.xml et file3.xml soient valides. On cherchera le schéma **le plus restrictif** possible. On cherchera un schéma sans description redondante, en utilisant la syntaxe ref/define lorsque c'est nécessaire.

```
<?xml version="1.0"?>
<!--file1.xml-->
<?oxygen RNGSchema="04.rng" type="xml"?>
<a>
<b><b1/></b>
</a>
```

```
<?xml version="1.0"?>
<!--file2.xml-->
<?oxygen RNGSchema="04.rng" type="xml"?>
<a>
<b><b1/></b>
<b><b1/><b2/></b>
<b><b1/></b>
</a>
```

```
<?xml version="1.0"?>
<!--file3.xml-->
<?oxygen RNGSchema="04.rng" type="xml"?>
<a>
<b><b1/></b>
<b><b1/><b2/></b>
<b><b1/></b>
<c/>
<c/>
<b><b1/></b>
<b><b1/></b>
</a>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<[REDACTED] xmlns="http://relaxng.org/ns/structure/1.0">
<[REDACTED]>
  <element name="[REDACTED]">
    <[REDACTED]>
      <ref name="B"/>
    </[REDACTED]>
    <[REDACTED]>
      <element name="[REDACTED]"><[REDACTED]/></element>
    </[REDACTED]>
    <[REDACTED]>
      <ref name="[REDACTED]" />
    </[REDACTED]>
  </element>
</[REDACTED]>
```

```
<[REDACTED] name="[REDACTED]">
  <element name="[REDACTED]">
    <element name="[REDACTED]"><[REDACTED]/></element>
    <[REDACTED]>
      <element name="[REDACTED]"><[REDACTED]/></element>
    </[REDACTED]>
  </element>
</[REDACTED]>
</grammar>
```

E.Exercices DTD

1.Exercice rédactionnel

Soit la DTD suivante.

```
<!ELEMENT document (entete,corps)>

<!ELEMENT entete (identification, motscles?, resume?)>
<!ELEMENT identification (titre, date, auteur, version)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT motscles (motitem+)>
<!ELEMENT motitem (#PCDATA)>
<!ELEMENT resume (paragraphe+)>

<!ELEMENT corps (introduction?, (div+ | contenu),
conclusion?)>
<!ELEMENT div (titre, introduction?, (div+ | contenu),
conclusion?)>
<!ELEMENT introduction (paragraphe+)>
<!ELEMENT conclusion (paragraphe+)>
<!ELEMENT contenu (paragraphe)+>

<!ELEMENT paragraphe (#PCDATA | important | etranger | note)*>

<!ELEMENT important (#PCDATA)>
<!ELEMENT note (#PCDATA)>
<!ELEMENT etranger (#PCDATA)>
<!ATTLIST etranger
langue CDATA #IMPLIED
>
```

Question 1

Produire un document XML valide par rapport à cette DTD

Question 2

Produire le plus petit document XML valide par rapport à cette DTD.

Question 3

Produire un document XML utilisant toutes les balises de la DTD.

Soit le document XML suivant.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <entete>
    <identification>
      <titre>Text</titre>
      <date>Text</date>
      <auteur>Text</auteur>
      <version>Text</version>
    </identification>
    <motscles>
      <motitem>Text</motitem>
      <motitem>Text</motitem>
      <motitem>Text</motitem>
    </motscles>
  </entete>
  <corps>
    <introduction>
      <paragraphe>Text</paragraphe>
      <paragraphe>Text
        <important>Text</important>Text</paragraphe>
      </introduction>
      <div>
        <titre>Text</titre>
        <introduction>
          <paragraphe>Text</paragraphe>
        </introduction>
        <contenu>
          <paragraphe>Text</paragraphe>
        </contenu>
        <conclusion>
          <paragraphe><note>Text</note></paragraphe>
        </conclusion>
      </div>
      <div>
        <titre>Text</titre>
        <div>
          <titre></titre>
          <contenu>
            <paragraphe>Text</paragraphe>
          </contenu>
        </div>
        <conclusion>
          <paragraphe>Text</paragraphe>
        </conclusion>
      </div>
      <conclusion>
        <paragraphe>Text</paragraphe>
      </conclusion>
    </corps>
  </document>
```

Question 4

Ce document est-il valide par rapport à la DTD ?

2.Exercice

Le fichier `file.xml` n'est-il pas valide par rapport à la DTD `schema.dtd`. Sélectionnez les éléments cause de cette non-validité.

```
<?xml version="1.0"?>
<!--file.xml-->
<!DOCTYPE papier SYSTEM "schema.dtd">

    <titre>Réinterroger les structures documentaires</titre>
    <auteur>Stéphane Crozat</auteur>
    <auteur>Bruno Bachimont</auteur>
    <resume>Nous proposons dans cet article d'aborder
    ...</resume>
    <abstract>In this paper we define...</abstract>
    <motsCles>
        <terme>Ingénierie des connaissances</terme>
        <terme>Document</terme>
    </motsCles>
    <version num='1' />
    <ressource src="sic_00001016.pdf"/>

```

```
<!-- schema.dtd-->
<!ELEMENT papier (titre, sousTitre?, auteur, resume, abstract,
motsCles, (version | ressource)*)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT sousTitre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT resume (#PCDATA)>
<!ELEMENT motsCles (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT ressource EMPTY>
```

- i - resume
- ii - motsCles
- iii - titre
- iv - abstract
- v - auteur
- vi - version
- vii - sousTitre
- viii - ressource

Éléments correctement spécifiés

Éléments incorrectement spécifiés

F.Quiz DTD

1. Préambule

Soit les fichiers XML et DTD suivants :

cours1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cours>
<definition>XML est un méta-langage</definition>
<exemple>XHTML est un langage XML</exemple>
<exemple>SMIL est un langage XML</exemple>
</cours>
```

cours2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cours>
<definition>XML est un méta-langage</definition>
<remarque>XML est hérité de SGML</remarque>
</cours>
```

cours3.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cours>
<definition>XML est un méta-langage</definition>
</cours>
<cours>
<exemple>XHTML est un langage XML</exemple>
</cours>
```

cours4.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cours>
<definition>XML est un méta-langage</definition>
<exemple>XHTML est un langage XML</exemple>
<remarque>XML est hérité de SGML</remarque>
</cours>
```

schCours1.dtd

```
<!ELEMENT cours (definition, exemple)>
<!ELEMENT definition (#PCDATA) >
<!ELEMENT exemple (#PCDATA) >
```

schCours2.dtd

```
<!ELEMENT cours (definition, remarque?)>
<!ELEMENT definition (#PCDATA) >
<!ELEMENT remarque (#PCDATA) >
```

2.Exercice : Fichiers bien formés

Quels sont les fichiers bien formés ?

- cours1.xml
- cours2.xml
- cours3.xml
- cours4.xml
- schCours1.dtd
- schCours2.dtd

3.Exercice : Fichiers valides

Quels sont les fichiers XML valides par rapport à quelles DTD ?

- i - cours2.xml
- ii - cours4.xml
- iii - cours1.xml
- iv - cours3.xml

Est valide par rapport à
schCours1.dtd

Est valide par rapport à
schCours2.dtd

N'est pas valide

4.Exercice : Schéma intégrateur

Proposer un schéma schCours3.dtd tel que tous les fichiers bien formés parmi cours1.xml, cours2.xml, cours3.xml et cours4.xml soient valides.

```
<!ELEMENT cours ( [REDACTED], [REDACTED],  
[REDACTED] )>  
<!ELEMENT definition (#PCDATA) >  
<!ELEMENT exemple (#PCDATA) >  
<!ELEMENT remarque (#PCDATA) >
```

G.Exercices XSLT et XPath

1.CV

Un schéma de CV

Soit le schéma XML suivant (formalisme DTD) :

```
<!ELEMENT cv (nom, prenom, age?, rubrique+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT rubrique (titre, contenu)>
<!ELEMENT contenu (#PCDATA)>
<!ELEMENT titre (#PCDATA)>
```

Question 1

Écrire un document XML valide par rapport à ce schéma avec au moins deux rubriques.

Indice :

```
<cv>
  <nom>...</nom>
  ...
  <rubrique>
    ...
  </rubrique>
  <rubrique>
    ...
  </rubrique>
</cv>
```

Question 2

Écrire un document HTML cible d'une transformation XSLT de ce document.

Les nom, prénom et age seront en italique, les titres de rubriques seront en gras.

Indice :

```
<html>
  <head>
    <title>...</title>
  </head>
  <body>
    <p><i>...</i></p>
    <p><i>...</i></p>
    <p><i>...</i></p>
    <p><b>...</b></p>
    <p>...</p>
    ...
  </body>
</html>
```

Question 3

Écrire le programme de transformation XSLT d'un fichier XML en fichier HTML.

Indice :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="cv">
    <html>
        <head>
            <title>CV de <xsl:value-of select="..."/> <xsl:value-
of select="..."/></title>
        </head>
        <body>
            <xsl:apply-templates/>
        </body>
    </html>
</xsl:template>
<xsl:template match="nom">
    <p><i>...</i></p>
</xsl:template>
...
<xsl:template match="rubrique">
    ...
</xsl:template>
<xsl:template match="titre">
    ...
</xsl:template>
...
</xsl:stylesheet>
```

2.Poème

Soit l'extrait de poème suivant écrit en XML :

```
<poeme titre="The Stone Troll" auteur="JRR Tolkien">
<strophe>
    <vers>Troll sat alone on his seat of stone,</vers>
    <vers>And munched and mumbled a bare old bone;</vers>
    <vers>For many a year he had gnawed it near,</vers>
    <vers>For meat was hard to come by.</vers>
    <vers>Done by! Gum by!</vers>
    <vers>In a cave in the hills he dwelt alone,</vers>
    <vers>And meat was hard to come by.</vers>
</strophe>
<strophe>
    <vers>Up came Tom with his big boots on.</vers>
    <vers>Said he to Troll: 'Pray, what is yon?</vers>
    <vers>For it looks like the shin o' my nuncle Tim.</vers>
    <vers>As should be a-lyin' in the graveyard.</vers>
    <vers>Caveyard! Paveyard!</vers>
    <vers>This many a year has Tim been gone,</vers>
    <vers>And I thought he were lyin' in the graveyard.</vers>
</strophe>
</poeme>
```

Question

Écrire un programme XSL-XSLT permettant de le transformer selon le format HTML suivant :

```
<html>
  <head>
    <title>The Stone Troll (JRR Tolkien)</title>
  </head>
  <body>
    <p>Troll sat alone on his seat of stone,</p>
    <p>And munched and mumbled a bare old bone;</p>
    <p>For many a year he had gnawed it near,</p>
    <p>For meat was hard to come by.</p>
    <p>Done by! Gum by!</p>
    <p>In a cave in the hills he dwelt alone,</p>
    <p>And meat was hard to come by.</p>
    <hr/>
    <p>Up came Tom with his big boots on.</p>
    <p>Said he to Troll: 'Pray, what is yon?</p>
    <p>For it looks like the shin o' my nuncle Tim.</p>
    <p>As should be a-lyin' in the graveyard.</p>
    <p>Caveyard! Paveyard!</p>
    <p>This many a year has Tim been gone,</p>
    <p>And I thought he were lyin' in the graveyard.</p>
  </body>
</html>
```

Indices :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:template match="poeme">
    <html>
      <head>
        <title>...</title>
      </head>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="strophe">
    <xsl:apply-templates/>
    ...
  </xsl:template>
  <xsl:template match="vers">
    ...<xsl:value-of select="..."/>...
  </xsl:template>
</xsl:stylesheet>
```

Pour gérer l'absence de <hr/> sur la dernière strophe, ajouter une règle qui sélectionne strophe[last()].

3.Glossaire

Soit le schéma RelaxNG suivant permettant de représenter un glossaire.

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <start>
    <element name="glossaire">
      <oneOrMore>
        <ref name="Definition"/>
      </oneOrMore>
    </element>
  </start>
  <define name="Definition">
    <element name="definition">
      <attribute name="id"><data type="ID"/></attribute>
      <element name="terme"><text/></element>
      <element name="explication"><text/></element>
      <zeroOrMore>
        <element name="voirAussi">
          <attribute name="ref"><data type="IDREF"/></attribute>
        </element>
      </zeroOrMore>
    </element>
  </define>
</grammar>
```

Question 1

Expliquer ce qu'exprime les *datatypes ID et REFID*. Préciser le format que doit respecter un attribut ou un élément de type *ID*.

Question 2

Instancier un document de type *glossaire*, sur une thématique au choix.

Question 3

Réaliser un programme XSLT permettant de publier les documents de type *glossaire* en HTML.

Indice :

Utiliser les ancrés en HTML pour gérer les "voir aussi".

4.Tester des expressions XPath

Les éditeurs XML permettent en général d'exécuter des expressions XPath.

Expression XPath

Nœud courant

Résultat

The screenshot shows the Oxygen XML Editor interface. The main window displays an XML file named "Sans titre1.xml". The XML content includes a document header, an entete section with identification details, and a corps section containing paragraphs and a remark. A cursor is positioned over the first paragraph. The bottom panel shows the execution results for the XPath expression "/document[1]/corps[1]/contenu[1]/paragraphe[1]". The results pane lists two elements: the paragraph itself and its contained remark element. The status bar at the bottom indicates the file path is "/home/stc/Bureau/Sans titre1.xml", the operation is "XPath -- réussie", and the time is "11:6".

Exécution d'expression XPath dans l'éditeur Oxygen

Soit le fichier XML suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<document modele="ULCoursGeneral" code="BP-
Incendie3_S1_E2_UL1">
<entete>
  <identification>
    <titre>L'assurance de la responsabilité de voisinage</titre>
    <date>21/02/01</date>
    <auteur>AEA</auteur>
    <version>1.00</version>
  </identification>
</entete>
<corps>
  <contenu>
    <paragraphe>Cette garantie est appelée : "recours des voisins et des tiers".</paragraphe>
    <remarque>
      <paragraphe>L'image suivante <ressource URIsrc="img07.jpg"
        titre="Recours des voisins et des tiers" type="image"/> montre la
        garantie</paragraphe>
    </remarque>
  </contenu>
</corps>
</document>
```

```
voisinage</titre>
  <date>21/02/01</date>
  <auteur>AEA</auteur>
  <version>1.00</version>
</identification>
</entete>
<corps>
  <contenu>
    <paragraphe>Cette garantie est appelée : "recours des
voisins et des tiers".</paragraphe>
    <remarque>
      <paragraphe>L'image suivante <ressource URIsrc="img07.jpg"
titre="Recours des voisins et des tiers" type="image"/> montre
la garantie.</paragraphe>
    </remarque>
  </contenu>
</corps>
</document>
```

Question

Exécuter les expressions XPath suivantes. Pour chacune choisissez au moins deux nœuds courants différents : un qui renvoie quelque chose et un qui ne renvoie rien (lorsque c'est possible).

```
/document/entete/identification/titre
/document/@modele
corps//contenu
contenu/*
contenu/remarque[1]
../paragraphe
@type
```

5.Namespaces et FO

Soit le fichier XSLT suivant, permettant de transformer un fichier XML en fichier *Formatting Objects* (standard W3C pour la publication de fichiers imprimables).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
>
<xsl:template match="doc">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="A4" page-width="297mm"
        page-height="210mm" margin-top="1cm" margin-bottom="1cm"
        margin-left="1cm" margin-right="1cm">
        <fo:region-body margin="3cm"/>
        <fo:region-before extent="2cm"/>
        <fo:region-after extent="2cm"/>
        <fo:region-start extent="2cm"/>
        <fo:region-end extent="2cm"/>
      </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="A4" format="A">
      <fo:flow flow-name="xsl-region-body">
```

```

<xsl:apply-templates select="para"/>
</fo:flow>
</fo:page-sequence>
</fo:root>
</xsl:template>
<xsl:template match="para">
<fo:block><xsl:value-of select=". "/></fo:block>
</xsl:template>
<xsl:template match="para[1]">
<fo:block><fo:inline font-weight="bold"><xsl:value-of
select=". "/></fo:inline></fo:block>
</xsl:template>
</xsl:stylesheet>

```

Question 1

Indiquer quels sont les *namespaces* et préfixes définis dans ce fichier XSLT, expliquer à quoi ils servent en général et dans ce cas précis, et en particulier pourquoi l'on ne pourrait pas s'en passer ici.

Question 2

Inventer un fichier XML source cohérent avec la transformation XSLT proposée, et produire le fichier FO correspondant au résultat de la transformation.

6.Bulletins météo

Soit le fichier f1.smi suivant (très proche de la syntaxe SMIL) :

```

<?xml version="1.0" encoding="UTF-8"?>
<smil>
  <body>
    <seq>
      <par>
        <text dur="1">Île de France</text>
      </par>
      <par>
        <text dur="1">8 janvier 2010</text>
      </par>
      <par>
        <text dur="3">5 cm de neige en moyenne sur la
        région</text>
      </par>
      <par>
        <text dur="3">Jusqu'à 10cm de neige sur le sud de la
        région</text>
      </par>
    </seq>
  </body>
</smil>

```

Pour rappel :

- `seq` décrit une séquence de parties `par`, contenant du contenu affiché pendant une durée `dur`.
- les `par` d'une `seq` sont présentes en même temps à l'écran (pour simplifier dans notre cas, les unes en dessous des autres)

Question 1

Ce fichier est-il un fichier XML bien formé (justifier) ?

Question 2

Soit la DTD `smilSuperLight.dtd` suivante :

```
<!ELEMENT smil (meta*, body)>
<!ATTLIST meta
  name (title) #REQUIRED
  content CDATA #REQUIRED>
<!ELEMENT body (seq)>
<!ELEMENT seq (par+)>
<!ELEMENT par (text)>
<!ELEMENT text (#PCDATA)>
<!ATTLIST text
  dur CDATA #REQUIRED>
```

Le fichier `f1.smi` est-il valide par rapport à cette DTD (justifier) ?

Question 3

Si `smilSuperLight.dtd` est un sous-ensemble du schéma SMIL officiel du W3C, le fichier XML `f1.smi` est-il valide par rapport à la DTD SMIL du W3C (expliquer) ?

Question 4

Pourquoi le fichier `f2.smi` n'est-il pas valide par rapport à la DTD `smilSuperLight.dtd` ?

```
<?xml version="1.0" encoding="UTF-8"?>
<smil>
  <body>
    <seq>
      <par>
        <text dur="1">Champagne</text>
        <text dur="1">8 janvier 2010</text>
      </par>
      <par>
        <text dur="3">Brouillard dominant et éclaircies en fin
        de matinée</text>
      </par>
    </seq>
  </body>
</smil>
```

Question 5

Modifier `smilSuperLight.dtd` afin que `f2.smi` soit à présent valide par rapport à une nouvelle DTD `smilSuperLight2.dtd`

Soit les deux fichiers XML suivants, structurés selon une sémantique métier :

```
<?xml version="1.0"?>
<meteo>
  <region>Picardie</region>
  <date>8 janvier 2010</date>
  <phenomene>Pluie et brouillard sur toute la
  région</phenomene>
  <temperature>
    <matin>-5</matin>
    <soir>-3</soir>
  </temperature>
```

Exercices

```
</meteo>
```

```
<?xml version="1.0"?>
<meteo>
    <region>Nord</region>
    <date>8 janvier 2010</date>
    <phenomene>Brouillard et neige sur le nord de la
région</phenomene>
    <phenomene>Vents violents sur le bord de mer</phenomene>
    <phenomene>Verglas sur toute la région</phenomene>
</meteo>
```

Question 6

Proposer un schéma XML `meteo.dtd` selon le formalisme des DTD de telle façon que ces **deux** fichiers soient valides par rapport à ce schéma.

Question 7

Proposer un programme XSLT permettant de transformer les fichiers de type `meteo` en `smilSuperLight2`, en affichant séquentiellement :

- la région avec la date dans une même première partie (1 seconde chacune, donc 2 secondes en tout),
- puis chaque phénomène (les températures ne sont pas affichées) pendant 3 secondes chaque.

Indices :

Cela correspond à l'exemple de `f2.smi`.

*Il y a une difficulté particulière en XSLT à regrouper `région` et `date` dans une même partie, si vous ne parvenez pas à résoudre cette difficulté, faites l'exercice **sans les regroupez**. Cela revient dans ce cas à faire une transformation en `smilSuperLight`, à l'image de `f1.smi`.*

H.Quiz XSLT et XPath

1.Exercice

Soit le fichier XML ci-dessous. Si le nœud courant est un des éléments `terme`, écrivez 4 expressions XPath différentes permettant de renvoyer le titre du document :

1. *Sachant que `titre` est unique dans tout le document.*
2. *Sachant que `titre` est le fils de l'élément racine `papier`.*
3. *Sachant que `titre` est le fils du père du père du nœud courant.*
4. *Sachant que `titre` est avant le nœud courant dans l'ordre du document.*

```
<?xml version="1.0" encoding="UTF-8"?>

    <titre>Réinterroger les structures documentaires</titre>
    <sousTitre>De la numérisation à l'informatisation</sousTitre>
    <auteur>Stéphane Crozat</auteur>
    <auteur>Bruno Bachimont</auteur>
    <resume>Nous proposons dans cet article d'aborder
    ...</resume>
```

```
<abstract>In this paper we define...</abstract>
<motsCles>
    <terme>Ingénierie des connaissances</terme>
    <terme>XML</terme>
    <terme>Document</terme>
</motsCles>
<keywords>
    <word>Knowledge engineering</word>
    <word>XML</word>
    <word>Document</word>
</keywords>
<publication date="2004-07-05"/>
<version maj='1' min='0' />
<ressource
uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/si
c_00001015.pdf"/>
</papier>
```

1. //
2. /
3. ..
4. preceding

2.Exercice

Écrire le programme XSLT permettant de transformer le fichier file.xml en result.xhtml.

```
<!--file.xml-->
<doc>
<para>Lorem ipsum dolor sit amet.</para>
<para>Consectetur adipiscing elit.</para>
<para>Nunc eu lectus in diam.</para>
</doc>
```

```
<!--result.xhtml-->
<xhtml>
<body>
<p><i>Lorem ipsum dolor sit amet.</i></p>
<p>Consectetur adipiscing elit.</p>
<p>Nunc eu lectus in diam.</p>
</body>
</xhtml>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="[">
    <[">
    <[">
    <xsl:apply-templates select="[">
        </[">
    </[">
```

```

</xsl:template>
<xsl:template match=" " >
    < ><xsl:value-of select=" " />< >
</xsl:template>
<xsl:template match=" " >
    < >< ><xsl:value-of
select=" " />< >< >
</xsl:template>
</xsl:stylesheet>

```

3.Exercice

Soit le fichier `file.xml`. Compléter le fichier XSLT `transf.xsl` afin de générer, pour chaque élément `terme`, une instruction SQL d'insertion dans une table relationnelle de schéma : `tMotsCles` (`terme`, `titre`, `url`) (où `terme` est le terme sélectionné, `titre` est le titre du document et `url` est l'adresse de la ressource associée).

Pour rappel, la syntaxe d'insertion de données dans une table relationnelle en SQL : `INSERT INTO <Nom de la relation> (<Liste ordonnée des propriétés à valoriser>) VALUES (<Liste ordonnée des valeurs à affecter>)`.

Soit le fichier XML de la première question.

```

<!--file.xml-->
<papier type="scientifique">
    <titre>Réinterroger les structures documentaires</titre>
    <sousTitre>De la numérisation à l'informatisation</sousTitre>
    <auteur>Stéphane Crozat</auteur>
    <auteur>Bruno Bachimont</auteur>
    <resume>Nous proposons dans cet article d'aborder
    ...</resume>
    <abstract>In this paper we define...</abstract>
    <motsCles>
        <terme>Ingénierie des connaissances</terme>
        <terme>XML</terme>
        <terme>Document</terme>
    </motsCles>
    <keywords>
        <word>Knowledge engineering</word>
        <word>XML</word>
        <word>Document</word>
    </keywords>
    <publication date="2004-07-05"/>
    <version maj='1' min='0' />
    <ressource
        uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/si
        c_00001015.pdf"/>
    </papier>

<!--transf.xsl-->
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="text"/>

```

```
<xsl:template match=" " >
<xsl:apply-templates select=" " />
</xsl:template>
<xsl:template match=" " >
  (terme, titre, url) (
    '<xsl:value-of select=" " />',
    '<xsl:value-of select="// " />',
    '<xsl:value-of select="// " />'
  );
</xsl:template>
</xsl:stylesheet>
```

I.Exercices UML, passage RelaxNG, XSLT

1.Standard W3C

Soit l'extrait modifié suivant du début d'un standard du W3C. L'objectif est de réaliser une chaîne éditoriale XML permettant la production contrôlée de ce type de documents.

Exercices





Extensible Stylesheet Language (XSL) Version 1.1

W3C Recommendation 05 December 2006

This version:

<http://www.w3.org/TR/2006/REC-xsl11-20061205/>

Latest version:

<http://www.w3.org/TR/xsl11/>

Previous version:

<http://www.w3.org/TR/2006/PR-xsl11-20061006/>

Editor:

Anders Berglund, IBM aalrb@us.ibm.com

Abstract

This specification defines the features and syntax for the Extensible Stylesheet Language (XSL), a language for expressing stylesheets. It consists of two parts:

1. a language for transforming XML documents (XSLT), and
2. an XML vocabulary for specifying formatting semantics.

An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.

Capture Web modifiée de la page <http://www.w3.org/TR/xsl>

Question 1

Proposer un modèle conceptuel UML permettant de modéliser ce document dans un langage orienté métier.

Question 2

Traduire ce modèle en schéma RelaxNG.

Question 3

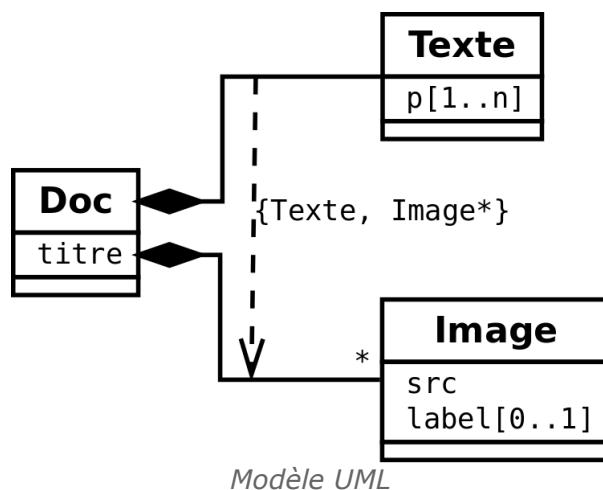
Rédiger un fichier XML valide correspondant à l'exemple.

Question 4

Écrire une feuille de transformation XSLT permettant de transformer les documents de ce type en HTML.

2. Contraintes d'ordre

Soit le modèle conceptuel ci-après.

**Question 1**

Expliquer la signification de la contrainte (`Texte, Image*`) ajoutée au schéma, et pourquoi de telles contraintes sont parfois indispensables dans le cas de la modélisation documentaire ?

Question 2

Proposez une traduction en schéma RelaxNG de ce modèle.

3.Cours en ligne

Question 1

Réaliser le modèle UML d'un des modules de cours de NF29, par exemple : <http://www4.utc.fr/~nf29/DOCS/cours/02ce/module>.

Question 2

Traduisez le en schéma XML (selon le formalisme de votre choix).

4.Diaporama

On souhaite réaliser une chaîne éditoriale XML permettant de créer et publier des diaporamas.

Soit l'exemple représentatif de contenu suivant :

Question 1

Réaliser le modèle conceptuel du document avec UML.

Question 2

Réaliser le schéma XML du document avec RelaxNG.

Question 3

Rédiger un exemple de contenu XML valide par rapport au schéma.

Question 4

Écrivez un programme XSL-XSLT qui permet de transformer une diapositive de type diapoImage en XHTML.

Pour ce premier programme simple on pourra écrire un seul template et l'on utilisera les XPATH pour remplir le patron XHTML.

Pour le corps du XHTML on pourra utiliser les balises h1 et img.

Indice :

Structure XHTML visée :

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>...</title>
</head>
<body>
    <h1>...</h1>
    
</body>
</html>
```

Programme XSLT à finir de remplir (remplacer les __XPATH__).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:td="http://utc.fr/nf29/td"
    xmlns="http://www.w3.org/1999/xhtml"
    exclude-result-prefixes="td"
    version="1.0">
    <xsl:output method="html" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title><xsl:value-of select="__XPATH__"/></title>
            </head>
            <body>
                <h1><xsl:value-of select="__XPATH__"/></h1>
                
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Question 5

Écrivez un programme XSL-XSLT qui permet de transformer une diapositive de type diapoTexte en XHTML.

Indice :

Écrivez six templates, un pour la racine /, un pour diapoTexte, un pour liste, un pour item, un pour paragraphe et un pour important.

Pour le XHTML, vous pourrez utiliser *ul*, *li*, *p* et *b*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:td="http://utc.fr/nf29/td"
    xmlns="http://www.w3.org/1999/xhtml"
    exclude-result-prefixes="td"
    version="1.0">
    <xsl:output method="html" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title><xsl:value-of select="__XPATH__" /></title>
            </head>
            <body>
                <xsl:apply-templates select="__XPATH__" />
            </body>
        </html>
    </xsl:template>
    <xsl:template match="td:diapoTexte">
        <h1><xsl:value-of select="__XPATH__" /></h1>
        <xsl:apply-templates select="__XPATH__" />
        <h2>Conclusion</h2>
        <xsl:apply-templates select="__XPATH__" />
    </xsl:template>
    <xsl:template match="td:liste">
        XSLT
    </xsl:template>
    <xsl:template match="td:item">
        XSLT
    </xsl:template>
    <xsl:template match="td:paragraphe">
        XSLT
    </xsl:template>
    <xsl:template match="td:important">
        XSLT
    </xsl:template>
</xsl:stylesheet>
```

Question 6

Écrivez un programme XSL-XSLT qui permet de transformer un diaporama complet en un unique fichier XHTML.

Indice :

Utiliser la fonction document().

```
<xsl:template match="td:diaporama">
    ...
    <xsl:apply-templates
        select="document(./td:refPartie)/td:partie" />
    ...
</xsl:template>
```

Question 7

Ajouter au programme précédent la génération préalable d'un sommaire avec des

ancres.

Indice :

Utiliser :

- La commande `xsl:foreach` pour le sommaire ;
- la fonction `generate-id` dans la balise `` pour les liens sources ;
- la fonction `generate-id` dans la balise `<h2 id="#{generate-id(.)}"><xsl:value-of select=".//td:titre"/></h2>` pour les ancrés cibles.

```
<xsl:template match="td:diaporama">
    ...
    <h2>Table des matières</h2>
    <xsl:for-each
        select="document(./td:refPartie)/td:partie">
        <p><a href="#{generate-id(.)}"><xsl:value-of
            select=".//td:titre"/></a></p>
        <xsl:for-each select="document(./td:refDiapo)/*">
            <li><a href="#{generate-id(.)}"><xsl:value-of
                select=".//td:titre"/></a></li>      </xsl:for-each>
        </xsl:for-each>
        ...
    </xsl:template>
```

```
<xsl:template match="td:partie">
    ...
    <h2 id="#{generate-id(.)}"><xsl:value-of
        select=".//td:titre"/></h2>
    ...
</xsl:template>
```

J.Travaux pratiques Wiki

1.Exécuter l'usage d'un Wiki

<http://scenari-platform.org/ici55/wiki/Accueil>

- Modifier une page existante
- Créer une nouvelle page
- Discuter une page
- Consulter l'historique d'une page
- Créer un compte utilisateur

K.Question théorique indexation

1.Exercice rédactionnel

[15 minutes]

En quoi peut-on rapprocher l'indexation de l'opération de balisage que l'on peut effectuer sur un document ? Ont-elles la même finalité ? Utilisent-elles les mêmes outils ? Peut-on dire que l'une est une partie de l'autre ? Un moyen pour l'autre ?

L.Travaux pratiques ECM

1.Utiliser un ECM : Nuxéo DM

Cet exercice a pour but l'expérimentation des bases de l'ECM Nuxéo.

Utiliser la documentation en ligne disponible depuis la page d'accueil du serveur : <https://doc.nuxeo.com/display/DMDOC>

Question 1

Administrer les utilisateurs (menu Utilisateurs et groupes)

1. Se connecter en tant qu'administrateur (Administrator ou utilisateurs du groupe administrators)
2. Identifier les groupes (administrators et members) et utilisateurs existants (Administrator)
3. Créer un nouvel utilisateur, lui affecter un groupe
4. Créer un nouveau groupe, lui affecter des utilisateurs

Question 2

Gérer les droits (onglet Administration)

1. Se connecter en administrateur
2. Identifier la structure générale par défaut d'un Nuxéo DM (un Espace personnel par utilisateur, une base documentaire commune Workspaces)
3. Créer un dossier Commun dans Workspaces, donner les droits complets aux membres de members

Question 3

Ajouter des documents (onglet Contenu)

1. Se connecter en tant qu'utilisateur standard (non administrateur).
2. Créer un dossier dans un dossier accessible en écriture de la base commune ou dans son espace personnel.
3. Mettre un document dans ce dossier.

Question 4

Renseigner les méta-données d'un document (onglet Modification).

1. Ajouter des **mots clés**, une **couverture** et d'autres méta-données
2. Visualiser les méta-données dans le résumé
3. Tester les fonctions de **recherche**
4. Tester les **navigations** par couverture et par sujet (en haut à gauche)

Question 5

Tester le verrouillage (onglet Résumé).

1. Verrouiller
2. Modifier
3. Déverrouiller

Question 6

Activer et tester le versionnage (*versioning*) (onglet Modification).

1. Activer la montée de version mineure
2. Enregistrer une modification
3. Vérifier la modification du numéro de version

Renouveler le test avec une montée de version majeure.

Question 7

Historique et restauration (onglet Historique).

1. Consulter l'historique des modifications du document
2. Restaurer une version antérieure

Que s'est-il passé au niveau du numéro de version ?

Question 8

Lier des documents (onglet Relations.)

Ajouter une relation de dépendance entre deux fichiers.

Question 9

Commenter un document (onglet Commentaires)

Ajouter des commentaires sur vos fichiers.

Question 10

Partager un dossier (onglet Administration).

Gérer les droits d'accès de tiers à vos contenus.

2.Documenter un ECM

L'objectif de cet exercice est de réaliser la documentation d'un ECM (Alfresco ou Nuxeo). Pour cela chaque groupe de projet réalisera une partie de la documentation sous la forme d'une mini-référence.

Chaque mini-référence traitera d'un cadre fonctionnel particulier. Elle détaillera les fonctions étudiées et pour chacune les modalités (comment faire ?) avec des procédures pas à pas (où cliquer ? quelles touches utiliser ? etc.), des copies d'écran correspondant à chaque étape (avant/après), des explications de concepts lorsque nécessaire (définitions des mots particuliers utilisés dans l'outil par exemple) et des renvois vers la référence quand elle existe.

La documentation sera réalisée au moyen soit d'un traitement de texte soit de Scenari/Dokiel. Dans le premier cas le livrable est constitué du document source (format .doc ou .odt), d'une version PDF et de l'ensemble des copies d'écran correctement référencées dans un fichier ZIP). Dans le second cas le livrable est constitué de la version HTML en ligne, de la version PDF et des sources Scenari zippées.

Les fonctions particulièrement importantes pour le projet pourront être mises en exergue.

Question 1

Fonctions d'administration de l'ECM : organisation des espaces de travail et des utilisateurs.

Indice :

- *Organisation des espaces de travail*
 - *Nuxéo : Domain, Workspace, Section,*
 - *Templates d'espaces (Nuxéo)*
- *Gestion des utilisateurs*
 - *Ajout d'utilisateur*
 - *Ajout de groupes d'utilisateurs*
 - *Ajout/Suppression de droits*

Question 2

Fonctions de gestion de document : Crédation, modification, suppression des fichiers et répertoires.

Indice :

- *Création de répertoires*
- *Création de documents*
- *Pré-visualisation de documents*
- *Accès aux document (visualisation, téléchargement)*
- *Relations entre documents (Nuxéo)*

Question 3

Fonctions de gestion de la concurrence et de versionning : accès concurrents, gestion de versions.

Indice :

- *Modification de document*
- *Check-in / Check-out*
- *Incrément de version (mineure, majeure)*
- *Historisation, visualisation/récupération de versions antérieures*
- *Publication de document (versions permanentes en lecture seule)*

Question 4

Fonctions de gestion de workflow et actions sur événements

Indice :

- *Utilisation de workflow*
- *Paramétrage de workflow*
- *Espaces "intelligents" (Alfreco)*

Question 5

Fonctions d'indexation et de recherche

Indice :

- *Ajout et suppression de métadonnées*
- *Modèles de métadonnées, schémas d'indexation*
- *Gestion des terminologies*
- *Interfaces de recherche*
- *Paramétrage de la recherche*

Question 6

Fonctions de collaboration

Indice :

- *Commentaires de documents*
- *Modération*
- *Abonnements, notifications*

M.Exercices SMIL et FO

1.Photos de vacances commentées

Écrire un fichier SMIL permettant d'enchaîner des photos de vacances avec pour chacune une phrase d'illustration.

- Les photos devront être de taille 640*480 (on ne gérera que des photos en "paysage")
- Le texte s'affichera dans une fenêtre de 640*200
- Chaque partie (photo + texte) s'affichera durant 3 secondes.

Question 1

Récupérer 2 ou 3 photos de vacances et les mettre au format 640*480.

Question 2

Définir le layout.

Question 3

Écrire les fichiers textes de commentaires (1 pour chaque photo).

Question 4

Écrire le fichier SMIL.

Question 5

Écrire un fichier FO permettant d'imprimer la même information. On imaginera que les fichiers SMIL et FO ont une même source XML (associée à des transformations XSLT différentes).



Glossaire

AMO

Assistance à maîtrise d'ouvrage, celui qui fait le lien entre MOA et MOE.

EAI

L'EAI (Enterprise Application Integration ou Intégration des applications d'entreprise) désigne à la fois les solutions et les méthodes destinées à assurer l'intégration des différentes composantes du système d'information. Il s'agit de gagner en souplesse et de baisser les coûts de maintenance des interfaces inter-applicatives. Les chantiers d'EAI sont souvent un préalable nécessaire à des projets e-business (source : Le Journal Du Net).

MOA

Maîtrise d'ouvrage, celui qui commande la réalisation du travail.

MOE

Maîtrise d'œuvre, celui qui réalise le travail.

XSL-FO

XSL-FO (FO pour Formatting Objects) est la seconde partie du standard W3C « Extensible Stylesheet Language Family ». Il propose un langage XML de mise en forme de documents imprimables.

Exemple d'extrait de code FO :

- <fo:block font-family="Times" font-size="12pt">Ceci est un paragraphe en police Times de taille 12pt</fo:block>.



Signification des abréviations

- AIIM Association for Information and Image Management
- CMS Content Management System
- CSS Cascading Style Sheet
- DAM Digital Asset Management
- DMAM Digital Media Asset Management
- DTD Document Type Definition
- EAI Enterprise Application Integration
- ebXML electronic business XML
- ECM Enterprise Content Management, ECM
- EDI Electronic Data Interchange ou Echange de Données Informatisé
- GED Gestion Électronique de Document
- GEIDE Gestion Électronique d'Informations et de Documents existants
- HTML HyperText Markup Language
- IA Intelligence Artificielle
- PGI Progiciel de Gestion Intégré
- RH Ressource humaine
- SGC Système de Gestion de Contenu
- W3C World Wide Web Consortium
- WCMS (ou WMS) Web Content Management System
- WYSIWYG What You See Is What You Get
- XML eXtensible Markup Language



Bibliographie

[Bachimont04]

BACHIMONT BRUNO, *Arts et sciences du numérique : ingénierie des connaissances et critique de la raison computationnelle*, Mémoire de HDR, Université de Technologie de Compiègne, 2004.

[Bachimont04b]

BACHIMONT BRUNO, CROZAT STÉPHANE, *Instrumentation numérique des documents : pour une séparation fonds/forme*, revue I3, 2004.

[Bachimont07]

BRUNO BACHIMONT, *Ingénierie des connaissances et des contenus : le numérique entre ontologies et documents*, Lavoisier, Hermès, 2007

[Bachimont96]

BACHIMONT BRUNO, *Herméneutique matérielle et artéfacture*, Thèse en épistémologie, 1996.

[Bachimont98]

BACHIMONT B, CHARLET J, *PolyTex : un environnement pour l'édition structurée de polycopiés électroniques multisupports*, actes du colloque EuroTex'98, France, 1998. [<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPS/28-29-bachimont.ps.gz>]

[Crozat07]

STÉPHANE CROZAT, *Scenari, la chaîne éditoriale libre*, Accès Libre, Eyrolles, 2007.

[Goody79]

GOODY JACK, *La raison graphique : La domestication de la pensée sauvage*, Les éditions de minuit, 1979.

[husserl50]

HUSSERL EDMUND, *Idées directrices pour une phénoménologie*, Gallimard, 1950 (Ideen I, 1913).

[Messenger09]

MESSAGER-ROTA VÉRONIQUE, *Gestion de projet : Vers les méthodes agiles*, Eyrolles, 2007.

[Serda Lab 2007]

SERDA LAB, *Documents et flux numériques : Marché et tendances 2007-2008*, Archimag, 2007.

[Soulez85]

SOULEZ ANTONIA, *Manifeste du Cercle de Vienne et autres récits*, Presses

Bibliographie

Universitaires de France, 1985.

[*Stiegler94a*]

STIEGLER BERNARD, *La technique et le temps, Tome I : La faute d'Épiméthée*, Galilée, 1994.

[*Stiegler94b*]

STIEGLER BERNARD, *La technique et le temps, Tome II : La désorientation*, Galilée, 1994.

[*Winograd89*]

WINOGRAD TERRY, FLORES FERNANDO, *L'intelligence artificielle en question*, Presses Universitaires de France, 1989.

[*AIIM*]

AIIM, *What is ECM ?* [<http://www.aiim.org/about-eclm.asp>]

[*Couillault et Le Foll 2007*]

COUILLAULT A, LE FOLL L, *Livre Blanc : Valorisation de l'information non-structurée*, octobre 2007.

[http://www.aproged.org/Portals/0/LivretBlanc_April_Aproged_Cigref_Oct2007s.pdf]

[*Le Grand Dictionnaire Terminologique*]

<http://www.granddictionnaire.com/>

[*OASIS, 2003*]

JAMES CLARK, JOHN COWAN, MURATA MAKOTO, *RELAX NG Compact Syntax Tutorial*, OASIS, 2003. [<http://www.relaxng.org/compact-tutorial-20030326.html>]

[*Vlist, 2000*]

ERIC VAN DER VLIST, *Guide de référence rapide W3C XML Schema*, 2000. [<http://xmlfr.org/documentation/tutoriels/001219-0001>]

[*Vlist, 2000b*]

ERIC VAN DER VLIST, *Tutoriel XML Schema :Introduction aux schémas W3C XML Schema*, 2000. [<http://xmlfr.org/documentation/tutoriels/001218-0001>]

[*w_scenari-platform.org02*]

SYLVAIN SPINELLI, *WYSIWYM (what you see is what you MEAN)*, <http://scenari-platform.org/trac/scenari/wiki/WYSIWYM>, 2006.

[*w_w3c.org/TR/2008/REC-xml*]

W3C, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, <http://www.w3.org/TR/2008/REC-xml-20081126/>, 2008.

[*w_w3c.org/XML*]

XML, <http://www.w3.org/XML>.

[*w_wikipediaA*]

Single Source Publishing, WIKIPÉDIA, consulté en mai 2008
[http://en.wikipedia.org/wiki/Single_source_publishing]

[*W3C, 2000*]

W3C, *XML Schema Part 1: Structures*, 2000. [<http://www.w3.org/TR/2000/CR-xmlschema-1-20001024/>]