Rapport d'étude

projet NF29 A2011

Sommaire

Introduction

Contexte

Ce rapport présente l'assistance à la maîtrise d'ouvrage effectuée dans le cadre de l'atelier projet NF29 du semestre A2011.

Cet atelier s'inscrit dans le projet **C2M**, qui étudie l'ajout de fonctions collaboratives au système de gestion de chaînes éditoriales **Scenari**, et plus précisément dans le sous-projet **Doctek**, qui s'intéresse au champ de la documentation technique comme processus d'écriture à la fois structurée et collaborative.

Organisation

Une première phase de travail s'est articulée autour de trois axes :

- l'analyse d'un corpus documentaire (manuels de vol de l'APM 20 Lionceau),
- un état de l'art des logiciels existants pour la rédaction de documentation technique,
- une veille technologique sur le langage DITA.

Preuve de concept

L'assistance à la maîtrise d'ouvrage a eu pour but la réalisation d'un *proof of concept* (POC) devant regrouper :

- un éditeur Scenari pour la rédaction de contenus structurés de documentation technique,
- des publications sur des supports variés (papier, web, mobile),
- un lien éventuel avec DITA.

Participants au projet

L'équipe de l'AP est composée de Morgane Hedin, Diane Wakim, Yacine Badiss, Sylvain Coulombel, Pierre Cuni, Léonard Dumas, Alexandre Fouchs, Clément Haudecoeur et Guillaume Swiatek. Le travail a été réalisé en collaboration avec Stéphane Crozat et Vincent Carpentier.

La preuve de concept et ce présent rapport sont à destination de la société **Kelis**.

I DITA

1 Darwin Information Typing Architecture

Qu'est-ce que DITA ?

DITA, ou Darwin Information Typing Architecture, est une architecture pour la création de documentation technique basée sur XML (standard XML public depuis 2001, standard OASIS depuis 2005).

Etant donné que le but du projet était de réaliser un preuve de concept permettant la création de documents techniques et que l'objectif était de se baser sur un éditeur Scenari, dont le format de sortie est le XML, il semble logique que DITA soit une solution envisagée pour servir de base à notre projet.

Cela dit il existe d'autres solutions pour faire de la documentation technique en XML, en particulier avec DocBook qui est à la fois un format SGML et XML.

Pourquoi DITA ?

DocBook a en réalité une approche narrative et continue, en permettant de décrire un travail de manière très détaillée.

Au contraire, DITA repose sur la notion de **topic**, c'est-à-dire un sujet, qui correspond à la granularité idéale pour permettre la réutilisation sans compromettre l'efficacité de l'auteur. Le fait d'avoir une approche par **topic** permet une production de contenus morcelés, et favorise ainsi la réutilisation, ce qui correspond tout à fait à la philosophie de Scenari, et justifie donc son utilisation au sein de ce projet.

2 La spécialisation en DITA

Un concept au cœur de l'architecture DITA

DITA signifie Darwin Information Typing Architecture. L'emploi de Darwin au sein de l'acronyme, en référence au célèbre naturalise anglais auteur de *«De l'origine des espèces»* et père de la théorie de l'évolution, montre toute l'importance portée au concept d'héritage rendu possible en DITA.

Grâce à la spécialisation on peut définir ses propres contenus, mais cela est déjà possible avec du XML classique, l'intérêt d'avoir un mécanisme d'héritage est qu'un moteur de publication prévu pour les éléments de base permettra également l'affichage des éléments spécifiques à une organisation ou à un projet.

Si vous désirez mieux comprendre la spécialisation ou approfondir le sujet, veuillez vous

référer à la notice technique de la spécialisation DITA en annexe *(cf. annexe 1 page 56)* qui présente en détail ce mécanisme et explique comment réaliser sa propre spécialisation.

Pourquoi la spécialisation est importante dans notre contexte

Nous n'aurons a priori pas besoin d'utiliser directement la spécialisation au sein de ce projet. En effet, la limite de la spécialisation est qu'on ne peut jamais ajouter d'éléments, car la spécialisation par définition va toujours vers le plus spécifique. Dans les faits, lorsque l'on créé un nouvel élément on fera toujours explicitement le lien vers l'élément déjà existant (père) auquel on se rapporte et dont on hérite.

Et donc si l'on est capable d'exporter des données depuis Scenari vers un modèle spécifique DITA à l'aide d'une transformation XSL, alors c'est qu'on est aussi capable de le faire pour un modèle plus général de DITA.

Cependant, il est très important dans le cadre de ce projet de bien comprendre la spécialisation. D'une part parce qu'il s'agit d'un concept fondamental de la DITA. D'autre part, lorsqu'un utilisateur utilise le mécanisme de spécialisation, il a ses motivations. Il le fait généralement pour 2 raisons :

- pour bénéficier d'un langage orienté métier
- pour bénéficier de ses propres publications adaptées à son usage

Dans le cas où un client qui a spécialisé son contenu se présente, le premier point est facilement gérable, puisqu'il s'agit d'un simple renommage. Cependant le second point est très important. En effet, il faut absolument être capable d'exporter le contenu Scenari vers le modèle DITA du client, sans quoi il perdra la possibilité d'utiliser ses publications et n'aura donc aucun intérêt à utiliser le modèle Doctek.

3 Thésaurus DITA

Voici un thésaurus réalisé lors de la première phase d'analyse afin de fixer les termes importants relatifs à la DITA.

Architecture

Une **architecture** est une description de la structure d'un document.

La structure est une décomposition du contenu documentaire en éléments de bases (titre, chapitre, section, exemple...)

L'**architecture** décrit de quelle manière (ordre, hiérarchie, cardinalité...) doivent s'arranger ces briques élémentaires pour former le document.

Non Descripteur

Voir : DITA, DocBook

Concept

Un **concept** permet d'exprimer une connaissance à mobiliser. La racine du fichier XML d'un **concept** est un élément *concept* et son corps est un élément *conbody*.

Est-un :Topic

Partie-de : Map, DITA

DITA

Darwin Information Typing Architecture.

Il s'agit d'une architecture pour la création de documentation technique basée sur XML. (standard XML public depuis 2001, standard OASIS depuis 2005)

DITA repose sur la notion de **topic** qui correspond à la granularité idéale pour permettre la réutilisation sans compromettre l'efficacité de l'auteur. Il convient donc a une production de contenus morcelés.

Utilisation :

- Pour décrire un ensemble complexe de **topics**.
- Pour les applications qui nécessitent de l'extensibilité et de l'interopérabilité.

Est-un : Architecture

Partie-de : Documentation technique

DocBook

DocBook est un langage à balise permettant de créer de la documentation technique par une approche narrative et continue, pour ainsi décrire un travail de manière détaillée.

C'est à la fois un format SGML et XML, et depuis la version 5.0 DocBook a également été porté en XML Schema et Relax NG.

Partie-de : Documentation technique

Documentation technique

Document comportant de l'information relative à un processus technique ou un produit, créé pour des utilisateurs ciblés (concernés par le processus ou utilisant le produit) au travers d'un média. L'information doit être particulièrement pertinente pour le lecteur.

DITA et **DocBook** sont des langages XML spécialisés dans la réalisation de documentations techniques.

Fragment

Partie signifiante d'un document.

Non Descripteur

Voir : Topic

Map

Une **map DITA** est un assemblage de différentes références de **topic** (*topicrefs*) pour une publication donnée (aide en ligne, manuel d'utilisation papier, etc.).

Les **topics** peuvent être organisés sous forme de liste ou de façon hiérarchique (*topicrefs* imbriqués). Un document **DITA** peut contenir plusieurs **map**, et une **map** peut contenir d'autres **map**.

Partie-de :DITA

Métadonnées

Les **métadonnées** sont des données servant à décrire un document (auteur, date, ...) ou de l'information au sein d'un document.

DITA permet de spécifier des **métadonnées** au niveau d'un **topic** ou d'une **map**. Si on en définit dans les 2 alors ce sont celles de la **map** qui compteront.

Partie-de :DITA

Reference

Une **reference** donne des informations relatives à un langage de programmation ou les faits à propos d'un produit (focus sur les propriétés et les relations entre items similaires). Par exemple des fonctions ou des commandes. De la même manière que les autres types de **topic** une **reference** comporte un titre, le plus souvent une courte description et un corps qui est un élément *refbody*. Ce dernier peut en particulier contenir un élément *refsyn* qui est une section spéciale pour la syntaxe ou la signature d'un contenu.

Est-un : Topic

Partie-de : Map,DITA

Spécialisation

Processus par lequel de nouvelles entités sont créées en se basant sur des entités déjà existantes, ce qui permet aux nouvelles entités d'être gérées par les fonctions déjà existantes, et qui se caractérise par l'héritage (les nouvelles entités étant plus restrictives que les entités déjà existantes). La **spécialisation** permet de définir de nouveaux types d'information en réutilisant aux maximum les structures existantes. Il existe 2 types de **spécialisation** : la **spécialisation de type** et la **spécialisation de domaine**.

Spécialisation de domaine

La **spécialisation de domaine** correspond à la création de nouveau domaine d'information. Contrairement à la **spécialisation de type**, la **spécialisation de domaine** s'applique à des éléments qui se trouvent au sein d'une **map** ou d'un **topic**. Par exemple on pourra spécialiser un élément existant au sein d'un **topic** et créer un *specializedParagraph* qui corresponde à notre domaine d'activité à partir d'un *p*. L'élément spécialisé ne pourra alors apparaître que là où l'élément déjà existant était

autorisé. (comme *p* est autorisé dans un *conbody*, dans l'*example* d'un *conbody* et dans le *postreq* d'un *taskbody* -entre autre- il en sera de même pour *specializedParagraph*).

Il existe 4 spécialisations de domaine de base : *highlight*, *programming*, *software* et *user interface* qui ajoutent de la sémantique aux éléments existant dans les **topics** et **map** de base.

Est-un :Spécialisation

Spécialisation de type

Une spécialisation de type correspond à la **dérivation** des types DITA standards (avec **map** ou **topic** en racine) pour des besoins spécifiques d'une organisation (formats de travail). Pour garantir l'échange de contenus entre deux organisations ayant des formats de travail différents, chaque type spécialisé est considéré comme le type général (commun aux deux organisations) le plus proche : il s'agit du **polymorphisme**.

Est-un : Spécialisation

Sujet

Brève description de ce que traite un contenu.

Non descripteur

Utiliser : Topic

Task

Une **task** définit les moyens de réaliser une tâche précise. La racine du fichier XML d'une task est un élément "task" et son corps est un élément "taskbody". Ce dernier contient les éléments "context" (contexte de la tâche), "prereq" (le pré-requis de la tâche), "steps" (élément constitué de plusieurs "step" - étape de la tâche) et "result" (résultat final de la tâche).

Synonymes : Tâche

Est-un : Topic

Partie-de : Map, DITA

Thème

Idée développée dans un contenu.

Non descripteur

Utiliser : Topic

Topic

Un **topic** est un ensemble d'informations relevant d'un même thème (ou sujet). Il peut être réutilisé et son contenu doit être signifiant par lui-même et ce indépendamment du contexte d'utilisation (de la **map** dans laquelle il est utilisé). Un **topic** possède trois

éléments obligatoires : un identifiant (*id*), un titre (*title*) et un corps (*body*). Concrètement il s'agit donc d'un titre suivit de texte ou d'images, éventuellement organisé en sections. Enfin, un ensemble de liens vers d'autres topics (*related-links*) peut être ajouté au **topic**. **DITA** fournit trois types standards qui spécialisent le type **topic** : **concept**, **task**, et **reference**.

Synonymes : Sujet, Thème

Est-un : Fragment

Partie-de : Map, DITA

4 DITA Open Toolkit

Au cours de cette phase, nous avons été amené à tester l'**Open Toolkit** : il s'agit d'un outil permettant de publier du contenu DITA à l'aide d'une librairie assez complète de schémas (DTD, XSD), d'exemples de fichiers DITA (*topics*, *maps*, etc.), de transformations xsl, de styles (css) et de publications exemplaires (HTML, PDF).

Publication standard des contenus DITA dans Oxygen

Avec Open Toolkit, la publication se fait à l'aide d'Ant et donc d'un fichier de configuration (*build.xml*) ; nous avons procédé ainsi pour lancer la génération des contenus exemplaires, sans réussir cependant à le faire pour nos propres contenus.

Par ailleurs dans Oxygen, les contenus DITA peuvent être publiés dans l'export standard HTML ou PDF de l'Open Toolkit, ce dont nous nous sommes finalement servis pour réaliser nos premiers essais de publication.

Intérêts pour la publication dans le cadre de notre projet ?

En premier lieu, nous avons pensé qu'il pouvait être utile de réaliser nos publications en nous basant sur ce qui existait déjà dans l'Open Toolkit, et ainsi reprendre les transformations "standard" pour les adapter comme nous le souhaitions.

Cependant, nous avons rapidement été confronté à l'architecture massive d'Open Toolkit et à la difficulté de réaliser le point ci-dessus ; c'est pourquoi nous avons finalement décidé de ne pas se baser sur Open Toolkit mais bien de réaliser entièrement nos publications (ce qui signifiait "repartir de zéro").

La DITA Open Toolkit est téléchargeable à cette *adresse*[1].

1 - http://dita-ot.sourceforge.net/

II Editeurs

1 Editeurs : Etat de l'art

Introduction

Nous présentons ici l'étude de plusieurs éditeurs permettant de faire de la documentation technique et/ou de la DITA. Voici une description détaillée des deux principaux éditeurs qui présentent selon nous, le plus de fonctionnalités intéressantes, à savoir Author IT et ArborText.

2 Author-IT

Informations générales

Author-IT est une solution de gestion de contenus intégrant les fonctionnalités nécessaires à la publication multi-supports et multi-formats.

Cinq versions majeures de l'éditeur ont vu le jour depuis sa première mise sur le marché en 1997.

L'outil est utilisé à travers 50 pays différents. Il est l'un des outils de gestion de contenus les plus largement utilisés dans le monde.

Sa principale caractéristique est de faciliter la création, la gestion et la publication de tous les contenus d'entreprise.

Le passage à Author-IT est envisageable à toute entreprise qui le souhaite grâce à la fonctionnalité d'import de contenus présent dans l'éditeur.

Les multinationales peuvent, quant à elles, profiter de la gestion des langues.

Fonctionnalités

Author-IT propose un large panel de fonctionnalités. Outre l'aspect **collaboratif** qui est détaillé dans le paragraphe "Fonctions de collaboration", cet éditeur propose de stocker différents objets dans une **base de données** (books, topics, file objects, hyperlinks, styles, glossaries, tables of contents, indexes, and publishing profiles). La volonté des développeurs de Author-IT est de donner le pouvoir à l'utilisateur de réutiliser des fragments de contenu. Cette fonctionnalité est bien présente et plutôt simple d'utilisation puisque la réutilisation du contenu se fait par un simple **glisser-déposer**. Par ailleurs, l'éditeur embarque un environnement de **gestion de projet** avec **gestion des profils** (pour l'aspect sécuritaire) et des fonctionnalités **d'audit et d'historique**. Enfin, il offre des outils pour la **gestion des métadonnées** et **des bases de données**.

Fonctions de collaboration

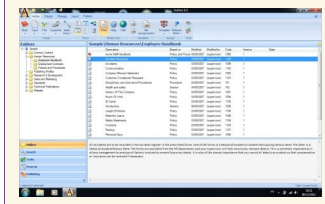
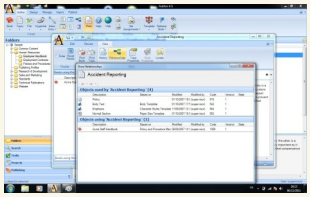
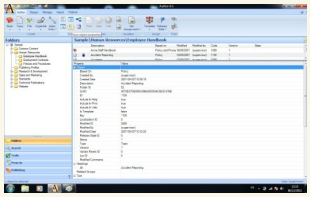
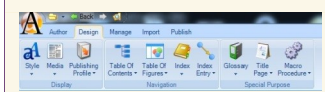
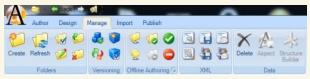
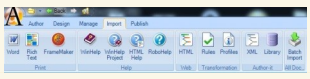
Author-IT dispose des fonctionnalités de base pour la rédaction (et validation) collaborative :

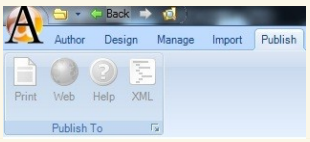
- Possibilité pour **plusieurs utilisateurs** de **voir** et **modifier plusieurs sujets en même temps**
- **Aucun logiciel à installer** pour revoir du contenu, s'ouvre dans un navigateur web
- Fonctionnalité de **fils de discussion** (possibilité de suggérer) et **mesure d'approbation** via des boutons "d'accord" ou "pas d'accord"
- **Fonctions de suivi** et **de notifications automatiques**

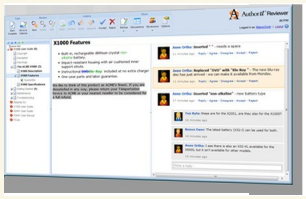
Interface

L'interface est conviviale et ressemble aux traitements de texte puisque les balises sont cachées à l'utilisateur.

Galerie d'images Author-It





Publication

Les formats de publication proposés par Author-IT sont nombreux :

- RTF, PDF ou Microsoft Word
- Microsoft WinHelp (Windows Help)
- Microsoft HTML Help
- JavaHelp
- Oracle Help for Java
- Web and browser based help
- XML
- DITA

Sources

- *Article Wikipédia*[2]
- *Site officiel*[3]

3 ArborText

Informations générales

Arbortext PTC est un ensemble de différents modules pour créer, gérer, publier de la documentation technique. Chacun des modules se place d'un point de vue utilisateur différent :

- ArborText Editor : création du contenu structuré ;
- ArborText IsoDraw et Creo Illustrate: création des illustrations techniques ;
- ArborText ContentManager : gestion de versions et du workflow ;
- ArborText Styler : création de feuilles de styles pour les différentes publications ;
- ArborText Publishing Engine et Advanced Print Publisher : publication sur différents supports.

2 - http://en.wikipedia.org/wiki/Author-it
3 - http://www.author-it.com/index.php?page=authoring

Il est possible de télécharger la *version d'évaluation d'ArborText Editor + Styler*[4] sur le site de PTC.

Fonctionnalités

Parmi les fonctionnalités de l'éditeur, on retrouve la création **composants réutilisables**, la possibilité de faire des **liens intra- et extra-document**. Il y a **différents modèles disponibles** ; on peut aussi créer ses propres modèle via une interface de définition de schémas, et l'éditeur supporte également les standards DITA et DocBook.

On retrouve également une **gestion des versions** des documents, ainsi que du workflow. Avec ArborText Content Manager, il est possible de gérer la **collaboration** d'équipes distantes, en ayant un dossier de dépôt gérant la gestion de versions.

Un élément notable est le fonctionnement d'ArborText IsoDraw, le module de création d'illustrations techniques. Les illustrations peuvent être crées à partir de zéro, ou bien automatiquement à partir de données de conception 2D ou 3D. Lorsque les données sont modifiées, les illustrations sont mises à jour automatiquement.

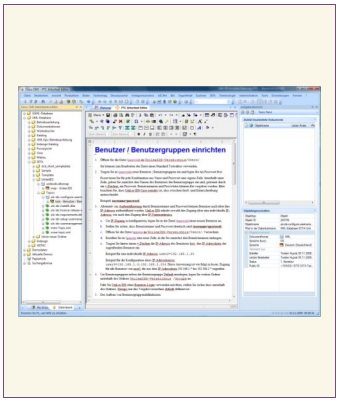
IsoDraw facilite le processus de création et de mise à jour pour des illustrations telles que des instructions d'assemblage, des coupes de pièces, etc.

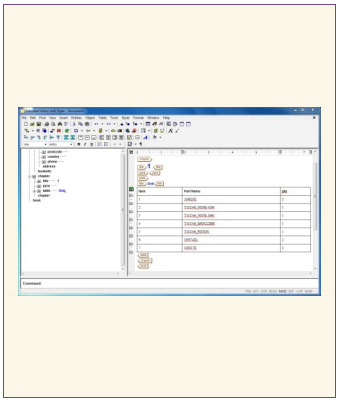
Creo Illustrate permet de créer ou d'interpréter des données 3D pour générer des illustrations techniques en 3D. Celles-ci peuvent être publiées sous forme d'animation.

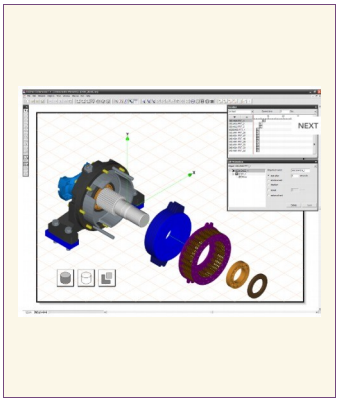
Interfaces

L'interface de l'éditeur est semblable à un traitement de texte classique. On peut avoir également une vue arborescente du document, ainsi que le marquage des balises XML.

Galerie d'images Arbortext PTC







4 - http://www.ptc.com/products/arbortext/editor/eval.htm

Publication

Différentes formes de publication sont possibles avec ArborText Publishing Engine :

- HTML, HTML Help ;
- PDF, Microsoft Word, EPUB.

Avec Advanced Print Publisher, il est apparemment possible de publier à partir de la même source en plusieurs langues. On a aussi la possibilité de modifier manuellement les publications papier a posteriori.

Sources

- *Site officiel*[5]
- *Pourquoi choisir Arbortext ?*[6]
- *Article Wikipédia (de)*[7]

4 Autres éditeurs

Les autres éditeurs de documentation technique que nous avons étudiés sont les suivants :

- *Adobe FrameMaker 10*[8] ;
- *Schema ST4*[9] ;
- *X-Metal*[10].

Ils proposent globalement tous les même fonctionnalités de base :

- Création de contenu structuré ;
- Structuration en contenus réutilisables ;
- Publication multi-supports à partir d'une source unique.

Leur principal problème est le **manque d'ergonomie** : l'auteur est très peu guidé lors de la rédaction. Il est difficile d'appréhender l'ensemble du document, et de comprendre quelles balises et quelles fonctionnalités utiliser à quel endroit.

5 - http://www.ptc.com/product/arbortext/
6 - http://www.4cad.fr/files/arbortext.pdf
7 - http://de.wikipedia.org/wiki/Arbortext_3B2
8 - http://www.adobe.com/fr/products/framemaker/features.html
9 - http://www.schema.de/index.php?option=com_k2&view=item&layout=item&id=21&Itemid=26&lang=en
10 - http://na.justsystems.com/content-xmetal-author

III Modèle conceptuel

1 Modèle documentaire du manuel de vol

Motivations

Avant l'élaboration de notre preuve de concept, il s'est révélé essentiel de déterminer un contenu exemplaire à confronter au modèle à produire. En effet, confronter la preuve de concept à un cas concret permet de tester son adéquation avec des conditions réelles.

Nous avons donc étudié différents corpus de documentations techniques (automobile, moto et portail électrique), mais deux problèmes ont émergé :

- Il n'était jamais précisé si les documents étaient libres de droit ou non, ce qui posait un problème majeur
- Ces documents ne présentaient que peu de concepts et de procédures, et ne présentaient aucune donnée complexe (tableaux,...)

Au vu de ces problèmes, et ayant abordé le thème de l'aviation en début de projet, nous avons donc finalement choisi d'étudier un corpus de manuels de vol de la société Issoire Aviation.

Les manuels de vol sont des documents techniques, qui décrivent en détail les caractéristiques de l'avion qui y est associé (masse, centrage,...) ainsi que ses procédures normales d'utilisation, d'entretien et d'urgence. Chaque document est composé de nombreux types d'objets, ce qui permet d'aborder le problème du modèle de la chaîne éditoriale de manière très vaste. Même s'il s'agit d'un document orienté métier, son modèle sera réutilisable dans d'autres cas de notices techniques, après spécialisation.

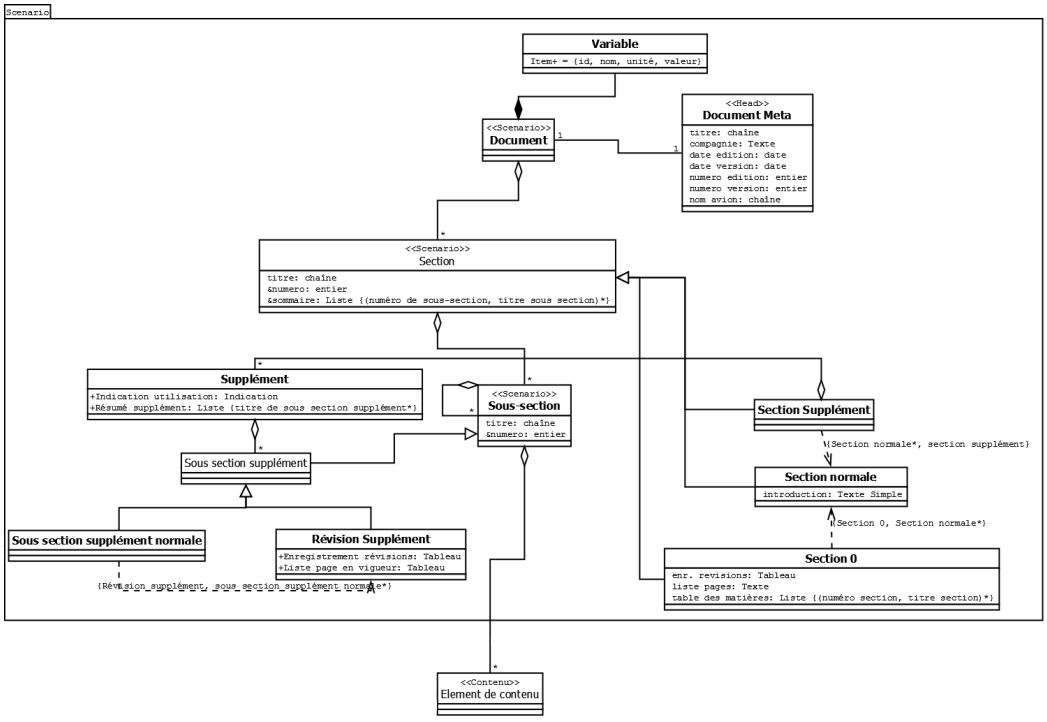
Organisation

Nous avons effectué l'analyse de ce corpus en parallèle de la veille technologique sur la DITA, sans en prendre connaissance pour ne pas introduire de biais. Ensuite, nous avons comparé nos deux approches afin de savoir si DITA était à même de décrire le manuel. Il s'est avéré qu'hormis la notion de variable/constante (cf. schémas UML ci-dessous), la DITA était effectivement suffisante pour représenter et structurer le corpus.

Scenario du manuel

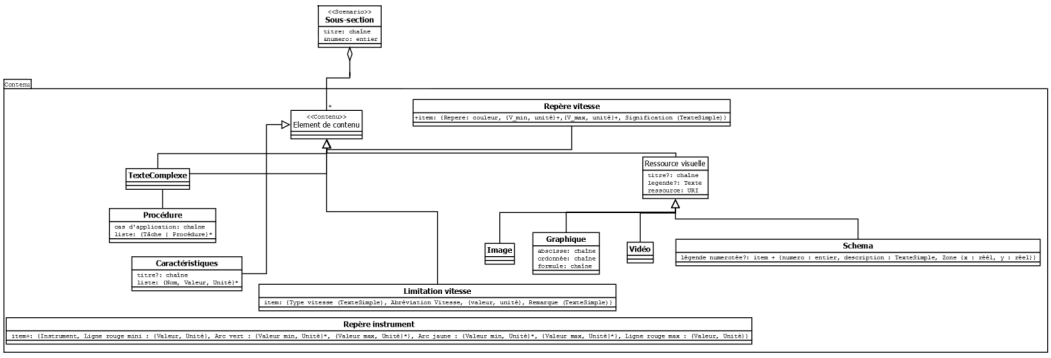
L'organisation globale du manuel de vol est décrite ci-après. Le manuel est divisé en sections, dont deux sont remarquables : la section0 qui regroupe les modifications ayant été opérées sur le manuel, et la section supplément qui regroupe de nombreuses

données, tel un second manuel inséré dans le premier. Le seul élément du scenario pouvant contenir du contenu à proprement parler est la sous-section. Chaque section contient des sous-sections.



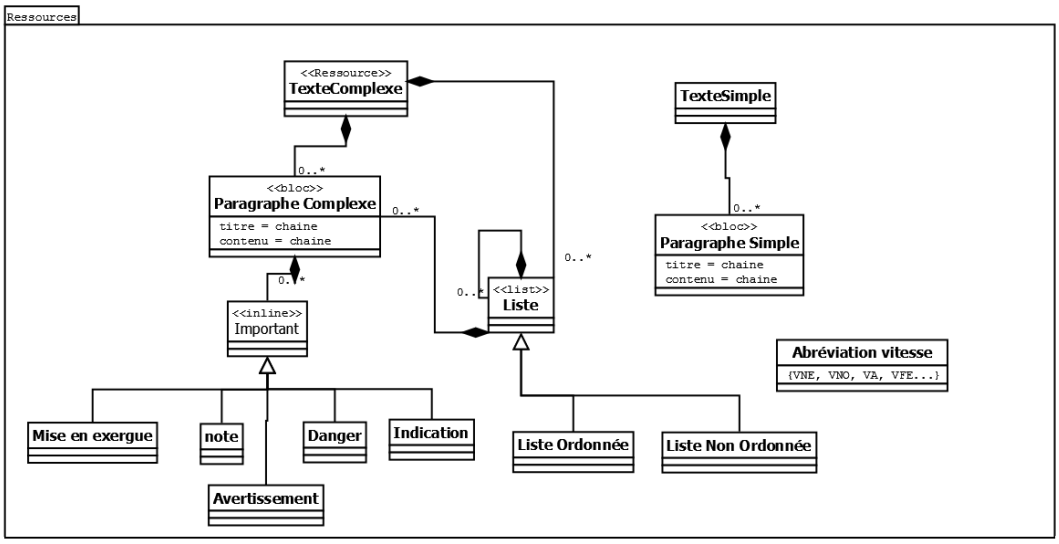
Contenu du manuel

Le manuel contient des informations communes à toutes les documentations techniques, telles que des ressources visuelles et des procédures. Cependant, il contient aussi des informations orientées métier, comme les repères et les limitations de vitesse. Comme dit précédemment, la sous-section est formée par le contenu du manuel. Elle peut contenir n'importe quel élément de contenu décrit ci-après.



Ressources

Les ressources constituent la forme du contenu du manuel. Il s'agit des types utilisés dans la partie contenu :



IV POC

Introduction

La réalisation du POC s'est faite à partir de l'éditeur Scenari-Doctek développé par Vincent Carpentier. La chaîne complète envisagée est la suivante :

- rédaction des éléments de documentation technique dans l'éditeur,
- transformation de la structure XML propre à Scenari en une structure de langage DITA,
- publication pour les trois supports à partir des fichiers DITA générés précédemment.

Si la première étape se fait au préalable, les deux dernières se font simultanément à l'aide d'un seul fichier de génération.

Nous commencerons par présenter l'architecture générale du système avant de détailler ses différentes fonctions.

1 Architecture générale du système

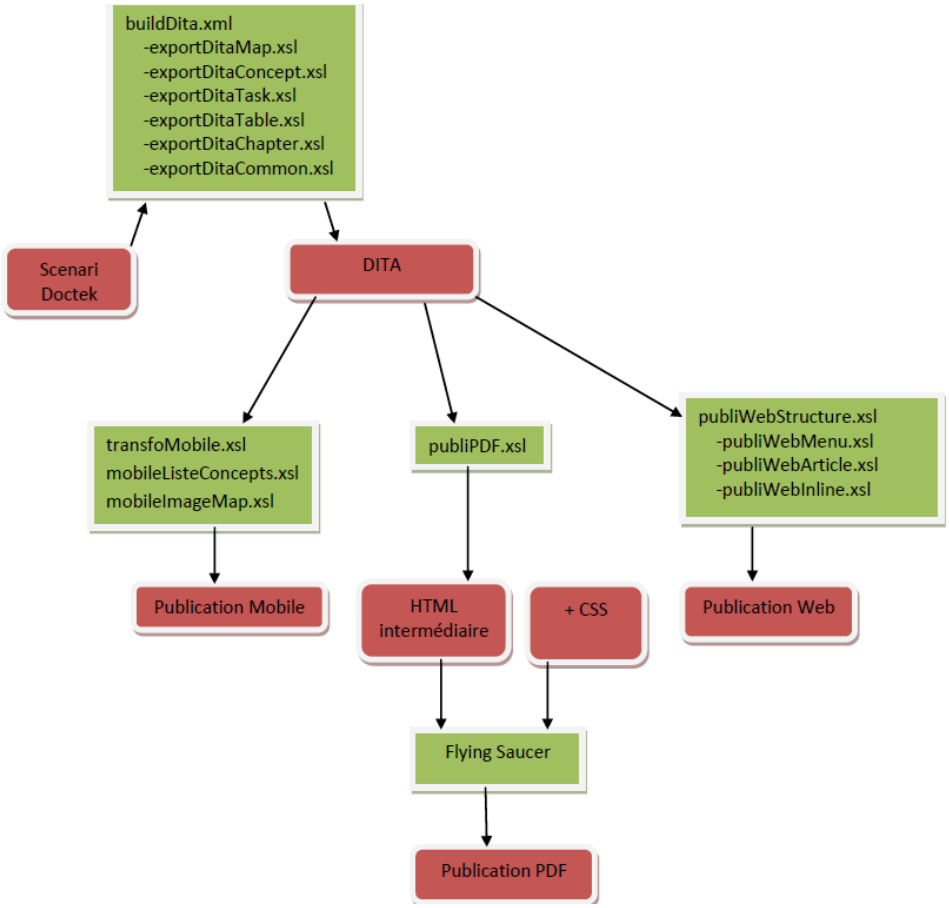
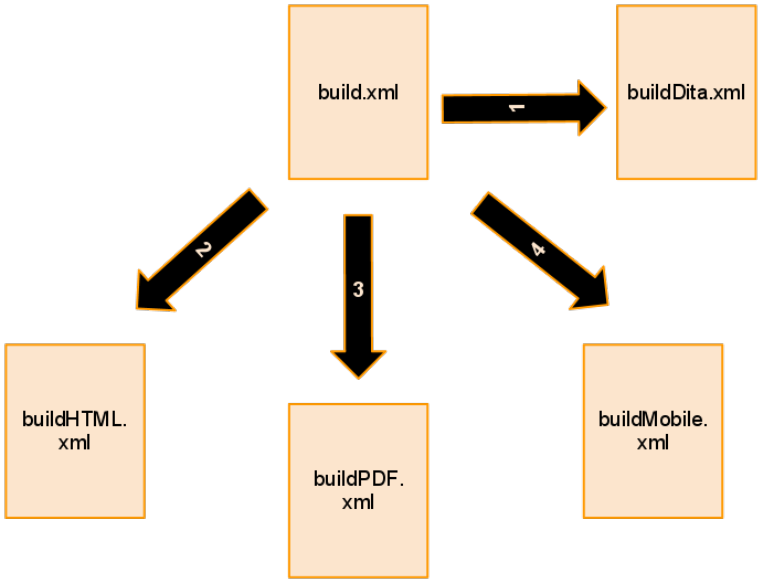


Schéma de l'architecture complète du système : rédaction sous Scenari, export Dita, publication sous trois supports

1.1 Publication globale avec Ant

Architecture générale des générations automatisées via ANT



2 Export du modèle Scenari-Doctek vers Dita

Introduction

Après l'analyse des manuels de vol et la phase de veille sur Dita, il nous a paru possible de structurer le contenu de cette documentation à l'aide de ce langage. D'un autre côté, Dita s'est imposé comme une solution assez souple pour les publications que nous souhaitions faire, en plus de l'avantage qu'on peut en tirer du fait que c'est un standard, à savoir la conformité et la comparabilité à d'autres documentations techniques structurées en Dita.

Ainsi, nous avons décidé de réaliser une transformation du modèle Scenari-Doctek vers Dita, puis de nous appuyer sur les fichiers Dita générés pour les publications finales.

L'éditeur Doctek et les fichiers .xml sources nous ont permis un export Dita résumé par le diagramme UML suivant :

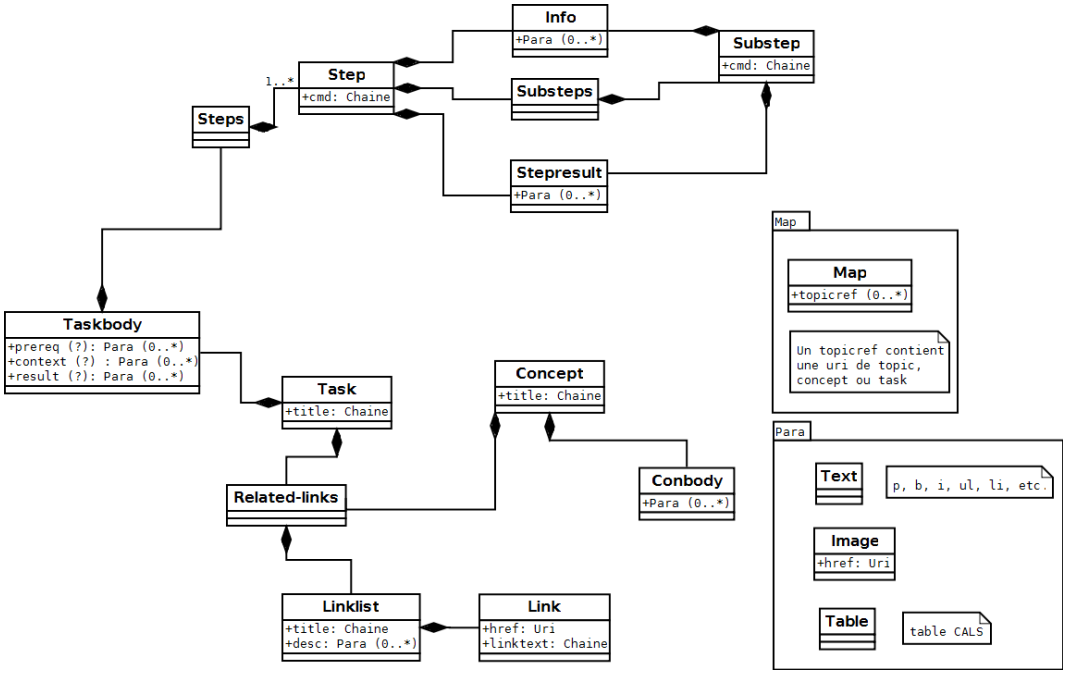


Diagramme UML de la Dita que nous utilisons

Démarche

Le modèle Scenari-Doctek propose entre autres la création de 4 types d'items, à savoir les maps, les chapters, les concepts et les tasks. Pour chacun d'entre eux, une transformation spécifique en langage Dita a été nécessaire, ce qui a mené aux fichiers suivants :

- exportDitaMap.xsl,
- exportDitaChapter.xsl,
- exportDitaConcept.xsl,
- exportDitaTask.xsl.

2.1 Export des concepts et des tâches Dita

Mise en commun des transformations similaires

Les transformations des *concepts* et des *tasks* ont a gérer un certain nombres de traitements similaires : en effet, le modèle Dita restreint que nous utilisons (cf. diagramme UML) met en jeu des éléments communs tels que ceux liés au titre (*title*, *titleats*, etc.), aux notes (*info*, *note*, *danger*, *tip*, etc.), aux liens et références (related-links, xref, etc.), ou encore au texte et aux balises "inline" (*p*, *b*, *li*, *image*, etc.).

Nous avons donc décidé de "mutualiser" les transformations de ces éléments en les incluant dans un fichier commun (exportDitaCommon.xsl) qui est référencé à la fois dans exportDitaConcept.xsl et dans exportDitaTask.xsl. Cela nous a permis de réutiliser les traitements similaires ainsi que de mieux modulariser notre code.

Transformation des liens

La transformation des liens (attributs *href* dans les balises *link* ou *xref* typiquement) a été un point particulièrement important : en effet dans le modèle Scenari, l'attribut *refUri* d'un item de l'atelier Doctek (ayant par exemple une extension "* .concept" ou "*.task") contient un chemin vers un fichier lui aussi situé dans l'atelier (ayant donc le même genre d'extension). Puisque nous voulons transformer le fichier référençant dans le langage Dita, il convient de transformer le lien en une référence à un fichier Dita.

Ceci est possible grâce à un jeu sur les extensions de fichier : nous procédons simplement en remplaçant l'extension de base par une extension "*.dita", tandis que le nom du fichier reste le même. Cette substitution fonctionne à condition que les hypothèses suivantes soient vérifiées :

- les fichiers référencés respectivement dans le modèle Doctek et en Dita doivent posséder le même nom,
- la transformation du fichier référencé depuis le modèle Doctek vers Dita doit aussi avoir lieu en parallèle, sans quoi le lien en Dita ne mène vers rien.

Ces deux conditions sont respectées grâce à l'exécution en parallèle de toutes les transformations avec Ant.

2.2 Export des Map et des Chapter en DITA

Schéma classique d'une map DITA

Une *map* DITA est un assemblage de différentes références vers des *topics* : des *topicrefs*. Ces *topics* peuvent être organisés hiérarchiquement, avec des *topicrefs* à l'intérieur d'un *topicref*. L'exemple ci-dessous du fragment d'une *map* DITA comporte, dans l'ordre :

- Une référence simple vers le *topic* Introduction.xml ;
- Une référence vers le *topic* donnesImportantes.xml, qui contient lui même les *sous-topics* distancesAterrissage.dita et etalonnage.dita ;
- Une référence simple vers le *topic* informationsSupplementaires.xml

Fragment d'une map DITA

```
<map title="Section 5 : Performances" format="" chunk="to-content">
<topicref href="Introduction.xml"/>
<topicref href="donneesImportantes.xml">
<topicref href="distancesAterrissage.dita"/>
<topicref href="etalonnage.dita"/>
</topicref>
<topicref href="informationsSupplementaires.xml"/>
```

```
</map>
```

Dans le modèle Scenari Doctek, la structure d'un fichier .map est quasiment la même que dans DITA. La différence principale est la gestion des sous-topics, sous la forme d'élément Scenari *chapter*. L'élément *chapter* de Doctek contient les références vers les *sous-topics*, qui ne sont donc pas dans le fichier Scenari de la map. La *map* Scenari correspondant à la *map* Dita présentée aura des références vers les *topics* simples, ainsi que vers le *chapter*, mais pas vers les *sous-topics* contenus dans le *chapter*.

Fragment de map Doctek

```
<kd:map xmlns:sp="http://www.utc.fr/ics/scenari/v3/primitive" xmlns:sc="http://www.utc.fr/ics/scenari/v3/core" xmlns:kd="kelis.fr:doctek">
<kd:mapM>
<sp:title>Section 5 : Performances</sp:title>
</kd:mapM>
<sp:topicRef sc:refUri="/EspaceDoctek/Introduction.task"/>
<sp:topicRef sc:refUri="/EspaceDoctek/chap_01.chapter"/>
<sp:topicRef sc:refUri="/EspaceDoctek/concept1.concept"/>
</kd:map>
```

La transformation d'une *map* Doctek vers une *map* DITA consiste en une réécriture des *topicrefs*, en utilisant la même transformation de liens que pour *concept* et *task*. Pour récupérer les références vers les *sous-topics* qui se trouvent dans *chapter*, il suffit de parser le fichier *chapter* correspondant avec la fonction *document* de XSL.

Notion de Chapter

Dans une *map* DITA, on peut avoir une organisation en *sous-topics*, comme décrit dans la *map* ci-dessus. Dans Doctek, cette hiérarchisation est mise en place par l'élément *chapter* : un *chapter* a un *title*, et peut avoir :

- des *topicrefs* vers des *task* ou *concept* ;
- un *body* avec des informations, paragraphes, etc. ;
- des *nestedTopics*, qui peuvent être des *task*, des *concept*, ou bien des *chapter* (ce qui peut permettre une organisation en sous-chapitres de la *map*). Les *nestedTopics* peuvent être internalisés ou externalisés.

Fragment de chapter Doctek

```
<kd:chapter xmlns:sp="http://www.utc.fr/ics/scenari/v3/primitive" xmlns:sc="http://www.utc.fr/ics/scenari/v3/core" xmlns:kd="kelis.fr:doctek">
```

```
<kd:chapterM>
<sp:title>Données importantes</sp:title>
</kd:chapterM>
<sp:body>
<kd:chapterChunk>
<sp:task sc:refUri="/EspaceDoctek/DistanceAterrissage.task"/>
<sp:task sc:refUri="/EspaceDoctek/etalonnage.task"/>
</kd:chapterChunk>
</sp:body>
<sp:nestedTopic sc:refUri="/EspaceDoctek/Performance.concept"/>
</kd:chapter>
```

Transformation de chapter vers DITA

Les *topicrefs* présents dans le fichier *chapter* de Doctek sont traduits par des *topicsrefs* au niveau de la *map*, au moment où la transformation de la *map* est effectuée.

La transformation du fichier *chapter* en lui-même produit un fichier DITA *topic* contenant tous les éléments du *chapter*, à savoir le *title*, le *body* et les *nestedTopics*. La transformation utilisée ne présente rien de particulier, excepté pour les *nestedTopics*. En effet, en transformant de cette façon les *chapter* Doctek, on se retroouve donc avec un fichier DITA *topic*, dans lequel on ne peut pas intégrer de *concept* ou *task*, selon la DTD de DITA, mais seulement d'autres *topics*. On a choisi ici de "dégrader" les *concept* et *task* des *nestedTopics* en simple *topics*, pour pouvoir les mettre dans le fichier *topic* correspondant au *chapter*.

Dans l'exemple de *chapter* ci-dessus, les deux *task* dans le corps du *chapter* se retrouveront dans le fichier de la Map DITA. Le *nestedTopic*, qui est le *concept* Performance.concept, sera intégré dans le fichier du *topic* correspondant au *chapter*, sous la forme d'un *sous-topic* qui contiendra le contenu de Performance.concept.

Transformation d'une map plus complexe

Pour l'instant, il n'y a pas de gestion des sous-sous-chapitres : si un *nestedTopic* est un *chapter*, il n'est pas traité. Deux solutions paraissent possibles :

- Les sous-chapitres qui se trouvent dans les *nestedTopics* sont parsés au moment de la transformation de la *map* : on aura dans la *map* la hiérarchie de tous les *topics* et *sous-topics*, et chaque *chapter* sera transformé en un fichier *topic*;
- La transformation de la *map* ne parse que le premier niveau de *chapter* : les sous-chapitres présents dans les *nestedTopics* sont alors transformés en *topics* internes (à savoir en balises *topic* au sein du fichier *topic* du *chapter*).

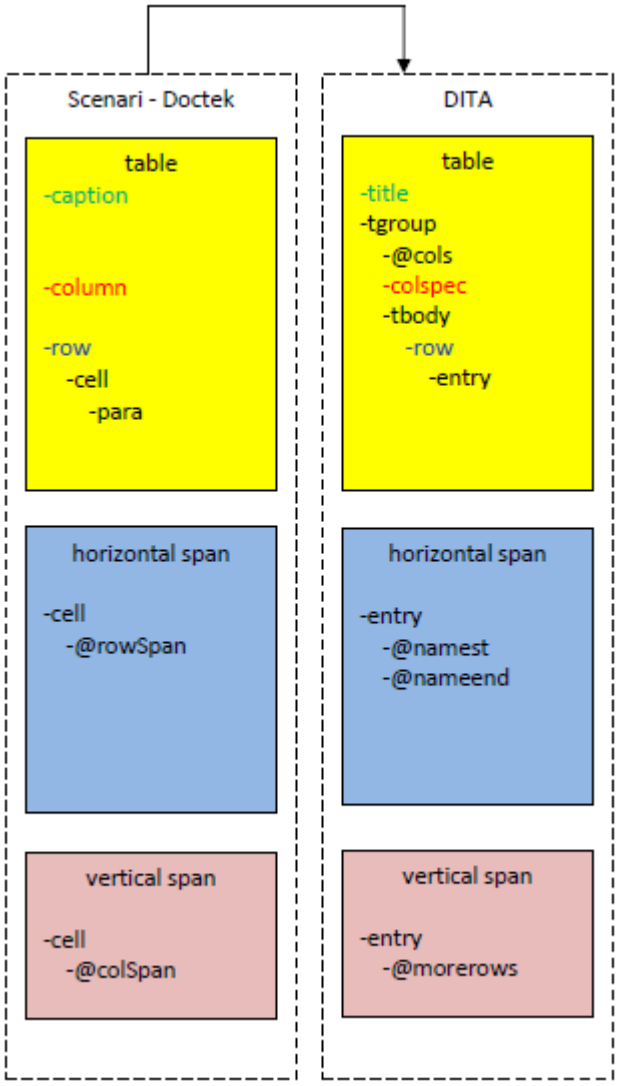
2.3 Export des tableaux de Doctek vers DITA

Transformation des tableaux

Transformation relativement complexe car les tableaux peuvent contenir des cellules fusionnées, ce qui se traduit par des *spans* horizontaux ou verticaux.

La difficulté principale est que le mode de représentation des *spans* horizontaux en *DITA* est différent de la façon de faire sous Scenari, et la transformation n'est donc pas triviale.

Voici un résumé de la transformation à effectuer :



cols correspond au nombre maximum de cellules « unitaires » (sans *span*) sur une ligne.

Les valeurs entrées dans *namest* et *nameend* sont des valeurs déclarées dans des *colspec* et doivent correspondre à la colonne de début et celle de fin suivant la plage sur laquelle on veut que la cellule s'étende.

Solution apportée

Voici la solution que nous avons adoptée pour résoudre le problème :

```
<xsl:template match="sc:column">
<colspec colname="{count(preceding::sc:column) +1}"/>
</xsl:template>
```

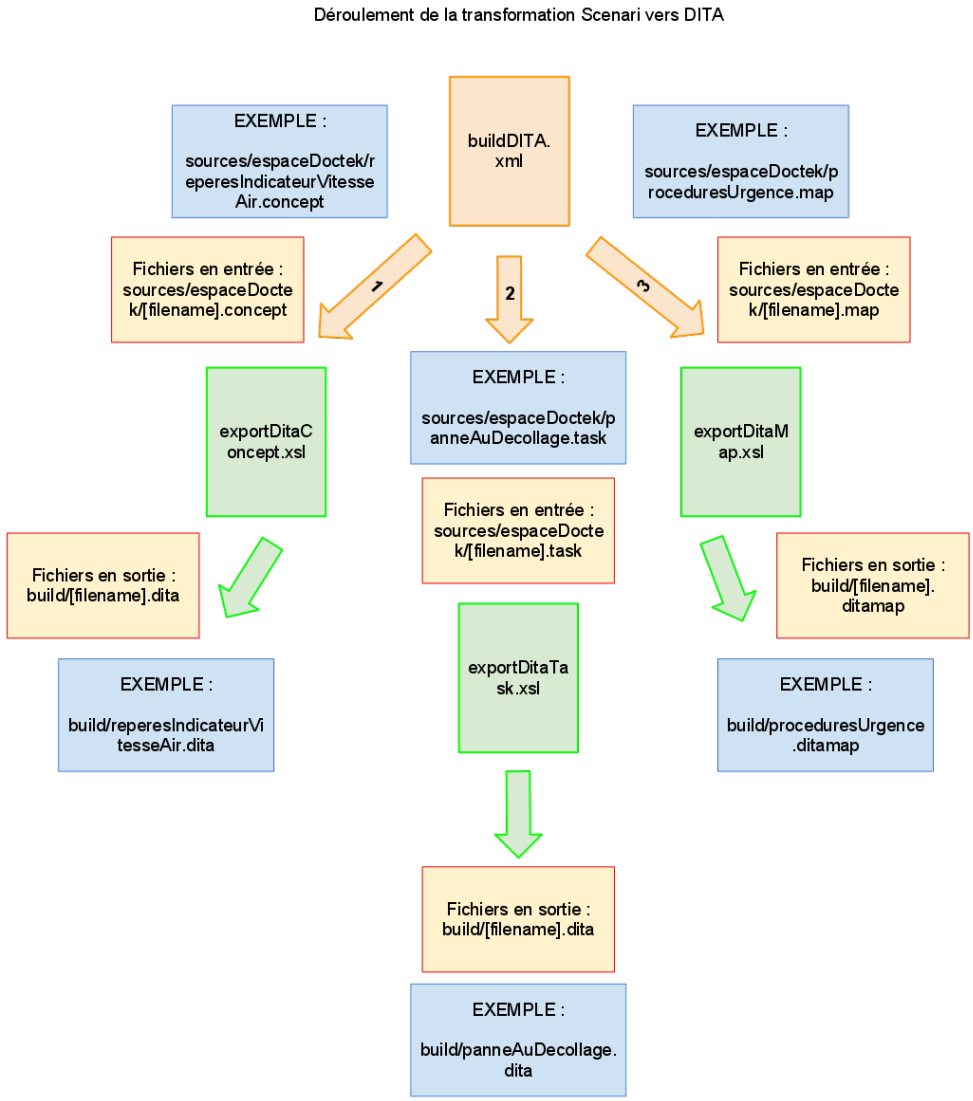
On obtient ainsi des *colspec* numérotées de 1 à n pour les n colonnes, ainsi on peut se resservir de ces numéros de colonne lorsque l'on veut définir des *spans* horizontaux (les *spans* verticaux sont assez simples et consistent en un changement de balise (*rowSpan* devient *morerows*) sans oublier de décrémenter la valeur de 1)

Pour les *spans* horizontaux, il faut savoir dans quelle colonne on se trouve. *Namest* spécifie la première colonne « logique » à partir de laquelle (incluse) le *span* horizontal s'étend, cela correspond donc à la somme des colonnes logiques précédentes plus 1. Dans la transformation il faut tenir compte du fait que les colonnes précédentes peuvent également comporter un *colSpan*. De même, nameend spécifie la dernière colonne « logique » incluse dans le *span* horizontal, c'est-à-dire la somme des colonnes logiques précédant le *span* plus la longueur du *span*. Voici comment cela peut s'écrire :

```
<xsl:template match="sc:cell[@colSpan > 1]">
<entry>
<xsl:attribute name="namest">
<xsl:value-of select=" count(preceding-sibling::sc:cell[not(@colSpan)]) + sum(preceding-sibling::sc:cell/@colSpan) + 1"/>
</xsl:attribute>
<xsl:attribute name="nameend">
<xsl:value-of select="count(preceding-sibling::sc:cell[not(@colSpan)]) + sum(preceding-sibling::sc:cell/@colSpan) + @colSpan"/>
</xsl:attribute>
...
```

2.4 Génération Dita avec Ant

Schéma de la génération Dita avec Ant



Passage du schéma Scenari au schéma DITA via des transformations XSL

2.5 Les ImageMap en DITA

ImageMap

L'*imagemap* dans *DITA* est un élément permettant d'associer des liens à une zone d'une image, et est adapté à la documentation technique dans le cadre d'un schéma avec légende par exemple.

Une *imagemap* nécessite une image, incluse grâce à un *href*, ainsi qu'autant de zones (*area*) que l'on veut pour qualifier l'image. Une *area* est définie par sa forme (rectangle, cercle ou polygone) et sa position (coordonnées en pixels) ainsi que le contenu associé. Cependant le contenu est en fait une référence vers un *topic* à partir entière à travers la balise xref qui comporte un lien.

Gestion des ImageMap dans le cadre du projet

Le modèle pour les *imageMap* dans Doctek n'étant pas encore en place au moment du projet, nous n'avons pas pu créer la transformation des *imageMap* de Scenari vers *DITA*. Cependant, cela aurait été possible en appliquant une transformation XSL au fichier XML issu de Scenari de la même manière que pour les autres transformations. Le point spécifique à prendre en compte est qu'en *DITA* les éléments de légende doivent se situer dans des fichiers externes.

La solution pourrait être de créer pour chaque élément de légende un concept dont le titre serait la légende en question. Etant donné qu'en *DITA* la zone référence un *topic* on pourra même imaginer des structures plus complexes qu'une simple légende (paragraphe, lien, etc.) et qui pourront être transformées de la même manière qu'un concept classique. Les formes et les façons de représenter les coordonnées étant a priori les mêmes dans Doctek et dans *DITA*, il suffira pour cette partie de changer le nom des balises lors de la transformation. Pour les contenus de légende on pourrait par exemple procéder en 2 passes. Lors de la 1ère passe on créé le fichier « classique » contenant l'*imagemap* et on met dans les xrefs des liens. Puis dans la 2ème passe on créé les fichiers pointés par les xref avec comme contenu le commentaire associé à la zone, en gérant la correspondance entre fichiers et références, par exemple en prenant pour nom de fichier le titre de la zone (cela impliquant l'hypothèse de titres différents pour chaque zone afin d'avoir unicité des fichiers). Mais comme nous n'avons pas directement expérimenté, cette solution reste une suggestion.

Un gros avantage d'utiliser Scenari plutôt que directement *DITA* est que cela permet de créer les zones à la souris plutôt que de renseigner à la main les coordonnées en pixel, ce qui est vite laborieux.

En conclusion nous n'avons pas pu mettre en place une transformation des *ImageMap* de Doctek vers *DITA* mais nous avons vu grâce au modèle Dokiel que cela sera possible une fois le modèle établi dans Doctek, et nous avons créé à la main une *imageMap DITA* pour permettre une transformation vers une version mobile et une version web.

3 Publication Web

Introduction - motivation de la publication Web

Avec la publication Web du manuel de vol de l'APM 20 Lionceau, nous souhaitons faire tirer parti au lecteur de la navigation hypertextuelle pour lui permettre de s'orienter dans le manuel suivant un compromis entre liberté (accès rapide à la connaissance recherchée) et structuration (possibilité de consulter les connaissances suivant un ordre, séquentiel ou hiérarchique).

Nous reprenons les éléments principaux de Dita, à savoir la *map* comme page de départ pour l'organisation des liens vers les *concepts* et les *tasks*, ainsi qu'un ensemble de

pages traitant chacune d'un *concept* ou d'une *task*.

3.1 Publication de la map

Mise en page générale

La page HTML de la map sera l'élément central du site web créé lors de la publication. Cette page sera composée d'un ensemble de liens, organisés sous la forme d'une table des matières. Tous les liens référencés par la map DITA créée lors de l'export DITA devront être présents dans cette table, et organisés en fonction de leur ordre d'apparition dans le document général. Cette page constituera l'index du site web.

Interactions avec l'utilisateur

Quand un utilisateur publiera son manuel au format web, la page d'accueil sera la map qu'il avait créée. En cliquant sur un lien de la table des matières, il pourra accéder à la page concernée – qui aura été précédemment transformée par la chaîne éditoriale -, que ce soit dans un ordre séquentiel ou par accès direct à une donnée quelconque.

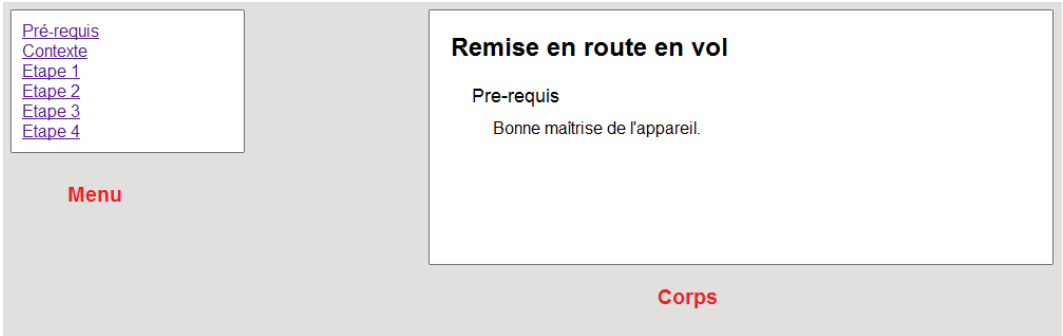
3.2 Publication web des concepts et des tâches

Mise en page générale

Les pages HTML présentant les concepts et les tasks se composent d'un menu et d'un corps, respectivement à gauche et au centre de la page. Le menu propose au lecteur de sélectionner la partie à afficher dans le corps à savoir (en fonction de l'obligation de présence ou non de l'élément en Dita, cf. diagramme UML) :

- pour un *concept*, sa définition (tout ce qui est contenu dans la balise *conbody* de l'élément) ou les liens proposés (*related-links*),
- pour une *task*, son pré-requis (*prereq*), son contexte (*context*), ses étapes numérotées (chaque *step* de premier niveau a une entrée dans le menu) , son résultat global (*result*) ou les liens.

Voici une capture d'écran illustrant cette mise en page pour une *task* :

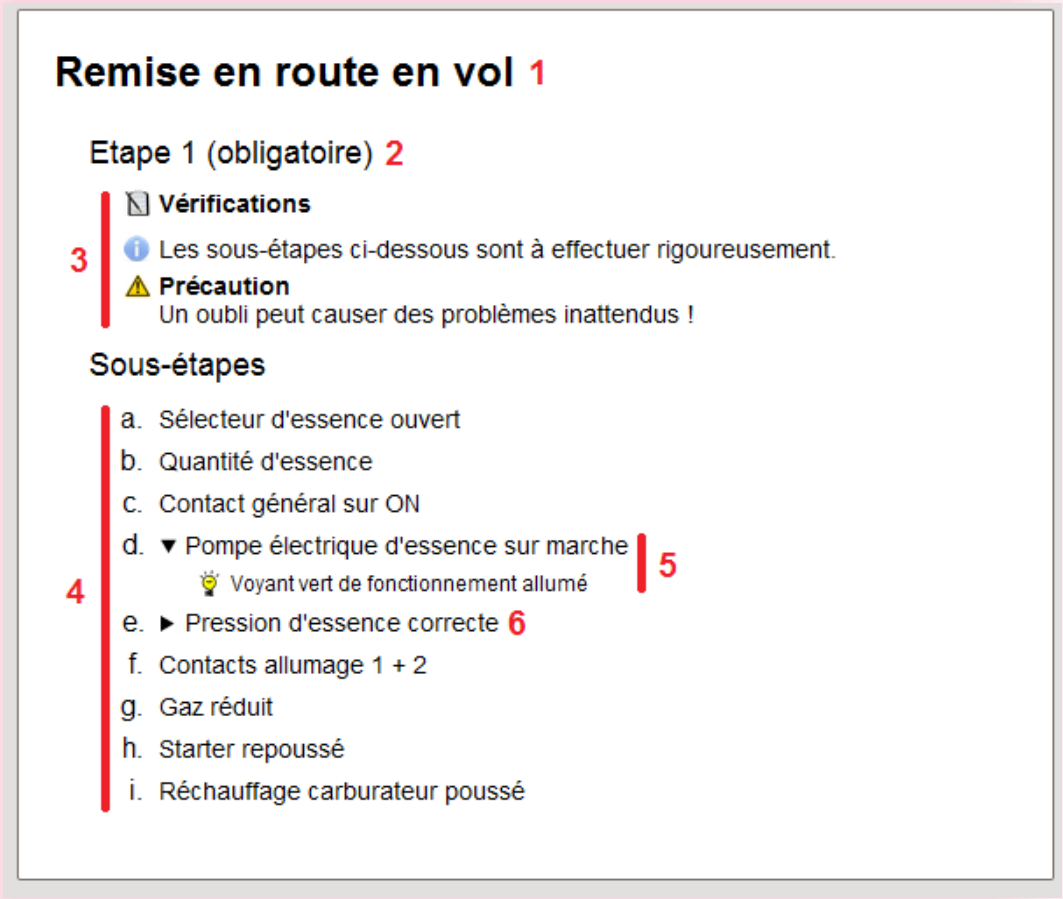


Dynamisation de la page

L'affichage du contenu d'une partie dans le corps à partir de la sélection via le menu est effectué grâce à l'appel d'une fonction Javascript qui modifie les propriétés CSS (*display: none;* ou *display: block;* selon le cas) des éléments en se basant sur la mécanique d'IDs (générés en XSL lors de la transformation).

Présentation d'une étape

Un des points particuliers de cette publication est la présentation d'une étape au sein d'une *task* ; en effet cet élément peut être assez volumineux en contenu (notes, sous-étapes contenant parfois elles-mêmes des notes, etc.), cependant il parait essentiel que tout ce contenu puisse se lire « en une fois » dans le corps de la page.



Capture d'écran de la présentation d'une étape dans la publication web

1. titre de la tâche rappelé à chaque étape,
2. entête de l'étape : numéro et, si renseignée en Dita, importance (obligatoire ou optionnel),
3. intitulé de l'étape (balise *cmd*) suivi des informations associées (blocs correspondant à des notes en Dita),
4. liste des sous-étapes, chacun étant représentée par le contenu de sa balise *cmd*,

5. exemple de sous-étape déployée : les informations (notes) sont affichées en supplément,
6. exemple de sous-étape repliée : les informations ne sont pas affichées.

L'affichage des informations liées aux sous-étapes en mode "déployé" ou "replié" est possible grâce à la balise HTML5 *details*, dont le contenu est caché par défaut et n'est affiché que lorsque le lecteur clique sur la flèche (cf. capture d'écran). Le contenu "entête" de cette balise (à savoir, dans notre cas, l'intitulé de la sous-étape), est quant à lui inséré dans la balise *summary*, elle-même contenue dans la balise *details* ; celui-ci restera affiché quelque soit l'action du lecteur.

3.2.1 Structure de la publication web des concepts et des tâches

La publication web des concepts et des tâches est structurée comme suit :

- le fichier publiWebInline.xsl regroupe toutes les transformations des éléments de contenu communs aux *concepts* et aux *tasks* tels que ceux liés au texte, aux images, aux tableaux, aux listes, etc.,
- le fichier publiWebArticle.xsl définit les transformations visant à structurer ce qui va être affiché dans le corps du *concept* (sa définition ou ses liens) ou de la *task* (son contexte, son prérequis, ses étapes, etc.) et appelle les transformations de publiWebInline.xsl pour les transformations plus fines,
- le fichier publiWebMenu.xsl définit les transformations visant à afficher le menu du *concept* ou de la *task* (les différentes entrée à sélectionner pour changer le contenu du corps),
- enfin, le fichier publiWebStructure.xsl définit la structure générale de la page HTML et appelle les transformations de publiWebMenu.xsl et de publiWebArticle.xsl.

Transformation des liens

Concernant les liens (related-links ou xref – liens "inline" – dans Dita), il a fallu faire le même traitement que pour la transformation de Doctek vers Dita, à savoir le remplacement de l'extension "*.dita" en "*.html". Là encore, l'exécution simultanée de toutes les publications HTML avec Ant permet de s'assurer que le fichier référencé a bien le même nom que le fichier Dita de base, et qu'il est bien lui-même publié en HTML.

3.3 Génération HTML à l'aide de Ant

Génération HTML des concepts et des tâches

Afin de générer les fichiers HTML des *concepts* et des *tasks,* le fichier buildHTML.xml applique la transformation publiWebStructure.xsl sur tous les fichiers du répertoire build ayant l'extension "*.dita" (qui ont été générés à l'aide du fichier buildDita.xml,

appelé en amont dans la génération globale build.xml).

En précisant différents attributs de la balise *xslt* dans buildHTML.xml, on est assuré que le nom du fichier source (situé dans le répertoire précisé par l'attribut *basedir*) sera repris pour le fichier généré (qui sera placé dans le *destdir*), qui aura pour extension "*.html" (valeur donnée à l'attribut *extension*). On peut également n'appliquer la transformation qu'aux fichiers "*.dita" grâce à la valeur de l'attribut *includes* (dans le cas où le répertoire source contient par exemple des *maps* - extension "*.ditamap").

Cette précaution permet d'assurer que les liens qui peuvent exister entre les différents fichiers Dita existeront et fonctionneront effectivement entre les fichiers HTML correspondant :

- d'un côté, les références au sein des fichiers Dita qui se font vers des fichiers Dita, se feront désormais des fichiers HTML vers des fichiers HTML ayant le même nom que les fichiers Dita d'origine,
- de l'autre, tous les fichiers Dita du répertoire seront transformés, d'où l'assurance que chaque passage d'un lien Dita à un lien HTML soit cohérent et mène à un fichier HTML existant.

4 Publication PDF

Introduction

La publication papier reste en tête des problématiques de diffusion des documents et ceci est d'autant plus vrai dans les milieux techniques et industriels, le pdf étant d'autre part le format de référence pour ce genre de publication.

Il existait à notre connaissance 2 méthodes principales de génération de pdf à partir de fichiers xml : les scripts FO et le moteur de rendu flyingsaucer. Nous nous somme donc concentré sur la deuxième solution du fait qu'elle était déjà utilisée dans Otpim Office par exemple.

Le dernier choix à souligner vient d'une réflexion sur la manière de publier un document technique de type DITA sous format papier. Nous en avons identifié deux : la publication d'une map complète sous forme de livre/fascicule/manuel bien entendu, mais également une publication séparé d'une tâche ou d'une fiche technique qui pourrait être directement affiché sur le poste/l'appareil qu'elle concerne.

4.1 Principe de publication PDF avec Flyingsaucer

Présentation de Flyingsaucer

Flyingsaucer est un moteur de rendu basé sur une librairie *Java*.

Pour fonctionner il est nécessaire de lui fournir 2 fichiers en entré :

- Un fichier *xhtml*, c'est à dire un fichier *html* répondant à la norme *xml*.

- Un fichier *css* (prise en charge complète de la norme CSS2.1 du *W3C*).

Le premier représentant notre contenu documentaire et le second la mise en forme qu'on souhaite lui appliquer lors de notre rendu PDF.

Une fois les 2 fichiers fournis le moteur calcul alors directement le rendu *pdf*.

Publication de plusieurs documents html sur un même pdf

Le moteur étant avant tout une basé sur une librairie java il est possible d'utiliser une routine java que l'on peut adapter à notre besoin.

La piste a été abandonné ici, car il est apparu qu'on pouvait obtenir le même résultat avec des scripts *Ant* et *xslt*.

4.2 Publication séparée des concepts et des tâches

Présentation général

L'idée est de pouvoir dans un premier temps publier au format PDF des fiches de concepts et de taches indépendamment du manuel complet via Flying Saucer. Nous avons alors écrit un XSLT permettant de publier des concepts et des tâches à partir du langage DITA vers le langage XHTML. Une feuille de style CSS est associé à ce document XHTML. Ensuite nous fournissons en entrée a Flying Saucer le fichier XHTML généré par la XSLT afin d'obtenir un rendu final au format PDF.

Le XSLT (publiPDF.xsl) ayant permit de générer les taches et les concepts sera réutilisable dans un second temps pour produire un manuel de vol complet ou un livret avec les fiches procédures à destination des pilotes par exemple. Notons aussi qu'il sera très facile de faire de nouveaux thèmes de publication PDF grâce à la puissance de CSS.

Rendu PDF d'une Task et d'un Concept

Le rendu PDF est proche du rendu HTML et la charte graphique est conservée pour apporter un confort à l'utilisateur final. Il n'y a pas de partie cachée ni de partie non publiée comme on pourrait le faire dans un format de publication dynamique.

Voici un extrait du rendu PDF d'une Task

En cas de panne au décollage

Contexte

Perte de puissance ou arrêt du moteur

Etape 1

Avant que les roues aient quitté le sol

Sous-étapes

a. Réduire les gaz
   Couper la pompe électrique
b. Maintenir l'avion au sol
   Manche secteur avant
c. Freiner au maximum
   S'aider de la poignée de frein
d. Si une sortie de piste possible, couper les contacts allumage et général, fermer l'essence

Etape 2

Analyser et remédier à la panne avant tout nouveau décollage

En haut de chaque page est rappelé le titre de la tache, puis ensuite sont introduit le contexte, les pré-requis s'il sont présent et les différentes étapes dans lesquels se trouve les sous-étapes.

De même voici un extrait du rendu d'un Concept en PDF

Performances de décollage

Les deux tableaux page suivante donnent les performances de décollage au centrage avant, avec un avion propre sans pluie et sans insectes, pour les masses de 510 kg et 634 kg sur une piste en dur horizontale en fonction de la température et de l'altitude pression.

L'interprétation linéaire pour une masse intermédiaire est conservative.

La pente de la piste doit être prise en compte si la piste n'est pas horizontale.

L'état de la piste (herbe, sable, boue, neige) peut aller jusqu'à doubler les distances de décollage. Sur une piste en herbe moyenne, compter + 25 % au minimum.

Des corrections en fonction de la vitesse du vent sont à apporter :

	Composante de la vitesse du vent						
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27
Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55

Quelque problème spécifique à la publication PDF à partir d'un HTML

Dans un fichier HTML "Simple" il n'y a pas vraiment de page avec des dimensions finies contrairement à une publication PDF. Cela soulève alors certains problème lors de l'utilisation de Flying Saucer.

Notamment lors de la publication de tableaux, ceux-ci sont coupés et donne un rendu pas très propre, pour corriger ce problème nous avons placés dans le CSS, à la classe .table, la propriété suivante :

page-break-inside: avoid ;

Cependant si on veut que le titre du tableau se répète à chaque début de page, il faut alors placer le titre entre les balises <thead> (et non dans tbody) et rajouter la propriété CSS suivante :

-fs-table-paginate: paginate;

Notons que cette dernière propriété n'est pas du CSS standard et à été rajoute par flying saucer pour justement régler ce problème.[11]

Voici respectivement un exemple de rendu sans traitement et avec traitement (répétition de l'en-tête) d'un tableau long :

Sans Traitement

en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27
Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27

Page 1

Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27
Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27

Avec traitement

11 - http://flyingsaucerproject.github.com/flyingsaucer/r8/guide/users-guide-R8.html

en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27
Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27

Page 1

	Composante de la vitesse du vent						
Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50
en kts	+ 5	0	- 5	- 11	- 16	- 22	- 27
Coefficient multiplicateur	1,2	1	0,95	0,85	0,75	0,65	0,55
en km/h	+ 10	0	- 10	- 20	- 30	- 40	- 50

Un problème similaire serait celui de la numérotation des pages.

Quelques spécificités de Flying Saucer

Nous avons mis dans le début du XSLT le code suivant <xsl:output method="xml" />, sinon lors de la génération du HTML, que nous avons réalise a partir de notre XSLT depuis Oxygen celui-ci génère en sortie <link rel="stylesheet" type="text/css" href="../styles/styles.css"> au lieu de <link rel="stylesheet" type="text/css" href="../styles/styles.css"/>, (ie) la balise auto-fermante est supprimé. Cela génère alors une erreur dans Oxygen.

Il faut respecter le codage UTF-8 sous peine d'erreur aussi dans Flying Saucer.

4.3 Publi PDF de la map

Exportation de la map vers html

L'exportation de la map vers un unique document html devrait se faire en utilisant un script Ant, lui même réutilisant les scripts xslt de conversion des topics Dita vers les html à destination de flyingsaucer.

Sur une exportation de type livre/manuel il ne suffit cependant pas de concaténer les éléments les un à la suite des autres. Le document se doit de posséder au moins deux parties supplémentaires : une page de garde et une table des matière.

Toute les informations de la page de garde devront être fournit dans les métadonnées de la Ditamap.

La table des matière quand à elle correspond plus ou moins à la Ditamap en elle même.

Note :

Par faute de temps et au vu de la complexité d'un xslt de transformation d'une map vers un document complet, nous avons ici préféré travailler sur une export réalisé par Oxygen sur laquelle nous avons rajouté les éléments qui nous ont semblé utile à la génération d'un pdf avec une mise en page correct.

Spatialisation du document

La principale différence entre le rendu d'un document html via un navigateur et via flyingsaucer est la gestion des sauts de page.

Une partie du problème a déjà était abordé concernant la publication des tâches et des concepts grâce aux propriétés css : *«page-break-inside : avoid»* et *«-fs-table-paginate : paginate»*.

Pour pouvoir gérer facilement les sauts de page et obtenir un document aéré nous utilisons la propriété *«page-break-brefore : always»* devant tout bloc représentant un niveau de la DitaMap. Ces blocs sont repéré dans le html par les classes *«nestedN»* où N représente le niveau de profondeur dans l'arbre de la map.

Cependant, afin d'éviter d'obtenir des pages avec un seul titre il est également nécessaire d'écrire une instruction contraire (*«page-break-before :avoid»*) pour chaque premier élément d'une suite de bloc de niveau supérieur à 1. C'est bloc étant repéré dans le html par la classe *«first»*

Table des matières et numérotation des pages

Quoique la structure global de la table des matières peut être facilement définit dans le html en suivant celle de la DitaMap il reste à ce moment un élément encore inconnu car dépendant uniquement de la css utilisée : l'association d'un contenu à une page donnée.

Le problème doit donc être résolu dans la css, en faisant donc ici une entorse à la philosophie générale de cette méthode en générant du contenu via le fichier censé représenter la forme.

La numérotation des pages s'effectue très simplement avec la règle CSS3 : *«content: "Page " counter(page);»* qui rajoute un contenu à une position de la page que l'on défini.

Pour référencer ce numéro de page dans la table des matière nous avons récupéré une instruction déjà présente dans le css de conversion en pdf d'*Optim Office* : content : *«leader('.') " " target-counter(attr(href),page);»*

Numérotation des chapitres

Bien que celle ci puisse également être effectué au niveau de l'xslt, nous avons ici utilisé une solution très simple basé sur les propriété css et la manipulation des compteurs via les propriétés : *«counter-increment»* et *«counter-reset»*.

La première incrémentation devant systématiquement être précédé d'une initialisation du compteur via la méthode reset cette dernière doit donc forcément être associée à une balise parente de la balise de l'incrémentation.

5 Publication Mobile

5.1 Choix de la technologie

jQuery Mobile est la bibliothèque javascript que nous avons choisi pour la version mobile du site du manuel de vol. C'est une technologie innovante basée sur jQuery (bibliothèque javascript très utilisée de nos jours) et HTML5. Elle est supportée par de nombreux sponsors tels qu'Adobe, BlackBerry, Nokia, et Mozilla. Les raisons qui ont motivé ce choix sont les suivantes :

- Les sites et applications mobiles développés grâce à cette bibliothèque fonctionnent sur la majorité des plateformes mobiles
- La bibliothèque prend en charge l'ensemble de l'interface graphique du site : la présentation graphique de tous les éléments principaux d'un site (barre d'outils, boutons, formulaires, contenu en accordéon) est directement optimisée pour une interface tactile. Le développeur n'a donc qu'à se soucier du contenu du site, la quasi-totalité de son interface est prise en charge par la bibliothèque (presque

aucune couche CSS à coder par exemple). Ceci permet un développement très rapide.

- Enfin, cette technologie nous était familière ce qui nous a permis de développer rapidement la maquette du site et de nous concentrer sur les transformations XSLT (sans perdre trop de temps sur du débogage et des recherches inhérents à la partie développement mobile).

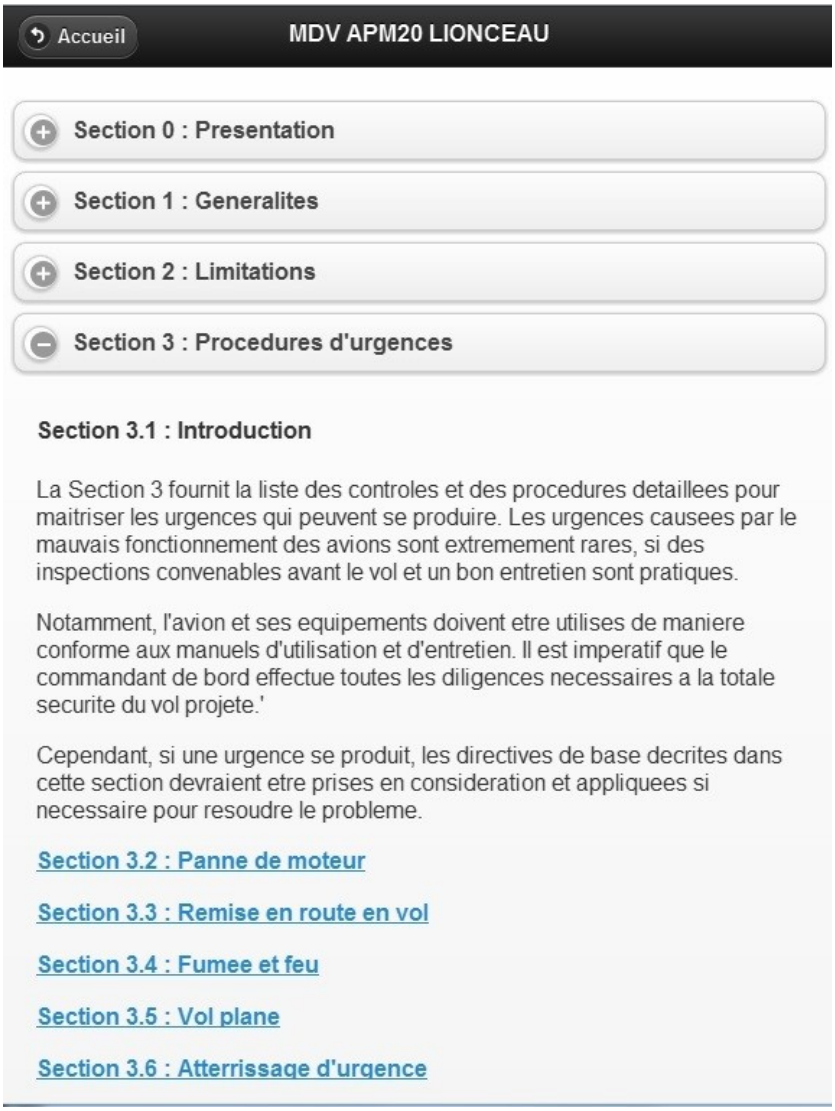
5.2 Création d'une maquette

Une fois la technologie choisie, nous avons développé une maquette du site mobile contenant le manuel de vol afin de s'entendre sur l'interface et le contenu du site dont la création serait automatisée par les transformations Dita vers jQuery Mobile. Le but de la démarche était d'être certain de savoir dans quelle direction s'orienter lors du développement des transformations xslt, en sus d'avoir un contenu exemplaire pour la version mobile du site du manuel de vol.

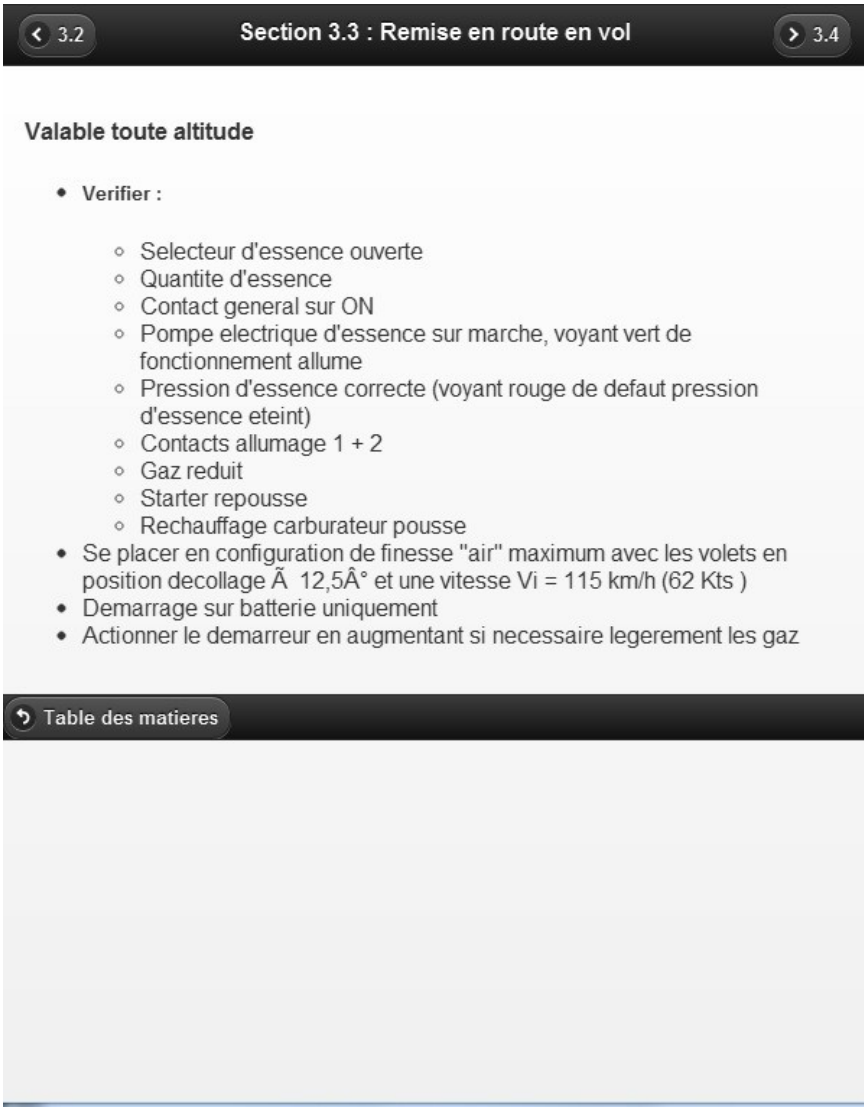
Voici une description de la maquette que nous avons créée :

- Le contenu du manuel est accessible via un sommaire dont chaque élément est une liste en accordéon.

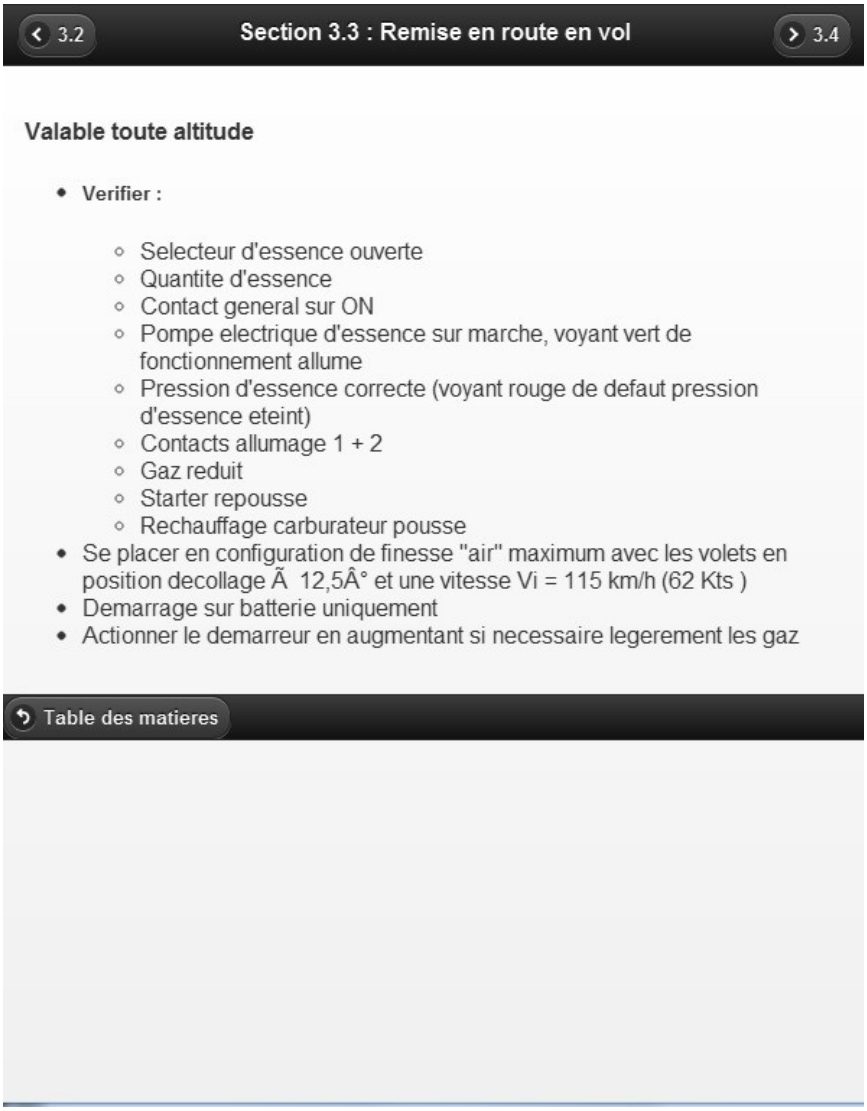
- Lorsqu'on touche une section, l'accordéon se déplie. On peut alors lire l'introduction de la section présentant son contenu ainsi que le reste de son sommaire dont chaque élément est un lien vers la sous-section correspondante.



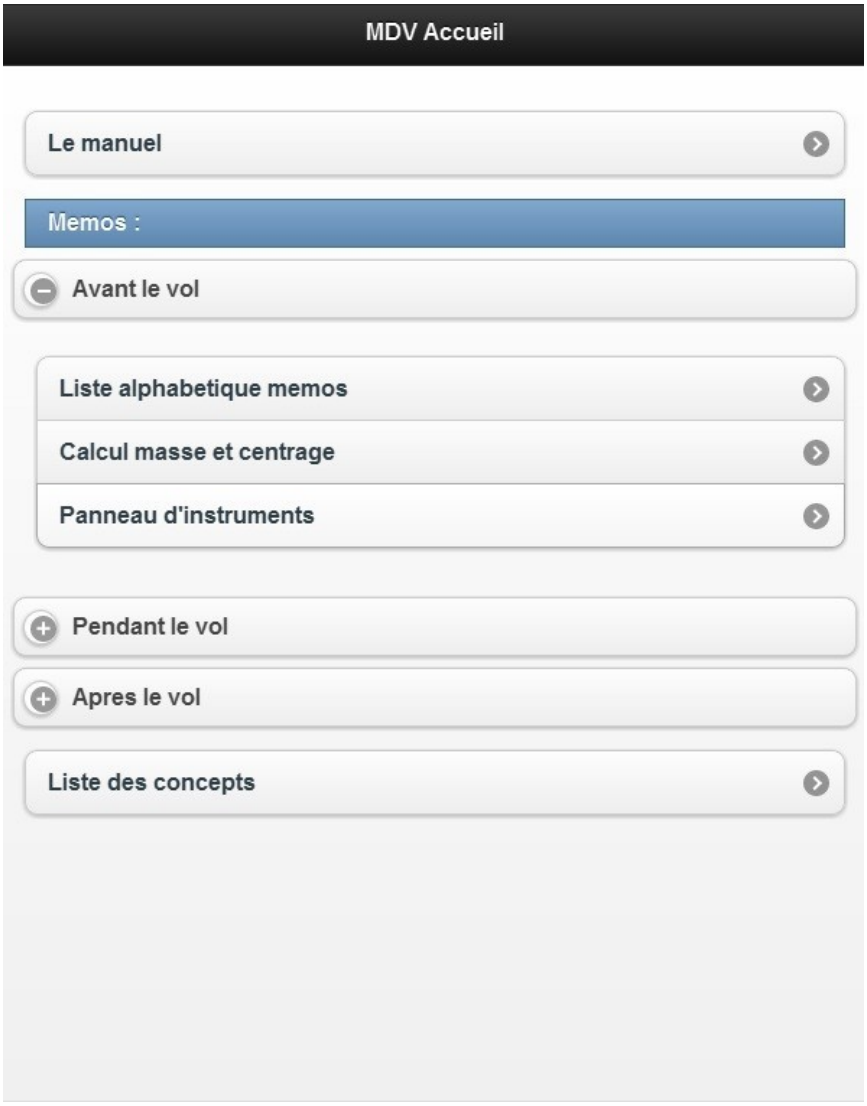
- Lorsqu'on accède à une sous-section, l'ensemble de son contenu est présenté dans la même page, on peut continuer à naviguer au sein de la section via les flèches de navigation situées dans le header ou revenir à la table des matières via le bouton retour situé dans le footer.



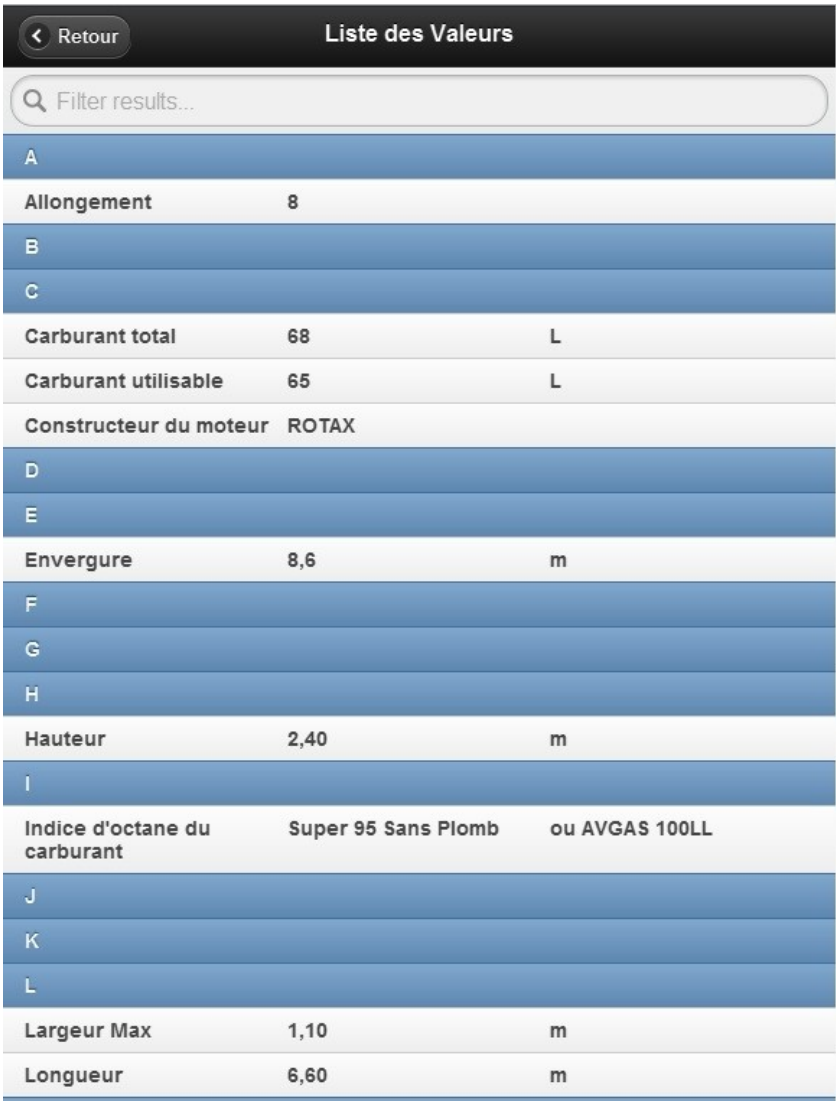
- Si au sein d'une sous-section un élément de contenu d'une autre sous-section est cité, un lien vers cette dernière permet d'y accéder immédiatement.



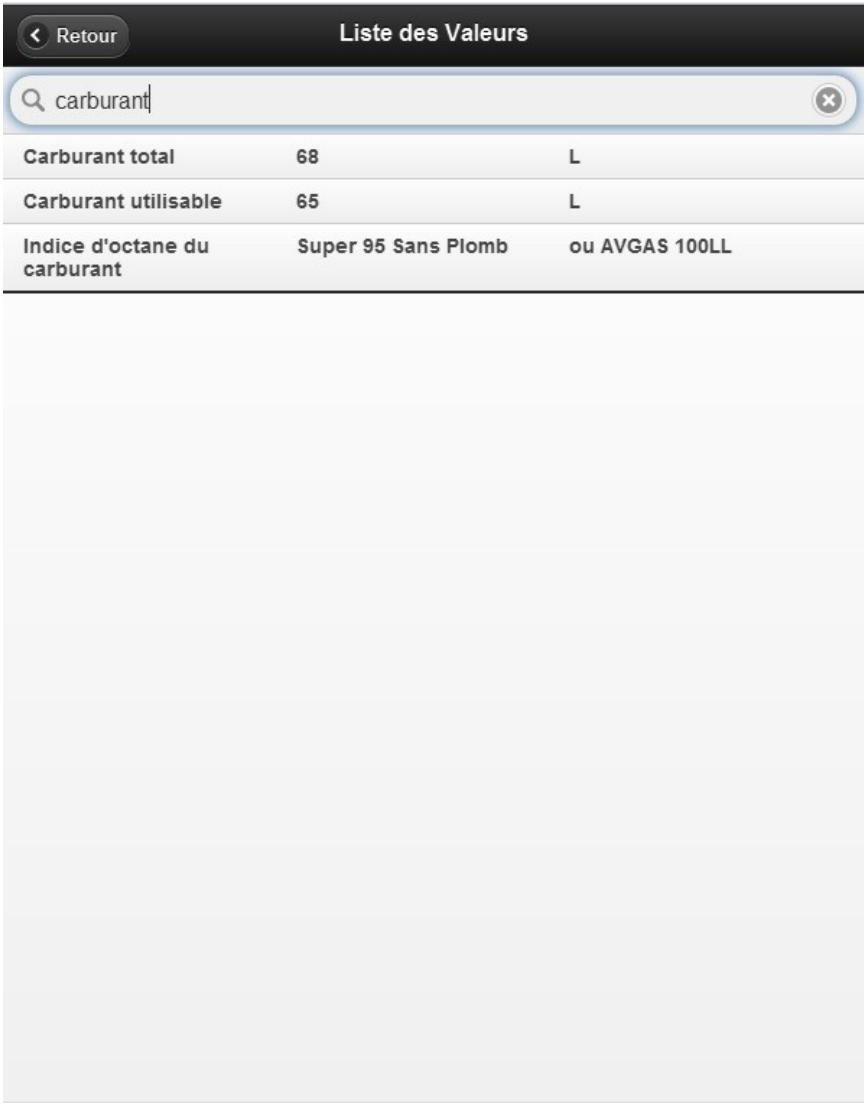
- Des raccourcis vers les éléments les plus consultés du manuel sont présents dans la page d'accueil du site.



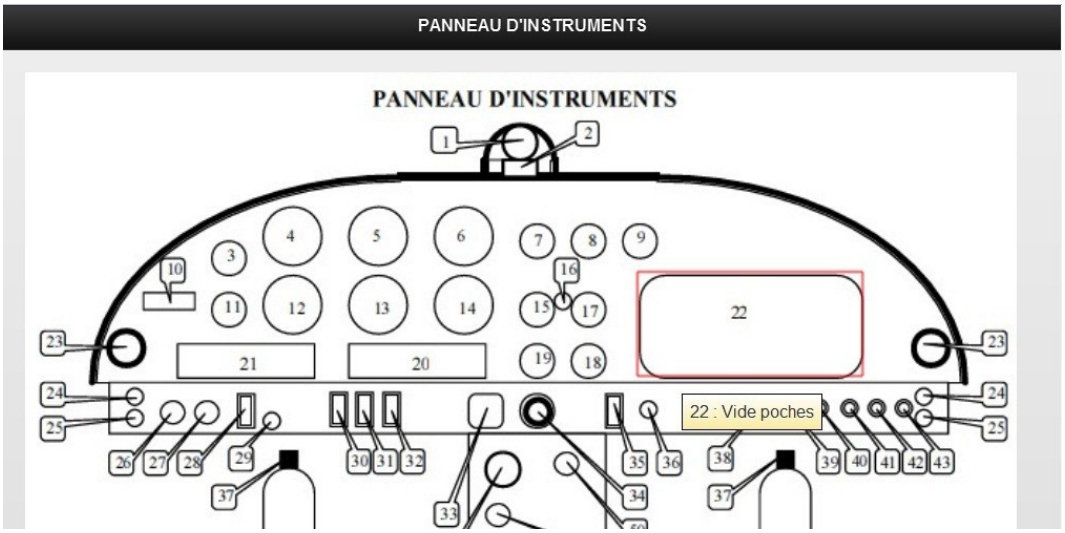
- La liste des concepts est disponible depuis la page d'accueil du site. Elle permet d'accéder à l'ensemble des constantes du manuel, triées par ordre alphabétique. La liste dispose d'un filtreur pour faciliter les recherches (sous la forme d'un champ de texte où l'on rentre des mots clés).



A		
Allongement	8	
B		
C		
Carburant total	68	L
Carburant utilisable	65	L
Constructeur du moteur	ROTAX	
D		
E		
Envergure	8,6	m
F		
G		
H		
Hauteur	2,40	m
I		
Indice d'octane du carburant	Super 95 Sans Plomb	ou AVGAS 100LL
J		
K		
L		
Largeur Max	1,10	m
Longueur	6,60	m



- Enfin, une image map du tableau de bord a été créée. Elle permet d'accéder à un schéma interactif de ce dernier : touchez du doigt un de ses éléments et sa description apparaitra à l'écran. Cette fonction n'étant pas inhérente à la bibliothèque jQuery mobile, il nous a fallu développer une interface optimisée pour mobile en javascript.
  Le principe est le suivant : On inscrit en dur dans le document html la description des différentes « notes » accessibles sur l'image. On donne toutes leurs caractéristiques (légende, position en pixel, forme). Au lancement de la page on récupère à l'aide de javascript les données ainsi accessibles et on crée les objets associés avant de charger l'image de fond de l'imagemap. Le tout est placé dans un canvas (HTML5) pour pouvoir y dessiner aisément des formes géométriques indiquant la limite des annotations.



5.3 Implémentation des transformations Dita vers jQuery

Une fois la maquette validée par le chef de projet, le reste de l'équipe, et les porteurs, nous sommes passés à la transformation Dita vers jQuery Mobile. Plutôt que d'automatiser la création de l'ensemble de notre maquette, nous nous sommes concentrés (pendant les 3 semaines qu'il nous restait) sur ses éléments les plus importants et donc les plus attendus pour la preuve de concept, à savoir : l'image map du tableau de bord, les sections (contenant des procédures), et la liste des concepts. L'implémentation des transformations des éléments qui ont été laissés de côté (page d'accueil du site, sommaire du manuel de vol en listes en accordéon, raccourcis depuis la page d'accueil) n'étant pas très intéressante techniquement parlant, nous avons considéré que la maquette suffirait à les illustrer.

Procédure

Les transformations des procédures pour plateforme mobile ont été réalisées à partir des fichiers d'extension .dita générées par les exports Scenari vers Dita. Nous avons tenté de suivre au maximum la maquette que nous avions produis précédemment. Le model était le suivant : Une page de menu permettant d'accéder à chacune des étapes de la procédure, puis une page par étape avec des boutons en haut de page permettant de passer d'une étape à l'autre.

Après avoir écrit un premier jet de transformation, nous nous sommes rendu compte que le résultat n'était pas particulièrement adapté à la visualisation sur plateformes mobiles à l'écran de petite taille. En effet, les étapes sont composées de nombreuses notes (Attention, Conseil, Détails) rallongeant la taille des pages. La solution que nous avons apportée à ce problème est l'utilisation de liste à accordéons. Le principe est simple : Une seule note par page peut être visualisée à la fois. Afficher une autre note fait disparaitre le contenu de la précédente limitant ainsi le manque de visibilité.

Image Map

La transformation pour les imagemap a sans doute été la moins évidente. En effet il nous fallait commencer par trouver un moyen facile et complet de décrire ce concept puis développer une couche javascript permettant de gérer efficacement l'interaction entre l'utilisateur et l'image. En dernier lieu seulement venait la transformation xls qui n'était plus alors qu'une formalité.

Pour l'expression du concept, nous avons estimé avoir besoin des informations suivantes : L'image cible ainsi qu'une description de chacune des notes. Pour chaque note, nous avons besoin de connaitre la forme de la note (cercle ou rectangle), sa position en pixel sur l'image ainsi que la légende associée. A terme, certaines améliorations pourraient être apportées comme un choix plus grand de formes ou alors un plus grand nombre d'informations relatives à la note (ex : un lien url).

Nous avons placé ces informations au sein d'un fichier html et nous avons alors ajouté la surcouche javascript. A l'aide de jQuery, nous avons implémenté un système récupérant ces informations dans le .html avant de créer les objets « notes » correspondant et le canvas (HTML5) qui contiendra l'image finale. On associe aussi une fonction à l'évènement de clique sur le canvas, récupérant et affichant la note présente à la position de l'évènement.

Liste des concepts

La liste des concepts est faite à partir d'une « map » créée sur scenariChain par l'auteur du manuel de vol. Celui-ci spécifie lors de la rédaction de son manuel les items qu'il considère comme étant des concepts. Scenari créé alors un fichier .map énumérant l'ensemble des concepts sélectionnés par l'auteur ainsi que le fichier .concept correspondant. Une transformation XSLT permet de changer le fichier .map en un fichier .dita (équivalent exact mais rédigé en dita, langage que nous avons sélectionné pour notre projet) et les fichiers .concept en fichiers .dita (idem).

C'est à partir de cette map rédigée en dita que nous créons la page mobile reprenant la liste des concepts. Notre transformation XSLT créée tout d'abord une page html jquery mobile classique dans laquelle elle insère une liste vide (balise <ul>). Puis elle parcourt la map dita et pour chaque concept qu'elle y trouve (encapsulé dans une balise topicref), elle ajoute un item de liste (balise <li>) contenant le nom du concept ainsi qu'un lien vers une page présentant sa définition, sa valeur, et éventuellement des liens vers d'autres concepts complémentaires.

La liste créée est triée par ordre alphabétique grâce à la fonction xsl:sort et possède un « filtreur » (sous la forme d'un champ de texte) permettant de rentrer des mots clés afin de retrouver plus rapidement un concept (objet javascript pris en charge par jQuerry mobile).

Enfin un script est ajouté à la page html créée afin d'insérer un list-divider (séparateur de liste) entre chaque ensemble de concepts commençant par la même lettre.


Filter items...
E
Étalonnage du système indicateur de vitesse-air
M
Masse
V
Vitesse-air

V Bilan de l'étude

Ce qui a été fait

Lors de cette étude, nous avons réalisé une chaîne complète depuis l'éditeur Doctek vers 3 types de publication : Web, Mobile et PDF.

La réalisation de cette preuve de concept a donc permis de montrer qu'il était possible de créer du contenu DITA à partir du modèle Doctek, et de réaliser des interfaces sur les 3 types de supports. Cela a également permis de lever certains verrous, puisque nous avons montré qu'il était possible de prendre en compte bien évidemment les éléments de base, mais également des éléments plus complexes tels que des tableaux comportant des cellules fusionnées.

L'un des points forts de cette étude a également été la prise en compte des *imageMaps* en particulier sur mobile, ce qui ajoute une plus value intéressante car ce type de représentation est particulièrement utile pour la documentation technique (schéma légendé) que ce soit pour le web ou le mobile, voire du PDF. Une fois le modèle pour les *imageMaps* en place du côté de Doctek, on pourra donc associer saisie aisée dans Scenari, compatibilité DITA et interface novatrice.

Ce qu'il reste à faire

Plusieurs points ont été envisagés pendant le projet, mais non développés :

- L'édition de l'*imageMap* existe dans l'éditeur Dokiel, mais pas encore dans Doctek, et la transformation XSL de Scenari vers DITA pour l'*imageMap* a été étudiée, mais pas implémentée.
- La transformation de l'élément *chapter* de Doctek vers DITA n'est pas évidente, et peut être faite de différentes manières. Cette élément n'existe pas à proprement parler dans DITA, et est utilisé dans Doctek pour structurer le contenu en chapitres, sous-chapitres. Dans DITA, cela se traduit pour l'essentiel par des sous-topics. Nous avons implémenté la transformation d'un *chapter* de Doctek vers DITA, mais pas la transformation de *chapters* contenus eux-mêmes dans des *chapters*, etc.
- Des éditeurs comme ArborText permettent d'inclure des plans en 2D, il serait intéressant de pouvoir en faire de même.
- L'idée a été émise de créer la notion de variable ou constante, qui correspondrait à une valeur particulière (par exemple la vitesse de vol de l'avion) qui ne serait rentrée qu'une fois par l'utilisateur puis utilisé ensuite dans le document technique. Cette notion n'existant a priori pas en DITA, il s'agirait d'une fonctionnalité utilisé lors de l'édition dans Scenari, puis lors du passage vers la DITA les constantes seraient remplacées par leur valeurs réelles.

Annexe 1 Annexes

1.1 Liens utiles

DITA FAQ : *http://www.ditausers.org/dita-ot/demo/faq/DITA-faq.html*

DITA reference book : *http://www.oasis-open.org/committees/download.php/11347/ditaref-book.pdf*

1.2 Notice technique de la spécialisation en DITA

Introduction

La spécialisation est une des caractéristiques essentielles de la DITA. Bien que nous ne l'ayons pas utilisée dans ce projet, une partie de l'état de l'art y a été consacrée. Cette notice a pour but de résumer l'intérêt et les mécanismes de spécialisation en DITA.

Spécialisation de type

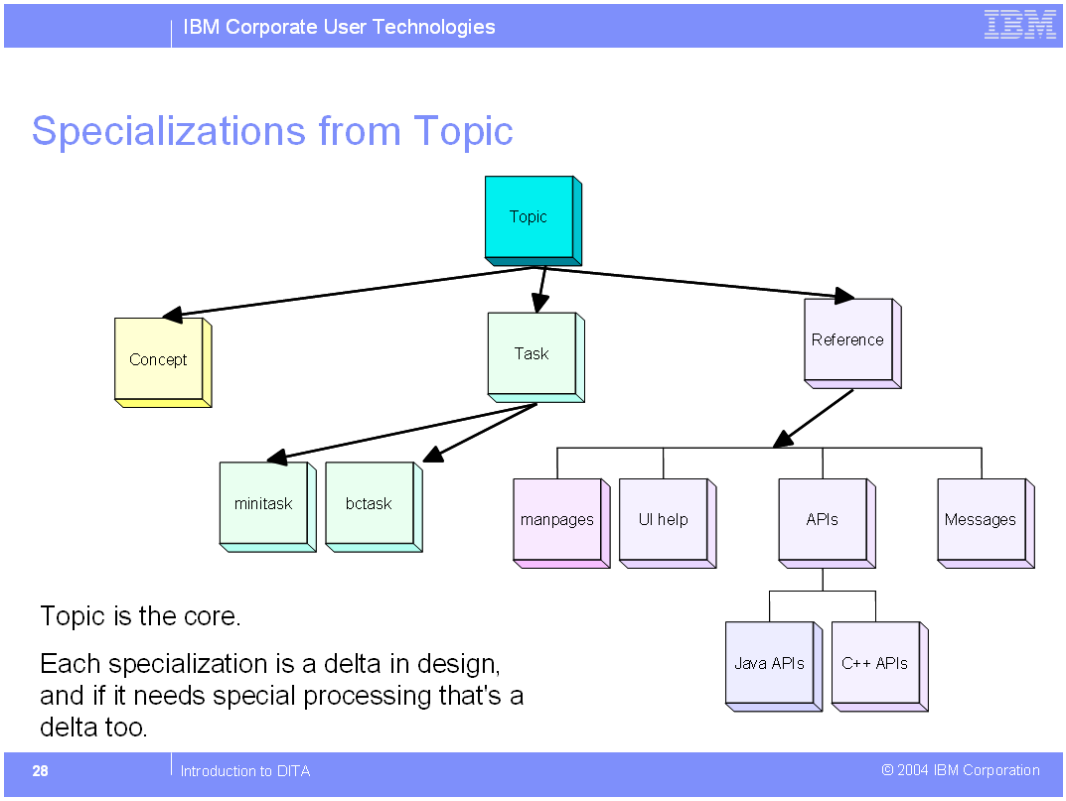
Les types de base de DITA ne sont pas toujours suffisants (trop généraux) pour la documentation technique, mais il existe un mécanisme de spécialisation qui repose fortement sur la notion d'héritage. Cela permet d'intégrer des éléments spécifiques tels que des procédures d'urgence ou des plans, qui peuvent faire l'objet de transformations spécifiques pour un affichage approprié. Cependant la force de la spécialisation est qu'un moteur de transformation prévu pour les éléments de base permettra quand même l'affichage des éléments spécifiques, en les affichant de la même manière que les éléments dont ils dérivent. Cette caractéristique est particulièrement intéressante dans le cas d'échange de contenus spécialisés entre deux sociétés/organismes utilisant le standard DITA (et n'ayant pas nécessairement connaissance des spécialisations de l'autre). De plus, on a la possibilité, grâce au renommage des balises, d'avoir un langage typé métier.

Spécialisation de domaine

Il ne semble pas que nous ayons besoin de la spécialisation de domaine.

Spécialisation de topic : concept, task et reference

Pour bien comprendre comment fonctionne la spécialisation, il est à noter que concept, task et reference sont des spécialisations de topic. Cela est illustré par le schéma suivant :



(source :*http://www.xml.gov/documents/completed/ibm/dita.ppt*)

Cela signifie qu'une nouvelle DTD a été créée : concept.dtd par exemple.

Celle-ci reprend des éléments de topic.dtd de cette manière :

```
<!-- Embed topic to get generic elements -->
<!ENTITY % topic-type PUBLIC
"-//OASIS//ELEMENTS DITA 1.2 Topic//EN"
"/base/dtd/topic.mod">
%topic-type;
```

Et d'autre part redéfinit ses propres éléments de cette manière :

```
<!-- Embed concept to get specific elements -->
<!ENTITY % concept-typemod
PUBLIC
"-//OASIS//ELEMENTS DITA 1.2 Concept//EN"
"concept.mod">
```

```
%concept-typemod;
```

C'est donc au sein de concept.mod que seront spécifiés les éléments et leur cardinalité. Pour chaque topic, il faut en effet au sein de DITA une DTD à laquelle sont liés des .mod (définition des éléments) et un .ent (déclaration des entités utilisées). De plus des % sont utilisés comme références (en fait les éléments références référencent des éléments du même nom).

Enfin, étant donné qu'on a renommé des éléments existants, il faut à l'aide de class faire le lien entre l'élément déjà existant (père, e.g. body) et celui qu'on vient de redéfinir (fils, e.g. conbody)

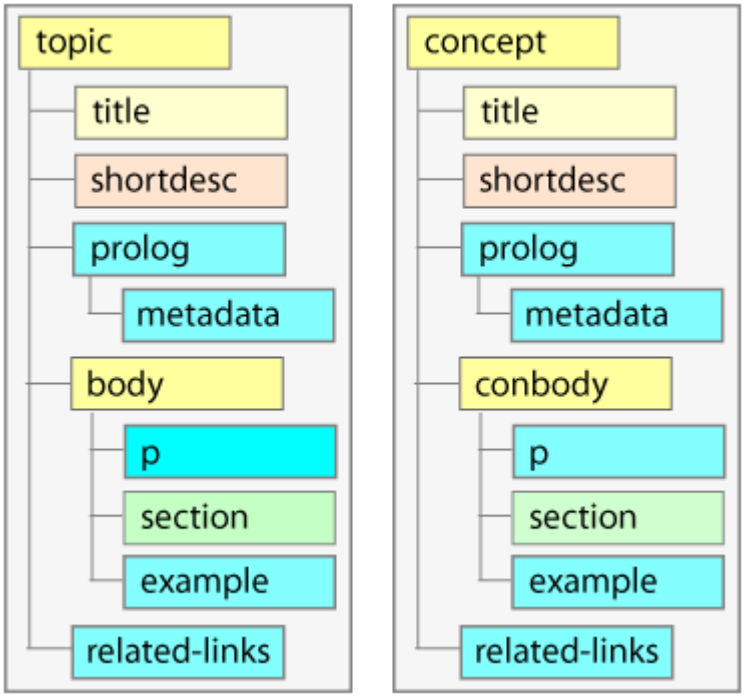
de cette manière :

```
<!ATTLIST concept %global-atts; class CDATA "- topic/topic concept/concept ">
<!ATTLIST conbody %global-atts; class CDATA "- topic/body concept/conbody ">
```

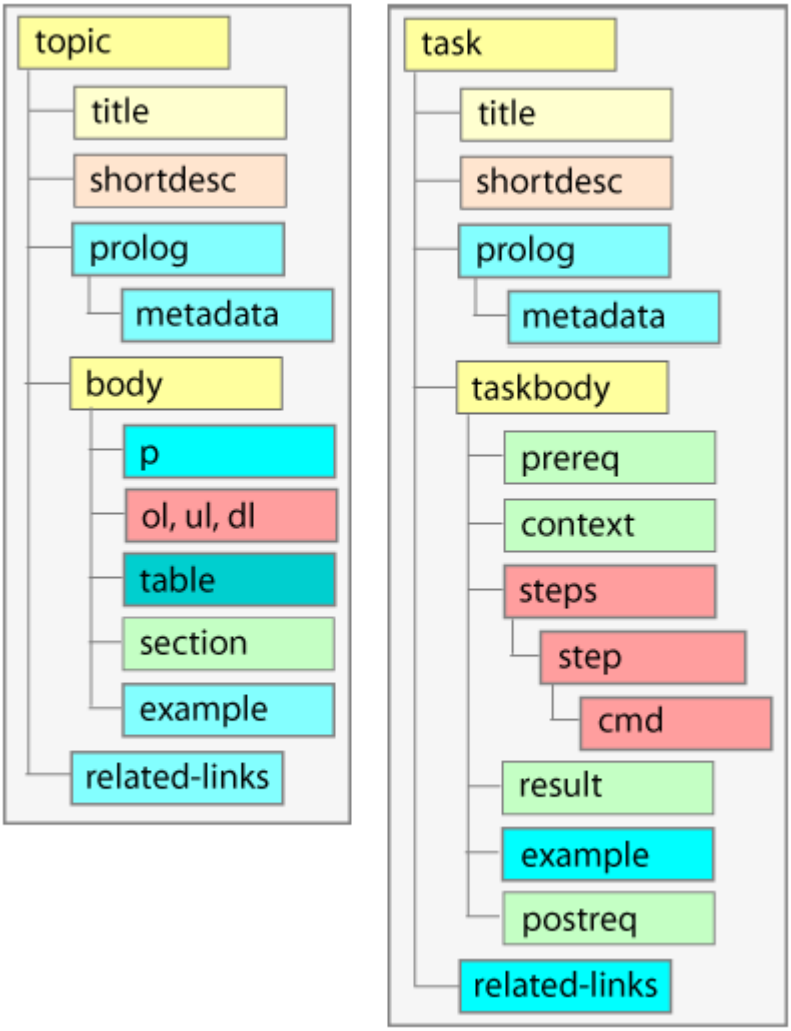
C'est ce mécanisme qui permet d'utiliser une transformation définie pour le type général sur le type spécialisé.

Une fois toutes les spécialisations définies au sein des nouvelles DTD, on se retrouve avec les configurations suivantes :

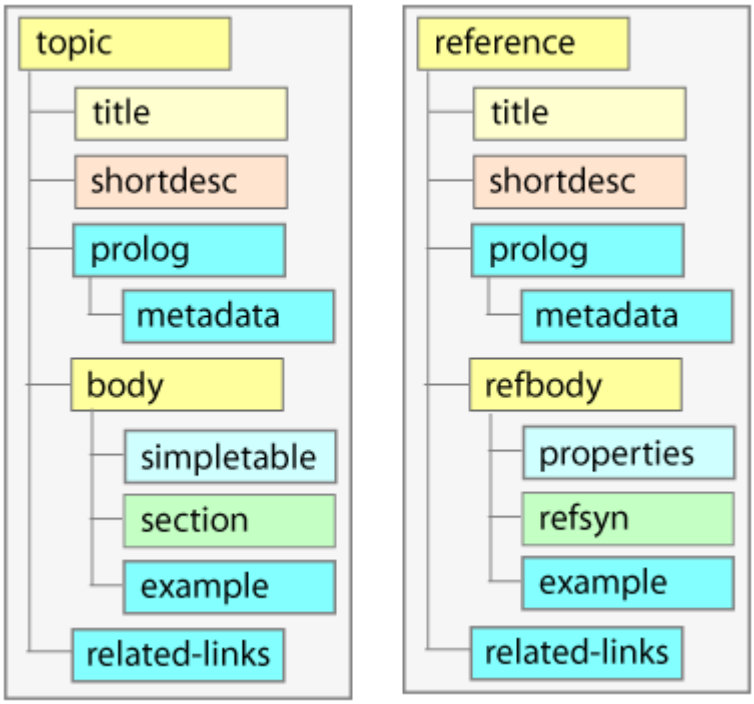
Concept topic :



Task topic :



Reference topic :



On peut alors appliquer ou non un XSL spécifique sur les éléments spécifiques. On peut également de la même manière aller plus loin et spécialiser Concept, Task ou Reference.

(source des images : *http://www.ditausers.org/training/DITATopics/*)

Créer sa propre spécialisation : outils

La création de documents DITA et de spécialisations nécessite à la fois un éditeur XML, un outil pour vérifier que les documents XML sont valides par rapport à la DTD créée, ainsi qu'un moteur de transformation de la DITA vers d'autres support (PDF, web). Oxygen répond à l'ensemble de ces besoins.

Créer sa propre spécialisation : règles

Lorsqu'on spécialise un type déjà existant, on peut :

- reprendre tout ou partie des éléments du type,
- renommer éventuellement les éléments repris,
- restreindre éventuellement les cardinalités des éléments repris.

En revanche, **on ne peut pas ajouter de nouveaux éléments : la spécialisation est restrictive.**

Le changement de cardinalité répond aux règles suivantes :

- un élément "unique et requis" **doit** être conservé,
- un élément "optional" peut rester "optional", devenir "unique et requis" ou être supprimé,
- un élément "one or more" peut rester "one or more", devenir "unique et requis" ou séparé en éléments renommés (exemple : a+ peut devenir a1, a2+, a3?, a4*, mais

pas a1? qui est moins restrictif),

- un élément "zero or more" suit les mêmes règles qu'un élément "one or more" et peut aussi devenir "optional" ou être supprimé,
- un choix de plusieurs éléments (exemple a | b) peut devenir un choix partiel de ces éléments,
- **dans tous les cas précédents, l'élément peut toujours être renommé.**

Créer sa propre spécialisation : exemple

1. Pour créer sa propre spécialisation il faut commencer par un travail d'analyse sur le corpus et une prise de connaissance de la DITA pour identifier à quel élément de base on peut se rapporter le plus. Par exemple s'agit-il plutôt d'un topic ? d'un concept ? d'une reference ? d'une task ? Afin de tester comment marche techniquement la spécialisation, on peut par exemple vouloir créer un contenu de type introduction, avec juste un titre et du texte, l'élément qu'on va alors utiliser est un topic. A noter que cet exemple n'est pas pertinent dans un contexte de production mais facile à comprendre.
2. Une fois qu'on a identifié l'élément qui fournit la meilleure base pour notre spécialisation, il faut commencer par créer du contenu en DITA en utilisant le dit élément. Dans notre cas nous créons donc une map, contenant divers topicref vers des éléments et en particulier un topic (l'introduction) qui comporte un title et un body contenant lui même des p. Dans la pratique il est recommandé de produire un nombre suffisamment significatif d'exemples de contenus afin de pouvoir bien identifier toute la variabilité dès le début et ainsi faire une spécialisation efficace. Si le contenu a été bien écrit on doit pouvoir transformer la map en un document PDF par exemple à l'aide d'Oxygen (gestionnaire de cartes DITA).
3. Il faut ensuite passer à l'étape de spécialisation. On commence par créer une DTD correspondante qu'on associe avec le fichier DITA en question.
    1. Dans notre exemple on veut spécialiser topic. Comme concept est une également spécialisation de topic et que l'on se limite à des éléments basiques (title et body) présents dans concept le plus simple est de modifier la DTD de concept selon nos besoins. Si l'on voulait spécialiser task on procéderait de la même manière en reprenant la DTD de task.
    2. On commence par changer les noms des attributs. Dans notre exemple on choisit intro pour la racine et introbody pour le corps. Puis on se restreint aux éléments dont on a besoin. Par exemple
    <!ENTITY % intro.content
    "((%title;),
    (%titlealts;)?,
    (%abstract; |
    %shortdesc;)?,
    (%prolog;)?,
    (%introbody;)?,
    (%related-links;)?,
    (%intro-info-types;)*)"
    >
    devient
    <!ENTITY % intro.content
    "((%title;),
    (%introbody;)?)"

>
en respectant la règle selon laquelle on ne peut que être que plus restrictif lorsque l'on spécialise un élément. (voir)
De cette manière, on définit l'ensemble des éléments dont on a besoin au sein du .mod

3. Une fois les éléments définis avec les cardinalités adéquates il faut spécifier le lien d'héritage des différents éléments de cette manière :
<!ATTLIST intro %global-atts; class CDATA "- topic/topic intro/intro ">
<!ATTLIST introbody %global-atts; class CDATA "- topic/body intro/introbody ">
Cela signifie que intro est une spécialisation de topic et introbody une spécialisation de body.

Il est important lors de la création de la DTD de procéder de manière itérative. En particulier il faut prendre le soin de vérifier régulièrement que le .dita est toujours valide par rapport à la DTD et que la transformation de la map DITA en PDF par exemple fonctionne bien. En effet, Oxygen ne comporte pas de debugger lorsque la génération échoue et il devient donc très difficile de trouver la source de l'erreur si l'on a tout édité d'un coup.

Cas des images : imagemap

L'exemple d'une image à laquelle on puisse associer des commentaires par zone a été évoqué lors de la séance du 30 novembre. La question était de savoir si DITA permettait de faire quelque chose de similaire à l'exemple suivant (tiré de ce qu'on peut faire avec la chaîne éditoriale Dokiel) :

```
<image href="...">
<zone id="..." x="..." y="..." width="..." height="...">
<p>un commentaire sur cette zone</p>
<p>un autre commentaire</p>
</zone>
<zone id="..." ...>
<p>...</p>
</zone>
</image>
```

Element imagemap dans DITA

Dans la référence de la DITA, il existe un élément nommé "ditamap", qui est utilisable au sein du "body" d'un "topic" et suit la structure suivante :

```
<imagemap>
<image href="...">
```

```
<area>
<shape>rect</shape>
<cords>0, 0, 100, 100</cords>
<xref href="..."></xref>
</area>
<area>
...
</area>
<image>
</imagemap>
```

Observations

Dans ce cas, il est impossible de spécialiser imagemap en "image" : en effet, l'élément "xref", qui est une url vers un fichier DITA externe, ne peut pas être changé en "p", qui lui correspond à un commentaire interne.

En revanche, l'idée serait de conserver le type "image" en entrée (dans l'éditeur Scenari) puis appliquer deux feuilles de transformation XSL pour obtenir un imagemap en sortie :

- une première pour générer le "squelette" de l'imagemap, c'est-à-dire tout, sans s'occuper des éléments "p" de commentaires (en générant néanmoins des url pour les "xref"),
- une seconde pour générer des fichiers DITA externalisés pour chaque ensemble de commentaires (les noms des fichiers doivent correspondre avec les noms des "xref" générés avec la première transformation).

Ainsi, le contenu, saisi en tant qu'"image" sera transformé en un imagemap DITA qui pourra lui-même être transformé via les publications standards de DITA (PDF, HTML).

Annexe 2 Liens utiles

DITA FAQ : *http://www.ditausers.org/dita-ot/demo/faq/DITA-faq.html*

DITA reference book : *http://www.oasis-open.org/committees/download.php/11347/ditaref-book.pdf*

Annexe 3 Notice technique de la spécialisation en DITA

Introduction

La spécialisation est une des caractéristiques essentielles de la DITA. Bien que nous ne

l'ayons pas utilisée dans ce projet, une partie de l'état de l'art y a été consacrée. Cette notice a pour but de résumer l'intérêt et les mécanismes de spécialisation en DITA.

Spécialisation de type

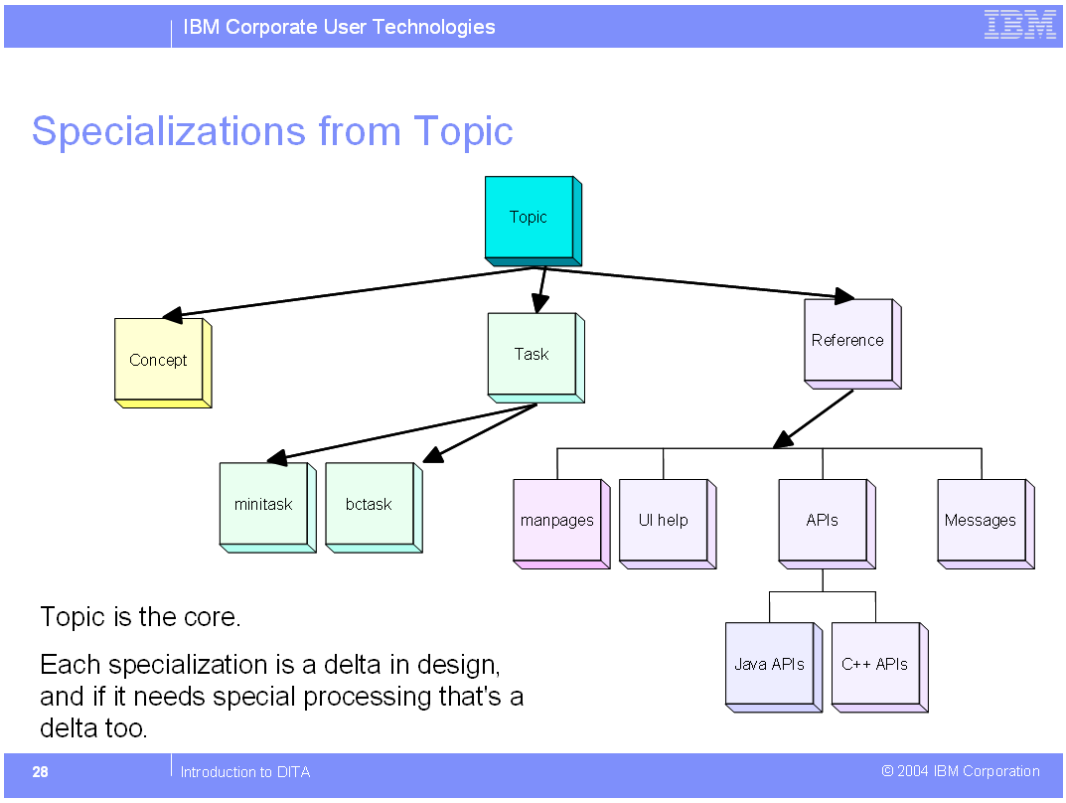
Les types de base de DITA ne sont pas toujours suffisants (trop généraux) pour la documentation technique, mais il existe un mécanisme de spécialisation qui repose fortement sur la notion d'héritage. Cela permet d'intégrer des éléments spécifiques tels que des procédures d'urgence ou des plans, qui peuvent faire l'objet de transformations spécifiques pour un affichage approprié. Cependant la force de la spécialisation est qu'un moteur de transformation prévu pour les éléments de base permettra quand même l'affichage des éléments spécifiques, en les affichant de la même manière que les éléments dont ils dérivent. Cette caractéristique est particulièrement intéressante dans le cas d'échange de contenus spécialisés entre deux sociétés/organismes utilisant le standard DITA (et n'ayant pas nécessairement connaissance des spécialisations de l'autre). De plus, on a la possibilité, grâce au renommage des balises, d'avoir un langage typé métier.

Spécialisation de domaine

Il ne semble pas que nous ayons besoin de la spécialisation de domaine.

Spécialisation de topic : concept, task et reference

Pour bien comprendre comment fonctionne la spécialisation, il est à noter que concept, task et reference sont des spécialisations de topic. Cela est illustré par le schéma suivant :



(source :*http://www.xml.gov/documents/completed/ibm/dita.ppt*)

Cela signifie qu'une nouvelle DTD a été créée : concept.dtd par exemple.

Celle-ci reprend des éléments de topic.dtd de cette manière :

```
<!-- Embed topic to get generic elements -->
<!ENTITY % topic-type PUBLIC
"-//OASIS//ELEMENTS DITA 1.2 Topic//EN"
"/base/dtd/topic.mod">
%topic-type;
```

Et d'autre part redéfinit ses propres éléments de cette manière :

```
<!-- Embed concept to get specific elements -->
<!ENTITY % concept-typemod
PUBLIC
"-//OASIS//ELEMENTS DITA 1.2 Concept//EN"
"concept.mod">
%concept-typemod;
```

C'est donc au sein de concept.mod que seront spécifiés les éléments et leur cardinalité. Pour chaque topic, il faut en effet au sein de DITA une DTD à laquelle sont liés des .mod (définition des éléments) et un .ent (déclaration des entités utilisées). De plus des % sont utilisés comme références (en fait les éléments références référencent des éléments du même nom).

Enfin, étant donné qu'on a renommé des éléments existants, il faut à l'aide de class faire le lien entre l'élément déjà existant (père, e.g. body) et celui qu'on vient de redéfinir (fils, e.g. conbody)

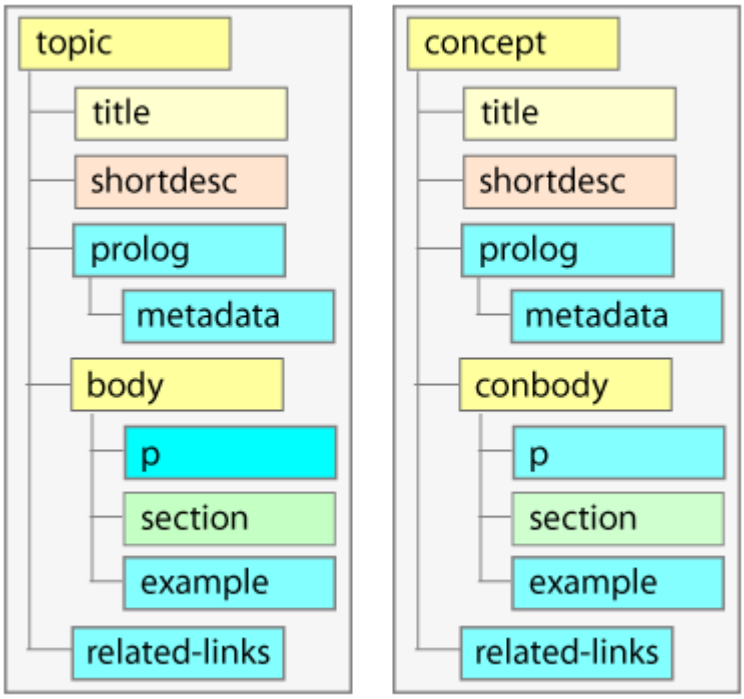
de cette manière :

```
<!ATTLIST concept %global-atts; class CDATA "- topic/topic concept/concept ">
<!ATTLIST conbody %global-atts; class CDATA "- topic/body concept/conbody ">
```

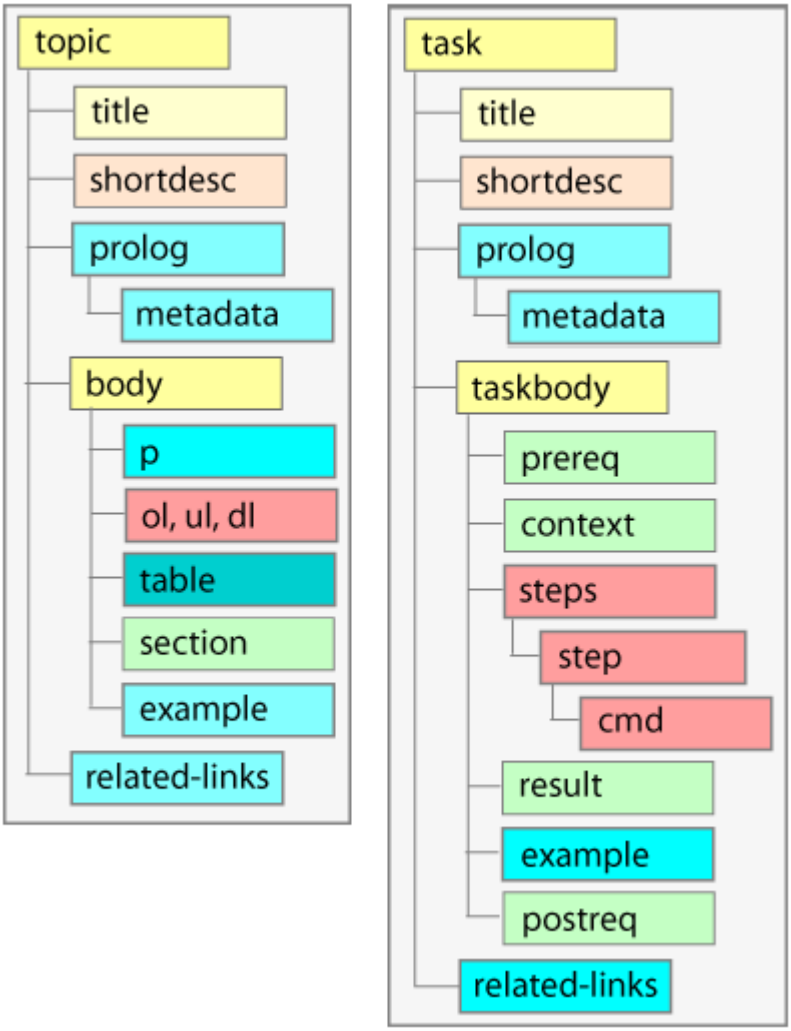
C'est ce mécanisme qui permet d'utiliser une transformation définie pour le type général sur le type spécialisé.

Une fois toutes les spécialisations définies au sein des nouvelles DTD, on se retrouve avec les configurations suivantes :

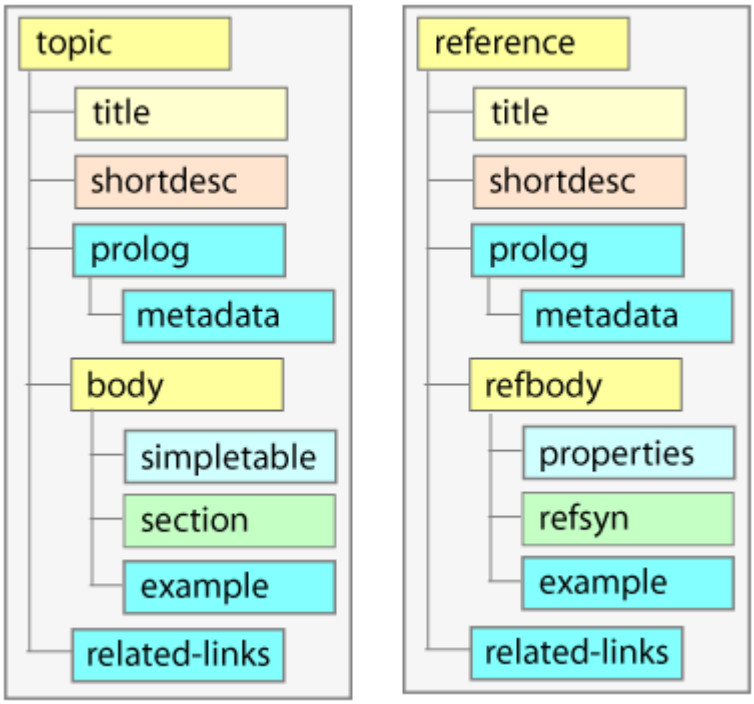
Concept topic :



Task topic :



Reference topic :



On peut alors appliquer ou non un XSL spécifique sur les éléments spécifiques. On peut également de la même manière aller plus loin et spécialiser Concept, Task ou Reference.

(source des images : *http://www.ditausers.org/training/DITATopics/*)

Créer sa propre spécialisation : outils

La création de documents DITA et de spécialisations nécessite à la fois un éditeur XML, un outil pour vérifier que les documents XML sont valides par rapport à la DTD créée, ainsi qu'un moteur de transformation de la DITA vers d'autres support (PDF, web). Oxygen répond à l'ensemble de ces besoins.

Créer sa propre spécialisation : règles

Lorsqu'on spécialise un type déjà existant, on peut :

- reprendre tout ou partie des éléments du type,
- renommer éventuellement les éléments repris,
- restreindre éventuellement les cardinalités des éléments repris.

En revanche, **on ne peut pas ajouter de nouveaux éléments : la spécialisation est restrictive.**

Le changement de cardinalité répond aux règles suivantes :

- un élément "unique et requis" **doit** être conservé,
- un élément "optional" peut rester "optional", devenir "unique et requis" ou être supprimé,
- un élément "one or more" peut rester "one or more", devenir "unique et requis" ou séparé en éléments renommés (exemple : a+ peut devenir a1, a2+, a3?, a4*, mais

pas a1? qui est moins restrictif),

- un élément "zero or more" suit les mêmes règles qu'un élément "one or more" et peut aussi devenir "optional" ou être supprimé,
- un choix de plusieurs éléments (exemple a | b) peut devenir un choix partiel de ces éléments,
- **dans tous les cas précédents, l'élément peut toujours être renommé.**

Créer sa propre spécialisation : exemple

1. Pour créer sa propre spécialisation il faut commencer par un travail d'analyse sur le corpus et une prise de connaissance de la DITA pour identifier à quel élément de base on peut se rapporter le plus. Par exemple s'agit-il plutôt d'un topic ? d'un concept ? d'une reference ? d'une task ? Afin de tester comment marche techniquement la spécialisation, on peut par exemple vouloir créer un contenu de type introduction, avec juste un titre et du texte, l'élément qu'on va alors utiliser est un topic. A noter que cet exemple n'est pas pertinent dans un contexte de production mais facile à comprendre.
2. Une fois qu'on a identifié l'élément qui fournit la meilleure base pour notre spécialisation, il faut commencer par créer du contenu en DITA en utilisant le dit élément. Dans notre cas nous créons donc une map, contenant divers topicref vers des éléments et en particulier un topic (l'introduction) qui comporte un title et un body contenant lui même des p. Dans la pratique il est recommandé de produire un nombre suffisamment significatif d'exemples de contenus afin de pouvoir bien identifier toute la variabilité dès le début et ainsi faire une spécialisation efficace. Si le contenu a été bien écrit on doit pouvoir transformer la map en un document PDF par exemple à l'aide d'Oxygen (gestionnaire de cartes DITA).
3. Il faut ensuite passer à l'étape de spécialisation. On commence par créer une DTD correspondante qu'on associe avec le fichier DITA en question.
    1. Dans notre exemple on veut spécialiser topic. Comme concept est une également spécialisation de topic et que l'on se limite à des éléments basiques (title et body) présents dans concept le plus simple est de modifier la DTD de concept selon nos besoins. Si l'on voulait spécialiser task on procéderait de la même manière en reprenant la DTD de task.
    2. On commence par changer les noms des attributs. Dans notre exemple on choisit intro pour la racine et introbody pour le corps. Puis on se restreint aux éléments dont on a besoin. Par exemple
    <!ENTITY % intro.content
    "((%title;),
    (%titlealts;)?,
    (%abstract; |
    %shortdesc;)?,
    (%prolog;)?,
    (%introbody;)?,
    (%related-links;)?,
    (%intro-info-types;)*)"
    >
    devient
    <!ENTITY % intro.content
    "((%title;),
    (%introbody;)?)"

>
en respectant la règle selon laquelle on ne peut que être que plus restrictif lorsque l'on spécialise un élément. (voir)
De cette manière, on définit l'ensemble des éléments dont on a besoin au sein du .mod

3. Une fois les éléments définis avec les cardinalités adéquates il faut spécifier le lien d'héritage des différents éléments de cette manière :
<!ATTLIST intro %global-atts; class CDATA "- topic/topic intro/intro ">
<!ATTLIST introbody %global-atts; class CDATA "- topic/body intro/introbody ">
Cela signifie que intro est une spécialisation de topic et introbody une spécialisation de body.

Il est important lors de la création de la DTD de procéder de manière itérative. En particulier il faut prendre le soin de vérifier régulièrement que le .dita est toujours valide par rapport à la DTD et que la transformation de la map DITA en PDF par exemple fonctionne bien. En effet, Oxygen ne comporte pas de debugger lorsque la génération échoue et il devient donc très difficile de trouver la source de l'erreur si l'on a tout édité d'un coup.

Cas des images : imagemap

L'exemple d'une image à laquelle on puisse associer des commentaires par zone a été évoqué lors de la séance du 30 novembre. La question était de savoir si DITA permettait de faire quelque chose de similaire à l'exemple suivant (tiré de ce qu'on peut faire avec la chaîne éditoriale Dokiel) :

```
<image href="...">
<zone id="..." x="..." y="..." width="..." height="...">
<p>un commentaire sur cette zone</p>
<p>un autre commentaire</p>
</zone>
<zone id="..." ...>
<p>...</p>
</zone>
</image>
```

Element imagemap dans DITA

Dans la référence de la DITA, il existe un élément nommé "ditamap", qui est utilisable au sein du "body" d'un "topic" et suit la structure suivante :

```
<imagemap>
<image href="...">
```

```
<area>
<shape>rect</shape>
<cords>0, 0, 100, 100</cords>
<xref href="..."></xref>
</area>
<area>
...
</area>
<image>
</imagemap>
```

Observations

Dans ce cas, il est impossible de spécialiser imagemap en "image" : en effet, l'élément "xref", qui est une url vers un fichier DITA externe, ne peut pas être changé en "p", qui lui correspond à un commentaire interne.

En revanche, l'idée serait de conserver le type "image" en entrée (dans l'éditeur Scenari) puis appliquer deux feuilles de transformation XSL pour obtenir un imagemap en sortie :

- une première pour générer le "squelette" de l'imagemap, c'est-à-dire tout, sans s'occuper des éléments "p" de commentaires (en générant néanmoins des url pour les "xref"),
- une seconde pour générer des fichiers DITA externalisés pour chaque ensemble de commentaires (les noms des fichiers doivent correspondre avec les noms des "xref" générés avec la première transformation).

Ainsi, le contenu, saisi en tant qu'"image" sera transformé en un imagemap DITA qui pourra lui-même être transformé via les publications standards de DITA (PDF, HTML).