



**Mastère Spécialisé en
Ingénierie des Systèmes Informatiques Ouverts
MSIO**

***Création d'un répertoire de
schémas XML***



Thèse professionnelle

Réalisée et présentée par :
Abderrazak MKADMI

Dirigée par :
Marc LANGLOIS (EDIFRANCE)
Philippe LEMARQUAND (École Centrale)

2001 - 2002

REMERCIEMENTS

Je tiens à remercier M. Bertrand Bruller, responsable du mastère SIO, de m'avoir donné l'occasion de suivre cette formation à l'école centrale Paris,

M. Philippe Lemarquand qui a accepté d'être mon maître de stage, et qui nous a donné les outils nécessaires pour pouvoir conduire un projet au sein d'une entreprise,

M. Marc Langlois, délégué général d'EDIFRANCE, de m'avoir accueilli et veillé au bon déroulement de mon stage,

Mme Juliette Dewavrin, Directrice de communication à EDIFRANCE pour ses conseils et ses orientations pédagogiques,

Mlle Nathalie Lesourd (Responsable informatique) de m'avoir aidé lors des moments de blocage et Mlle Maria Chaves (Responsable comptabilité) pour sa sympathie et son sourire le matin tout en m'ouvrant la porte,

M. Dominique Richard pour sa sympathie et pour les discussions fructueuses que nous avons eu pendant les pauses, concernant l'EDI, la mondialisation, le pétrole en Irak, la coopération en Afrique noire ...

Tous les enseignants et les élèves du mastère SIO qui m'ont accompagnés depuis le 11 septembre 2001 !!!

TABLE DES MATIERES

LISTE DES FIGURES	6
RÉSUMÉ.....	7
INTRODUCTION GÉNÉRALE.....	8
1- DÉFINITION DES CONCEPTS	10
1-1 XML ?	10
1-2 DTD, XML schéma, XSL et XSLT	11
1-2-1 DTD : définition d'un document	11
1-2-2 Schéma-XML.....	12
1-2-3 XSL.....	13
1-2-4 XSLT	13
1-3 ebXML	14
1-4 Core components.....	14
1-5 UML.....	15
2- CONTEXTE DU STAGE.....	15
2-1 Présentation du mastère SIO	15
2-2 Présentation de l'entreprise (EDIFRANCE).....	16
2-2-1 Missions d'EDIFRANCE	16
2-2-2 Groupes de travail chez EDIFRANCE	16
1 ^{ÈRE} PARTIE : CONCEPTION DU PROJET	18
1- CONTENU DU PROJET.....	19
2- ÉNONCÉ FONCTIONNEL DUBESOIN	20
2-1 Énoncé du besoin	20
2-2 Cycle d'utilisation et environnement	20
2-3 Analyse fonctionnelle des besoins	20
A- Fonctions liées à l'alimentation de la base	21
B- Fonctions liées à l'identification de l'utilisateur	22
C- Fonctions liées à la consultation du répertoire	22
D- Fonctions liées à la navigation dans un <i>modèle</i>	23
E- Fonctions liées à la récupération des documents	23
2-4 Contraintes	23
3- MODÉLISATION DU PROJET	24
3-1 Modèles des données des <i>modèles</i> XML	24
3-1-1 modèle simple	25
3-1-2 Modèle sophistiqué	26
3-1-3 Modèle pour les données élémentaires (Core Components)	27
3-2 Modèle de données de la base	28
3-2-1 Description du Modèle des données	29
3-3 Interface d'accès.....	35
3-3-1 Modèle d'accès au public	35
3-3-2 Modèle d'accès au participant d'un groupe de travail	36
3-3-3 Modèle d'accès au soumissionnaire du <i>modèle</i>	37
3-4 Architecture logique	39
3-5 Architecture physique de base	39

3-6 Architecture réelle utilisée	40
4- OUTILS DE DÉVELOPPEMENT ET D'EXPLOITATION	41
4-1 Serveur : Logiciels de bases	41
4-2 Langage de programmation.....	41
4-3 Clients	42
5- CONCLUSION.....	42
2^{ÈME} PARTIE : RÉALISATION DU PROJET	43
1- LES OUTILS DE DÉVELOPPEMENT.....	44
1-1 Système d'exploitation.....	44
1-2 Serveur WEB	44
1-3 Moteur de servlets : Tomcat.....	45
1-4 Cocoon.....	45
1-4-1 Relation entre Cocoon et Tomcat	45
1-5 Mysql.....	45
1-5-1 Justification de l'utilisation d'une base de données :	46
2- INSTALLATION ET CONFIGURATION	46
2-1 Installation du JDK	46
2-2 Installation de Tomcat.....	47
2-3 Intégration avec Apache	48
2-4 Installation de cocoon.....	49
2-5 Démarrage et arrêt de Tomcat Avec Apache	49
3- CONCEPTION DE LA BASE DE DONNÉES	49
4- CONNEXION ET ACCÈS À LA BASE DE DONNÉES	51
4-1 Principes de fonctionnement	52
4-2 Scripts de communication avec Mysql	52
5- PRÉSENTATION DU SITE WEB	53
5-1 Consultation des modèles.....	55
5-2 Recherche des modèles	55
5-3 Exportation des core components	56
5-4 Soumission et modification des modèles	56
5-4-1 Formulaires de soumission de modèles	59
5-5 Révision des modèles.....	59
6- SCRIPTS DE TRAITEMENT DES FORMULAIRES	59
6-1 Traitement des champs simples	60
6-1-1 traitement lors de la soumission.....	60
6-1-2 traitement lors de la modification (mise à jour).....	61
6-1-3 traitement lors de la consultation.....	61
6-2 Traitement des champs à choix multiple :.....	61
6-2-1 traitement lors de la soumission :	62
6-2-2 traitement lors de la modification.....	62
6-2-2 traitement lors de la consultation.....	63
6-3 Traitement des champs répétitifs :	64
6-3-1 traitement d'Insertion des données dans la table principale	65
6-3-2 traitement pour l'affichage des outils existants dans la base.....	66
6-3-2 traitement pour la récupération des Ids	66
6-3-3 insertion des IDs dans la table intermédiaire OUTILASSOCIE_DM	67

6-4 Traitement des fichiers attachés :	67
7- TESTS DE VALIDATION DES FONCTIONNALITÉS DU SITE.....	69
CONCLUSION GÉNÉRALE.....	74
1- Suite du projet.....	75
2- Intérêts du projet	75
3- connaissances acquises lors du stage	75
RÉFÉRENCES BIBLIOGRAPHIQUES	77
ANNEXES	79
ANNEXE 1 : PLAN QUALITÉ.....	80
ANNEXE 2 : GLOSSAIRE.....	83
ANNEXE 3 : CRÉATION DES TABLES DANS LA BASE DE DONNÉES	84
ANNEXE 4 : DIAGRAMMES	89
1 Diagramme de classes :	89
2 Diagramme de cas d'utilisation.....	90
2-1 Consultation	90
2-2 Création.....	90
2-3 Révision.....	91
3 Diagrammes de séquences.....	91
3-1 Consultation du public	91
3-2 Consultation du participant au groupe du travail	92
3-3 Révision du modèle	92
ANNEXE 5 : FORMULAIRES DE DÉCLARATION DES MODÈLES	93
1 Formulaire de déclaration d'un modèle simple	93
2 Formulaire de déclaration d'un modèle core component	97
3 Formulaire de déclaration d'un core component à partir d'un fichier Excel	102

LISTE DES FIGURES

Figure 1 : Modèle XML simple	25
Figure 2 : Modèle XML sophistiqué	26
Figure 3 : Modèle XML du composant	27
Figure 4 : Modèle de données de la base	28
Figure 5 : Modèle d'accès au public	36
Figure 6 : Modèle d'accès à un participant de groupe de travail	37
Figure 7 : Modèle d'accès au soumissionnaire d'un modèle	38
Figure 8 : Architecture logique du site web	39
Figure 9 : Architecture physique de base du site	39
Figure 10 : Architecture physique réellement utilisée	41
Figure 11 : Interface d'accueil du site	54
Figure 12 : Interface de recherche	56
Figure 13 : Interface d'identification pour les inscrits	57
Figure 14 : Interface d'identification pour les nouveaux propriétaires	57
Figure 15 : Interface sommaire du site	58

RÉSUMÉ

Français

XML est devenu le format standard de structuration et d'échange de données. Ce format associe aux données une structure sémantique qui permet de comprendre le sens des données. Cette structure est définie selon un modèle sous forme de schémas ou de DTD qui donne les règles d'assemblage et d'ordonnement des données.

De ce fait, Il devient alors intéressant de pouvoir partager ces modèles, ainsi que toutes les informations associées qui permettront de les utiliser de façon rapide et efficace. C'est dans ce contexte que le projet « Création de répertoire de modèles XML » a vu le jour.

Ce rapport essaie donc de définir le contexte du projet, ainsi que tout son déroulement de la conception à la réalisation technique. Nous avons présenté dans un premier lieu une partie théorique sur le contenu du projet, les différents acteurs, ainsi que les différentes réactions et activités qui peuvent se dérouler entre eux, ou entre eux et le système. Dans cette partie, nous avons essayé de définir les spécifications des besoins et des différents outils de développement.

Nous avons présenté également la méthodologie suivie pour l'installation et la configuration de la plate-forme de développement y compris la base de données.

Enfin et dans une deuxième partie qui traduit la réalisation technique du projet, nous avons présenté de façon détaillée la base de données et les différentes interfaces de soumission, d'accès, de modification et de révision des modèles XML.

Mots clés : Répertoire de modèles XML, Schémas XML, Composant élémentaire, Base de données MYSQL, développement PHP, site web ...

Anglais

XML became the standard format of structuring and data exchange. This format associates the data a semantic structure which makes it possible to understand the significance of the data. This structure is defined according to a model in the form of schemas or DTD which gives the rules of data's gathering and organization. So It then becomes interesting to be able to share these models, as all associated information which will make it possible to use them in a fast and effective way. It is in this context that the project "Creation of repository of models XML" was born.

This report tries to define the context of the project and the processus of the design. We presented in a first place a theoretical part on the contents of the project, the various actors, as well as the various reactions and activities that it is between them, or between them and the system. In this part, we tried to define the specifications of the needs and the various development tools.

We also presented the methodology followed for the installation and the configuration of the platform of development including the data base.

Finally and in a second part which translates the technical realization of the project, we presented the data base and the various interfaces of declaration, access, modification and revision of models XML.

Key words : Repository of XML models, XML Schemas, Core component, MYSQL Data Base, PHP development, web site ...

INTRODUCTION GÉNÉRALE

Les échanges de données informatisées (EDI) classiques sont complexes de mise en oeuvre ; les coûts des logiciels spécifiques restent élevés. La plus grande partie des PME et la totalité des ménages en sont exclus. La généralisation d'Internet tant dans le secteur des entreprises que des ménages est un facteur déterminant de l'évolution.

Dans ces conditions, XML (eXtensible Markup Language), formellement standardisé par le W3C, apparaît comme le standard d'échange pertinent et mondial des données. En effet, les travaux de normalisation menés par le W3C, les développements des éditeurs de logiciels et les préconisations de différents groupes et consortium (ebXML, OASIS) conjuguent leurs efforts pour définir, promouvoir et utiliser XML dans différentes situations. Ce méta-langage est utilisé aujourd'hui par tous : les fournisseurs d'ERP, les éditeurs de middleware, les fournisseurs de bases de données, etc. Les raisons de ce consensus sont à chercher du côté de la simplicité et de la richesse d'expression d'XML.

Par ailleurs, les technologies de l'Internet permettent la mise en oeuvre de solutions intégrées entre les différentes entreprises et administrations ou avec leurs partenaires qui reposent sur une gestion cohérente des échanges entre les acteurs et une intégration avec les systèmes d'information existants. Néanmoins, les délais de développement et les coûts associés peuvent être élevés pour les propriétaires et les gestionnaires de données. C'est dans cette perspective que XML offre une possibilité de réduction des délais et des coûts en facilitant la compatibilité et la réutilisation des données. Il est aujourd'hui perçu comme une solution « prometteuse » qui doit être envisagée lors des projets traitant de gestion et d'échange de document et de données informatisées.

Cependant, pour que les objectifs d'XML, à savoir permettre l'échange généralisé intersectoriel quel que soit le type d'acteur, soient réellement atteints, il faut que les éléments et les structures d'informations d'intérêt commun soient identifiés et réutilisés par les entreprises, les administrations et les communautés d'intérêt.

C'est dans ce même ordre d'idées que le projet de la création d'un répertoire de schémas XML a vu le jour pour pouvoir identifier, partager et réutiliser les *modèles* de documents par les différentes applications XML. Cette initiative est prise par EDIFRANCE (Association pour le développement des échanges électroniques professionnels), MUTU-XML et le GFII (Groupement Français de l'Industrie de l'Information), avec le soutien de la FING (Fondation Internet Nouvelle Génération). Elle a pour but de promouvoir l'utilisation d'XML en milieu ouvert¹.

« Un répertoire de modèles pour XML est une base de données permettant à tout un chacun de prendre connaissance des modèles qui existent dans un domaine d'activité particulier, pour un besoin particulier. Il répond aux questions que peut se poser un utilisateur, une entreprise : quels sont les modèles qui me permettent d'échanger avec mes partenaires ? Quelles sont les pratiques habituelles dans mon domaine d'activité ? ...

Mais ce répertoire doit aussi répondre à des questions plus précises telles que : je sais que je dois être compatible avec le modèle XXX mais au fait, qu'y a-t-il dans ce modèle ? Comment est-il organisé et que signifient tous les objets qui y sont référencés ? »²

Ce répertoire de *modèles*, tel qu'il est présenté par ses auteurs, est une base de données qui permet de :

¹⁻² EDIFRANCE, Mutu-XML, FING, GFII. -<http://www.repertoire-modeles.org/defProj/decl/defProj.html>

- favoriser l'interopérabilité et la dématérialisation des échanges entre les différents utilisateurs de la technologie XML ;
- faciliter la conservation des documents dématérialisés échangés ;
- informer les gens de l'existence de tel modèle dans tel domaine d'activité, pour un besoin particulier, ainsi de ses différents composants ...

Le présent travail s'appuie sur l'ensemble des travaux menés par des différentes organisations et sur les idées des personnes ressources dans le domaine³. Il tient compte également des travaux réalisés par OASIS (Organization for the Advanced of Structured Information Systems).

Le projet n'intègre pas de manière formelle la problématique de validation du contenu des schémas. Si dans certains domaines, cette validation formelle est nécessaire, l'outil permet de la mettre en œuvre.

Le projet a une fonction de registre qui permet de retrouver un modèle stocké sur le même serveur ou ailleurs. L'accès au modèle dans ce dernier cas se fait par un lien URI. Cette possibilité évite à une organisation qui gère son propre répertoire d'être contrainte à maintenir deux sources d'information. Il permet également de stocker directement sur le serveur des modèles accessibles par l'intermédiaire de la fonction registre. Cette possibilité permet aux organisations et/ou aux groupes de travail de profiter librement d'un espace de travail.

1- DÉFINITION DES CONCEPTS

1-1 XML ?

Le langage XML (Extensible Markup Language) a été développé par un groupe de travail constitué sous les auspices du Consortium du World Wide Web (W3C). Il représente, selon la « Recommandation W3C du 10 février 1998 », un sous-ensemble de SGML dont le but est de permettre au SGML générique d'être servi, reçu, et traité sur le web de la manière qui est maintenant possible avec le langage HTML. XML a été conçu pour la facilité de l'exécution et pour l'interopérabilité avec le SGML et le HTML.

Comme le HTML (Hypertext Markup Language) il s'agit d'un langage de balisage (Markup), c'est-à-dire un langage qui présente de l'information encadrée par des balises. Mais contrairement à HTML, qui présente un jeu limité de balises orientées présentation (titre, paragraphe, image, lien hypertexte, etc.), XML est un métalangage, qui va permettre d'inventer à volonté de nouvelles balises pour isoler toutes les informations élémentaires (titre d'ouvrage, prix d'article, numéro de sécurité sociale, référence de pièce...), ou agrégats d'informations élémentaires, que peut contenir une page Web.

Exemple :

```
<?xml version = »1.0 « encoding= »ISO-8859-1 « ?>
<Societe>
<Nom>EDIFRANCE</Nom>
<Adresse>
<Rue>8 Rue Saint Marc</Rue>
<CodePostal>75002</CodePostal>
<Ville>Paris</Ville>
<Telephone>33 1 55 34 96 96</Telephone>
</Adresse>
</Societe>
```

³ Je cite notamment Marc Langlois (EDIFRANCE) et Pierre Attar (Mutu-XML).

XML utilisant un format texte et des balises pour délimiter les données, les fichiers XML sont presque toujours d'une taille plus importante que les formats binaires équivalents.

Les objectifs de la [recommandation du 10 février 1998](#) sont les suivants :

- XML devrait pouvoir être utilisé sans difficultés sur Internet ;
- XML devrait soutenir une grande variété d'applications ;
- XML devra être compatible avec SGML ;
- Il devrait être facile d'écrire des programmes traitant les documents XML ;
- Le nombre d'options dans XML doit être réduit au minimum, idéalement à aucune ;
- Les documents XML devraient être lisibles par l'homme et raisonnablement clairs ;
- La conception de XML devrait être préparée rapidement ;
- La conception de XML sera formelle et concise ;
- Il devrait être facile de créer des documents XML ;
- La concision dans le balisage de XML est de peu d'importance.

Ces spécifications permettent la mise en oeuvre d'échanges s'appuyant sur XML qui se démarquent des EDI classiques notamment par la capacité d'adaptation au contexte de l'échange qui est plus souple que celle nécessitée par les traducteurs EDI classiques.

Comparé à l'EDI conventionnel, XML offre la capacité de :

- transmettre des données multimédia (image et vidéo pour illustrer un catalogue en ligne par exemple) ;
- afficher des données sous une forme humaine par l'utilisation de feuilles de style (pour les entreprises qui ne voudraient pas intégrer les messages directement dans leurs systèmes d'information) ;
- convertir facilement une structure de message en une autre structure de message (ce qui facilite l'intégration des données dans des applications existantes)... et cela tout en utilisant les technologies bon marché d'Internet.

C'est pourquoi XML est retenu comme le langage d'avenir pour la généralisation des échanges électroniques, que ce soit entre les grandes entreprises, entre les PME ou entre les deux groupes.

1-2 DTD, XML SCHÉMA, XSL ET XSLT

1-2-1 DTD : DÉFINITION D'UN DOCUMENT

Le rôle d'une DTD est de définir la structure d'un document XML. Une DTD est caractérisée par un ensemble de règles qui spécifient les éléments du document XML, ainsi que leur ordre et leur fréquence d'apparition.

La DTD est une caractéristique optionnelle d'un fichier XML. Si le fichier XML n'a pas de DTD, ce dernier devra alors être **bien formé**" (well-formed) et respecter les règles imposées par XML. Si une DTD est rattachée au fichier XML, ce dernier est automatiquement considéré comme **valide** puisque la définition des marqueurs du document est intégrée.

La DTD peut être intégrée au fichier XML, on dit alors qu'elle est interne ou externe et dissociée du fichier XML ; le nom du fichier contenant la DTD lui est alors associé.

Une DTD commence toujours par `<!DOCTYPE` et se termine par `>`. La racine du document est renseignée après `<!DOCTYPE` et doit être également définie comme un `ELEMENT` de la DTD.

La DTD est donc construite à partir d'un ensemble de déclarations permettant de définir le type, la nature et les contraintes liées à chaque nouvelle balise:

- déclaration de types d'éléments : Ils permettent de définir le contenu du fichier XML ;
- la déclaration de listes d'attributs : Ils permettent d'enrichir la sémantique des éléments ;
- la déclaration d'entités ;
- la déclaration de notations.

Le XML Working Group travaille actuellement sur le remplacement des DTDs par les *schémas-XML*.

1-2-2 SCHÉMA-XML

Une alternative aux DTDs a été proposée au W3C sous le nom de XML-Data, dont les schémas XML sont un sous-ensemble.

De même qu'une DTD, un schéma permet de définir un ensemble de règles visant à définir un document XML, et notamment les marqueurs autorisés, leurs attributs et relations les uns par rapport aux autres. Mais contrairement à une DTD, un schéma permet de définir des types pour les données. De plus, un schéma XML est un document XML à part entière et peut donc être édité et manipulé à partir de n'importe quel outil d'édition ou de traitement XML.

Par exemple, avec une DTD on pourra définir un marqueur <NOM>, dont le contenu pourra être n'importe quel type de données. Toute la puissance des schémas réside dans leur faculté à donner un type aux balises permettant ainsi de renforcer le contrôle syntaxique du message. On pourra ainsi facilement obliger un utilisateur à ne saisir que des caractères pour le marqueur.

On préfère les schémas XML aux DTDs pour diverses raisons dont les plus importantes sont :

- une DTD est difficile à lire ;
- une DTD est non extensible, car ce n'est pas un document XML ;
- une DTD ne permet pas de typer les données ;
- une DTD ne supporte pas les espaces de noms (Namespace) ;
- une DTD est plus concise... mais moins riche qu'un schéma XML.

La définition d'un schéma XML se réalise par l'intermédiaire de plusieurs éléments chargés de représenter l'arborescence et les informations d'un document XML :

- **L'élément racine** d'un schéma XML peut incorporer des schémas extérieurs avec un espace de noms identique ou différent.

Les **espaces de noms** permettent à un même document d'utiliser plusieurs langages de balisage XML, par exemple XHTML + MathML + SVG + RDF

Exemple :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:svg="http://www.w3.org/2000/svg"
      xmlns:m="http://www.w3.org/1998/Math/MathML">
<head>...</head>
<body>
<h1>Les données</h1>
<m:math>
  <m:mfrac>...</m:mfrac>
</m:math>
<h1>Le graphe</h1>
<svg:svg width="4cm" height="8cm">
  <rect x="0.5cm" y="0.5cm" width="2cm" height="1cm"/>
</svg>
</body>
</html>
```

```
...
</svg:svg>
</html>
```

- **Les éléments et les attributs XML** sont chacun spécifiquement représentés dans un schéma. Il est possible de les regrouper pour des raisons de commodités, ou encore de représenter des nœuds XML indéfinis par des éléments de schéma génériques.
- **Le contenu des éléments XML** est également décrit précisément tout d'abord par un type de données pouvant être soit simple, soit complexe.

Un type complexe signifie qu'un élément XML pourrait posséder des attributs, des éléments et une valeur simple.

```
<element attribut="valeur">
  Valeur de l'élément
  <element_enfant_1/>
  <element_enfant_N/>
</element>
```

L'ordre et l'apparition des éléments enfants ainsi que des **alternatives** peuvent être de la même façon définis dans le schéma XML.

Un type simple signifie quant à lui que l'élément ou l'attribut XML ne contient qu'une valeur simple.

```
<element>
  Valeur de l'élément
</element>
```

Certains éléments de schéma XML permettent d'exprimer des contraintes d'unicité et de référencement par rapport au contenu de plusieurs éléments et attributs XML. Ces contraintes d'identité sont semblables aux clés primaires ou étrangères des bases de données SQL.

Enfin, **les éléments d'annotation** offrent la possibilité soit d'insérer des commentaires, soit des informations destinées à des applications.

1-2-3 XSL

Parce que des données brutes n'ont jamais été exploitables en tant que telles, XML fait appel à un langage de présentation pour assurer la mise en forme des données qu'il formalise et transporte ; le XSL (eXtensible Stylesheet Language).

Un peu comme les feuilles de style CSS (Cascade Style Sheet) du langage HTML et DSSSL (Document Style Semantics and Specification Language) pour des documents SGML, XSL permet de manipuler les documents XML.

En effet, les fonctions principales d' XSL sont :

- la transformation d'un document XML en un autre document XML ;
- le formatage d'un document XML en un format d'affichage.

XSL est à XML ce que les feuilles de styles (les fameuses "CSS") sont à HTML: un moyen de formater les documents.

1-2-4 XSLT

XSLT est conçu pour être utilisé comme une partie de XSL, le langage des feuilles de style de XML. XSLT est aussi conçu pour être utilisé indépendamment de XSL. Cependant, XSLT n'est pas censé être utilisé comme un langage de transformation XML à vocation

générale. Il a surtout été conçu pour les types de transformations nécessaires lorsque XSLT est utilisé comme une partie de XSL.

1-3 ebXML

ebXML est le nouveau standard des échanges électroniques professionnels basé sur l'utilisation des technologies de l'Internet et plus particulièrement du langage XML. Il s'agit d'une initiative des Nations Unis et d'OASIS pour créer une infrastructure basée sur XML et l'Internet ouvrant le commerce électronique à toutes les entreprises, petites ou grandes, sur tous les continents.

Signifiant « e-business XML », ebXML définit des structures facilitant les échanges et le commerce électronique, il représente une architecture d'échange fondée sur des scénarios d'affaires types et voulant aboutir à une refondation de l'EDI. Une première plate-forme a été implémenté en mai 2001.

Loin de devoir remplacer EDIFACT, ebXML se positionne dans la complémentarité et dans la continuité. EDIFACT est particulièrement adapté aux échanges de gros volumes avec des partenaires stables, alors que ebXML doit répondre, en plus, à la problématique des petits échanges entre partenaires épisodiques.

De fait, en s'appuyant sur les technologies d'Internet et sur le métalangage XML, ebXML permet de couvrir deux domaines qui n'avait pas pu l'être avec EDIFACT :

- Les échanges électroniques entre PME et / ou entre PME et grandes entreprises ;
- Les échanges de données non textuelles.

1-4 CORE COMPONENTS⁴

Traduit en Français : les composants élémentaires, les core components sont « des pièces d'assemblage, chacune avec une définition unique en sémantique d'affaire. L'assemblage de ces pièces se fait dans des contextes particuliers (différents processus d'affaires) où elles acquièrent une signification particulière : par exemple, le composant élémentaire « date » peut être employé dans l'entité d'information d'affaires « date de publication ». »

Trois catégories de composants élémentaires peuvent être distinguées :

- *CCT* : *Core Component Type* (traduit en français : **types de composant élémentaire**) : brève liste, par exemple date, servant à former des composants élémentaires, mais sans signification d'affaires avant d'être combinés ;
- *BCC* : *Basic Core Component* (en français : **Composant Élémentaire de Base**) : bâti avec un type de composant élémentaire, représente un seul concept d'affaires avec une définition unique en sémantique d'affaires ; et
- *ACC* : *Aggregate Core Component* (en français : **Composant Élémentaire Assemblé**) : bâti avec plusieurs pièces d'information d'affaires, représente un seul concept d'affaires avec sa propre définition unique en sémantique d'affaires. Exemple : adresse postale.

⁴ Toutes les définitions et les présentations de ce concept sont tirées du document : *UN/CEFACT . - Core Components Technical Specifications, Part 1*//Spécification technique des composants élémentaires, Partie 1. – traduit par Richard Parent. - <http://www.autoroute.gouv.qc.ca/publica/normes/liste.htm>

1-5 UML

UML (Unified Modeling Language) traduit en français "**langage de modélisation objet unifié**" est une notation permettant de modéliser un problème de façon standard. Il est né de la fusion des trois méthodes qui ont le plus influencé la modélisation objet au milieu des années 90 : OMT, Booch et OOSE. Issu "du terrain" et fruit d'un travail d'experts reconnus, UML est devenu un standard incontournable et une référence en terme de modélisation objet. Il s'agit d'un langage qui permet de :

- représenter des concepts abstraits (graphiquement par exemple),
- limiter les ambiguïtés (parler un langage commun, au vocabulaire précis, indépendant des langages orientés objet),
- faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions),
- définir les vues qui permettent de décrire tous les aspects d'un système avec des concepts objets.

UML est devenu fin 1997 une norme **OMG**⁵ (Object Management Group).

2- CONTEXTE DU STAGE

Le stage que j'effectue entre dans le cadre de la formation que je suis cette année à l'école centrale Paris sous le titre du « Mastère spécialisé en Ingénierie des Systèmes Informatiques Ouverts ». Cette formation a pour objectif de promouvoir l'utilisation des Systèmes Ouverts dans le domaine des technologies de l'information. Par systèmes Informatiques ouverts, nous entendons :

- Les différentes variantes d'Unix ;
- Les différents types d'interfaces standardisées et publiées telles que POSIX (appels systèmes), TCP/IP (réseaux) ...etc ;
- Les différents langages de programmation normalisés tels que C, C++, JAVA, SQL (bases de données)... etc;

Mon stage représente donc la fin de cette formation qui est sous forme d'une mission en entreprise d'une durée de 5 mois et demi. Cette entreprise s'appelle EDIFRANCE. Dans ce qui suit, je vais présenter d'abord la formation (mastère MSIO) que j'ai suivie et après je présenterai l'entreprise.

2-1 PRÉSENTATION DU MASTÈRE SIO

Le mastère des Systèmes Informatiques Ouverts (MSIO) est une formation spécialisée de haut niveau, délivrée par l'**École Centrale Paris** et accréditée par la **Conférence des Grandes Ecoles**⁶. Cette formation est axée sur «les systèmes Internet - intranets basés sur les technologies ouvertes et standardisées: systèmes d'exploitation Unix (FreeBSD, Linux, HP-UX, Solaris...), réseaux TCP/IP, environnements distribués CORBA et Services WEB. Le programme comporte également des modules de formation sur les langages de programmation C/C++, Java et la méthode UML, les bases de données SQL, ainsi que des cours sur la normalisation et le droit informatique »⁷.

⁵ Voir le site : <http://www.omg.org>

⁶ Pour plus d'informations, veuillez voir le site : <http://www.cge.asso.fr>

⁷ Présentation du mastère tirée du site : <http://www.msio.ecp.fr/>

Ce mastère est une formation est de 12 mois à temps plein, y compris un stage en entreprise de 4 mois minimum. Elle a pour ambition d'apporter une réelle compétence dans le domaine des Systèmes Informatiques Ouverts (basés sur les normes et les standards largement publiés). En fin de formation, les étudiants diplômés sont en mesure d'appréhender tous les aspects de l'intégration de systèmes, depuis la conception, la réalisation, la documentation y compris les aspects réglementaires.

2-2 PRÉSENTATION DE L'ENTREPRISE (EDIFRANCE)

[EDIFRANCE](#) est une association (loi 1901) créée en 1990. Elle a pour vocation la promotion et le développement des Échanges Électroniques Professionnels (EEP) auprès des sociétés et administrations françaises.

2-2-1 MISSIONS D'EDIFRANCE

EDIFRANCE a pour missions principales de :

- faire la *veille technologique* et proposer des études de marché, des analyses d'offre et un panorama des usages du commerce électronique B2B⁸ ;
- mobiliser les acteurs des différentes communautés sectorielles autour de sujets communs ;
- mettre en place des séminaires de formation sur des points précis des Échanges Électroniques Professionnels (EEP) en fonction des besoins des professionnels et développer un catalogue de l'offre de solutions, de services et de prestations utilisables dans les EEP ...

Avec plus de 150 adhérents, EDIFRANCE rassemble les entreprises, administrations, communautés sectorielles, organismes professionnels, fournisseurs de solutions et prestataires de services intéressés ou impliqués dans la mise en place de systèmes d'échanges électroniques.

Par communautés sectorielles, on entend des groupes qui représentent des instances responsables du développement des Échanges Électroniques Professionnels en France, dans un secteur bien déterminé (exemple : la santé, la construction) ou pour une fonction (exemple : fonctions commerciales, douanes ou TVA).

2-2-2 GROUPES DE TRAVAIL CHEZ EDIFRANCE

Quant aux groupes de travail, on peut les classer en deux catégories :

LES GROUPES DE TRAVAIL AUTOUR DE XML

Ces groupes de travail ont pour mission principale de valider les traductions qui sont proposées par EDIFRANCE, de mettre en commun les efforts pour une compréhension globale des spécifications d'ebXML, d'échanger les expériences réalisées par les uns et les autres autour d'ebXML.

Ces groupes sont classifiés selon les thèmes suivants :

- Données élémentaires ;
- Nommages XML
- Processus d'affaires ;
- Architecture technique

⁸ B2B ou BtoB : Business to Business : qualifie un logiciel ou un site web de professionnel à professionnel. Par opposition à B2C ou BtoC : Business to Consumer qui qualifie un logiciel ou un site web destiné au grand public.

- CPP / CPA ;
- Messages et sécurité ;
- UMM / UML ;
- Méthode d'analyse des processus et données d'affaires ;
- frXML

LES AUTRES GROUPES DE TRAVAIL

- Commission internationale
- frEDIFACT
- XML/X400
- Aspects légaux et contractuels
- L'interopérabilité des certificats
- Club des offreurs
- SDC : Dictionnaire standard de description et de structuration des données des produits de construction...

Après cette introduction très générique qui a pour objectif de définir le contexte du stage, ainsi que les concepts clés du projet, nous allons présenter la suite de notre document comme suit :

- Première partie : conception du projet : une étude qui détermine les besoins en termes de concepts, de fonctionnalités et d'outils, ainsi que leurs spécifications techniques. Elle permet également de définir le contenu du projet, et ce en passant par une phase de modélisation qui nous permet d'extraire le modèle de données, ainsi que les interfaces utilisateurs nécessaires. Cette modélisation est faite par le biais du langage UML ;
- Deuxième partie : réalisation du projet. Cette partie essaie d'expliquer et de détailler, dans un premier lieu, toutes les étapes d'installation et de configuration des outils de développement. Elle présente dans un deuxième lieu les étapes et la façon de la création de la base de données, ainsi que le développement des interfaces et les liens entre elles. Enfin, je présente, à travers ces interfaces toutes les fonctionnalités offertes par cette application ;
- Une conclusion générale qui synthétise l'état d'avancement du projet et qui donne les éventualités de son évolution selon les besoins et l'avancement des travaux dans ce domaine ;
- Une partie des annexes, là où je vais présenter tous les diagrammes de modélisation, le plan qualité, un glossaire des termes utilisés dans le domaine de l'échange de données informatisé, etc.

1^{ÈRE} PARTIE : CONCEPTION DU PROJET

Avant de parler du projet, je tiens à préciser que je définis dans mon document un **modèle** (écrit en Italique et en gras) comme étant la définition d'une structure permettant de décrire un schéma XML ou une DTD, c'est à dire toutes les informations permettant de qualifier et de documenter un schéma XML ou une DTD ainsi que la représentation du schéma XML ou la DTD. Ce terme est donc à ne pas confondre avec « modèle » (écrit en normal) dans ce document qui correspond à l'utilisation courante de ce mot dans la langue française.

Cette partie définit les différents besoins dans le cadre du projet : Création d'un répertoire de schémas XML. Ces besoins sont exprimés en termes de fonctionnalités et de contraintes.

Elle représente à la fois le cahier des charges et les spécifications techniques du projet en question, et donc elle consiste à étudier la circulation de l'information dans la base et jusqu'au poste de travail, à déterminer les saisies à effectuer, les services rendus par la base de données, les besoins potentiels des utilisateurs, ainsi que déterminer les outils et les techniques à utiliser pour développer le présent projet. Ce travail permet de fixer les objectifs à atteindre et les moyens à employer.

1- CONTENU DU PROJET

L'objectif du projet est de mettre à disposition, sur Internet, l'ensemble des *modèles* XML francophones publics existants, mais aussi les *modèles* existants dans d'autres langues, en les rendant compréhensibles par les lecteurs francophones, et ceci par une description sommaire selon des critères bien définis et obligatoirement en français.

Le présent projet a pour objet donc de développer un site web qui contient :

- des formulaires d'accès et de mise à jour de la base de *modèles* XML ;
- une base de données qui contient les *modèles* de documents XML (schémas, DTD, autres) ;
- un outil permettant une recherche textuelle rapide dans la base ;
- une documentation servant comme un guide d'utilisation et de maintenance du site ;
- un espace de travail des groupes spécifiques sous forme d'une base de données.

En d'autres termes, il s'agit de créer un serveur d'application permettant le stockage des *modèles* XML. Ces *modèles* devront être accessibles à tous par le biais d'une Interface Web.

Pour cela, trois acteurs, au moins se distinguent : l'utilisateur, le participant au groupe du travail et l'administrateur. L'utilisateur devra avoir accès aux *modèles* par le biais d'une interface web à partir de n'importe quel navigateur. Un système de session personnel devra être mis en place. Il devra pouvoir faire une recherche suivant différents champs ou codes ou à partir de mot-clefs. Il pourra afficher sur écran et/ou télécharger n'importe quel document.

Cet utilisateur pourra être n'importe qui, mais il pourra être aussi un soumissionnaire d'un modèle dans la base.

Les données concernant ces modèles (domaines d'utilisation, méta-modèles, mots clés, schémas XML, DTD, ...) devront être stockées dans une base de données relationnelles.

Le participant au groupe du travail est une personne qui se dote d'un mot de passe pour accéder à un espace de travail bien particulier (et qui est sous forme de base de données).

L'administrateur devra pouvoir gérer cette base de données (ajouter, modifier ou supprimer un champ) mais il devra aussi pouvoir modifier les différents modules du serveur.

2- ÉNONCÉ FONCTIONNEL DU BESOIN

2-1 ÉNONCÉ DU BESOIN

Mettre en place une base de données de *modèles* XML accessible à travers un site web. Ce site devra offrir les fonctionnalités suivantes :

? Offrir aux différents utilisateurs, selon leurs catégories, un certain nombre de fonctionnalités qui leur permettent d'accéder à la base de données (répertoire de schémas), ainsi qu'à la documentation associée.

Cet accès aura a priori trois objectifs : déclarer (ou soumettre) un *modèle*, accéder à un *modèle*, modifier un *modèle* ;

? Offrir aux propriétaires du site, et/ou comité de maintenance, la possibilité de mettre à jour la base et éventuellement de développer de nouvelles fonctionnalités d'accès et de mise à jour.

2-2 CYCLE D'UTILISATION ET ENVIRONNEMENT

La solution retenue devra :

- permettre une maintenance et mise à jour rapide et facile du site ;
- être évolutive pour permettre d'intégrer de nouvelles fonctionnalités et de supporter une fréquentation plus importante ;
- utiliser des produits qui ont prouvé leur maturité et dont la question de la pérennité ne se pose pas ;
- s'intégrer à l'environnement existant ;
- s'intégrer dans un environnement graphique et prendra en compte certaines particularités dues à la nature des informations manipulées ; il conviendra par exemple d'éviter la saisie répétitive d'informations.
- permettre aux utilisateurs d'accéder facilement à la base selon des critères de recherches plus au moins précis pour que le résultat soit le plus pertinent possible. Cet accès leur permettra d'avoir le schéma XML approprié à leur domaine, ainsi que la documentation associée qui permet de leur expliquer les différents éléments du schéma.
- Elle doit permettre également au comité éditorial de mettre à jour cette base sans aucune difficulté.

2-3 ANALYSE FONCTIONNELLE DES BESOINS

En se référant à la Norme AFNOR NF X 50-151, notre démarche d'analyse fonctionnelle de notre projet consiste à «*rechercher, ordonner, caractériser, hiérarchiser et valoriser les fonctions*» du produit à mettre à la disposition des utilisateurs. En effet ce système n'a d'intérêt et de finalité qu'en fonction des services qu'il rendra.

Afin de déterminer ces fonctions, nous avons opté dans un premier lieu à une modélisation du projet. Cette modélisation consiste à établir :

- un diagramme de classes qui représente la structure statique et globale de l'application. Ce diagramme représente graphiquement les classes, les attributs, les opérations et les relations associées ;
- des diagrammes de cas d'utilisation qui montrent toutes les relations possibles entre les différents acteurs de l'application au sein du système. Ils comprennent les acteurs, l'ensemble des cas d'utilisation et bien sûr les associations entre les acteurs et les cas d'utilisation ;

- des diagrammes de séquences qui montrent les différentes interactions entre les objets de l'application avec les messages qu'ils échangent de façon ordonnancée dans le temps⁹.

À partir de cette modélisation, nous avons pu déceler les fonctions que nous allons analyser dans les tableaux qui suivent. Ces fonctions peuvent être classifiées comme suit :

- **A** : Fonctions liées à l'alimentation de la base de *modèles* XML (mise à jour) ;
- **B** : Fonctions liées à l'identification de l'utilisateur (notamment pour les participants des groupes de travail) ;
- **C** : Fonctions liées à la consultation du répertoire (schémas XML et documents associés) ;
- **D** : Fonctions liées à la navigation dans les *modèles* de schémas XML ;
- **E** : Fonctions liées à la récupération (téléchargement) des documents du répertoire.

À travers ces tableaux nous mettons en évidence les noms des fonctions à assurer par notre système et nous déterminons leurs niveaux :

- E** Essentiel (cette fonctionnalité se doit d'être disponible)
- I** Important (nous y tenons, mais si)
- S** Souhaitable (de préférence).

À la suite de chaque tableau, nous expliquons l'utilité de chaque fonction. Nous déterminons par la suite les contraintes liées à la définition de ces fonctionnalités. Mais d'abord, serait-il intéressant de définir quelques besoins généraux avec leurs degrés d'importance :

Besoins généraux	Niveau
Site disponible tout le temps et accessible librement (pas de notion de confidentialité)	E
Le site permet l'accès aux <i>modèles</i> dans plusieurs langues : le français étant la langue obligatoire pour le descriptif du <i>modèle</i> , les autres documents peuvent être dans d'autres langues.	I
La mise à jour du répertoire se fait dans deux étapes : - les utilisateurs enregistrent leurs modèles pour approbation ; - un comité éditorial approuve le modèle et met à jour le répertoire.	
Les langues recommandées pour les autres documents de la base sont le français et l'anglais	I
Aucun lien inactif de disponible	S

A- FONCTIONS LIÉES À L'ALIMENTATION DE LA BASE

N°	Fonction	Niveau
A1	Enregistrer un <i>modèle</i>	E
A2	Soumettre un <i>modèle</i> à un réviseur	E
A3	Modifier un <i>modèle</i>	E
A4	Supprimer un <i>modèle</i>	S

A1 : Enregistrer un *modèle* : Le système doit offrir à tous les utilisateurs la possibilité de déposer des propositions de *modèles* dans la base sans aucune difficulté. Il s'agit d'une fonction principale qui permet d'enrichir le répertoire de *modèles*.

⁹ Voir en annexes les différents diagrammes (diagramme de classes, diagrammes de cas d'utilisation et diagrammes de séquences) réalisés par le biais du logiciel « Objecteering »

A2 : Soumettre un *modèle* au réviseur : cette fonction doit être offerte par le système en même temps que la première fonction. Elle consiste à engager une procédure de revue sous forme de vérification et de contrôle préalables à l'ajout définitif du *modèle* dans la base et à le rendre public (accessible à tout le monde).

A3 et A4 : Modifier ou supprimer un *modèle* existant : ces fonctions doivent permettre aux seuls propriétaires de modifier ou de supprimer leurs *modèles* s si nécessaire.

B- FONCTIONS LIÉES À L'IDENTIFICATION DE L'UTILISATEUR

N°	Fonction	Niveau
B1	S'identifier pour soumettre, ou modifier un modèle	E
B2	S'identifier pour réviser un modèle	E

B1 : S'identifier pour soumettre ou ajouter un modèle : c'est une fonction essentielle pour la déclaration d'un modèle, et ce pour garder d'une part un nom unique du propriétaire, et d'autre part pour assurer que seul le propriétaire puisse modifier son ou ses modèles.

B2 : S'identifier : il s'agit d'une fonction que le système doit assurer et qui consiste à protéger certains travaux qui sont en cours (qui ne sont pas complets). Seuls les participants à ces travaux peuvent y accéder et doivent avoir un mot de passe. Cette fonction n'est obligatoire que pour ces groupes du travail et pour accéder à la zone citée ci-dessus. Pour le public, il n'y a pas besoin de s'identifier.

C- FONCTIONS LIÉES À LA CONSULTATION DU RÉPERTOIRE

N°	Fonction	Niveau
C1	Choisir le mode de recherche	I
C2	Saisir des mots clés	I
C3	Choisir un code de classification	I
C4	Choisir un contexte	I
C5	Choisir un <i>modèle</i>	E
C6	Choisir une langue	S

C1 : choisir le mode de recherche : notre système doit offrir aux utilisateurs (visiteurs) différentes manières d'effectuer leurs recherches, à savoir :

- une recherche contextuelle, avec laquelle l'utilisateur choisit parmi une liste d'éléments de contexte celui qui l'intéresse ;
- une recherche classificatoire qui permet de choisir un code de classification ;
- une recherche par mot clé qui permet d'introduire librement un mot ou une expression de recherche.

C2, C3, C4 ce sont des fonctions facultatives et dépendent de la première fonction.

C5 : choisir un *modèle* : cette fonction qui vient juste après l'introduction de la requête de l'utilisateur et la présentation des résultats permet de choisir à partir d'une liste de *modèles* celui qui répond au besoin.

C6 : Choisir une langue : cette fonction doit répondre aux besoins des utilisateurs qui préfèrent introduire leurs mots clés et/ou choisir les *modèles* recherchés dans une langue bien déterminée (a priori, ça sera l'anglais ou le français).

D- FONCTIONS LIÉES À LA NAVIGATION DANS UN *MODÈLE*

N°	Fonction	Niveau
D1	Accéder au résumé, à l'objectif et au principe du <i>modèle</i>	I
D2	Accéder aux autres méta-modèles	I
D3	Accéder au schéma XML	E

D1 : accéder au résumé, à l'objectif et au principe du *modèle* : cette fonction liée à la navigation dans un *modèle* permet d'afficher son résumé, son objectif, son principe et ses fonctionnalités.

D2 : Accéder aux méta-modèles : le système doit permettre aux utilisateurs d'aller voir si nécessaire les méta-modèles servant comme des moyens de contrôle et de validation des *modèles* choisis. Ces méta-modèles permettent de valider des schémas XML par exemple (par rapport à d'autres schémas).

D3 : accéder au schéma XML : il s'agit de la fonction essentielle de l'application, et c'est d'ailleurs le point central de l'application de pouvoir accéder au schéma XML appropriée.

E- FONCTIONS LIÉES À LA RÉCUPÉRATION DES DOCUMENTS

N°	Fonction	Niveau
E1	Imprimer	I
E2	Enregistrer	I
E3	Télécharger	I

E1 : Imprimer : après avoir navigué dans le *modèle* et trouvé ce qu'il cherche, l'utilisateur a besoin certainement de récupérer l'information pertinente recherchée. Un des moyens de récupération est l'impression. Il faut donc que le système donne la possibilité d'imprimer les documents.

E2 : enregistrer : une autre possibilité de récupérer une information du répertoire est de pouvoir l'enregistrer sur un support quelconque.

E3 : Télécharger : c'est aussi une autre fonction de récupération de documents qui consiste à télécharger le document (si par exemple, le schéma XML ou une feuille de style est attaché, l'utilisateur pourra le télécharger indépendamment du modèle).

2-4 CONTRAINTES

- Le site devra être accessible depuis un navigateur universel (Netscape, Internet Explorer, etc...) que ce soit depuis les plate-formes Windows ou Unix (et en particulier Linux) ;
- Le site doit être ouvert et évolutif : facile à mettre à jour et à y ajouter d'autres fonctionnalités répondant à de nouveaux besoins ;
- Les propriétaires des *modèles* sont munis des mots de passe pour accéder à leurs espaces de travail ;
- Aucun utilisateur et/ou propriétaire de modèles ne peut modifier un *modèle* qui ne lui appartient pas ;
- Les *modèles* déclarés doivent subir une opération de révision et de contrôle de la part d'un comité spécial pour assurer la présence de certaines informations liées à chaque modèle.
- Ni le logiciel serveur, ni le logiciel client ne devront être d'une technologie propriétaire.

3- MODÉLISATION DU PROJET

Pour bien comprendre les spécificités du projet, et déterminer ainsi les spécifications techniques, nous avons opté pour une modélisation par le biais du langage UML¹⁰, en utilisant l'outil «Objecteering» pour définir le modèle de données de la base de données, à travers les diagrammes de classes, ainsi que tous les diagrammes de cas d'utilisations possibles et les diagrammes de séquences. Cette modélisation va nous servir pour déterminer les modèles de données, et créer ainsi la base de données, mais aussi pour déterminer toutes les interfaces d'accès à cette base.

Les modèles de données de la base se proposent de définir de manière précise et exhaustive l'ensemble des données qui devront être stockées au niveau de la base de données et les relations qui les lient.

À partir du modèle Entités-Associations établi au départ, et vu la nature de notre projet qui a pour but d'une part de publier les modèles XML et de les mettre en accès public, et de l'autre de permettre à des groupes de travail de travailler sur des modèles qui ne sont pas encore complets et prêts à être publiés, nous avons commencé par définir d'abord les données du *modèle* XML avant de concevoir le modèle de la base.

Pour définir les données du *modèle* XML, nous avons utilisé l'outil XMLSPY¹¹ qui est un éditeur de documents XML et un designer de feuilles de styles. Cet outil nous a permis de définir toutes les entités, les attributs, les relations entre eux et les différentes caractéristiques de ceux-ci.

3-1 MODÈLES DES DONNÉES DES *MODÈLES* XML

Ces modèles des *modèles* ont pour objectif de présenter toutes les données qui existent dans un *modèle* XML stocké dans le répertoire. Vu aussi la complexité du sujet, nous avons décidé d'établir trois modèles différents :

- Un modèle simple valable pour tout utilisateur qui n'est pas forcément un spécialiste XML ;
- un modèle plus complet et plus sophistiqué pour les spécialistes XML ;
- un troisième modèle pour les données élémentaires (Core Components)¹².

¹⁰ Pour bien comprendre le langage UML, veuillez consulter le site : <http://uml.free.fr/>. là, où il ya le contexte et l'historique de l'apparition de ce langage, des cours, la norme, une bibliographie, des forums et d'autres liens intéressants.

¹¹ Pour plus d'information, veuillez consulter le site : <http://www.xmlspy.com/>

¹² Dans la partie réalisation de notre projet, vu la contrainte du temps et la complexité du modèle sophistiqué qui nécessite une infrastructure et des moyens importants, nous nous sommes intéressés, que par le modèle simple et le modèle des données élémentaires.

3-1-1 MODÈLE SIMPLE

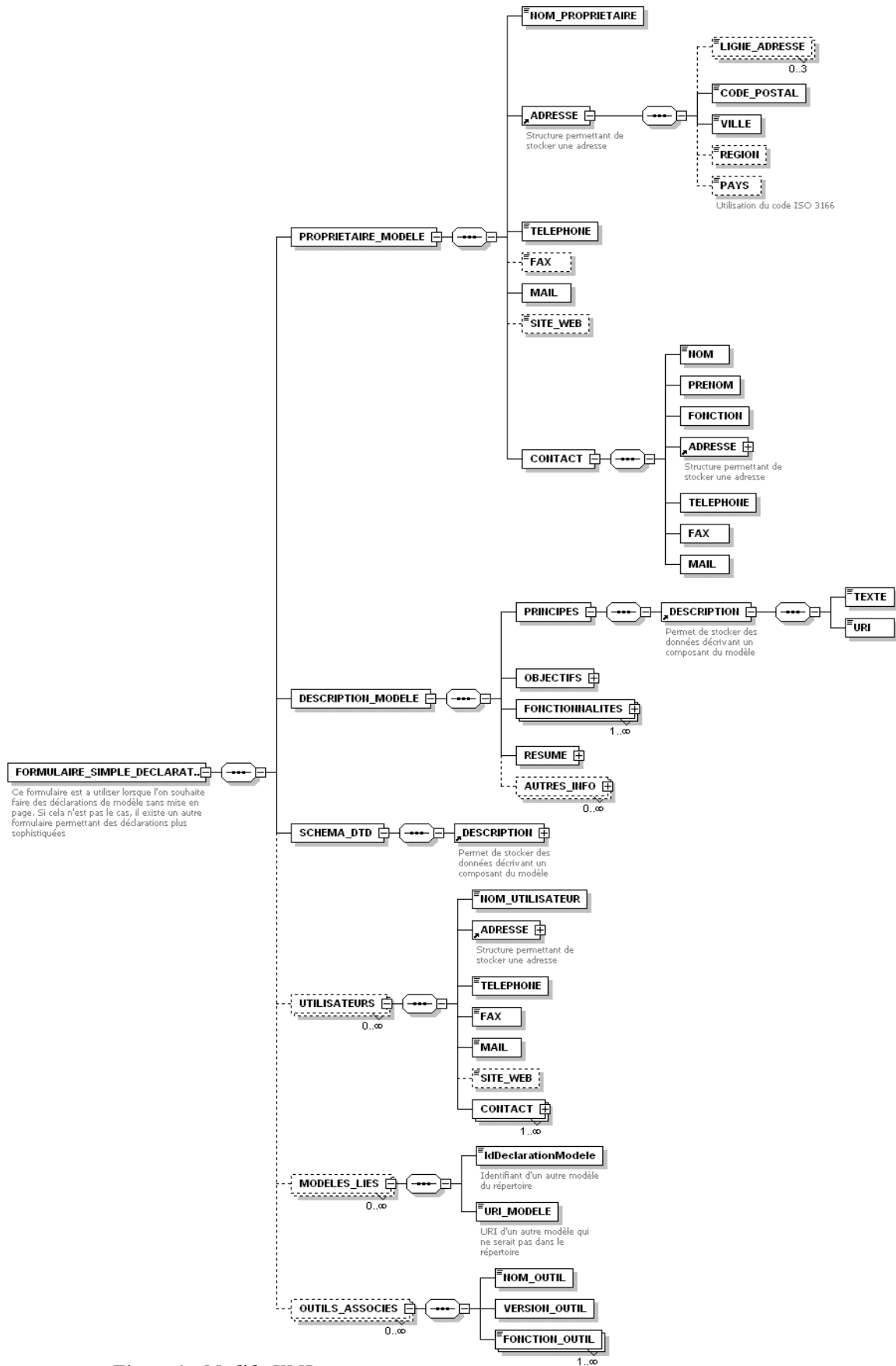


Figure 1 : Modèle XML

3-1-2 MODÈLE SOPHISTIQUE

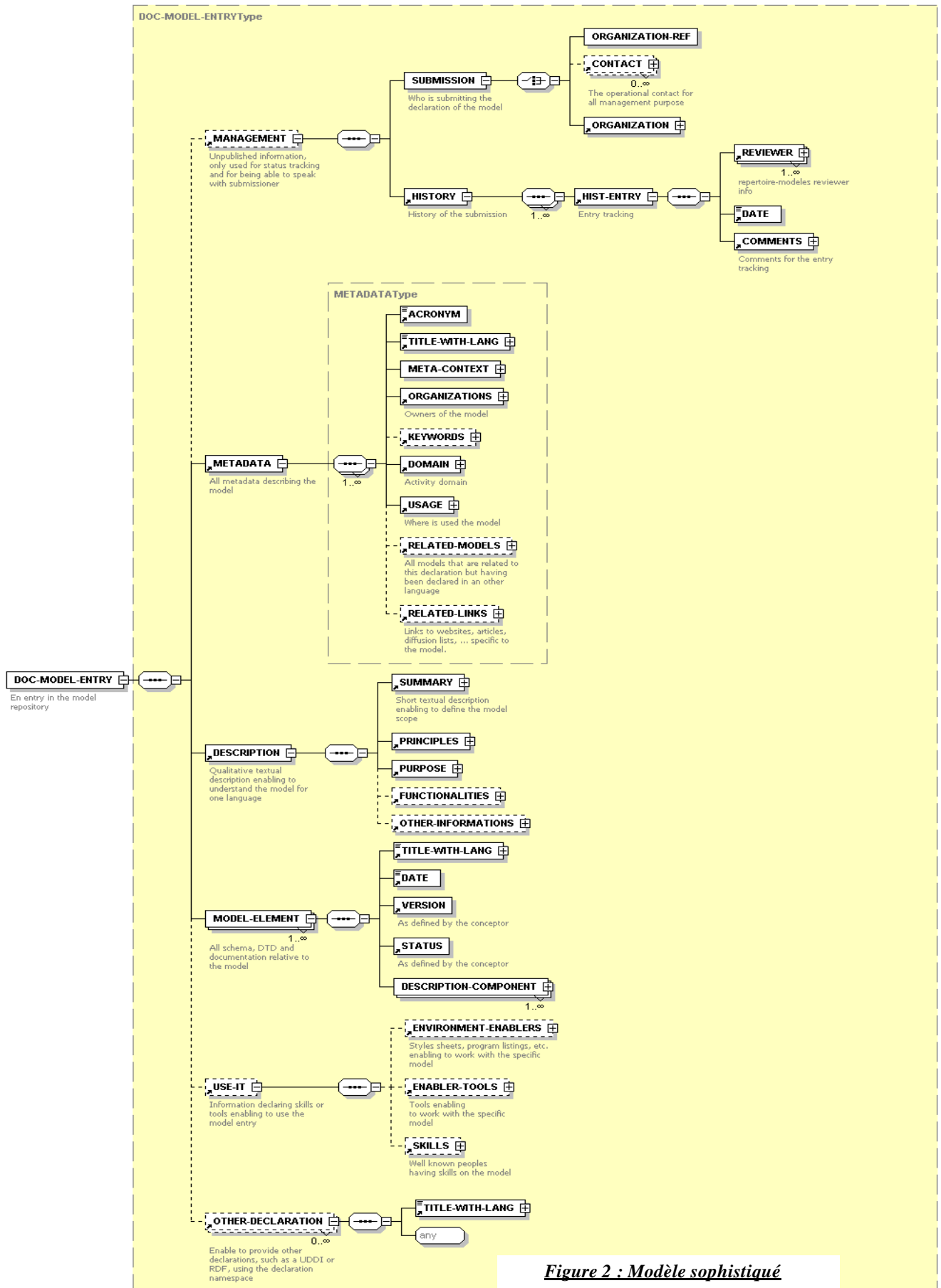


Figure 2 : Modèle sophistiqué

3-1-3 MODÈLE POUR LES DONNÉES ÉLÉMENTAIRES (CORE COMPONENTS)

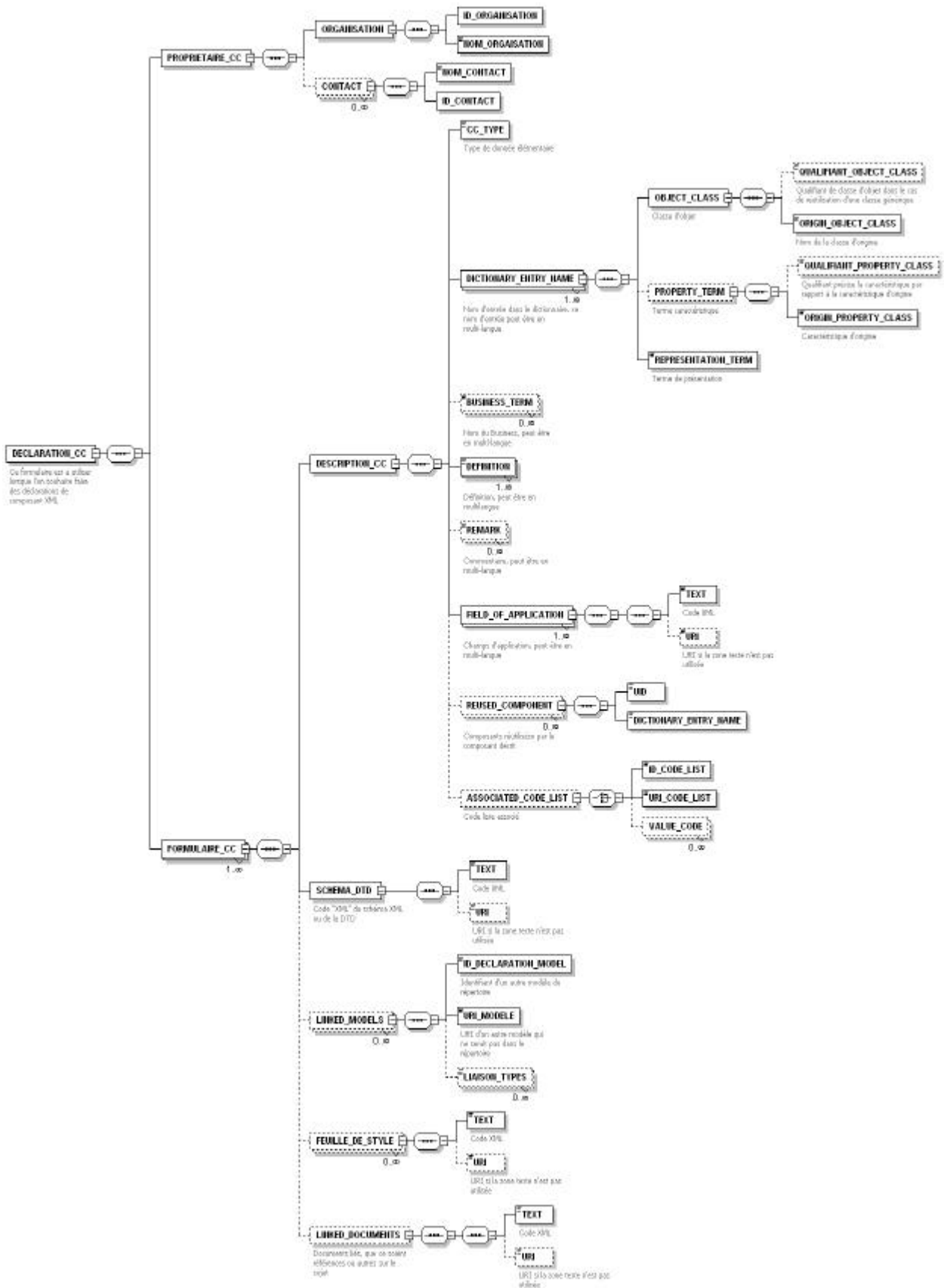
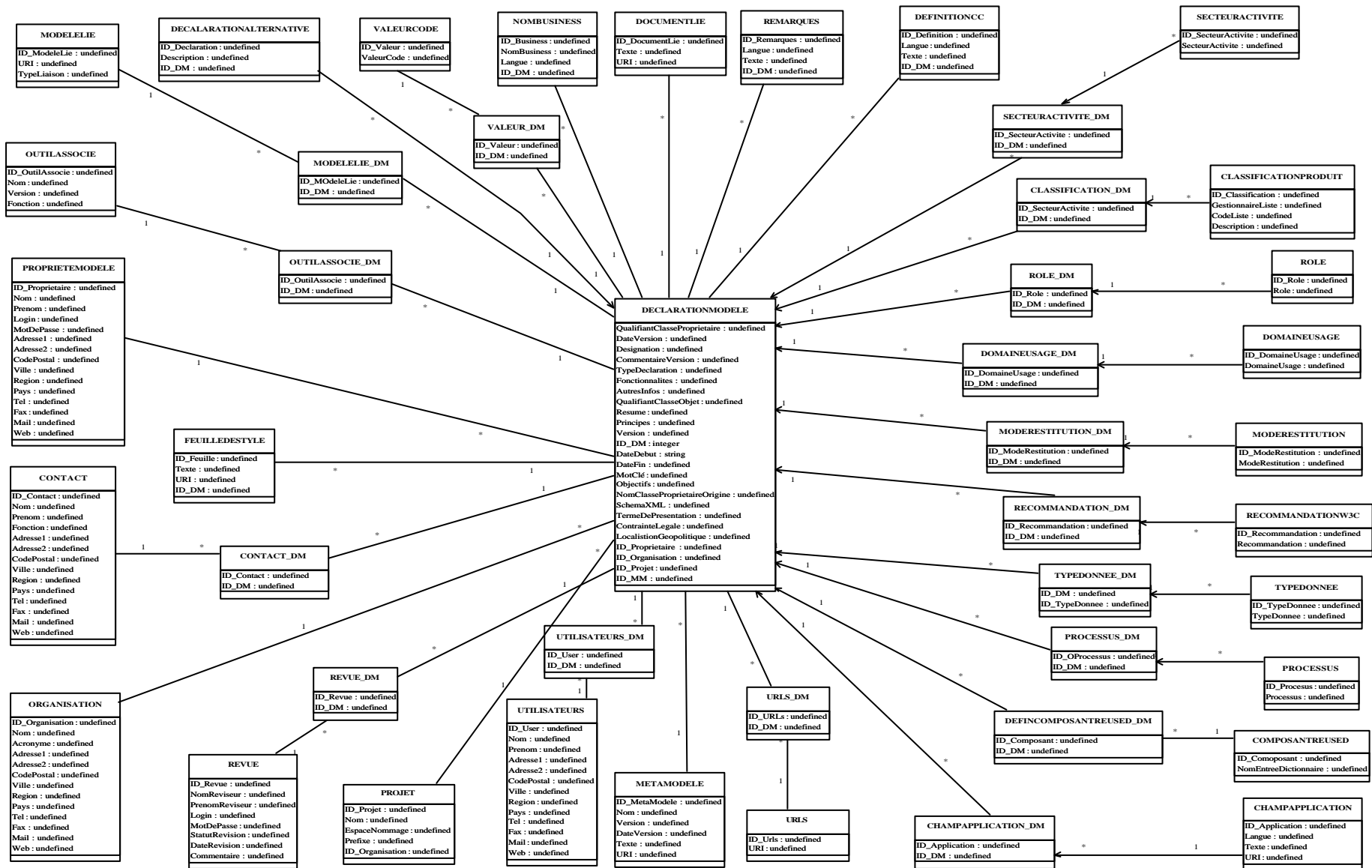


Figure 3 : Modèle core component

3-2 MODÈLE DE DONNÉES DE LA BASE



3-2-1 DESCRIPTION DU MODÈLE DES DONNÉES

Notre modèle de données contient les classes suivantes :

DECLARATIONMODELE	CLASSIFICATIONPRODUIT
DECLARATIONALTERNATIVE	URLS
METAMODELE	PROPRIETEMODELE
DOMAINEUSAGE	OUTILASSOCIE
RECOMMANDATIONW3C	MODELELIE
MODERESTITUTION	UTILISATEURS
TYPEDONNEE	COMPOSANTREUTILISE
REVUE	FEUILLEDESTYLE
CONTACT	CHAMPAPPLICATION
ORGANISATION	REMARQUES
PROJET	DEFINITIONCC
PROCESSUS	NOMBUSINESS
ROLE	DOCUMENTLIE
SECTEURACTIVITE	VALEURCODE

Ces classes peuvent être classées en deux catégories à savoir les classes communes aux deux modèles (modèle simple et modèle du core component) et classes spécifiques au modèle du core component.

3-2-1-1 CLASSES COMMUNES AUX DEUX MODÈLES

Classe "DECLARATIONMODELE" :

Cette classe est le point central de l'application. À partir d'elle, l'internaute peut avoir accès aux différentes documentations du modèle et au modèle lui-même. Les modèles peuvent être liés entre-eux.

Elle contient les attributs suivants¹³ :

- Identifiant (ID_DM) ;
- Nom du modèle (DésignationModèle) ;
- Version du modèle (Version) ;
- Date de la version (DateVersion) ;
- Commentaires sur l'évaluation (CommentaireVersion) ;
- Date de début du modèle (DateDébut) ;
- Date de fin du modèle (DateFin) ;
- Objectifs du modèles (ObjectifModele_Texte, ObjectifModele_URI) ;
- Résumé du modèle (ResumeModele_texte, ResumeModele_URI) ;
- Fonctionnalités du modèles (FonctionnalitesModele_texte, FonctionnalitésModèle_URI) ;
- Autres informations concernant le modèle (AutreInfoModele_texte, AutreInfoModele_URI) ;
- Mots clés du modèle (MotClé) ;
- Type du core component (TypeCC) ;
- Qualifiant de la classe d'objet (QualifiantClasseObject) ;
- Qualifiant de la classe propriétaire (QualifiantClassePropriétaire) ;
- Nom de la classe propriétaire d'origine (NomClassePropriétaireOrigine) ;
- Termes de présentation (TermeDePresentation) ;
- Schema XML ou DTD (SchemaXML, URISchemaXML) : permet d'enregistrer le schéma XML ou la DTD lié au modèle sous forme de texte ou sous forme de lien URI, ou même sous forme de fichier attaché qu'on garde son URI dans le deuxième champ) ;

¹³ Les noms qui sont entre parenthèses sont les noms tels qu'ils existent dans la base.

- Contraintes légales (ContraintesLegales) permet de définir les caractéristiques des situations d'affaires influencées par des exigences légales ou réglementaires (lois, règlements, conventions, traités...). Cette classe se caractérise par un identifiant et une désignation dont la définition est libre ;
- Localisation géopolitique (Geopolitique) : permet de caractériser la région, la nationalité et tout autre facteur culturel lié à la localisation géographique ou se déroule le processus d'affaires. la classification de ces localisations permettent de donner une valeur à chaque message d'affaires ou à chaque composant.;
- Identifiant du propriétaire du modèle (ID_Proprietaire) ;
- Identifiant du méta-modèle (ID_MetaModele) ;
- Identifiant du projet lié à ce modèle (IdProjet) ;

Classe « PROPRIETEMODELE »

Cette classe permet de stocker toutes les données relatives au propriétaire du modèle, et qui lui permettent d'accéder et de modifier ses modèles à savoir :

- Identifiant du propriétaire (ID_Proprietaire)
- Nom du propriétaire (NomProprietaire)
- Prénom (PrenomProprietaire)
- Login (Login)
- Mot de passe (MotDePasse)
- Adresse (Adresse1, Adresse2)
- Ville (Ville)
- Code postal (CodePostal)
- Région (Region)
- Pays (Pays)
- Téléphone (Tel)
- Fax (Fax)
- Mail (Mail).

Classe « FEUILLEDESTYLE »

Cette classe permet de stocker la ou les feuille(s) de style correspondante(s), sachant que le propriétaire peut envoyer la feuille de style en fichier attaché. Elle se compose des attributs suivants :

- Identifiant de la feuille de style (ID_Feuille)
- Texte de la feuille de style (FeuilleDeStyle_Texte)
- URI de la feuille de style (FeuilleDeStyle_URI)

Classe "DOMAINEUSAGE"

Permet de classer le modèle par domaine d'utilisation. Le domaine reste au niveau le plus générique (ex : Santé, Finances, Economie, Banque, Education, Presse ...).Elle contient les attributs :

- Identifiant du domaine d'usage (ID_Domaine) ;
- Description du domaine (DomaineUsage).

Classe "PROCESSUS"

La catégorie Processus d'Affaires donne un moyen d'identifier sans ambiguïté l'activité d'affaire en question. Elle s'appuie sur une classification enregistrée dans le Catalogue des Processus d'Affaires Communs de l'UN/CEFACT. Elle contient les attributs :

- Identifiant du processus d'affaires (ID_Processus) ;
- Description du processus d'affaires (Processus) .

Classe "SECTEURACTIVITE"

Permet de définir très spécifiquement le secteur ou le sous secteur d'activité dont dépend le processus d'affaires. Cette catégorie prend ses valeurs d'un ensemble des valeurs recommandées et qui sont :

- ISIC : International Standard Industrial Classification
- UNSPSC : Universal Standard Product and service Specification.

Elle contient les attributs :

- Identifiant du secteur d'activité (ID_SecteurActivite) ;
- Description du secteur (Secteur).

Classe "CLASSIFICATIONPRODUIT"

Permet de décrire les caractéristiques de la situation d'affaires relatives aux produits ou services échangés ou référencés dans le processus d'affaires. Elle contient les attributs :

- Identifiant de la classification produit (ID_Classification) ;
- La liste gestionnaire (GestionnaireListe) ;
- Code dans la liste (CodeDansLaListe) ;
- Description de la classification (DésignationClassification).

Classe "ROLE"

Permet de définir le rôle dans un processus d'affaires. cette classe prend ses valeurs dans l'ensemble des valeurs des rôles fournies par le catalogue de Processus d'Affaires Communs. Elle contient les attributs :

- Identifiant du rôle (ID_Role) ;
- Description du rôle (Role) : c'est un champ énuméré qui peut être (client, fournisseur, transporteur, distributeur, autre).

Classe "METAMODELE"

Permet de contrôler un schéma ebXML par rapport au méta modèle ebXML.

Utilisable pour les données élémentaires, les processus d'affaires, les protocoles de collaboration. Elle contient les attributs suivants:

- Identifiant du méta-modèle (ID_MétaModèle) ;
- Nom du méta-modèle (Titre) ;
- Version du méta-modèle (Version) ;
- Date de version (DateVersion) ;
- URI du méta-modèle (LienURI) ;
- Texte du méta-modèle (Texte).

Classe "ORGANISATION"

L'organisation est responsable d'un modèle. C'est à dire qu'un modèle est toujours rattaché à une organisation. Elle contient les attributs :

- Identifiant de l'organisation (ID_Organisation) ;
- Nom de l'organisation (NomOrganisation) ;
- Acronyme de l'organisation (Acronyme) ;
- Adresse mail (Mail) ;
- Site Web (web) ;
- Téléphone (Tel) ;
- Fax (Fax) ;
- Adresse ligne 1 (Adresse1)
- Adresse ligne 2 (Adresse2) ;
- Ville de l'organisation (Ville) ;

- Code postal (CodePostal)
- Région (Region)
- Pays (Pays).

Classe "CONTACT"

Un contact est une personne physique qui est liée généralement à une organisation. Selon son niveau de sécurité, un contact peut avoir ou pas accès aux modèles Elle contient les attributs suivants :

- Identifiant du contact (ID_Contact) ;
- Nom (NomContact)
- Prénom (PrenomContact)
- Mail (MailContact)
- Téléphone (TelContact)
- Fax (FaxContact)
- Adresse ligne 1 (Adresse1Contact)
- Adresse ligne 2 (Adresse2Contact)
- Code postal (CodePostalContact)
- Ville (VilleContact)
- Région (RegionContact)
- Pays (PaysContact)

Classe "DECLARATIONALTERNATIVE"

Cette catégorie permet de fournir des déclarations alternatives, dans d'autres méthodes de description.. Elle contient les attributs :

- Identifiant de la déclaration (ID_DA)
- Type de déclaration (TypeDéclaration)
- Description de la déclaration (Description)

Classe "MODERESTITUTION"

Cette entité permet d'informer l'utilisateur sur les modes possibles de restitution du document en question. Les valeurs possibles de ces restitution sont : Édition papier, Wap, HTML, TELETEL, SGML, XML

Elle contient les attributs :

- Identifiant du mode de restitution (ID_ModeRestitution)
- Désignation du mode de restitution (ModeRestitution).

Classe "RECOMMANDATIONW3C"

Cette classe a pour objectif de catégoriser les données selon les recommandations utilisées lors de leur création. Ces recommandations peuvent être : DTD, Schema XML, Xpath, Xlink, XSLT, XSLFO...

Elle contient les attributs suivants :

- Identifiant de la recommandation W3C (ID_Recommandation)
- Désignation de la recommandation (Recommandation).

Classe "TYPEDONNEE"

Cette classe permet de classer les données stockées dans le répertoire selon leurs types :

Donnée texte structurée

Donnée texte non structurée

Donnée multimédia structurée

Donnée multimédia non structurée

Donnée mixte (texte et multimédia) structurée
Donnée mixte (texte et multimédia) non structurée.

Elle contient les attributs :

- Identifiant du type de données (ID_TypeDonnée)
- Désignation du type de données (TypeDonnée).

Classe "REVUE"

Cette classe est utilisée par les gestionnaires du répertoire. Elle permet de suivre l'introduction de nouveaux modèles dans le répertoire.

Les gestionnaires du répertoire vérifient un certain niveau de qualité des soumissions. Elle contient les attributs

- Identifiant de la revue (ID_Revue)
- Réviseur
- Statut Révision
- Date de Révision
- Commentaire
- Identifiant du modèle

Classe "PROJET"

Permet de définir la désignation du projet, l'espace de nommage (name space), ainsi que le préfixe utilisés. Elle contient les attributs :

- Identifiant du projet (ID_Projet)
- Nom du projet (NomProjet)
- Espaces de noms (EspaceNommage)
- Préfixe utilisé dans le projet (Préfixe)
- Identifiant de l'organisation (ID_Organisation).

Classe « OUTILASSOCIE »

Cette classe permet de définir tous les outils qui sont utilisés et associés au modèle XML tels que par exemple l'éditeur du document XML, ou le designer de la feuille de style ... (ex : XMLSPY). Elle contient les attributs suivants :

- Identifiant de l'outil associé (ID_OutilAssocie)
- Nom de l'outil (NomOutil)
- Version de l'outil (Version)
- Fonction de l'outil (Fonction)

Classe « UTILISATEURS »

Cette classe définit tous les utilisateurs d'un modèle avec toutes les informations qui permettent de les identifier :

- | | |
|--|-----------------------------------|
| - Identifiant de l'utilisateur (ID_USER) | - Pays (PaysUser) |
| - Nom de l'utilisateur (NomUser) | - Téléphone (TelUser) |
| - Prénom (PrenomUser) | - Fax (FaxUser) |
| - Adresse (Adresse1User, Adresse2User) | - Adresse électronique (MailUser) |
| - Ville de l'utilisateur (VilleUser) | - Site Web (WebUser) |
| - Région (RegionUser) | - Société (SocieteUser) |

Classe « DOCUMENTLIE »

Ceci permet de préciser tous les documents liés à un modèle, et par conséquent aider à bien comprendre ce dernier. Elle contient les attributs suivants :

- Identifiant du document (ID_DocumentLie)

- Texte du document (DocumentLie_Texte)
- URI du document (DocumentLie_URI)

Classe « MODELELIE »

Cette classe permet de savoir tous les modèles liés à un modèle donné, tout en indiquant l'identifiant du modèle lié s'il existe déjà dans la base, ou en indiquant l'URI s'il est ailleurs et le type de liaison. Les attributs de cette classe sont donc :

- Identifiant du modèle lié (ID_ML)
- URI du modèle lié (URI)
- Type de liaison (TypeLiaison)

Classe "URLs"

Cette catégorie permet de rassembler toutes les adresses URL utilisés dans le modèle., et faciliter, par conséquent , la recherche de tous les documents XML utilisés dans un tel schéma... Elle contient les attributs :

- Identifiant de l'URL (ID_URL)
- Chemin absolu de l'URL (CheminAbsolu)
- Chemin relatif (CheminRelatif)
- Nom du fichier (NomFichier)

3-2-1-2 CLASSES SPÉCIFIQUES AU MODÈLE CORE COMPONENT

Classe "VALEURCODE"

Cette classe permet de définir les valeurs possibles du code dans la liste gestionnaire du composant. Elle contient les attributs suivants :

- Identifiant de la valeur code (ID_ValeurCode)
- Description de la valeur (Valeur).

Classe « DEFINITIONCC »

Elle permet de donner une définition du composant dans différentes langues, à savoir le français, l'anglais, l'italien, l'espagnol... Elle contient les champs suivants :

- Identifiant de la Définition (ID_Definition)
- Texte de la définition (Definition)
- Langue de la définition (Langue)

Classe « NOMBUSINESS »

Cette classe permet de préciser le nom du core component (nom d'entrée dans le dictionnaire) dans une autre langue. Elle contient donc :

- Nom (NomBusiness)
- Langue du nom (Langue)

Classe « REMARQUES »

Ceci permet aux propriétaires de core components de mettre un commentaire quant à leurs modèles. Ce commentaire peut être dans plusieurs langues. Les champs de cette table sont :

- Identifiant de la remarque (ID_Remarques)
- Texte de la remarque (Remarques)
- Langue (Langue)

Classe « COMPOSANTREUTILISE »

Cette classe permet de préciser les composants se trouvant dans la base et qui sont réutilisés par un autre core component. On trouve donc :

- Identifiant du composant réutilisé (ID_Composant)
- Identifiant du modèle (ID_DM)

Classe « CHAMPAPPLICATION »

Cette classe définit le champ d'application du core component, elle se compose des trois champs suivants :

- Identifiant du champ d'application (ID_Application)
- Texte du champ d'application (ChampApplication)
- Langue du texte (Langue).

3-3 INTERFACE D'ACCÈS

L'interface utilisateur sera basique ; c'est à dire qu'elle implémentera toutes les fonctionnalités demandées sans contraintes ergonomiques et esthétiques (cette interface devant être retravaillée ultérieurement).

Cette interface doit contenir trois modèles d'accès selon les types d'utilisateurs :

1. modèle d'accès au public ;
2. modèle d'accès aux participants des groupes de travail ;
3. modèle d'accès au soumissionnaire du *modèle*.

3-3-1 MODÈLE D'ACCÈS AU PUBLIC

Ce modèle contient plusieurs modes de recherches à savoir :

- Recherche par mot-clé
- Recherche par secteur d'activité ;
- Recherche par domaine d'usage ;
- Recherche par rôle d'affaires ;
- Recherche par nom du modèle ;
- Etc.

Cet accès est libre et aucun mot de passe est nécessaire.

On peut donc schématiser ce modèle dans la figure suivante :

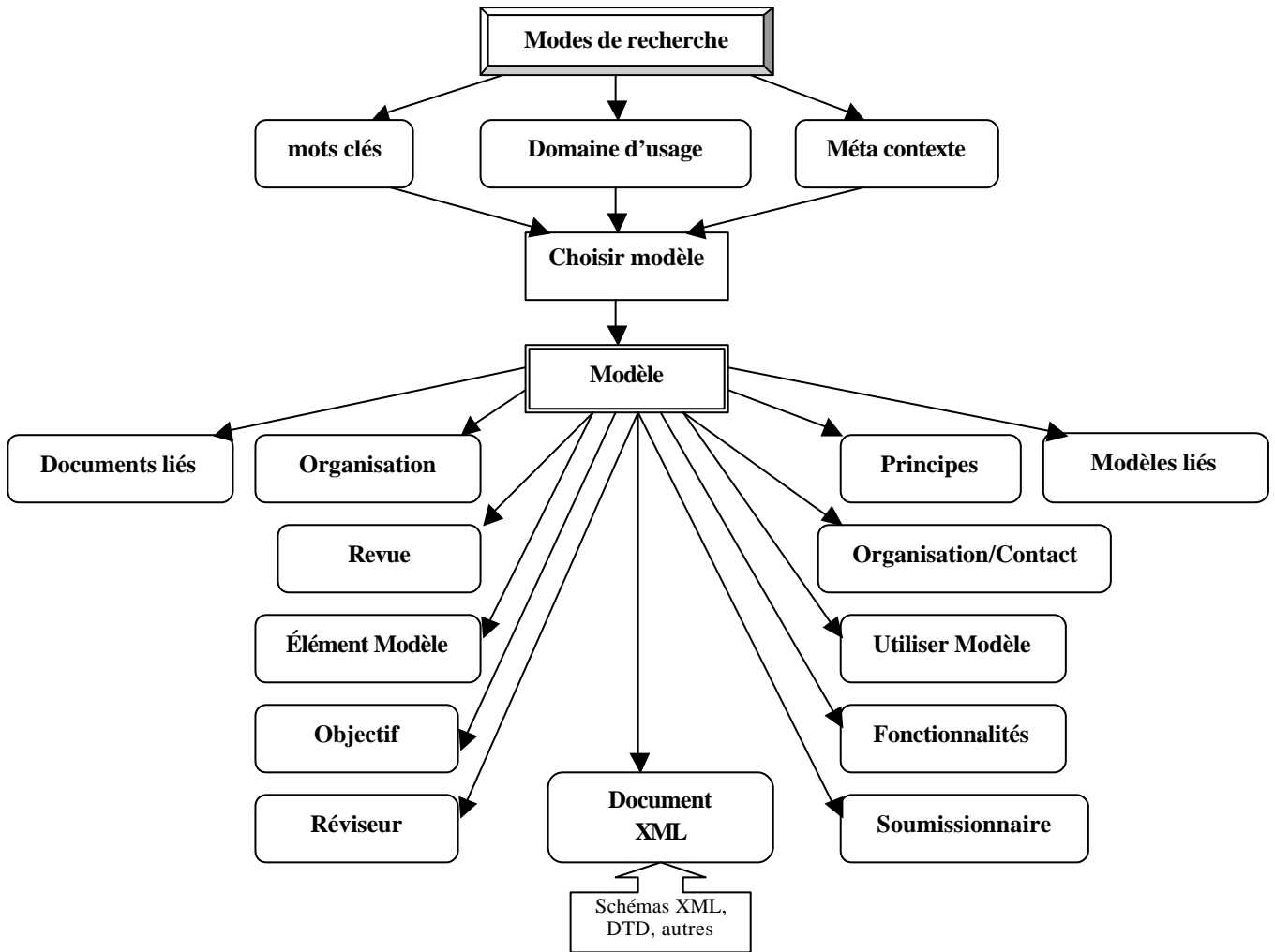


Figure 5 : Modèle d'accès au public

3-3-2 MODÈLE D'ACCÈS AU PARTICIPANT D'UN GROUPE DE TRAVAIL

Ce modèle est très simple, mais l'accès est sécurisé, et donc il exige un mot de passe.

Le participant tape son mot de passe et accède directement à la liste des *modèles* qui sont associés à son groupe de travail.

Un participant peut avoir plusieurs groupes de travail, et donc, il a besoin de plusieurs mots de passe (autant de mots de passe que de groupes de travail).

On peut schématiser ce modèle comme suit :

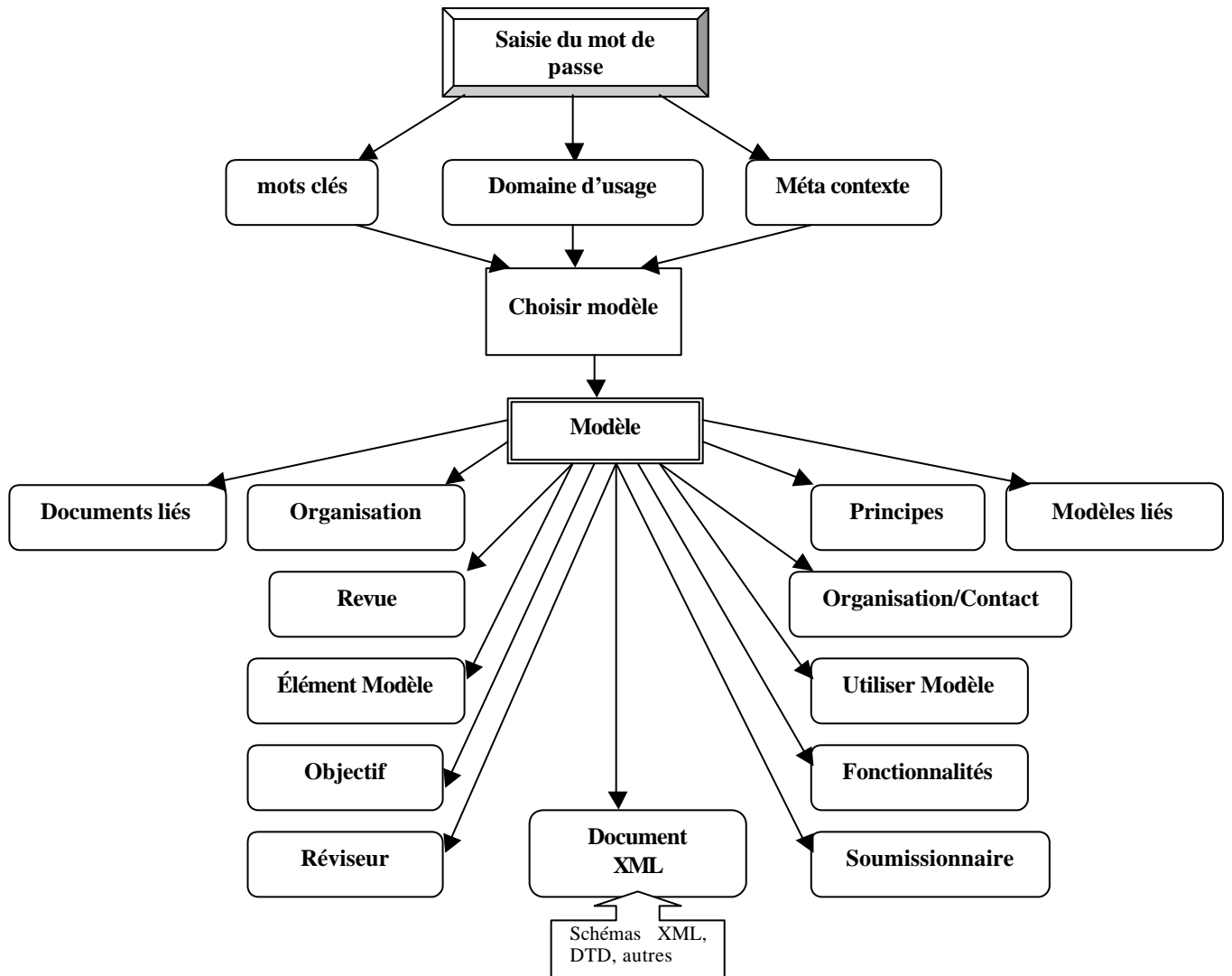


Figure 6 : Modèle d'accès au participant au groupe de travail

3-3-3 MODÈLE D'ACCÈS AU SOUMISSIONNAIRE DU MODÈLE

Ce modèle permet à celui qui veut soumettre un *modèle* pour la mise en accès, après revue, en public.

Le soumissionnaire a besoin d'accéder à un squelette du *modèle* vide pour le remplir par les données de son document. Il est donc obligé de passer par un mot de passe pour renseigner le domaine et l'usage du document à soumettre, et il reçoit ensuite le squelette entier du *modèle*.

Ce modèle d'accès peut être donc schématisé de la façon suivante :

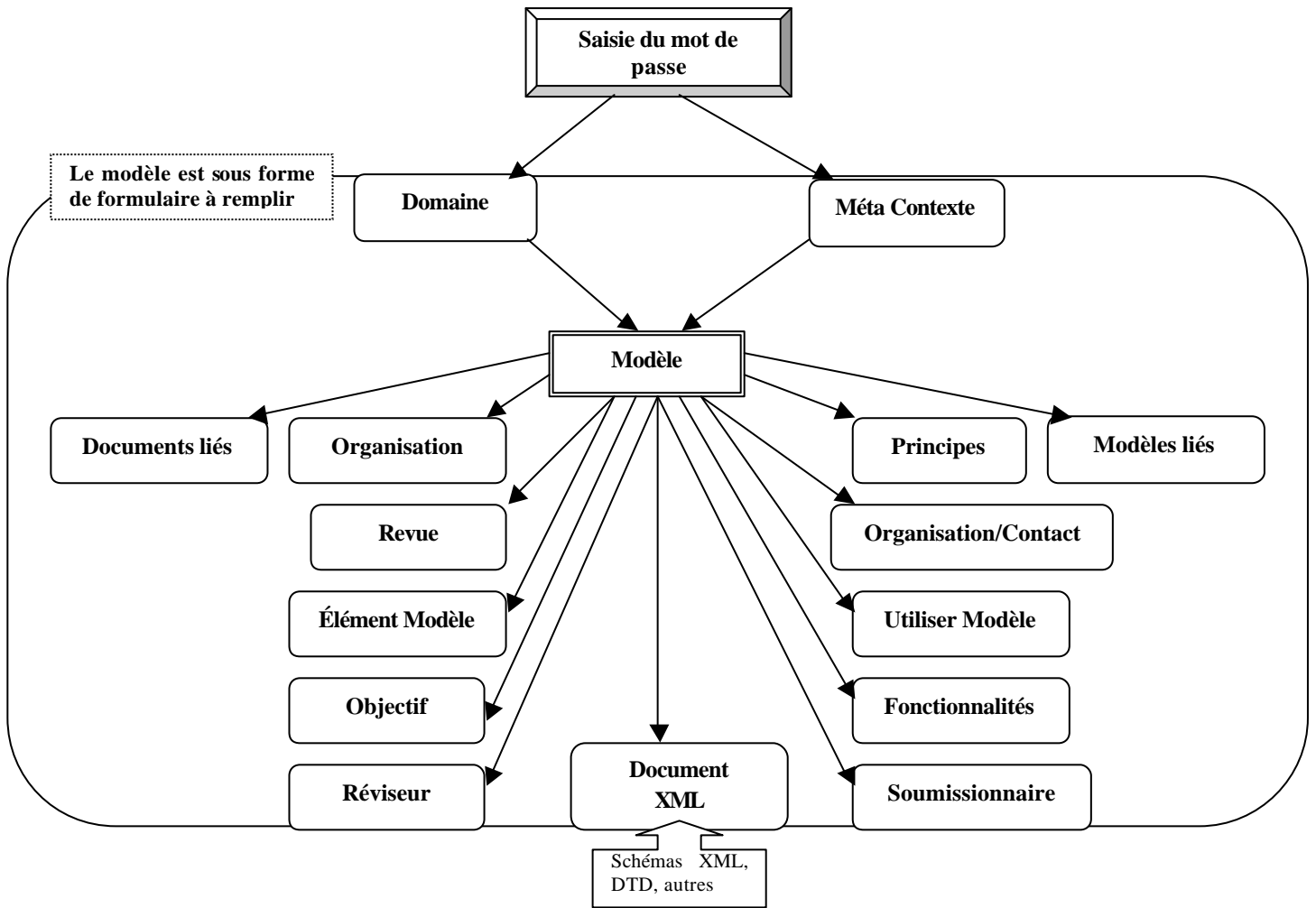


Figure 7 : Modèle d'accès au soumissionnaire d'un modèle

3-4 ARCHITECTURE LOGIQUE

Le choix s'est porté sur une architecture trois tiers (3/3) comportant d'un coté le serveur Web traitant les requêtes et où est stocké le site web, d'un serveur de base de données, et de postes de travail munis d'un browser permettant la visite du site et donc la formulation des requêtes.

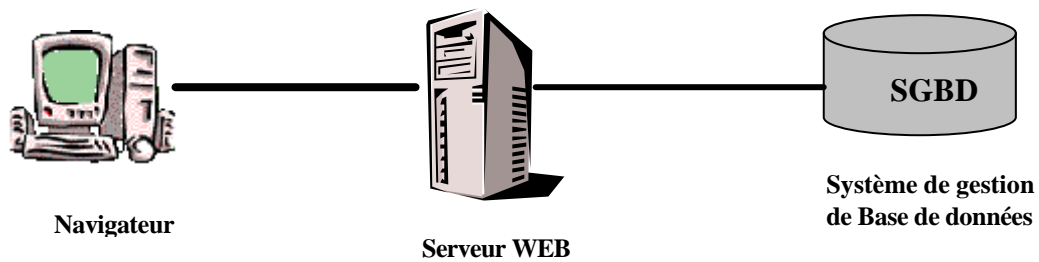


Figure 8 : Architecture logique

3-5 ARCHITECTURE PHYSIQUE DE BASE

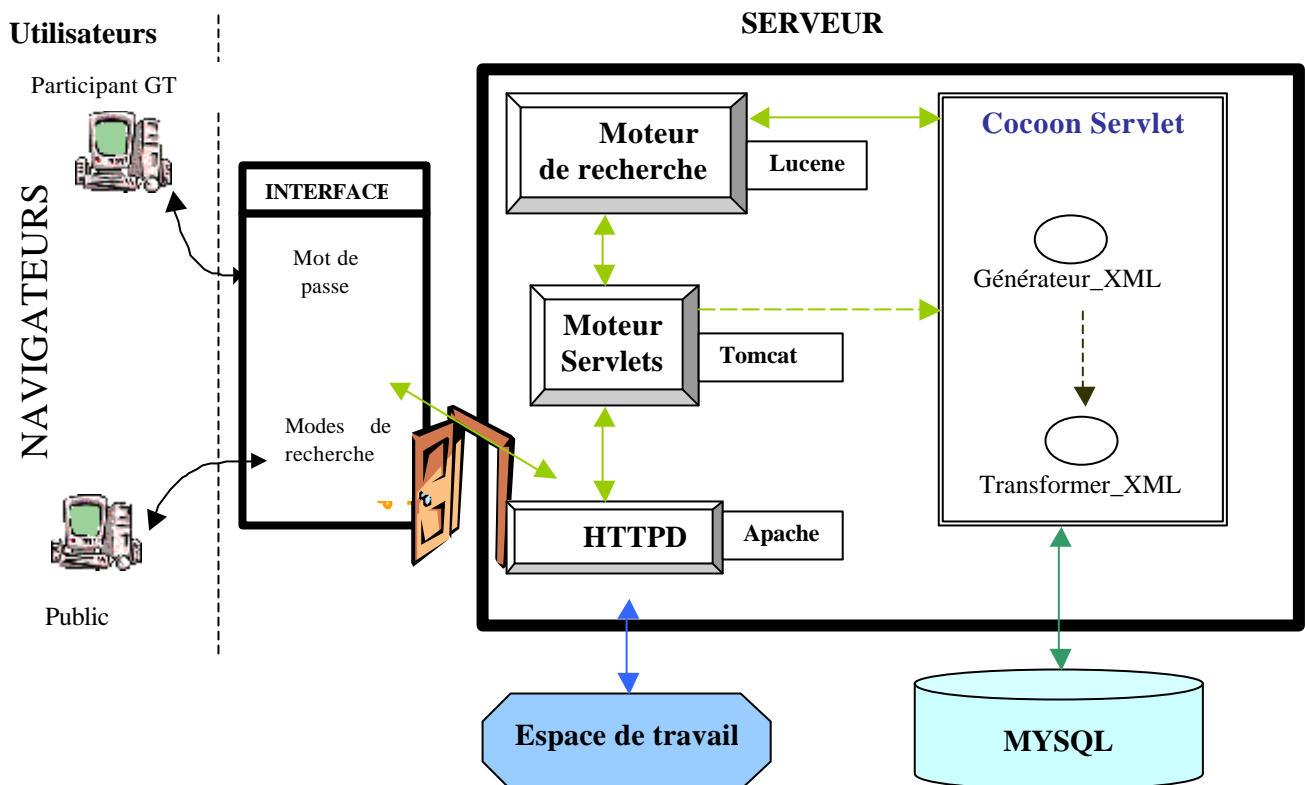


Figure 9 : Architecture physique de base

Comme nous pouvons le constater à travers ce schéma, nous avons trois grands composants du système :

- des navigateurs pour émettre des requêtes ;

- des serveurs pour traiter les requêtes ;
- un SGBD qui gère le répertoire de modèles XML ;

Par ailleurs, il nous paraît quand même important d'expliquer, avec un peu plus de détails, le fonctionnement du composant «serveurs » par rapport aux autres. En effet, du côté serveur, le mécanisme du traitement est un peu compliqué et ce du fait que nous avons à gérer des documents qui peuvent être de tous types, surtout des documents XML qui exigent un traitement spécifique pour leur transformation en documents affichables sur écran.

Pour répondre à cette spécificité du projet, nous avons donc choisi, une «*architecture Apache/Tomcat/Cocoon* » qui représente, à notre avis, une solution robuste et ouverte.

En d'autres termes, nous avons choisi deux serveurs web qui tournent en parallèle : Apache et Tomcat. Chaque serveur sera en écoute sur un port bien particulier (Exemple :Apache sur le port 80 et Tomcat sur le port 8080).

Apache est le serveur web classique le plus utilisé, qui prend en charge des requêtes des utilisateurs. Il joue aussi le rôle d'un intermédiaire entre les clients et les différents serveurs qui composent notre système.

Tomcat est lui aussi un serveur web (écrit en java) qui coopère avec le serveur http apache capable de charger des objets java (les servlets) qui vont prendre en charge certaines requêtes HTTP et engendrer dynamiquement un document HTML constituant la réponse.

Cocoon est une servlet qui va être appelée par Tomcat. Associée à Tomcat, cette servlet fournit un environnement de publication à base de documents XML/XSLT.

Les requêtes de l'utilisateur sont adressées au serveur : s'il s'agit de demandes http pour des documents HTML ou PHP, Apache les prend en charge sans recourir à Tomcat. En revanche des demandes qui impliquent des documents XML sont transmises par Apache à Tomcat. Ce dernier fait appel à Cocoon, qui lui fait une utilisation de la technologie XML et des outils associés tels que XSL, DOM et FOP. Il prend donc en charge l'ensemble du processus de production du document HTML.

3-6 ARCHITECTURE RÉELLE UTILISÉE

L'architecture que je viens de présenter représentait une architecture de base que nous avons établie au départ pour introduire les modèles en totalité, comme ils sont présentés plus haut, mais réellement, nous n'avons pas utilisé sa configuration en totalité parce que nous avons modifié notre modèle de données de façon à ne plus avoir besoin de Tomcat et de cocoon. Par ailleurs, même si nous avons gardé la même architecture, avec les mêmes configurations, nous avons travaillé en réalité avec une configuration plus simple qui ressemble à la suivante :

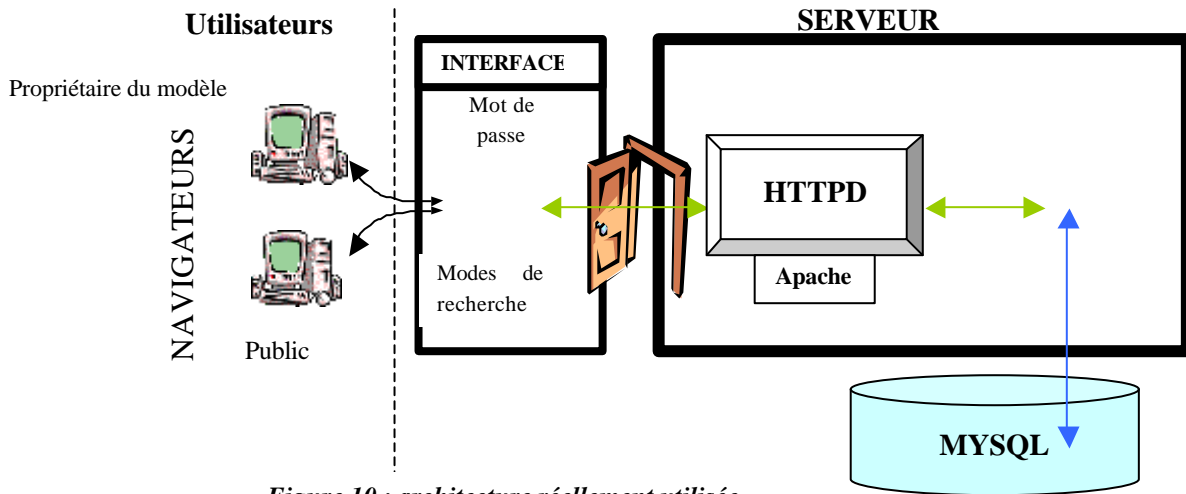


Figure 10 : architecture réellement utilisée

4- OUTILS DE DÉVELOPPEMENT ET D'EXPLOITATION

4-1 SERVEUR : LOGICIELS DE BASES

Le serveur de documents doit s'appuyer sur :

- **LINUX** comme système d'exploitation ;
- **Apache** comme serveur http. Choisi au vu de sa notoriété sur le marché et sa stabilité prouvée, tout en sachant bien sûr qu'il est gratuit. Il prend en charge tous les documents de type classique HTML, PHP ou autre ;
- **Tomcat** comme serveur web (moteur de servlets) qui prend en charge les documents XML.
- **Mysql** comme logiciel de gestion de base de données. Il offre une grande rapidité, d'autant plus qu'il est gratuit, simple d'utilisation et parfaitement adapté à la plupart des applications Web.
- **Cocoon** pour la transformation (XSL/XSLT) de documents XML.

Cocoon permet de concevoir une architecture de site Web parfaitement structurée, en réduisant les efforts de duplication et les coûts de maintenance en permettant différents types de publication et présentation possibles (HTML, PDF, WML,..) des mêmes données, et en séparant ces dernières de leur présentation.

N.B : S'il est nécessaire d'ajouter des logiciels complémentaires à ceux précédemment cités, ceux-ci sont obligatoirement des logiciels libres.

4-2 LANGAGE DE PROGRAMMATION

Le langage prévu d'utilisation pour le développement du site était Java. Ce langage permet d'envisager une portabilité maximale sous différents systèmes d'exploitation. Nous avons changé, après avoir renoncé à l'idée d'utiliser tout le modèle en format XML, et ce pour une raison de contrainte de temps et de disponibilité des personnes sur le projet. Le langage utilisé

est donc PHP, et ce pour sa simplicité et sa souplesse d'une part et d'autre part son intégration en tant que module dans Apache et avec Mysql.

4-3 CLIENTS

Les clients sont obligatoirement des navigateurs (Netscape ou Internet Explorer (I.E.)) sous LINUX et sous WINDOWS (à partir de 95).

En principe, aucun autre logiciel ne doit être nécessaire sur les clients.

5- CONCLUSION

Cette partie nous a permis de définir les différents besoins du projets, ainsi que les différents services qu'il peut offrir aux utilisateurs. Elle nous a donné une idée très claire sur les différentes possibilités de développement du projet et jusqu'à quel point nous pouvons tracer les limites et les contraintes éventuelles pour atteindre notre objectif. Elle constitue ainsi la partie la plus délicate et la plus difficile à réaliser. Pour le faire, il fallait de multiples réunions avec les différents collaborateurs du projet, ainsi qu'avec des spécialistes intéressés par le projet, des différentes modélisations et remodelisations, ...

Cependant, malgré notre concertation, et notre volonté d'être méticuleux dans la définition des besoins et des spécifications, nous allons voir par la suite qu'au niveau réalisation, il y a beaucoup de modifications et de changements, même au niveau des outils de développement.

2^{ÈME} PARTIE : RÉALISATION DU PROJET

Cette partie représente la partie du développement et de réalisation technique du projet. Elle essaie de répondre à tous les besoins définis dans la première partie et consiste à :

- installer toute la plate-forme nécessaire, et de la configurer de façon à ce que le développement du projet se déroule dans les meilleures conditions ;
- créer la base de données pour le stockage des modèles XML ;
- développer les pages HTML et les codes PHP nécessaires pour l'accès à cette base de données ;
- effectuer les tests nécessaires pour le bon fonctionnement du site.

Cependant, comme j'ai cité plus haut, cette partie a subi quelques changements, notamment après les phases d'installation et de configuration, et ce pour plusieurs raisons dont les plus importantes sont celle du temps consacré à la réalisation du projet et celle de la lourdeur de la mise en place de notre première méthodologie. La première grande modification consiste à séparer les champs du modèle et les mettre dans des champs séparés dans la base de données, et de ne laisser que le schéma XML comme un champs texte en entier, et qui n'a pas besoin d'être transformé parce que les utilisateurs auront besoin de le voir tel qu'il est. Ceci nous a poussé à abandonner Tomcat et Cocoon, puisqu'on n'en avait plus besoin (On n'a pas des documents XML à transformer). La deuxième modification est d'abandonner java au profit de PHP, et ce pour une raison de suivi du site¹⁴. Néanmoins, je vais continuer à parler de ces outils pour donner au moins une idée sur leur façon de fonctionnement et aussi pour une éventuelle évolution du projet.

1- LES OUTILS DE DÉVELOPPEMENT

La première étape de notre projet consiste à installer les outils de développement.

Comme c'était prévu dans notre cahier des charges, tous les outils font partie des logiciels libres :

1-1 SYSTÈME D'EXPLOITATION

Notre choix a été fixé sur la distribution « Mandrake de linux », et ce pour plusieurs raisons :

- Sa gratuité ;
- Sa disponibilité au moment du développement du projet ;
- Les différents outils qu'elle englobe, à savoir le serveur Mysql, le module PHP ...
- Sa documentation détaillée (How to), notamment en langue française.

Nous avons installé Linux (mandrake) dans une partition sur notre ordinateur qui héberge aussi Windows 98.

1-2 SERVEUR WEB

Nous avons choisi pour cela le serveur httpd d'Apache sous linux, (version 1.3.22) au vu de sa notoriété sur le marché et sa stabilité prouvée, tout en sachant bien sûr qu'il est gratuit.

Ce serveur est inclus dans linux, il suffit de l'installer lors de l'installation de Linux et le configurer pour qu'il démarre directement avec l'OS.

¹⁴ la personne qui va continuer à entretenir le site travaille déjà avec du PHP. En plus, l'intégration du PHP avec Mysql nous permet de faire ce que nous avons à faire.

1-3 MOTEUR DE SERVLETS : TOMCAT¹⁵

Tomcat est un serveur applicatif JAVA, il permet d'exécuter des servlets et des JSP. C'est également un serveur web qui supporte le «ssl», les «virtuals hos», les «cgi», etc. Mais sa spécialité reste les servlets et les JSP. La configuration la plus souple et la mieux adaptée à une configuration où fonctionne déjà un serveur http apache est probablement de cantonner Tomcat à servir les servlets/JSP et à laisser apache faire le reste.

Pour ce faire, on a eu recours à un module qui s'appelle «**mod_webapp**» qui s'ajoute à apache pour qu'il reçoive toutes les requêtes HTTP et redirigera de manière transparente les requêtes destinées à Tomcat.

1-4 COCOON

Cocoon est basé sur un modèle-vue-contrôleur, et permet ainsi au lieu d'utiliser des solutions compliquées avec JavaBeans, Servlets et JSP de centraliser le traitement tout en respectant le modèle MVC, ce qui facilite la maintenance du site Web et accélère son développement.

Cocoon permet de concevoir une architecture de site Web parfaitement structurée en réduisant les efforts de duplication et les coûts de maintenance en permettant différents types de publication et présentation possibles (HTML, PDF, WML,..) des mêmes données, et en séparant ces dernières de leur présentation.

1-4-1 RELATION ENTRE COCOON ET TOMCAT

Tomcat reçoit une requête HTTP et la passe à la servlet Cocoon qui la traite. Cocoon permet aussi d'envoyer des requêtes à la base de données et récupérer les résultats sous la forme d'un fichier XML en utilisant un module déjà intégré à l'intérieur de Cocoon lui même.

Cocoon permet aussi d'emballer du code java à l'intérieur de fichiers XML, et on pourra donc avoir des contenus dynamiques en utilisant la technologie très poussée XSP.

Cocoon fournit aussi plusieurs outils très pratiques pour notre projet, notamment un module d'internationalisation, ce qui nous permettra de publier les données sur les modèles XML dans la langue souhaitée.

1-5 MYSQL

MySQL est un gestionnaire de bases de donnée qui fonctionne sur les systèmes Unix et qui est diffusé sous une licence légèrement différente de la GPL mais néanmoins disponible en code source. Ce logiciel offre de bonnes performances et supporte les bases de données relationnelles et le langage d'interrogation SQL. En terme de requête, son utilisation apparaît proche de celle d'autres logiciels équivalents Postgres ou Oracle. Destiné à fonctionner dans un contexte de serveur réseau, le fonctionnement de MySQL repose sur un démon, mysqld, qui réalise les accès effectifs à la base et qui gère une file d'attente des diverses requêtes.

Notre base de donnée sera utilisée pour contenir les différentes données concernant les modèles XML disponibles telles que le nom du modèle, la version, le réviseur, l'organisation, le responsable, le mode de restitution, le type de donnée, etc..

¹⁵ Les outils Tomcat et Cocoon sont installés et configurés, mais nous ne les avons pas utilisés, et ce pour des raisons de simplification du projet d'une part, et par contraintes du temps d'autre part. Cependant, j'ai bien voulu les garder dans ce document pour donner une idée, même sommaire sur ces outils, et pour une éventuelle évolution du projet.

1-5-1 JUSTIFICATION DE L'UTILISATION D'UNE BASE DE DONNÉES :

Pour gérer les données concernant les modèles archivés, une base de donnée a été préférée à un système de fichier.

En effet, le système de fichier aurait pu être intéressant pour un petit nombre de données figées. Cela aurait pu permettre un lien direct sur le modèle sélectionné. Seulement, l'ajout de champs ou la modification ou la suppression d'un champ serait moins évidente à mettre en oeuvre qu'une simple mise à jour pour la base de données. De plus, la gestion des accès sera gérée grâce à un pool manager intégré, ainsi que le système de vues est géré automatiquement par la base de données et son pilote de connexion. Enfin, le système de vue permettra de visualiser les données sous n'importe quelle forme, un système de sécurité peut être définie et l'intégrité des données peut être maîtrisée. En résumé, pour optimiser la réutilisation des données, nous avons opté pour une base de données.

Comme système de gestion de base de données, nous avons choisi Mysql car il est gratuit, simple d'utilisation et parfaitement adapté à la plupart des applications Web.

2- INSTALLATION ET CONFIGURATION

Cette configuration va nous permettre de créer un environnement de développement de sites web qui nous permettent de publier des documents XML :

Prérequis :

- une machine linux
- un serveur apache
- un SGBD MySQL

Ce qu'on va rajouter :

- Le JDK (environnement Java) ;
- Le moteur de servlet d'apache Tomcat
- Le module apache qui joue l'intermédiaire entre Apache et Tomcat
- Cocoon : qui permet de transformer les documents XML en documents HTML ou autres.

2-1 INSTALLATION DU JDK

Nous avons téléchargé une version GNUZIP Tar shell script à partir d'Internet. Une fois téléchargé, il faut exécuter le shell script, pour ce faire, il faut modifier les droits sur le fichier:

- `chmod u+x ./j2sdk-1_4-linux-i386.bin`

puis exécuter le script:

- `./j2sdk-1_3_1_01-linux-i386.bin`

répondre « **yes** » à la licence, et hop le TGZ va simplement décompresser le jdk dans le répertoire courant et créer le répertoire `jdk1.4`.

Nous avons donc installé java dans le répertoire courant.

Pour le rendre disponible à tous les utilisateurs, nous avons préféré le placer dans un endroit standard accessible de tous, C'est à dire dans `/usr/java/`, après avoir créé ce répertoire.

Les commandes utilisées sont :

- `mkdir /usr/java`
- `mv ./jdk1.4 /usr/java` (pour déplacer le répertoire `jdk1.3.1_01` dans le nouveau répertoire)

Nous avons fait ensuite un lien du répertoire vers «java» pour ne pas avoir à se rappeler le nom exact du JDK) :

- `ln -s /usr/java/jdk1.3.1_01 /usr/java/java`
- `chown -R root:root /usr/java` (pour que tout les fichiers du JDK appartiennent à root)

Pour vérifier que tout marche bien il suffit de taper:

- `/usr/java/java/bin/java -version`

et voir apparaître un message:

```
java version "1.4" Java(TM) 2 Runtime
Environment, Standard Edition (build 1.3.1_01) Java HotSpot(TM)
Client VM
(build 1.3.1_01, mixed mode)
```

2-2 INSTALLATION DE TOMCAT

Pour installer Tomcat, il suffit de récupérer le fichier «jakarta-tomcat 4.0.4b1 LE with JDK 1.0.4.tar.gz» correspondant sur le site du <http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.1/bin/jakarta-tomcat-4.0.1.tar.gz>

puis de faire un :

```
% tar zxvf jakarta-tomcat 4.0.4b1 LE with JDK 1.0.4.tar.gz
```

pour décompresser Tomcat dans le répertoire `jakarta-tomcat-4.0.4b1`, et hop Tomcat est installé. Maintenant on appelle `TOMCAT_HOME` le répertoire `jakarta-tomcat-4.0.1`.

Et on fait comme on a fait pour le JDK un lien vers un nom qui n'est pas difficile à retenir : `% ln -s jakarta-tomcat-4.0.4b1 Tomcat`

Le script de démarrage s'attend à avoir la variable `JAVA_HOME` positionnée sur l'endroit où nous avons installé java. Il faut :

- 1) éditer le «`.bash_profile`» et rajouter la ligne :
`JAVA_HOME=/usr/java/java; export JAVA_HOME` **ou**
- 2) éditer le fichier «`/etc/profile`» et rajouter la ligne :
`JAVA_HOME=/usr/java/java;export JAVA_HOME` **ou**
- 3) éditer le script `$TOMCAT_HOME/bin/catalina.sh` et rajouter la ligne :
`JAVA_HOME=/usr/java/java`

Ceci pour faire en sorte que le script de démarrage sache où trouver JAVA. Maintenant pour tester que Tomcat fonctionne on tape:

```
$TOMCAT_HOME/bin/startup.sh
```

Le message suivant apparaît si tout marche bien :

```
Using CATALINA_BASE: /home/Tomcat
Using CATALINA_HOME: /home/Tomcat
Using CATALINA_TMPDIR: /home/Tomcat/temp
Using JAVA_HOME: /usr/java/java
```

mais surtout en faisant un

```
tail -f $TOMCAT_ROOT/logs/catalina.out
```

voir le message

```
Starting service Tomcat-Apache
Apache Tomcat/4.0.3
```

```
Server 1.6 is running
Press [Ctrl]+[C] to abort
Starting service Tomcat-Standalone
Apache Tomcat/4.0.3
Starting service Tomcat-Apache
Apache Tomcat/4.0.3
Server 1.6 is running
Press [Ctrl]+[C] to abort
```

Pour arrêter tomcat, il suffit de taper :

```
$TOMCAT_HOME/bin/shutdown.sh
```

2-3 INTÉGRATION AVEC APACHE

Maintenant JAVA est en place, Tomcat est en place. Il ne reste plus qu'à brancher apache sur tomcat (via le port 8008).

Pour ce faire, il faut télécharger le module « `mod_webapp.so` ». à partir du site : <http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.2-b2/src/webapp-module-1.0.2-tc402-src.tar.gz>

Puis, il faut faire :

```
- tar zxvf webapp-module-1.0.2-tc402-src.tar.gz
- cd webapp-module-1.0.2-tc402/
```

Et on obtient ainsi le module « `mod_webapp.so` » qu'il faut le placer dans le répertoire `/etc/httpd/modules`).

```
- cp apache-1.3/mod_webapp.so /etc/httpd/modules
```

On va ensuite modifier le fichier de configuration de Apache qui est `/etc/httpd/conf/httpd.conf` pour prendre en compte ce nouveau module de la manière suivante:

On cherche la chaîne `LoadModule`

il faut placer la ligne suivante après le dernier `LoadModule` et rajouter la ligne suivante

```
LoadModule webapp_module modules/mod_webapp.so
```

Puis rajouter `AddModule mod_webapp.c` à la fin de la liste des `AddModule`

Puis rajouter les lignes suivantes pour faire le lien avec Tomcat :

```
WebAppConnection conn warp localhost:8008
WebAppDeploy examples conn examples
WebAppInfo /webapp-info
```

La première ligne définit une connexion qui s'appelle « `conn` » vers « `tomcat` » qui écoute sur le port 8008.

La deuxième ligne définit le mapping entre l'url `http://VHOST/examples/` et la `webapp examples` qui existe dans Tomcat.

La troisième ligne est optionnelle et sert à obtenir des infos sur le connecteur `mod_webapp` via l'URL `http://VHOST/webapp-info/`.

Il faut maintenant relancer apache, mais avant ça, il faut vérifier qu'on n'a pas fait trop de bêtises avec un `configtest`:

```
/usr/sbin/apachectl configtest
```

Si la réponse est `syntax ok`, alors on peut le relancer avec :

```
/usr/sbin/apachectl restart
```

Pour vérifier que tout a bien marché, nous avons fait des tests en accédant à l'adresse :

```
http://localhost/examples/
```

2-4 INSTALLATION DE COCOON

Cocoon, comme, nous avons dit plus haut, se présente sous forme d'une servlet intégrée à un serveur web qui prend en charge des requêtes http. En association avec Tomcat, Cocoon fournit un environnement pour la publication des documents XML.

Pour l'installer, il suffit de télécharger le fichier « .tar.gz » du site <http://xml.apache.org> et l'installer par exemple dans le répertoire /usr/local. On décompressant l'archive, nous avons obtenu un répertoire cocoon-2.0.2.

Pour installer cocoon dans Tomcat, il y a quelques configurations à faire :

1. Supprimer les .jar de \$TOMCAT_HOME/lib
2. Copier tous les .jar de \$COCOON_HOME/lib dans \$TOMCAT_HOME/lib
3. Copier le fichier \$COCOON_HOME/cocoon.war dans \$TOMCAT_HOME/webapps
4. Définir les paramètres de l'« extra-Classpath » dans le fichier : cocoon/src/webapp/WEB-INF/web.xml comme suit :

```
<init-param>
  <param-name>extra-classpath</param-name>
  <param-value>/Tomcat/lib/xalan-xxx.jar:
    /Tomcat/lib/xercesImpl-xxx.jar:
    /Tomcat/lib/xml-apis.jar :
    /Tomcat/lib/batik-libs-xxx.jar</param-value>
</init-param>
```

5. Redémarrer Tomcat : /Tomcat/bin/startup.sh ; ou plutôt si on est dans le répertoire /Tomcat/bin, on fait : ./startup.sh
6. Ouvrir la page d'accueil de Cocoon pour tester si tout va bien : <http://localhost:8080/cocoon>. Si la page s'affiche, c'est gagné.

2-5 DÉMARRAGE ET ARRÊT DE TOMCAT AVEC APACHE

Après avoir installé tous les outils nécessaires pour le développement, il ne reste qu'à trouver une solution pour démarrer et/ou arrêter Tomcat avec Apache, ce qui évitera de les lancer manuellement. Pour le faire, il suffit de créer un script qui sera placé dans le répertoire /etc/init.d/ et qui sera chargé du démarrage de Tomcat et d'Apache. (Démarrer Tomcat en premier, puis Apache).

Ce script, nous l'avons appelé tomcatapache.sh, et nous l'avons placé dans /etc/init.d après avoir donné les droits qu'il faut : `chmod 755 /etc/init.d/tomcatapache.sh`¹⁶.

3- CONCEPTION DE LA BASE DE DONNÉES

Notre base de données s'appelle «MODELESXML», un nom qui veut exprimer le projet qui consiste à la création d'un répertoire de modèles XML.

Cette création se fait avec la commande : create database MODELESXML dans le client mysql que l'on obtient en tapant simplement mysql.

¹⁶ Voir contenu du fichier en annexe

Ceci suppose bien sûr que le serveur et le client du SGBD MySQL soient démarrés. (La vérification se fait avec la commande « ps -ax »)

L'organisation des tables de la base de données se fait suivant le schéma Entité-Association établi lors de la modélisation. Il en résulte 44 tables, dont la table principale est DECLARATIONMODELE.

Cette base se compose de 44 tables (27 tables principales et de 17 tables intermédiaires (ou de jonction)). Par « intermédiaires », je veux dire les tables qui font les liens entre deux tables principales (et qui contiennent leurs clés primaires).

Pour simplifier le projet, nous avons préféré que la base de données soit valable pour les deux modèles que nous allons en parler par la suite. De ce fait, plusieurs tables concernent un modèle et non pas l'autre.

DECLARATIONMODELE;	SECTEURACTIVITE_DM;
DECLARATIONALTERNATIVE;	CLASSIFICATIONPRODUIT;
METAMODELE;	CLASSIFICATIONPRODUIT_DM;
DOMAINEUSAGE;	URLS;
DOMAINEUSAGE_DM ¹⁷ ;	URLS_DM;
RECOMMANDATIONW3C;	PROPRIETEMODELE;
RECOMMANDATIONW3C_DM;	OUTILASSOCIE;
MODERESTITUTION;	OUTILASSOCIE_DM;
MODERESTITUTION_DM;	MODELELIE;
TYPEDONNEE;	MODELELIE_DM;
TYPEDONNEE_DM;	UTILISATEURS;
REVUE;	UTILISATEURS_DM;
REVUE_DM;	COMPOSANTREUTILISE;
CONTACT;	FEUILLEDESTYLE;
ORGANISATION;	CHAMPAPPLICATION;
CONTACT_DM;	CHAMPAPPLICATION_DM;
PROJET;	REMARQUES;
PROCESSUS;	DEFINITIONCC;
PROCESSUS_DM;	NOMBUSINESS;
ROLE;	DOCUMENTLIE;
ROLE_DM;	VALEURCODE_DM;
SECTEURACTIVITE;	VALEURCODE;

Pour la création des tables, nous avons préféré les créer dans un fichier «.sql» pour nous permettre par la suite, de les modifier facilement et de garder toujours des traces de l'évolution de la base.

Le fichier de création des tables s'appelle *MODELESXML.SQL*¹⁸.

¹⁷ DM = Declaration Modele. Pour chaque table qui fait le lien entre la table principale DECLARATIONMODELE et une autre table principale, nous avons ajouté «_DM» au nom de la deuxième table

¹⁸ Voir le fichier de création des tables dans la base de données en Annexe

La commande qui permet de créer la base de données à travers ce fichier est la suivante :

```
% mysql < MODELESXML.SQL
```

Cette tâche peut se faire aussi à travers l'outil PhpMyAdmin (voir paragraphe suivant : modalités d'installation et de configuration de cet outil).

4- CONNEXION ET ACCÈS À LA BASE DE DONNÉES

L'alimentation de la base (comme d'ailleurs la création et la mise à jour) peut se faire en mode de ligne, exactement comme on a fait pour la création, et ce par la commande :

```
insert into table() values('value1', 'value2')
```

Elle pourra se faire aussi par le biais de l'outil **PhpMyAdmin**, qui se présente comme étant une application écrite en PHP qui permet d'administrer, depuis un navigateur, des bases de données de type MySQL. Il nous permet donc en quelques clics, de créer, modifier et supprimer des bases de données ou des tables, exécuter des requêtes SQL, exporter et importer des fichiers ...

Nous avons pu le récupérer à l'URL <http://www.phpwizard.net/phpMyAdmin>. En détail **phpMyAdmin** permet de:

- créer et supprimer des bases de données,
- éditer, ajouter ou supprimer des champs,
- taper des commandes **SQL**,
- gérer les clés de champs,
- ...

L'archive se présente sous la forme d'un tarball **phpMyAdmin_2.0.6.tar.gz**, pour le décompresser, il suffit de taper :

```
tar xvfz phpMyAdmin_2.0.6.tar.gz
```

Cela va créer dans le répertoire de travail un répertoire **phpMyAdmin**. Dans ce répertoire on va éditer le fichier **config.inc.php3**, on va utiliser la méthode d'authentification "basique" :

```
$cfgServers[1]['adv_auth'] = false;// Use advanced authentication?
$cfgServers[1]['stduser'] = 'root';// MySQL standard user (only needed with advanced auth)
$cfgServers[1]['stdpass'] = '';// MySQL standard password (only needed with advanced auth)
$cfgServers[1]['user'] = 'mkadmi';// MySQL user(only needed with basic auth)
$cfgServers[1]['password'] = 'mot-de-passe';// MySQL password (only needed with basic auth)
```

Puis, pour avoir la version française, on va mettre dans le même fichier la phrase suivante au lieu de:

```
Require ("french.inc.php3");    au lieu de :
require("english.inc.php3");
```

Et enfin on doit rendre accessible le répertoire **phpMyAdmin** d'une page web, pour cela, nous avons placé **phpMyAdmin** dans **/var/www/html**.

4-1 PRINCIPES DE FONCTIONNEMENT

Une des utilisations les plus répandues du Web dynamique est l'accès distant aux bases de données à travers un navigateur. Nous présentons ici notre méthode qui est basée sur l'utilisation conjointe de PHP et MySQL..

Le langage PHP supporte MySQL par une série de fonctions. La plus complexe d'utilisation est sans doute **mysql_db_query** qui ouvre une base et soumet cette requête SQL. Cette requête est une chaîne de caractères qui sera en général calculée lors de l'exécution du script. La requête transmise, le script PHP récupère du démon mysqld les enregistrements de la base correspondants à travers des fonctions comme **mysql_fetch_row** ou **mysql_fetch_array**. Les divers champs de ces enregistrements servent à constituer le document HTML qui sera servi en réponse à la requête.

Le système de protection associé à MySQL peut paraître contraignant. Celui ci est néanmoins indispensable pour assurer le bon fonctionnement des applications dans un contexte réseau. Aussi il est déconseillé de le désactiver en autorisant toutes les opérations.

4-2 SCRIPTS DE COMMUNICATION AVEC MYSQL

Dès qu'on commence à produire des pages à partir de scripts communiquant avec Mysql, on se trouve amené à programmer de manière répétitive certaines parties du code qui correspond le plus souvent à des opérations routinières telles que la connexion à la base, l'exécution d'une requête ... ce qui alourdit le code et le rendre illisible.

Pour ces raisons, nous avons commencé par la programmation de ces opérations routinières à travers des utilitaires qui sont basés soit sur les fonctions, soit sur la programmation objet.

- Le premier fichier est destiné à définir le serveur, la base, l'utilisateur et le mot de passe. Ce fichier s'appelle : **connect.php**.

Il est présenté comme suit :

```
<?php
define (NOM, "root");
define (PASSE, "");
define (SERVEUR, "localhost");
define (BASE, "essai");
?>
```

- Un autre fichier destiné à définir la connexion avec la base Mysql avec la fonction connexion qui prend comme paramètres les valeurs nécessaires pour se connecter à un serveur sous un compte utilisateur, et se placer ensuite dans la base si la connexion s'établit. Ce fichier est appelé : **Connexion.php**.

```
<?php
if (!isset ($FichierConnexion))
{
$FichierConnexion=1;
// Fonction Connexion : connexion à Mysql
function Connexion ($pNom, $pMotPasse, $pBase, $pServeur)
{
// Connexion au serveur
$connexion = mysql_pconnect ($pServeur, $pNom, $pMotPasse);
if (!$connexion)
{
echo "Désolé, connexion au serveur $pServeur impossible\n";
```

```

exit;
}
// Connexion à la base
if (!mysql_select_db ($pBase, $connexion))
{
echo "Désolé, accès à la base $pBase impossible\n";
echo "<B>Message de MySQL :</B>" .mysql_error($connexion);
exit;
}
// On renvoie la variable de connexion
return $connexion;
}
// Fin du test sur $fichierconnexion
}
?>

```

- Un troisième fichier a été créé pour définir les fonctions pour exécuter une requête avec Mysql. Ce fichier contient deux fonctions, une pour exécuter la requête et l'autre pour récupérer le résultat. Ce fichier est appelé : **ExecRequete.php**

```

<?php
if (!isset ($FichierExecRequete))
{
$FichierExecRequete = 1;
// Exécution d'une requête avec MySQL
function ExecRequete ($requete, $connexion)
{
$resultat = mysql_query ($requete, $connexion);
if ($resultat)
return $resultat;
else
{
echo "<B>Erreur dans l'exécution de la requête '$requete'.</B><BR>";
echo "<B>Message de MySQL : </B>" . mysql_error($connexion);
exit;
}
} //fin de la fonction ExecRequete
//recherche de la ligne suivante
function LigneSuivante ($resultat)
{
return mysql_fetch_object ($resultat);
} // Fin de la fonction LigneSuivante
} // Fin du test
?>

```

Ces trois fichiers et éventuellement d'autres peuvent être inclus dans n'importe quel script avec la commande « include », ou « require ».

5- PRÉSENTATION DU SITE WEB

Pour pouvoir accéder et mettre à jour la base de données, l'utilisateur doit passer par une première interface, ou plutôt une page d'accueil qui contient outre une présentation du projet, une liste de choix, par lesquels, il peut accéder à la ou les rubriques qui l'intéresse(nt).

Cette page d'accueil se présente comme suit :

[Réviser les modèles](#)

RÉPERTOIRE DES MODÈLES POUR LES APPLICATIONS XML

Pour permettre le partage et l'échange de documents XML en milieu ouvert, [EDIFRANCE](#) (Association pour le développement des échanges électroniques professionnels), le [GFII](#) (Groupement Français de l'Industrie de l'Information) et [Mutu-XML](#), avec le soutien de la [FING](#) (Fondation Internet Nouvelle Génération), lancent l'initiative de la création d'un répertoire des modèles pour les applications XML, privés et publics.

L'objectif de ce projet est de mettre à disposition, sur Internet, l'ensemble des modèles XML francophones publics existants ; l'objectif est également de mettre à disposition les modèles existants dans d'autres langues, en les rendant compréhensibles par les lecteurs francophones.

ACCÉDER AU RÉPERTOIRE DE MODÈLES XML :

- Pour **consulter** la liste complète :
 - des [modèles simples](#)
 - des [modèles core components](#)
- Pour **chercher** :
 - un [modèle](#)
 - un [core component](#)
- Pour [exporter un core component](#)
- Pour **soumettre** ou **modifier** un modèle, veuillez vous [identifier](#) d'abord

Figure 11 : page d'accueil du site

Comme, nous constatons, à travers cette interface, les rubriques, aux quelles les utilisateurs peuvent accéder peuvent se résumer en cinq grandes fonctionnalités :

- Consultation de la liste des modèles de la base de données ;
- Soumission ou modification des modèles ;
- Recherche des modèles ;
- Exportation des core components dans des fichiers texte ;
- Révision des modèles.

5-1 CONSULTATION DES MODÈLES

Cette première rubrique, ne nécessite aucune authentification, et n'importe quel utilisateur peut y accéder par un simple clic et consulter la liste des modèles existants dans la base. Cette liste contient deux sous rubriques : liste de modèles simples et liste de modèles core components. Chaque liste contient le numéro d'identifiant du modèle dans la base et son nom. Pour les modèles simples, ce nom est donné par le propriétaire à sa guise, mais pour le core component, le nom représente le nom de son entrée dans le dictionnaire (Dictionary Entry Name).

À partir de cette liste, l'utilisateur peut cliquer sur le nom pour accéder au contenu du modèle qui à la même forme d'un formulaire de soumission ou de modification.

Dans le cas, où il y a des fichiers attachés à ce modèle (exemple Schéma XML ou des feuilles de style...dont la taille dépasse la taille maximale autorisée par un Mysql), l'utilisateur trouve un lien vers ce fichier, tout en ayant la possibilité de le télécharger ou de l'ouvrir.

5-2 RECHERCHE DES MODÈLES

Cette rubrique est comme la première, n'exige aucune authentification et permet aux utilisateurs de saisir des critères de recherche pour accéder à tel ou tel modèle qui l'intéressent. Les critères de recherche prédéfinis sont :

- Le nom du modèle ;
- La version ;
- Le secteur d'activité ;
- Le domaine d'usage du modèle ;
- Le rôle d'affaire ;
- Le processus ;
- Les mots clés.

Nous pouvons bien sûr établir des équations booléennes entre ces critères en utilisant des opérateurs tels que « ET » et « OU ».

Cette rubrique est représentée par l'interface suivante :

FORMULAIRE DE RECHERCHE

Ce formulaire vous permet d'indiquer des paramètres pour la recherche des modèles

Vous pouvez chercher un modèle par :

SECTEUR D'ACTIVITÉ	<input style="width: 90%;" type="text"/>
DOMAINE D'USAGE	<input style="width: 90%;" type="text"/>
RÔLE	<input style="width: 90%;" type="text"/>
PROCESSUS	<input style="width: 90%;" type="text"/>
GESTIONNAIRE LISTE	<input style="width: 90%;" type="text"/>
NOM DU MODÈLE	<input style="width: 90%;" type="text"/>
VERSION	<input style="width: 90%;" type="text"/>
MOTS CLÉS	<input style="width: 90%;" type="text"/>

ET
OU ?

Figure 12 : Interface de recherche

5-3 EXPORTATION DES CORE COMPONENTS

Quant à la quatrième rubrique, celle de l'exportation des core components dans des fichiers, elle représente une facilité spécifique qui n'était même pas définie dans le cahier des charges, mais qui s'avérait intéressante, dans le sens où tous les professionnels d'EDI presque stockent leurs « core components » dans des fichiers « Excel ». Elle ne nécessite aucune authentification.

5-4 SOUMISSION ET MODIFICATION DES MODÈLES

À la différence des autres rubriques précédentes, cette rubrique exige une certaine identification de la part de l'utilisateur, et ce pour deux raisons : identifier le propriétaire de chaque modèle d'une part et d'autre part éviter les duplications au niveau des propriétaires de modèles et seul le propriétaire du modèle pourra modifier son modèle.

Cette authentification se fait à travers un login et un mot de passe pour les personnes qui sont déjà inscrits dans la base. Pour les autres, ils peuvent s'identifier à travers un formulaire spécifique, là où ils renseignent le nom, le prénom, le login, le mot de passe, l'adresse, le téléphone, le fax ...

Login :
Mot de passe :

Si vous êtes nouveau, veuillez vous [identifier](#)

Figure 13 : Interface d'identification pour les propriétaires déjà inscrits

Si l'utilisateur est nouveau, il suffit de cliquer sur « identifier » pour qu'il reçoive un formulaire qui lui permet de s'inscrire. Ce formulaire a la forme suivante :

PROPRIETAIRE DU MODELE

* Nom :	<input type="text"/>
* Prénom :	<input type="text"/>
Fonction :	<input type="text"/>
* Login :	<input type="text"/>
* Mot de passe :	<input type="password"/>
* Confirmation Mot de passe :	<input type="password"/>
Adresse:	<input type="text"/>
	<input type="text"/>
* Code Postal :	<input type="text"/>
* Ville :	<input type="text"/>
Région :	<input type="text"/>
Pays :	<input type="text"/>
* Tél. :	<input type="text"/>
Fax :	<input type="text"/>
* Mail :	<input type="text"/>
Site Web :	<input type="text"/>
<input type="button" value="Submit"/>	<input type="button" value="Annuler"/>

Figure 14 : Interface d'identification pour les nouveaux propriétaires de modèles

Pour soumettre un nouveau modèle, il suffit, comme nous avons expliqué plus haut, de cliquer sur la rubrique « **Ajouter (soumettre) un nouveau modèle ou modifier un modèle déjà existant** » tout en choisissant le type de modèle à soumettre ou à modifier. S'il s'agit d'une soumission, il accède directement au formulaire de soumission (voir Annexe 5 la présentation de ce formulaire, p.92). Cependant, quand il s'agit d'une modification, l'utilisateur accède à sa liste de modèles contenus dans la base, puis il choisit le modèle à modifier pour qu'en fin puisse accéder au formulaire (qui a la même présentation que celui de soumission, mais il contient des données).

Après chaque soumission ou modification, le propriétaire valide et envoie le modèle au serveur qui le traite à travers un script de mise à jour et reçoit une confirmation de son action, et une nouvelle interface sommaire qui lui permet d'accéder à d'autres rubriques sans passer par une nouvelle identification tout en gardant toujours sa session. Cette page sommaire se présente comme suit.

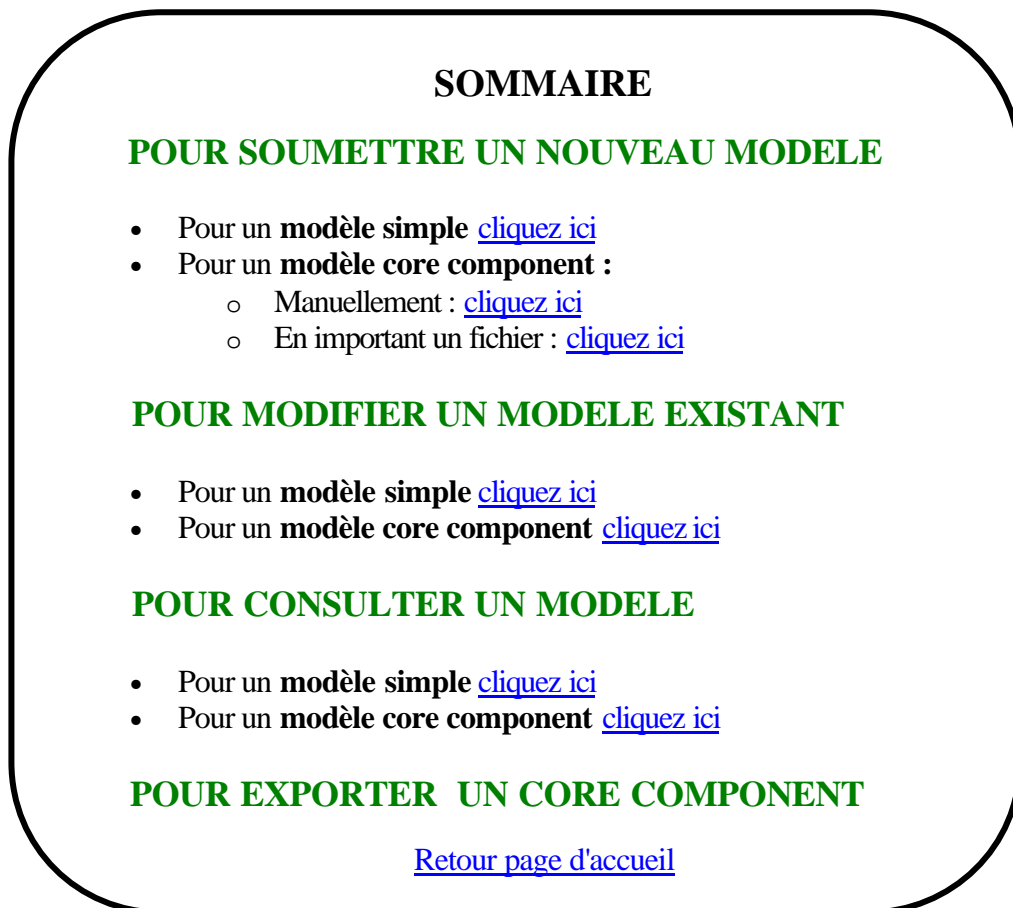


Figure 15 : Interface sommaire du site

Le soumissionnaire donc choisit la première rubrique « Soumission » , et selon son besoin, il choisit le type du modèle (« simple » ou « core component ») à déclarer.

Quant au modèle core component, le soumissionnaire a deux choix : remplir à la main tout le modèle ou s'il a déjà des core components dans un fichier Excel, il peut les importer directement et ne remplir les données communes entre tous les core components qu'une seule fois. Quant aux formulaires spécifiques à ces deux choix, je vais les présenter dans le paragraphe qui suit.

5-4-1 FORMULAIRES DE SOUMISSION DE MODÈLES¹⁹

Pour soumettre un modèle, nous avons créé trois formulaires :

- un pour déclarer un modèle simple (modelsimple.php);
- un pour déclarer un modèle du core component (modelcc.php);
- et un troisième pour importer un core component sous forme d'un fichier Excel (importcc.php).

Ces formulaires sont en langage HTML, mais ils contiennent du code PHP, notamment pour la sauvegarde des variables de sessions. Ils ont donc l'extension « .php » pour qu'ils puissent être interprétés par le serveur.

À partir de ces formulaires, l'utilisateur remplit les données, et surtout les champs qui sont obligatoires à renseigner et les envoie. Le formulaire envoyé sera traité par un script PHP (dont le nom est mentionné dans la rubrique action du formulaire). Ce script récupère les données entrées et les insère dans la base tout en gardant les identifiants des tables principales pour les insérer dans les tables intermédiaires. Ces identifiants sont généralement des « integers » qui s'auto-incrémentent.

Quant au troisième formulaire, il est rempli de la même manière, mais puisqu'il ne contient pas beaucoup de champs et en plus il contient une rubrique d'importer un fichier, il y a un traitement supplémentaire pour séparer les éléments de la feuille Excel, de les mettre dans un tableau, de les récupérer par la suite et les insérer dans la base.

5-5 RÉVISION DES MODÈLES

À la différence des autres rubriques de soumission et de consultation, la révision est une fonction réservée à peu de personnes qui se chargent de l'assurance de la qualité du site (pas au niveau du contenu), mais plutôt au niveau de la présence de certaines informations, et des fautes qui peuvent intervenir.

Le réviseur donc se trouve inscrit par l'administrateur de la base et aura son login et son mot de passe pour accéder à la liste des modèles à réviser. Cependant, sa mission est de faire la vérification nécessaire et de noter ses commentaires, il ne peut pas modifier lui-même aucune donnée.

Le propriétaire, à la lumière des commentaires laissés par le réviseur, modifie son modèle. Le principe est de ne pas laisser les utilisateurs accéder qu'aux modèles révisés, mais à l'état actuel du projet, nous avons laissé tous les modèles même ceux qui ne sont pas révisés en accès libre, tout en affichant pour chacun d'entre eux les commentaires du réviseur qui sont liés, et qui présentent le statut du modèle (soumis, en révision ou révisé), le nom du réviseur et les commentaires associés.

6- SCRIPTS DE TRAITEMENT DES FORMULAIRES²⁰

Les formulaires que ce soient de soumission ou de modification de modèles sont envoyés à des scripts de traitement de données. Ces scripts ont pour rôle de récupérer les valeurs des champs entrées par l'utilisateur et les insérer dans la base et donnent un résultat du traitement. Pour un besoin de clarté, j'ai essayé de séparer les scripts au maximum possible. Les scripts les plus importants sont donc répartis comme suit :

¹⁹ Voir la présentation des formulaires dans le CD-ROM ci-joint

²⁰ Voir tous les formulaires et les scripts associés dans le CD-ROM ci-joint

1- Scripts de traitement de formulaires lors de la déclaration (soumission)

- script de traitement du formulaire de déclaration d'un modèle simple ;
- un scripts de traitement du formulaire de déclaration d'un d'un modèle core component ;

2- Scripts traitement lors de la consultation des modèles :

- script de consultation d'un modèle simple
- script de consultation d'un modèle core component ;

3- scripts de traitement lors de la modification (mise à jour) des modèles :

- script de traitement de la mise à jour d'un modèle simple ;
- script de traitement de la mise à jour d'un modèle core component.

En outre ces scripts, il y a plusieurs autres scripts qui sont communs aux deux modèles et qui traitent les champs multiples (qu'on peut remplir autant de fois pour un seul modèle), et qui ont généralement des interfaces à part.

Vu la longueur des scripts de traitement des modèles, je me contente de présenter ci-dessous quelques exemples de scripts de traitement de différents champs existants dans notre projet, avec leurs présentations HTML.

6-1 TRAITEMENT DES CHAMPS SIMPLES

Un champ simple est un champ unique et qui se remplit une seule fois par modèle. Il a un nom (dans le formulaire) et un nom de la base de données. Son traitement est très facile. Il suffit de récupérer sa valeur et l'insérer dans la base.

Je présente en ce qui suit trois champs simples d'un formulaire, suivis de leur code HTML, puis je présente leur traitement :

Exemple :

Qualifiant de la classe d'objet :	<input type="text"/>
Nom de la Classe d'origine :	<input type="text"/>
Qualifiant de la classe propriétaire :	<input type="text"/>

Le code HTML de ce formulaire est le suivant :

```
<HTML><BODY>
<<form name="form1" method="post" action="traitementCC.php">
<table border="0" cellspacing="0" cellpadding="0">
<tr><td width="250">Qualifiant de la classe d'objet :</td><td><input name="qualifiant_object_class" type="text"
size="50"></td></tr>
<tr><td width="250"><font color="#FF0000">*</font> Nom de la Classe d'origine:</td><td><input
name="origin_object_class" type="text" size="72"></td></tr>
<tr><td width="250">Qualifiant de la classe propriétaire :</td><td><input name="qualifiant_property_class"
type="text" size="72"></td></tr>
</table>
</body></html>
```

6-1-1 TRAITEMENT LORS DE LA SOUMISSION

```
<?
// On initialise les sessions
session_start();
?>
<HTML>
<BODY>
// Appel des fichiers de définition, de connexion et d'exécution des requêtes
require ("connect.php");
require ("Connexion.php");
```

```
require ("ExecRequete.php");
// Connexion à la base de données MODELESXML
$connexion = Connexion(NOM, PASSE, BASE, SERVEUR);
$requete="INSERT INTO DECLARATIONMODELE (QualifiantClasseObjet, ";
$requete .= "NomClasseOrigine, QualifiantClasseProprietaire" ;
$requete .= "values ";
$requete .= "('$qualifiant_object_class', '$origin_object_class', ";
$requete .= "'$qualifiant_property_class')";
$resultat=ExecRequete($requete, $connexion);
```

6-1-2 TRAITEMENT LORS DE LA MODIFICATION (MISE À JOUR)

```
// Après l'appel des fichiers de connexion à la base
// Mise à jour des champs modifiés
$requete="UPDATE DECLARATIONMODELE SET "
    . " QualifiantClasseObjet ='$qualifiant_object_class', "
    . " NomClasseOrigine = '$origin_object_class', "
    . " QualifiantClasseProprietaire = '$qualifiant_property_class' "
    . "WHERE ID_DM = $ID_MODIF ";
$resultat=ExecRequete($requete, $connexion);
// Sauvegarde du dernier Id inséré
$ID_DM=" " .mysql_insert_id();
```

6-1-3 TRAITEMENT LORS DE LA CONSULTATION

```
<?
// On initialise les sessions
session_start();
?>
<HTML>
<BODY>
< ?
// Récupération des données de la table (là où se trouve les champs) «DECLARATIONMODELE»
$requete = ("SELECT * FROM DECLARATIONMODELE where ID_DM = $ID_MODIF ");
$resultat=ExecRequete($requete, $connexion);
while($data = mysql_fetch_array($resultat)){
    // Mise en session de ID_DM
    $session_declaration_modele = $data["ID_DM"];
    session_register("session_declaration_modele");
    ?>
<table border='0' cellspacing='0' cellpadding='0'>
<tr><td height='16' colspan='2'><h5>Classe d'objects</h5></td></tr>
<tr><td width='250'>Qualifiant de la classe d'objet :</td><td><input name='qualifiant_object_class'
type='text' size='72' value="<?echo $data["QualifiantClasseObjet"]?>"></td></tr>
<tr><td width='250'><font color='#FF0000'>*</font> Nom de la Classe d'origine:</td><td><input
name='origin_object_class' type='text' size='72' value="<?echo $data["NomClasseOrigine"]?>"></td></tr>
<tr><td width='250'>Qualifiant de la classe propriétaire :</td><td><input name='qualifiant_property_class'
type='text' size='72' value="<?echo $data["QualifiantClasseProprietaire"]?>"></td></tr>
</table><hr>
</body></html>
```

6-2 TRAITEMENT DES CHAMPS À CHOIX MULTIPLE :

Un champ à choix multiple est un champ qui permet à un utilisateur de choisir une ou plusieurs valeurs parmi une liste prédéfinie. Le traitement de ce type de champs est un peu compliqué que celui d'un champ simple.

Exemple :

SECTEUR D'ACTIVITE

<input type="checkbox"/> Santé	<input type="checkbox"/> Finances	<input type="checkbox"/> Économie
<input type="checkbox"/> Éducation	<input type="checkbox"/> Banques	<input type="checkbox"/> Industrie
<input type="checkbox"/> Presse	<input type="checkbox"/> Autres	

Le fichier HTML de ce champ est le suivant :

```
<html><body>
<form name="form1" method="post" action="traitementMS.php">
<table border="0" cellspacing="0" cellpadding="0">
<tr><td width="200"><font color="#FF0000">*</font size="5"><b> SECTEUR D'ACTIVITE</b></td></tr>
<tr><td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Santé">Santé</td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Finances">Finances</td>
<td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Economie">Economie</td></tr>
<tr><td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Education">Education</td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Banques">Banques
<td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Industrie">Industrie</td></tr><td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Presse">Presse</td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Autres">Autres</td></tr>
</table>
</form></body></html>
```

6-2-1 TRAITEMENT LORS DE LA SOUMISSION :

```
// INSERTION DANS LA TABLE PRINCIPALE : SECTEURACTIVITE
// Boucle sur les secteurs d'activité (choisis)
for ($i=0;$i<count($secteur_activite);$i++){
  // Test sur l'existence du secteur dans la base
  if (isset ($secteur_activite[$i])){
    $requete=("SELECT DISTINCT (ID_SecteurActivite) FROM SECTEURACTIVITE
WHERE Secteur = '$secteur_activite[$i]' ");
    $resultat=ExecRequete($requete, $connexion);
    // Nombre de lignes retournees
    $nb = mysql_num_rows($resultat);
    // Recuperation de l'id du secteur d'activite
    $ID_SA = @mysql_result($resultat,0,"ID_SecteurActivite");
    // Test sur nb
    if($nb != 1){
      // Insertion du secteur d'activite dans la BDD
      $requete2=("INSERT INTO SECTEURACTIVITE (Secteur) values
('$secteur_activite[$i]')");
      $resultat2=ExecRequete($requete2, $connexion);
      $ID_SecteurActivite="" .mysql_insert_id();

      // Insertion du secteur d'activite dans la table de jonction
      $requete3=("INSERT INTO SECTEURACTIVITE_DM (ID_SecteurActivite,
ID_DM) values ($ID_SecteurActivite, $ID_DM)");
      $resultat3=ExecRequete($requete3, $connexion);

    } else {
      //echo "Existe :". $secteur_activite[$i]."<br>";
      // Insertion du secteur d'activite dans la table de jonction
      $requete3=("INSERT INTO SECTEURACTIVITE_DM
(ID_SecteurActivite, ID_DM) values ($ID_SA, $ID_DM) ");
      $resultat3=ExecRequete($requete3, $connexion);
    }
  }
}
```

6-2-2 TRAITEMENT LORS DE LA MODIFICATION

```
// Initialisation de la session
// Appel des fichiers de connexion à la base
// MODIFICATION DANS LA TABLE PRINCIPALE : SECTEURACTIVITE
```

```

// On supprime d'abord les données dans SECTEURACTIVITE
$requete1="DELETE FROM SECTEURACTIVITE_DM WHERE ID_DM = $ID_MODIF ";
$resultat1=ExecRequete($requete1, $connexion);
// Boucle sur les secteurs d'activite
for ($i=0;$i<count($secteur_activite);$i++){
// Test sur l'existence du secteur dans la base
if (isset ($secteur_activite[$i])){
$requete=("SELECT DISTINCT (ID_SecteurActivite) FROM SECTEURACTIVITE WHERE Secteur =
'$secteur_activite[$i]' ");
$resultat=ExecRequete($requete, $connexion);
// Nombre de lignes retournees
$nb = mysql_num_rows($resultat);
// Recuperation de l'id du secteur d'activite
$ID_SA = @mysql_result($resultat,0,"ID_SecteurActivite");
// Test sur nb
if($nb != 1){
// Insertion du secteur d'activite dans la BDD
$requete2=("INSERT INTO SECTEURACTIVITE (Secteur) values ('$secteur_activite[$i]')");
$resultat2=ExecRequete($requete2, $connexion);
$ID_SecteurActivite="" .mysql_insert_id();

// Insertion du secteur d'activite dans la table de jonction
$requete3=("INSERT INTO SECTEURACTIVITE_DM (ID_SecteurActivite, ID_DM) values
($ID_SecteurActivite, $ID_MODIF)");
$resultat3=ExecRequete($requete3, $connexion);
} else {
// Insertion du secteur d'activite dans la table de jonction
$requete3=("INSERT INTO SECTEURACTIVITE_DM
(ID_SecteurActivite, ID_DM) values ($ID_SA, $ID_MODIF) ");
$resultat3=ExecRequete($requete3, $connexion);
}
}
}
}

```

6-2-2 TRAITEMENT LORS DE LA CONSULTATION

```

// Initialisation de la session
// Appel des fichiers de connexion à la base
<table border="0" cellspacing="0" cellpadding="0">
<?
// Récupération des données de la table SECTEURACTIVITE
$requete = ("SELECT * FROM SECTEURACTIVITE S, SECTEURACTIVITE_DM SM where
SM.ID_DM = $ID_MODIF AND S.ID_SecteurActivite = SM.ID_SecteurActivite");
$resultat=ExecRequete($requete, $connexion);
// test sur les secteurs cochés
while($data = mysql_fetch_array($resultat)){
if($data['Secteur'] == "Santé"){
    $sante = 1;
}
elseif($data['Secteur'] == "Finances"){
    $finance = 1;
}
elseif($data['Secteur'] == "Economie"){
    $economie = 1;
}
elseif($data['Secteur'] == "Education"){
    $education = 1;
}
elseif($data['Secteur'] == "Banques"){
    $banques = 1;
}
}
}

```

```

    }
    elseif($data['Secteur'] == "Industrie"){
        $industrie = 1;
    }
    elseif($data['Secteur'] == "Presse"){
        $presse = 1;
    }
    elseif($data['Secteur'] == "Autres"){
        $autres = 1;
    }
}
mysql_free_result($resultat);
?>
<tr><td width="200"><font color="#FF0000">*</font size="5"><b> SECTEUR
D'ACTIVITE</b></td></tr>
<tr><td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Santé" <?if($sante == 1){
echo "CHECKED";}?>>Santé<td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Finances" <?if($finance == 1){ echo
"CHECKED";}?>>Finances<td>
<td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Economie" <?if($economie ==
1){ echo "CHECKED";}?>>Economie<td></tr>
<tr><td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Education" <?if($education
== 1){ echo "CHECKED";}?>>Education<td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Banques" <?if($banques == 1){ echo
"CHECKED";}?>>Banques
<td><INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Industrie" <?if($industrie == 1){
echo "CHECKED";}?>>Industrie<td></tr><td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Presse" <?if($presse == 1){ echo
"CHECKED";}?>>Presse<td>
<INPUT TYPE=CHECKBOX NAME="secteur_activite[]" VALUE="Autres" <?if($autres == 1){ echo
"CHECKED";}?>>Autres<td></tr>
</table><hr>

```

6-3 TRAITEMENT DES CHAMPS RÉPÉTITIFS :

À part ces deux types de champs, il y en a plusieurs autres types, dont l'exemple des champs répétitifs est le plus répandu. Il s'agit d'un champ qu'on le remplit autant de fois qu'on veut. Ces champs sont généralement présentés dans des interfaces à part, et leur traitement est un peu spécifique, car il se fait sur plusieurs étapes :

- afficher le formulaire de ces champs dans une nouvelle fenêtre ;
- remplir et envoyer à un script de traitement qui récupère les valeurs, en les insérant dans la base et en gardant les identifiants des champs dans des variables de session ;
- réafficher le formulaire pour remplir une autre fois
- Une fois terminé, on ferme la fenêtre et on revient au modèle pour finir la déclaration ;
- Une fois, la déclaration est finie, on envoie le formulaire principal au script qui récupère les variables de sessions gardées et les insère dans les tables de jonction (tables intermédiaires qui contiennent les identifiants des tables principales liées).


```

<p ALIGN="center"><a href="ListeOutilAssocie.php">Revenir </a>à la liste pour insérer <br>les outils
que vous avez déjà introduits</p>
<hr>
</table></BODY></HTML>

```

6-3-2 TRAITEMENT POUR L’AFFICHAGE DES OUTILS EXISTANTS DANS LA BASE

```

<?
// On initialise la session
session_start();
?>
<HTML>
<BODY bgcolor="#E6ECEE" link="#0066FF" vlink="#0066FF">
<h2 ALIGN="center">Liste des outils</h2>
<p ALIGN="center"><font size="3" color="blue">Si l'outil n'apparait pas dans la liste, veuillez l'<font
color="blue" size="4"><b><a href="AjoutOutilAssocie.html">ajouter</a></b></font></p>
<form name="ajout_util_associe" method="POST" action="ListeOutilAssocie2.php">
<table BGCOLOR="#669999" link="#0066FF" vlink="#FFCC99" ALIGN="center" border="1"
cellspacing="0" cellpadding="0">
<?php
// Appel des fichiers de connexion à la base
require ("connect.php");
require ("Connexion.php");
require ("ExecRequete.php");
$connexion = Connexion(NOM, PASSE, BASE, SERVEUR);
// Sélectionner la liste des outils associés dans la base
$requete = ("SELECT * FROM OUTILASSOCIE");
$resultat= ExecRequete ($requete, $connexion);
// Affichage de la liste
while ($modele = LigneSuivante ($resultat))
{
echo "<tr><td><b><INPUT TYPE='checkbox' name='id_util_associe[' value='$modele-
>ID_OUTILASSOCIE'>&nbsp;<ID_OUTILASSOCIE=$modele->ID_OUTILASSOCIE>$modele-
>ID_OUTILASSOCIE</b></td><td width='200'><b>$modele->NomOutil</td><td
width='200'><b>$modele->Version</td><td width='200'><b>$modele->Fonction</td></tr>";
}
?>
</table><br><br>
<center><input type="submit" value="Valider"></center>
</form><hr>
</body></HTML>

```

6-3-2 TRAITEMENT POUR LA RÉCUPÉRATION DES IDS

```

<?
// On initialise la session
session_start();
// RAZ de session_OutilAssocie
session_unregister('session_util_associe');
// On recupere le nombre d'outils associés
$nb_outils_associes = count($id_util_associe);
// On boucle sur le tableau pour créer la variable de session
for($temp=0;$temp<$nb_outils_associes;$temp++){
    $session_util_associe_temp .= $id_util_associe[$temp].";";
}
// Changement de variables
$session_util_associe = $session_util_associe_temp;
// On met en session session_util_associe
session_register('session_util_associe');
?>
<HTML><HEAD>

```

```

<TITLE>Ajout Outils Associés</TITLE>
<BODY bgcolor="#E6ECEE" link="#0066FF" vlink="#0066FF">
// Confirmation de l'envoi des outils choisis (parmi la liste)
<p ALIGN="center"><font size="3"><font color="blue">La liste des outils a bien &eacute;t&eacute;e;
envoy&eacute;e.</font><br><br>
// Rappel à l'utilisateur du prochain champ à remplir en revenant à l'interface principale du modèle
<p align="center"><font>Prochain champ à remplir est : <font
color="green"><b>URLS</b></font><br><br>
// Fermeture de la fenêtre et retour à l'interface précédente pour finir la déclaration du modèle
<A HREF="#" onClick="top.close()">Fermez</A> cette fenêtre et revenez au modèle pour terminer votre
déclaration.<br></p>
</BODY></HTML>

```

6-3-3 INSERTION DES IDS DANS LA TABLE INTERMÉDIAIRE OUTILASSOCIE_DM

```

// Insertion dans la table de jonction : OUTILASSOCIE_DM
// Test pour savoir si il y a des outils à insérer
if($session_outil_associe != ""){
// Récupération des ID_OutilsAssociés
$array = split(";", $session_outil_associe);
// Nombre d'éléments dans le tableau
$nb = count($array);
// Insertion dans la table OUTILASSOCIE_DM
for($i=0;$i<$nb;$i++){
// Pour supprimer les éléments vides
if($array[$i] != ""){
$requete1="INSERT INTO OUTILASSOCIE_DM (ID_OUTILASSOCIE, ID_DM)
values ($array[$i], $ID_DM)";
$resultat1=ExecRequete($requete1, $connexion);
}
}
session_unregister('session_outil_associe');
}

```

6-4 TRAITEMENT DES FICHIERS ATTACHÉS :

Dans nos deux modèles, nous avons deux champs (schéma XML et feuille de style), où on peut, à la place d'insérer du texte dans le formulaire, attacher un fichier, et ce pour une raison de contrainte liée au système de gestion de base de données Mysql, dont le champ texte est limité à 65535 caractères. Nous estimons que plusieurs schémas XML ou feuilles de style dépassent de loin cette taille.

Exemple :

Le fichier HTML de ce formulaire est le suivant :

```

<HTML>
<HEAD>
<TITLE>Fichier attaché</TITLE>
</HEAD>
<BODY bgcolor="#E6ECEE" link="#0066FF" vlink="#FFCC99">
<h2 ALIGN="center">Fichier attaché (Schéma XML ou DTD)</h2>

```

```

<form action="userfile.php" method="post" ENCTYPE="multipart/form-data">
<hr>
<table BGCOLOR="#669999" link="#0066FF" vlink="#FFCC99" ALIGN="center" border="0" cellspacing="0"
cellpadding="0">
<tr><td width="100" ALIGN="center">&nbsp;<br>&nbsp;<br>&nbsp;<br>&nbsp;<br>&nbsp;<b>Joindre :
</b></td><td>&nbsp;<br>&nbsp;<br>&nbsp;<br>&nbsp;<INPUT TYPE=FILE NAME="userfile"
size="35"></td></tr>&nbsp;<br>&nbsp;<br>&nbsp;<br>&nbsp;<br>&nbsp;<INPUT TYPE=SUBMIT><br>
</table>
<hr>
</form>
</body></html>

```

Script de traitement :

```

<?
// On initialise la session
session_start();
?>
<HTML>
<BODY bgcolor="#E6ECEE" link="#0066FF" vlink="#0066FF">
<p align="center"><b>Fichier soumis : </b></p><br><br>
<p align="center">Nom initial : <? echo($userfile_name); ?><br>
<p align="center">Taille : <? echo($userfile_size);
echo "<br><br><br>";
// Mise en session de URI
$session_attach = "upload/".$userfile_name;
session_register("session_attach");
// copier le fichier
if(copy($userfile, "upload/".$userfile_name))
{
echo("<br><center>Copie du fichier <font color='blue'>'$userfile_name'</font> réussie</center></b>");
}
else {
echo("<center><b>La copie du fichier a échoué ...</b></center>");
}
// supprimer le fichier après la copie
unlink($userfile);
?>
<br><br><p align="center"><p align="center"></font>Prochain champ à remplir est : <font
color="green"><b>Feuille de style</b></font><br><br>
<br><br><p align="center"><A HREF="#" onClick="top.close()">Fermez</A> cette fenêtre et revenez au modèle
pour terminer votre déclaration.<br></b></p>
</body>
</html>

```

7- TESTS DE VALIDATION DES FONCTIONNALITÉS DU SITE

Les tests de validation sont en fait effectués au fur et au mesure du développement du projet. Ces tests concernent chaque fonctionnalité créée et chaque lien établi, ainsi que chaque requête effectuée à destination de la base de donnée, à travers le navigateur. En outre, ces tests qui sont effectués individuellement, nous (Monsieur Langlois (Responsable du projet et moi même)) avons prévu de faire l'état des choses après chaque réalisation d'une grande fonctionnalité à savoir :

- la soumission des modèles
- la modification des modèles
- l'accès à la liste des modèles
- la recherche des modèles
- l'importation et l'exportation des core components
- la révision des modèles...

Enfin, nous avons organisé plusieurs réunions pour faire l'état d'avancement du projet et pour tester toutes les fonctionnalités offertes par le site dont la matrice qui suit essaie de résumer la description et l'environnement de ces tests, ainsi que les résultats obtenus.

FICHE DE TESTS DE VALIDATION				
<i>PROJET</i>	<i>TESTS DÉFINIS LE</i>	<i>TESTS DÉFINIS PAR</i>	<i>TESTS EFFECTUÉS LE</i>	<i>TESTS EFFECTUÉS PAR</i>
Création d'un répertoire de schémas XML	31/05/2002	Abderrazak MKADMI Marc Langlois Pierre Attar (Mutu-XML) Luth Martinez (GFII)	17 - 09 -2002	- Abderrazak MKADMI - Marc Langlois - Pierre Attar (Mutu-XML) - Luth Martinez (GFII)
	18/09/2002	Abderrazak MKADMI Jhon Lutz (UIC ²¹)	24 - 09 -2002	- Abderrazak MKADMI - Jhon Lutz (UIC)

<i>DESCRIPTION DU TEST</i>	<i>ENVIRONNEMENT DE TEST</i>	<i>CONDITIONS INITIALES</i>	<i>MOYENS DE VERIFICATION</i>
Tester toutes les fonctionnalités de l'outil développé	S.E : Linux (Mandrake 8.1) Navigateur : Mozilla	- le site est en accès local	- Navigateur Mozilla - PhpMyAdmin : outil d'administration de Mysql

²¹ UIC : Union Internationale de Chemins de fer (Jhon Lutz est chargé de mission : e-commerce)

N°	VERIFICATIONS A EFFECTUER	OPERATIONS A EFFECTUER	REMARQUES / Résultats	Validation
1	vérifier la cohérence de l'accès authentifié des propriétaires avec un login et un mot de passe, avec leurs identifiants dans la base.	introduire toutes les données du propriétaire, puis vérifier son accès par un login et un mot de passe	<ul style="list-style-type: none"> - Envoi des données à la base réussie - Accès par le login et le mot de passe réussi 	OK
		Vérifier l'identifiant dans la base de données par l'outil PhpMyAdmin	<ul style="list-style-type: none"> - Authentification et enregistrement dans la base réussie 	OK
2	Vérifier la soumission des modèles	<ul style="list-style-type: none"> - remplir le modèle simple et l'envoyer - retourner à la page sommaire - remplir un modèle core components et l'envoyer 	<ul style="list-style-type: none"> - Envoi et enregistrement des données dans la base réussis - Réception d'une confirmation d'enregistrement 	OK
3	Vérifier l'accès aux modèles	<ul style="list-style-type: none"> - cliquer sur la liste des modèles simples - cliquer su la liste des modèles core components 	<ul style="list-style-type: none"> - Affichage de la liste - Accès sans mot de passe 	OK
4	Vérifier l'importation des core components	<ul style="list-style-type: none"> - ouvrir le modèle core components - cliquer sur le lien "importer" - parcourir les disques pour choisir le fichier sous format CSV (le fichier sous format CSV est existant) - envoyer le fichier 	<ul style="list-style-type: none"> - Parcours et choix du fichier faciles - Enregistrement dans la base réussie - Réception d'une confirmation 	OK

5	Vérifier l'exportation des core components dans un fichier TXT	<ul style="list-style-type: none"> - Cliquer sur le lien d'exportation à partir de la page d'accueil - Choix parmi une liste des modèles à exporter - Saisie du répertoire, dans lequel, on va enregistrer le fichier (le nom du fichier est prédéfini) - valider 	<ul style="list-style-type: none"> - Affichage de la liste des core components réussi - Enregistrement dans le fichier réussi - Réception d'une confirmation, avec le nom du fichier et le chemin d'accès 	OK
6	Vérifier la modification des modèles	Accès sans mot de passe	- Accès impossible	OK
		<ul style="list-style-type: none"> - Accès avec mot de passe - Choix du type de modèles (simple ou Core components) - Affichage de la liste de modèles dont il est le propriétaire - Choix du modèle à modifier - Modification et validation du modèle 	<ul style="list-style-type: none"> - Affichage de la liste de tous les modèles qui lui appartiennent (seulement) réussi - Modification et envoi des données à la base réussis - Réception d'une confirmation 	OK
7	Vérifier la recherche des modèles	<ul style="list-style-type: none"> - Recherche par : <li style="padding-left: 20px;">Secteur d'activité ou <li style="padding-left: 20px;">Domaine d'usage ou <li style="padding-left: 20px;">Mot clé ou <li style="padding-left: 20px;">Rôle d'affaires ou <li style="padding-left: 20px;">Classification du produit ou <li style="padding-left: 20px;">Nom du modèle 	- Accès à la liste des modèles correspondante au critère choisi réussi	OK

		- Combinaison entre ces différents critères par OU	- Accès à la liste des modèles correspondante au critère choisi réussi	OK
		- Combinaison entre ces différents critères par ET	- Accès à la liste des modèles correspondante au critère choisi réussi	OK
		- Combinaison entre deux termes du même critère	- Accès impossible	NON
		- Recherche avec un moteur de recherche	- Fonctionnalité non disponible	NON

CONCLUSION GÉNÉRALE


Le projet «création d'un répertoire de schémas XML», tel qu'il a été présenté, représente un pas exceptionnel dans le domaine de l'EDI. Il était depuis un bon moment, (en tous cas, depuis le développement et l'introduction du métalangage XML dans le domaine du commerce électronique et dans l'administration française) le souci de tous les organismes qui travaillent dans le domaine de la normalisation et la standardisation. En effet, EDIFRANCE, Mutu-XML, GFII et la FING (Partenaires du projet), ainsi que l'UIC (Union Internationale de Chemins de Fer) se trouvent déjà avec un outil qui leur permettent de publier leurs modèles XML et leurs core components, et d'en profiter de l'existence de différents modèles liés à différentes activités. Ceci évitera certainement des travaux redondants d'une part, et encouragera d'autres à utiliser la technologie XML, tout en s'inspirant des modèles déjà publiés. Il devient alors possible de recevoir n'importe quel document issu d'un modèle particulier, de l'exploiter avec des applications génériques, ou encore de le visualiser sur un navigateur Web standard.

Quant à la réalisation du projet, elle était un vrai exercice pour comprendre à la fois la technologie XML et l'échange de données informatisé, la technologie web à savoir les langages HTML, PHP, SQL, ainsi que les méthodes de modélisation des projets (langage UML). La partie la plus difficile et celle qui a pris le plus de temps était celle de la conception, notamment la modélisation et la définition des modèles de données.

1- SUITE DU PROJET

- Installer un moteur de recherche plus puissant qui permet une recherche full-texte ;
- Possibilité d'exportation des données des modèles en format XML ;
- Créer une version du site en anglais ;

2- INTÉRÊTS DU PROJET

- Stockage des schémas XML de chaque organisme de façon organisée, et qui permet de les retrouver facilement ;
 - Découverte de tous les schémas XML existants et dans tous les domaines ;
 - Réutilisation des schémas déjà utilisés et déjà prêts, et donc gain du temps ;
- 
 - ***faciliter l'échange de données informatisé entre professionnels ;***
 - ***accélérer et élargir l'intégration B2B et le commerce sur Internet" .***

3- CONNAISSANCES ACQUISES LORS DU STAGE

- Pratiquer réellement la modélisation avec le langage UML ;
- Utiliser des logiciels et des outils qui permettent la modélisation avec ce langage : Objecteering ;
- Créer des documents XML, des schémas XML et des feuilles de styles ;
- Utiliser des éditeurs et des concepteurs de feuilles de style : XMLSPY ;

- Conduire un projet du début à la fin (Compréhension du domaine, préparation du cahier des charges et des spécifications techniques, et enfin réalisation du projet) ;
- Bien comprendre le mode de fonctionnement du langage PHP avec Mysql ...
- Se familiariser avec des concepts tels que : EDI, XML schémas, ebXML...

RÉFÉRENCES BIBLIOGRAPHIQUES

- ATICA.** - Le répertoire des schémas XML de l'administration.-
<http://www.atica.pm.gouv.fr/XML/repertoire.shtml>
- ATTAR, Pierre.** - Documentation Technique du modèle permettant l'échange de descriptions de *modèles*. - Version 0.1, 1^{er} Janvier 2002
- CASTAGNETTO, Jesus ; RAWAT, Harish ; SCHUMANN, Sascha .** - PHP Professionnel, avec 4 études de cas détaillées. - France, Eyrolles, 2000
- EDIFRANCE, GFII, FING, MUTU-XML.** - Un répertoire de *modèles* pour les applications d'XML. - <http://www.repertoire-modeles.org/defProj/decl/defProj.html>
- GENCOD EAN & Edifrance.** - Le commerce électronique pour l'entreprise : guide de mise en œuvre des échanges électroniques professionnels. - France, janvier 2002
- GENCOD EAN & Edifrance.-** comprendre XML pour les échanges électroniques professionnels (cd-rom). - France, mai 2002
- CHAUVET, Jean-Marie.** - Services Web avec SOAP, WSDL, UDDI, ebXML. - Paris : Eyrolles, 2002
- HAROLD, Elliotte Rusty.** - XML : Le guide de l'utilisateur. - France : O'reilly, 1999
- L'Altruiste : Le guide des langages du Web.** - <http://www.laltruiste.com/coursschema/sommaire.html>
- OASIS.** - OASIS Registry/Repository Technical Specification, Working Draft 1.1 December 20, 2000.
- LOPES, Frédéric.** - Tout savoir sur XML. - <http://www.xmltechno.com/tutoriels/>
- MKADMI, Abderrazak.** - Plan qualité du projet, document validé par Marc Langlois. - EDIFRANCE, avril 2002
- MKADMI, Abderrazak ; LANGLOIS, Marc.** - Cahier des charges du projet : création d'un répertoire de schémas XML. - <http://www.edifrance.org>
- PARENT, Richard.** - Normes ouvertes en technologies de l'information. - traduction de Richard Parent. - *UN/CEFACT . - Core Components Technical Specifications, Part 1*//Spécification technique des composants élémentaires, Partie 1. - traduit par Richard Parent. - <http://www.autoroute.gouv.qc.ca/publica/normes/liste.htm>
- RIGAUX, Philippe.** - Pratique de MySQL et PHP. - France : O'reilly, 2000
- VAN DER VLIST, Eric.-** Le répertoire de schémas XML de la MTIC ouvre ses portes. - <http://xmlfr.org/actualites/decid/010413-0002>

ANNEXES

ANNEXE 1 : PLAN QUALITÉ

1. Introduction

1.1. Résumé

Le projet consiste à réaliser un répertoire des schémas XML dans le cadre du stage effectué par Abderrazak MKADMI (mastère « Systèmes Informatiques Ouverts » à l'école Centrale Paris) chez EDIFRANCE (Association pour le développement des échanges électroniques professionnels), et sous la responsabilité de Monsieur Marc LANGLOIS (Délégué général d'EDIFRANCE)

Le but est de mettre en place un répertoire de modèles pour les différentes applications XML sous forme d'une base de données qui a pour objectif de permettre à « tout un chacun de prendre connaissance des modèles qui existent dans un domaine d'activité particulier, pour un besoin particulier », de connaître « les modèles qui lui permettent d'échanger avec ses partenaires et les pratiques habituelles dans son domaine d'activité²² ».

1.2. Produits du projet

- un répertoire de modèles XML, avec une base documentaire associée ;
- un site web sous forme de formulaires d'accès et de consultation de la base ;
- une documentation pour la maintenance.

2. Organisation du projet

2.1. Structure de l'organisation

Maître d'ouvrage : M. LANGLOIS

Maître d'œuvre : Abderrazak MKADMI

2.2 Planning de Travail

Entre le maître d'ouvrage et le maître d'œuvre : Réunions hebdomadaires Le but est d'établir et de modifier tout avancement du projet (Cahier Des Charges, Spécifications Techniques des Besoins, réalisation...).

2.3. Documents à produire

Documents à produire	Dates
• Cahier Des Charges Fonctionnel	20 mai 2002
• Choix de l'environnement de développement et d'exploitation	30 mai 2002
• Plan Test	15 juin 2002
• Conception et réalisation	15 juin-15 août 2002
• Fiches test	15 août – 15 septembre

3. Gestion du projet

3.1. Objectifs et priorités

Notre objectif principal, comme il est défini ci-dessus, est de mettre en place une base de données de schémas XML accessible a priori en local en premier lieu. Pour le faire, on s'est fixé des sous objectifs qu'on peut les définir comme suit :

²² GFII, FING XML, MUTU-XML, EDIFRANCE : Un répertoire de modèles pour les applications d'XML. – <http://www.repertoire-modeles.org/defProj/decl/defProj.html>

- Mener la partie gestion et conception du projet du 15 avril au 15 juin 2002 (Plan Qualité, Cahier Des Charges Fonctionnel, Spécifications Techniques des Besoins) ;
- Réaliser et rendre fonctionnelle la base de données : 15 juin au 15 août 2002 ;
- Faire des tests pendant la période du 15 août au 15 septembre ;
- Préparer une documentation pour la maintenance du site.

3.2. Hypothèses et contraintes

Plusieurs contraintes peuvent intervenir et peser sur notre projet (disponibilité des personnes à interviewer, l'ampleur de la base de données, etc.)

Le succès de notre projet repose sur le respect de notre calendrier de travail et de réunions.

3.3. Planning

Phase 1 : L'établissement du plan qualité

(réalisé du 15 avril au 21 avril 2002)

Ce plan qualité sera notre guide d'avancement du projet. Il définit les objectifs, les responsabilités, les tâches et planning du travail. Il est réalisé par moi-même et coordonné par le responsable du projet.

Phase 2 : L'établissement du Cahier des charges Fonctionnel : CDCF

(réalisé du 21 avril au 20 mai 2002)

Le Cahier Des Charges Fonctionnel est le document qui définit les objectifs et les contraintes du projet et dans quelles conditions il sera mené à bien. Il fournit des renseignements détaillés sur nos besoins (ce que nous allons faire exactement) et les différents aspects techniques. Il nous servira comme un document de référence pour proposer des solutions.

Phase 3 : Choix de l'environnement de développement et d'exploitation

(réalisé du 20 mai au 30 mai 2002)

Le choix de l'environnement de développement et d'exploitation sera déterminé selon les possibilités existantes et surtout selon les spécifications des tâches à effectuer, tout en prenant en compte les aspects fiabilité, accessibilité et robustesse des outils, et bien sûr l'ampleur du projet. Il sera fait en bref pour répondre au mieux aux besoins qui ont été définis dans le Cahier Des Charges Fonctionnel.

Phase 4 : Plan test

(réalisé du 30 mai au 15 juin 2002)

Le plan test définit les différents tests effectués du début de la réalisation jusqu'à la fin (fonctionnement du projet). On peut synthétiser les différents tests comme suit :

- Les tests effectués au fur et à mesure de la réalisation et en phase finale : contrôle des diverses fonctionnalités ;
- Validation du produit en phase finale de développement par une équipe d'expertise du produit.
- Test de fonctionnalité : le système fait-il ce qu'il est censé faire ? Le test de fonctionnalité permet de détecter les erreurs et les oublis lors de la réalisation du produit.

Phase 5 : Réalisation

- Réalisation du projet : du 15 juin au 15 août 2002

Phase 6 : Tests

- Réalisation des tests définis dans le Plan Test du 15 août jusqu'au 15 septembre

Phase 7 : Documentation

- Préparation d'une documentation tout au long du projet pour servir d'un outil de maintenance du site à développer. Sa finalisation sera effectuée du 15 septembre au 30 septembre

ANNEXE 2 : GLOSSAIRE

B2B (B to B)	(<i>Anglais</i> : Business to Business) : qualifie un logiciel ou un site web de professionnel à professionnel.
B2C (B to C)	(<i>Anglais</i> : Business to Consumer) : qualifie un logiciel ou un site web destiné au grand public
DOM	(<i>Anglais</i> : Document Object Model) : Spécification permettant aux logiciels clients certains traitements sur les documents consultés.
DTD	(<i>Anglais</i> : Document Type Definition) : une DTD contient les spécifications formelles de structuration de documents au format XML.
EDI	(<i>Anglais</i> : Electronic Data Interchange) : Échange de Données Informatisé. Terme généralisant la pratique de la transmission et de l'échange de données informatiques par des connexions point à point basées sur des formes, des messages et des données standards.
E-commerce	(<i>Anglais</i> : Electronic commerce). Le terme de commerce électronique englobe tous les produits et services qui sont vendus sur internet. La commande peut s'effectuer par paiement sécurisé, par téléphone, fax ou courrier. Le terme « Electronic Business » est souvent utilisé.
Élément	Un élément est une unité de structuration de document déclarée dans un schéma de données.
FAQ	(<i>Anglais</i> : Frequently Asked Questions) : Regroupements de questions posées fréquemment sur un sujet, constituant généralement les bases minimales de la connaissance à avoir sur ce sujet. On trouve également « Foire Aux Questions »
Feuille de style	Une feuille de style détermine l'apparence et la mise en forme d'éléments dans un ensemble de documents qui la référencent. Les documents XML ne contiennent en principe aucune information de présentation des données à l'affichage. Un document XML peut spécifier une feuille de style pour l'affichage de ses données, dans un navigateur par exemple.
Instance	Production d'un document selon un schéma de données.
Méta données	Données décrivant des données. Un nom d'élément ou de champ, par exemple.
Schéma de données	spécification formelle de la structure d'un document sous forme d'arbre. Les spécifications peuvent être exprimées au moyen d'une DTD ou d'un X-schéma
Schéma XML(ou X-Schéma)	langage de description de schéma de données exprimé en XML (recommandations X-schéma du W3C) remplaçant le formalisme des DTDs. Ce langage permet d'inclure des contrôles de syntaxe et de type dans les schémas de données.
SVG	(<i>Anglais</i> <i>Scalable Vector Graphics</i>) : langage de représentation graphique en 2 dimensions exprimé en XML.
Validation	D'un point de vue formel, vérifier qu'un document est conforme à la structure et à la syntaxe définie dans un schéma de données.
XML	(<i>Anglais</i> <i>eXtensible Markup Language</i>) : Langage de marquage de documents.
XSL	(<i>Anglais</i> : <i>eXtended Stylesheet Language</i>) Langage utilisé pour écrire des feuilles de style permettant de mettre en forme ou de modifier des données dans les documents XML.
XSLT	(<i>Anglais</i> : <i>eXtended Stylesheet Language Transformations</i>) : Élément du langage de mise en forme XSL permettant d'effectuer des transformations de données. Utiliser une feuille de style XSLT par exemple pour convertir des données XML en HTML.
XQL	(<i>Anglais</i> : <i>eXtended Query Language</i>) : Langage de recherche dans une arborescence XML. Il reconnaît la structure du document, et décrit ses propres critères de recherche : sous-éléments à rechercher dans un élément, par exemple.

ANNEXE 3 : CRÉATION DES TABLES DANS LA BASE DE DONNÉES

```

##      FICHER DE CREATION DE LA BASE : MODELESXML.SQL
##      AUTEUR : ABDERRAZAK MKADMI

##### CONNEXION A LA BASE #####

USE MODELESXML;

DROP TABLE IF EXISTS DECLARATIONMODELE;
DROP TABLE IF EXISTS DECLARATIONALTERNATIVE;
DROP TABLE IF EXISTS METAMODELE;
DROP TABLE IF EXISTS DOMAINEUSAGE;
DROP TABLE IF EXISTS DOMAINEUSAGE_DM;
DROP TABLE IF EXISTS RECOMMANDATIONW3C;
DROP TABLE IF EXISTS RECOMMANDATIONW3C_DM;
DROP TABLE IF EXISTS MODERESTITUTION;
DROP TABLE IF EXISTS MODERESTITUTION_DM;
DROP TABLE IF EXISTS TYPEDONNEE;
DROP TABLE IF EXISTS TYPEDONNEE_DM;
DROP TABLE IF EXISTS REVUE;
DROP TABLE IF EXISTS REVUE_DM;
DROP TABLE IF EXISTS CONTACT;
DROP TABLE IF EXISTS ORGANISATION;
DROP TABLE IF EXISTS CONTACT_DM;
DROP TABLE IF EXISTS PROJET;
DROP TABLE IF EXISTS PROCESSUS;
DROP TABLE IF EXISTS PROCESSUS_DM;
DROP TABLE IF EXISTS ROLE;
DROP TABLE IF EXISTS ROLE_DM;
DROP TABLE IF EXISTS SECTEURACTIVITE;
DROP TABLE IF EXISTS SECTEURACTIVITE_DM;
DROP TABLE IF EXISTS CLASSIFICATIONPRODUIT;
DROP TABLE IF EXISTS CLASSIFICATIONPRODUIT_DM;
DROP TABLE IF EXISTS URLS;
DROP TABLE IF EXISTS URLS_DM;
DROP TABLE IF EXISTS PROPRIETEMODELE;
DROP TABLE IF EXISTS OUTILASSOCIE;
DROP TABLE IF EXISTS OUTILASSOCIE_DM;
DROP TABLE IF EXISTS MODELELIE;
DROP TABLE IF EXISTS MODELELIE_DM;
DROP TABLE IF EXISTS UTILISATEURS;
DROP TABLE IF EXISTS UTILISATEURS_DM;
DROP TABLE IF EXISTS COMPOSANTREUTILISE;
DROP TABLE IF EXISTS FEUILLEDESTYLE;
DROP TABLE IF EXISTS CHAMPAPPLICATION;
DROP TABLE IF EXISTS CHAMPAPPLICATION_DM;
DROP TABLE IF EXISTS REMARQUES;
DROP TABLE IF EXISTS DEFINITIONCC;
DROP TABLE IF EXISTS NOMBUSINESS;
DROP TABLE IF EXISTS DOCUMENTLIE;
DROP TABLE IF EXISTS VALEURCODE_DM;
DROP TABLE IF EXISTS VALEURCODE;

##### CRÉATION DES TABLES #####

CREATE TABLE DECLARATIONMODELE (ID_DM BIGINT(100) NOT NULL AUTO_INCREMENT,
DesignationModele VARCHAR(255) NOT NULL, Version VARCHAR(255) NOT NULL, DateVersion

```

DATE NOT NULL, CommentaireVersion TEXT, DateDebut DATE NOT NULL, DateFin DATE, MotClé TEXT, TypeDeclaration ENUM ('ModèleSimple', 'ModèleCC', 'ModèleSophistiqué'), PrincipeModele_texte text, PrincipeModele_URI VARCHAR(255), ObjectifModele_texte text, ObjectifModele_URI VARCHAR(255), ResumeModele_texte text, ResumeModele_URI VARCHAR(255), FonctionnalitesModele_texte text, FonctionnalitesModele_URI VARCHAR(50), AutresInfoModele_texte text, AutresInfoModele_URI VARCHAR(255), TypeCC VARCHAR(255), QualifiantClasseObjet VARCHAR(255), NomClasseOrigine VARCHAR(255), QualifiantClasseProprietaire VARCHAR(255), NomClasseProprietaireOrigine VARCHAR(255), TermeDePresentation VARCHAR(255), NomCodeListe VARCHAR(255) NOT NULL, URICodeListe VARCHAR(255) NOT NULL, SchemaXML TEXT, URISchemaXML VARCHAR(255), ContrainteLegale TEXT, Geopolitique Text, ID_Proprietaire BIGINT NOT NULL, ID_Organisation BIGINT(50) NOT NULL, ID_MetaModele BIGINT(50) NOT NULL, ID_Projet BIGINT(50) NOT NULL, PRIMARY KEY (ID_DM));

CREATE TABLE **VALEURCODE** (ID_ValeurCode BIGINT(50) NOT NULL AUTO_INCREMENT, ValeurCode VARCHAR(255), PRIMARY KEY (ID_ValeurCode));

CREATE TABLE **VALEURCODE_DM** (ID_ValeurCode BIGINT(50), ID_DM BIGINT(100));

CREATE TABLE **FEUILLEDESTYLE** (ID_FeuilleDeStyle BIGINT(50) NOT NULL AUTO_INCREMENT, FeuilleDeStyle_Texte TEXT, FeuilleDeStyle_URI VARCHAR(255), ID_DM BIGINT(100), PRIMARY KEY (ID_FeuilleDeStyle), FOREIGN KEY (ID_DM) REFERENCES DECLARATIONMODELE);

CREATE TABLE **CHAMPAPPLICATION** (ID_Application BIGINT(50) NOT NULL AUTO_INCREMENT, Langue ENUM('Français', 'Anglais', 'Espagnol', 'Italien', 'Autres'), Application_Texte TEXT, Application_URI VARCHAR(100), PRIMARY KEY (ID_Application));

CREATE TABLE **CHAMPAPPLICATION_DM** (ID_Application BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL, PRIMARY KEY (ID_Application, ID_DM));

CREATE TABLE **COMPOSANTREUTILISE** (ID_ComposantReutilise BIGINT(50) NOT NULL AUTO_INCREMENT, ID_DM BIGINT(100), PRIMARY KEY (ID_ComposantReutilise));

CREATE TABLE **PROPRIETEMODELE** (ID_Proprietaire BIGINT(50) NOT NULL AUTO_INCREMENT, NomOwner VARCHAR(35) NOT NULL, PrenomOwner VARCHAR(35) NOT NULL, FonctionOwner VARCHAR(35) NOT NULL, Login VARCHAR(35) NOT NULL, MotDePasse VARCHAR(20) NOT NULL, Adresse1 VARCHAR(35), Adresse2 VARCHAR(35), CodePostal VARCHAR(9) NOT NULL, Ville VARCHAR(35) NOT NULL, Région VARCHAR(35), Pays ENUM ('France', 'Allemagne', 'Belgique', 'Brésil', 'Bulgarie', 'Canada', 'Chine', 'Colombie', 'Corée du Nord', 'Corée du Sud', 'Danemark', 'Egypte', 'Espagne', 'États-Unis', 'Finlande', 'Algérie', 'Grèce', 'Inde', 'Indonésie', 'Irlande', 'Islande', 'Italie', 'Japon', 'Maroc', 'Norvège', 'Nouvelle-Zélande', 'Pays-Bas', 'Pérou', 'Pologne', 'Portugal', 'Roumanie', 'Royaume-Uni', 'Russie', 'Salvador', 'Suède', 'Suisse', 'Tunisie', 'Turquie', 'Uruguay'), Tél VARCHAR(20) NOT NULL, Fax VARCHAR(20), Mail VARCHAR(250) NOT NULL, Web VARCHAR(250), PRIMARY KEY (ID_Proprietaire));

CREATE TABLE **MODELELIE** (ID_ML BIGINT(50) NOT NULL AUTO_INCREMENT, URI VARCHAR(255), TypeLiaison VARCHAR(255), PRIMARY KEY (ID_ML));

CREATE TABLE **MODELELIE_DM** (ID_DM BIGINT(100) NOT NULL, ID_ML BIGINT(50) NOT NULL);

CREATE TABLE **DOCUMENTLIE** (ID_DocumentLie BIGINT(50) NOT NULL AUTO_INCREMENT, DocumentLie_Texte TEXT, DocumentLie_URI VARCHAR(255), ID_DM BIGINT(100), PRIMARY KEY (ID_DocumentLie), FOREIGN KEY (ID_DM) REFERENCES DECLARATIONMODELE);

CREATE TABLE **REMARQUES** (ID_Remarques BIGINT(50) NOT NULL AUTO_INCREMENT, Remarques TEXT, Langue ENUM('Français', 'Anglais', 'Espagnol', 'Italien', 'Autres'), ID_DM BIGINT(100), PRIMARY KEY (ID_Remarques), FOREIGN KEY (ID_DM) REFERENCES DECLARATIONMODELE);

CREATE TABLE **DEFINITIONCC** (ID_Definition BIGINT(50) NOT NULL AUTO_INCREMENT, Definition TEXT, Langue ENUM('Français', 'Anglais', 'Espagnol', 'Italien', 'Autres'), ID_DM BIGINT(100), PRIMARY KEY (ID_Definition), FOREIGN KEY (ID_DM) REFERENCES DECLARATIONMODELE);

```
CREATE TABLE NOMBUSINESS (NomBusiness VARCHAR(100) NOT NULL, Langue ENUM ('Français', 'Anglais','Espagnol', 'Italien', 'Autres'), ID_DM BIGINT(100));
```

```
CREATE TABLE OUTILASSOCIE (ID_OUTILASSOCIE BIGINT(50) NOT NULL AUTO_INCREMENT, NomOutil VARCHAR(35), Version VARCHAR (35), Fonction VARCHAR(35), PRIMARY KEY (ID_OUTILASSOCIE));
```

```
CREATE TABLE OUTILASSOCIE_DM (ID_OUTILASSOCIE BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE DECLARATIONALTERNATIVE (ID_DA BIGINT(50) NOT NULL AUTO_INCREMENT, TypeDeclaration VARCHAR(100) NOT NULL, Description TEXT, ID_DM BIGINT(100), PRIMARY KEY(ID_DA), FOREIGN KEY(ID_DM) REFERENCES DECLARATIONMODELE);
```

```
CREATE TABLE METAMODELE (ID_MetaModele BIGINT(50) NOT NULL AUTO_INCREMENT, Titre VARCHAR(255) NOT NULL, Version VARCHAR(50) NOT NULL, DateVersion DATE NOT NULL, LienURI VARCHAR(50), Texte TEXT, PRIMARY KEY (ID_MetaModele));
```

```
CREATE TABLE TYPEDONNEE (ID_TypeDonnee BIGINT(50) NOT NULL AUTO_INCREMENT, TypeDonnee ENUM ('Données Texte Structurées', 'Données Texte Non Structurées', 'Données Multimédia Structurées', 'Données Multimédia Non Structurées', 'Données Mixtes Structurées', 'Données Mixtes Non Structurées') NOT NULL, PRIMARY KEY (ID_TypeDonnee));
```

```
CREATE TABLE TYPEDONNEE_DM(ID_TypeDonnee BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE RECOMMANDATIONW3C (ID_Recommandation BIGINT(50) NOT NULL AUTO_INCREMENT, Recommendation ENUM ('SCHEMA XML', 'DTD', 'XPATH', 'XSLT', 'XLINK', 'XSLFO', 'AUTRES') NOT NULL, PRIMARY KEY (ID_Recommandation));
```

```
CREATE TABLE RECOMMANDATIONW3C_DM (ID_Recommandation BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE MODERESTITUTION (ID_ModeRestitution BIGINT(50) NOT NULL AUTO_INCREMENT, ModeRestitution ENUM('EDITION PAPIER', 'HTML', 'WAP', 'SGML', 'XML', 'TELETEL', 'AUTRES') NOT NULL, PRIMARY KEY (ID_ModeRestitution));
```

```
CREATE TABLE MODERESTITUTION_DM (ID_ModeRestitution BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE DOMAINEUSAGE (ID_DomaineUsage BIGINT(50) NOT NULL AUTO_INCREMENT, DomaineUsage ENUM ('Commerce électronique', 'Procédure administrative', 'Catalogue', 'Production Information', 'Gestion Information', 'Edition Information', 'Partage Information', 'Echange Information', 'Archivage Information') NOT NULL, PRIMARY KEY (ID_DomaineUsage));
```

```
CREATE TABLE DOMAINEUSAGE_DM (ID_DomaineUsage BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE CLASSIFICATIONPRODUIT (ID_Classification BIGINT(50) NOT NULL AUTO_INCREMENT, GestionnaireListe VARCHAR(100) NOT NULL, CodeDansListe VARCHAR(50) NOT NULL, DesignationClassification TEXT, PRIMARY KEY (ID_Classification));
```

```
CREATE TABLE CLASSIFICATIONPRODUIT_DM (ID_Classification BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE SECTEURACTIVITE (ID_SecteurActivite BIGINT(50) NOT NULL
AUTO_INCREMENT, Secteur ENUM ('Santé', 'Finances', 'Economie', 'Education', 'Banques', 'Industrie',
'Presse', 'Autres') NOT NULL, PRIMARY KEY (ID_SecteurActivite));
```

```
CREATE TABLE SECTEURACTIVITE_DM(ID_SecteurActivite BIGINT(50), ID_DM BIGINT(100));
```

```
CREATE TABLE PROCESSUS (ID_Processus BIGINT(50) NOT NULL AUTO_INCREMENT, Processus
VARCHAR(255) NOT NULL, PRIMARY KEY (ID_Processus));
```

```
CREATE TABLE PROCESSUS_DM (ID_Processus BIGINT(50) NOT NULL, ID_DM INT(255) NOT
NULL);
```

```
CREATE TABLE ROLE (ID_Role BIGINT(50) NOT NULL AUTO_INCREMENT, Role ENUM ('Client',
'Fournisseur', 'Transporteur', 'Distributeur', 'Autres') NOT NULL, DesignationRole TEXT, PRIMARY KEY
(ID_Role));
```

```
CREATE TABLE ROLE_DM(ID_Role BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE CONTACT (ID_Contact BIGINT(50) NOT NULL AUTO_INCREMENT, NomContact
VARCHAR(35) NOT NULL, PrenomContact VARCHAR(35), FonctionContact VARCHAR(50), MailContact
VARCHAR(250), TelContact VARCHAR(20), FaxContact VARCHAR(20), Adresse1Contact
VARCHAR(35), Adresse2Contact VARCHAR(35), CodePostalContact VARCHAR(9), VilleContact
VARCHAR(35), RegionContact VARCHAR(35), PaysContact ENUM ('France', 'Allemagne', 'Belgique',
'Brésil', 'Bulgarie', 'Canada', 'Chine', 'Colombie', 'Corée du Nord', 'Corée du Sud', 'Danemark', 'Egypte', 'Espagne',
'États-Unis', 'Finlande', 'Algérie', 'Grèce', 'Inde', 'Indonésie', 'Irlande', 'Islande', 'Italie', 'Japon', 'Maroc', 'Norvège',
'Nouvelle-Zélande', 'Pays-Bas', 'Pérou', 'Pologne', 'Portugal', 'Roumanie', 'Royaume-Uni', 'Russie', 'Salvador',
'Suède', 'Suisse', 'Tunisie', 'Turquie', 'Uruguay'), ID_Organisation BIGINT(50), PRIMARY KEY
(ID_Contact));
```

```
CREATE TABLE CONTACT_DM(ID_Contact BIGINT(50) NOT NULL, ID_DM BIGINT(100));
```

```
CREATE TABLE ORGANISATION (ID_Organisation BIGINT(50) NOT NULL AUTO_INCREMENT,
NomOrganisation VARCHAR(35) NOT NULL, Acronyme VARCHAR(35), Tel VARCHAR(20), Mail
VARCHAR(50), Fax VARCHAR(20), Web VARCHAR(255), Adresse1 VARCHAR(35), Adresse2
VARCHAR(35), CodePostal VARCHAR(9), Ville VARCHAR(35), Region VARCHAR(35), Pays ENUM
('France', 'Allemagne', 'Belgique', 'Brésil', 'Bulgarie', 'Canada', 'Chine', 'Colombie', 'Corée du Nord', 'Corée du
Sud', 'Danemark', 'Egypte', 'Espagne', 'États-Unis', 'Finlande', 'Algérie', 'Grèce', 'Inde', 'Indonésie', 'Irlande',
'Islande', 'Italie', 'Japon', 'Maroc', 'Norvège', 'Nouvelle-Zélande', 'Pays-Bas', 'Pérou', 'Pologne', 'Portugal',
'Roumanie', 'Royaume-Uni', 'Russie', 'Salvador', 'Suède', 'Suisse', 'Tunisie', 'Turquie', 'Uruguay'), PRIMARY
KEY (ID_Organisation));
```

```
CREATE TABLE PROJET (ID_Projet BIGINT(50) NOT NULL AUTO_INCREMENT, NomProjet
VARCHAR(255), EspaceNommege VARCHAR(255) NOT NULL, Prefixe VARCHAR(255) NOT NULL,
ID_Organisation BIGINT(50) NOT NULL, PRIMARY KEY (ID_Projet));
```

```
CREATE TABLE URLS (ID_URL BIGINT(50) NOT NULL AUTO_INCREMENT, CheminAbsolu
VARCHAR (255), CheminRelatif VARCHAR(255), NomFichier VARCHAR(255), PRIMARY
KEY(ID_URL));
```

```
CREATE TABLE URLS_DM(ID_URL BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE REVUE (ID_Revue BIGINT(50) NOT NULL AUTO_INCREMENT, NomReviseur
VARCHAR(255) NOT NULL, PrenomReviseur VARCHAR (255) NOT NULL, LoginReviseur
VARCHAR(255), MotDePasseReviseur VARCHAR (255), StatutRevision ENUM ('Soumis', 'En Révision',
'Révisé') NOT NULL, DateRevision DATE, Commentaire TEXT, PRIMARY KEY(ID_Revue));
```

```
CREATE TABLE REVUE_DM(ID_Revue BIGINT(50) NOT NULL, ID_DM BIGINT(100) NOT NULL);
```

```
CREATE TABLE UTILISATEURS (ID_USER BIGINT(50) NOT NULL AUTO_INCREMENT, NomUser
VARCHAR(35) NOT NULL, PrenomUser VARCHAR(35) NOT NULL, Adresse1User VARCHAR(35),
```

```
Adresse2User VARCHAR(35), CodePostalUser VARCHAR(9), VilleUser VARCHAR(35), RegionUser
VARCHAR(35), PaysUser ENUM ('France', 'Allemagne', 'Belgique', 'Brésil', 'Bulgarie', 'Canada', 'Chine',
'Colombie', 'Corée du Nord', 'Corée du Sud', 'Danemark', 'Egypte', 'Espagne', 'États-Unis', 'Finlande', 'Algérie',
'Grèce', 'Inde', 'Indonésie', 'Irlande', 'Islande', 'Italie', 'Japon', 'Maroc', 'Norvège', 'Nouvelle-Zélande', 'Pays-Bas',
'Pérou', 'Pologne', 'Portugal', 'Roumanie', 'Royaume-Uni', 'Russie', 'Salvador', 'Suède', 'Suisse', 'Tunisie',
'Turquie', 'Uruguay), TelUser VARCHAR(20), FaxUser VARCHAR(20), MailUser VARCHAR(250) NOT
NULL, WebUser VARCHAR(250), SocieteUser VARCHAR(100), PRIMARY KEY (ID_USER));
```

```
CREATE TABLE UTILISATEURS_DM (ID_DM BIGINT(100) NOT NULL, ID_USER BIGINT(50) NOT
NULL);
```

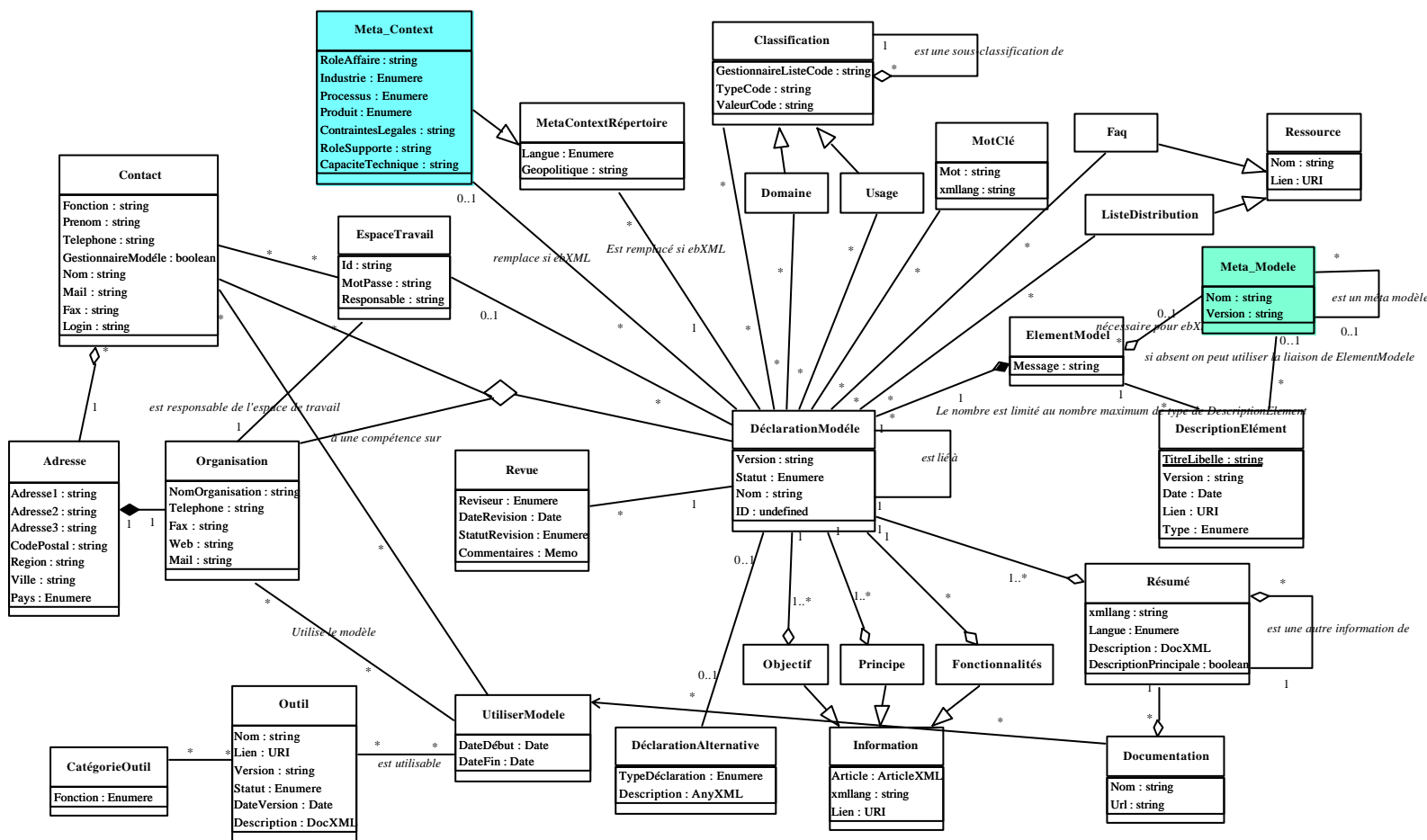
```
##### QUITTER MYSQL #####
```

```
EXIT
```

ANNEXE 4 : DIAGRAMMES

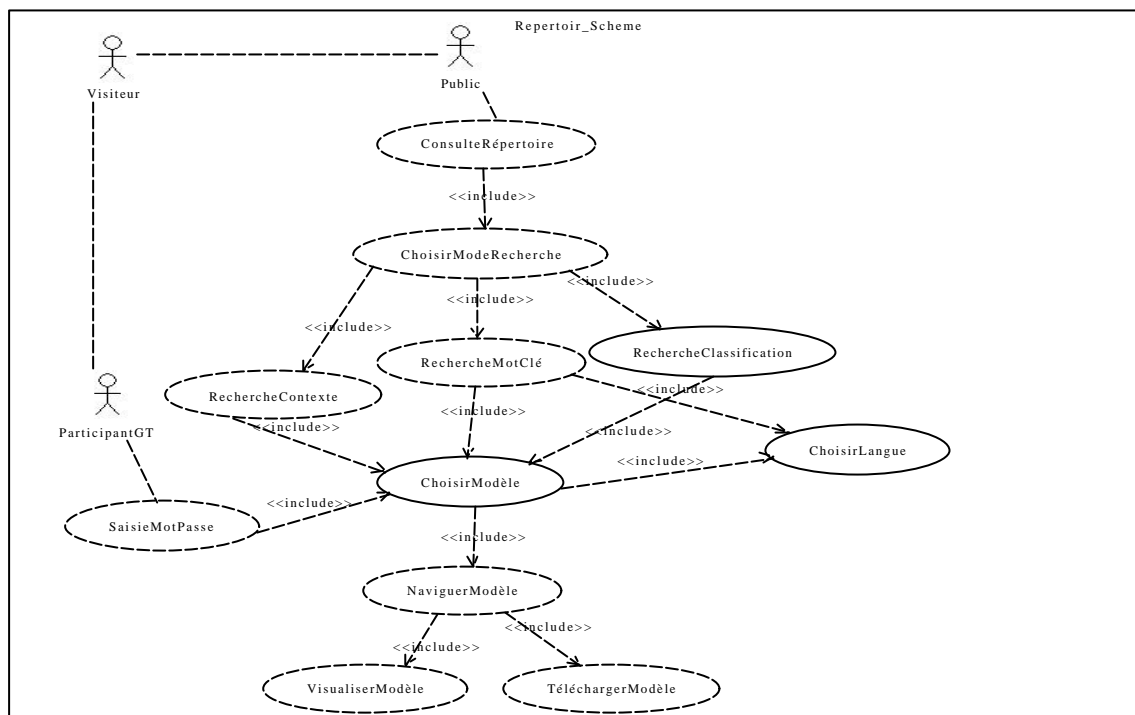
1 DIAGRAMME DE CLASSES :

Ce diagramme de classe est celui qui a été utilisé au départ pour l'analyse fonctionnelle du projet.
Il est différent du diagramme présenté en début de document qui sera celui qui sert de base à la réalisation.

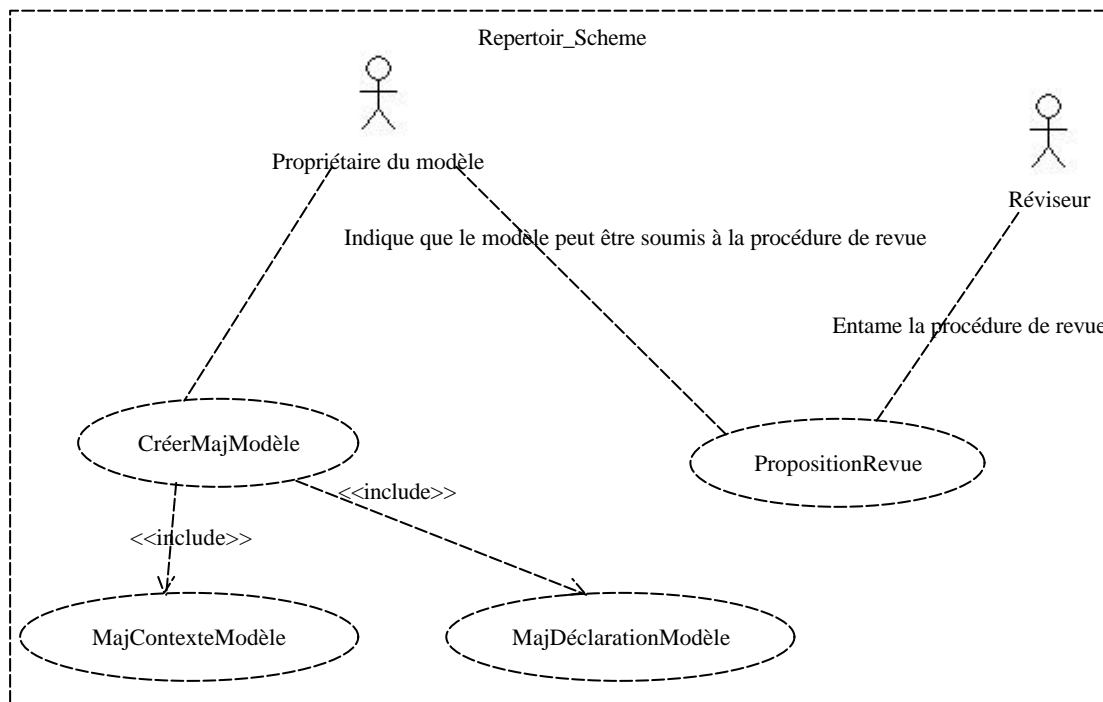


2 DIAGRAMME DE CAS D'UTILISATION

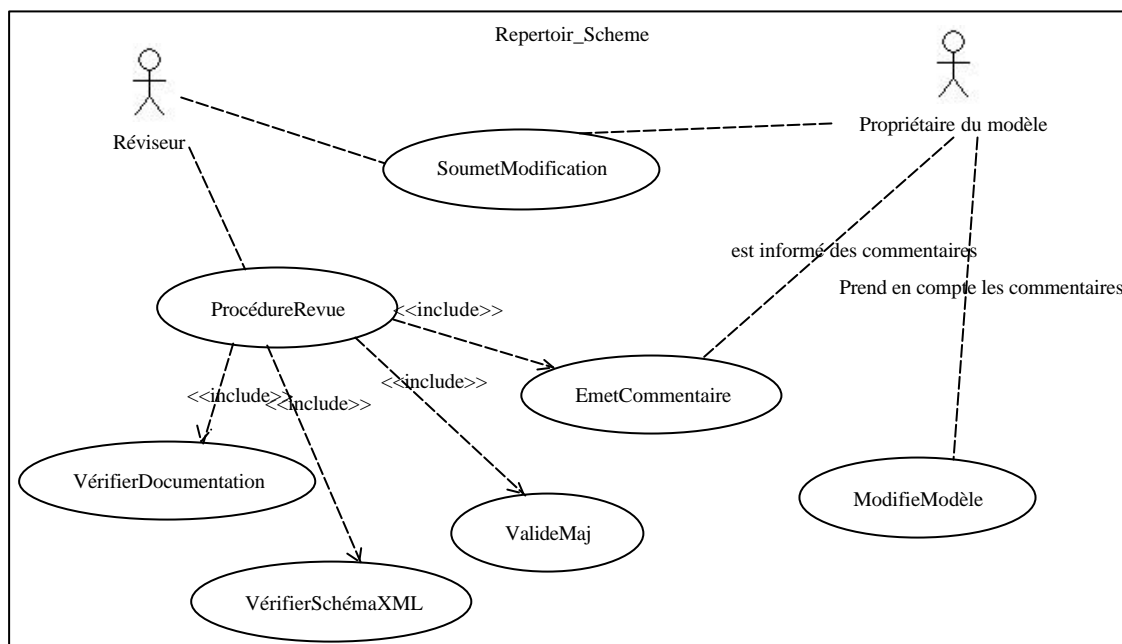
2-1 CONSULTATION



2-2 CRÉATION

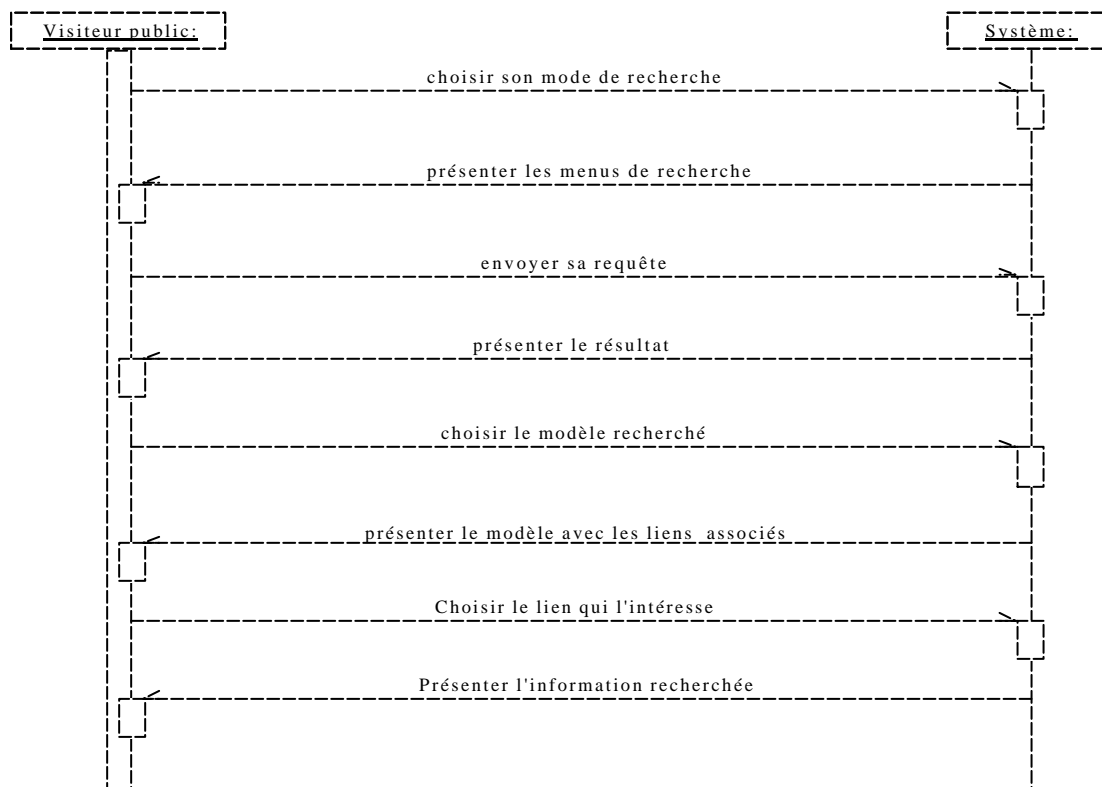


2-3 RÉVISION

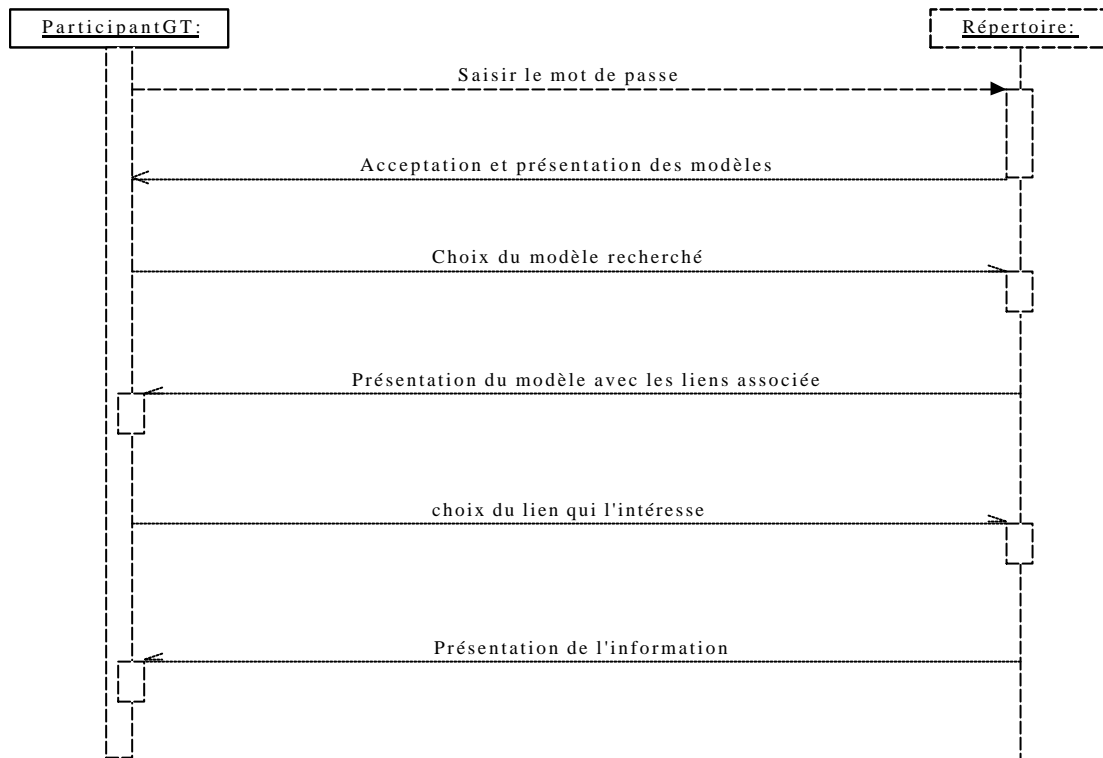


3 DIAGRAMMES DE SÉQUENCES

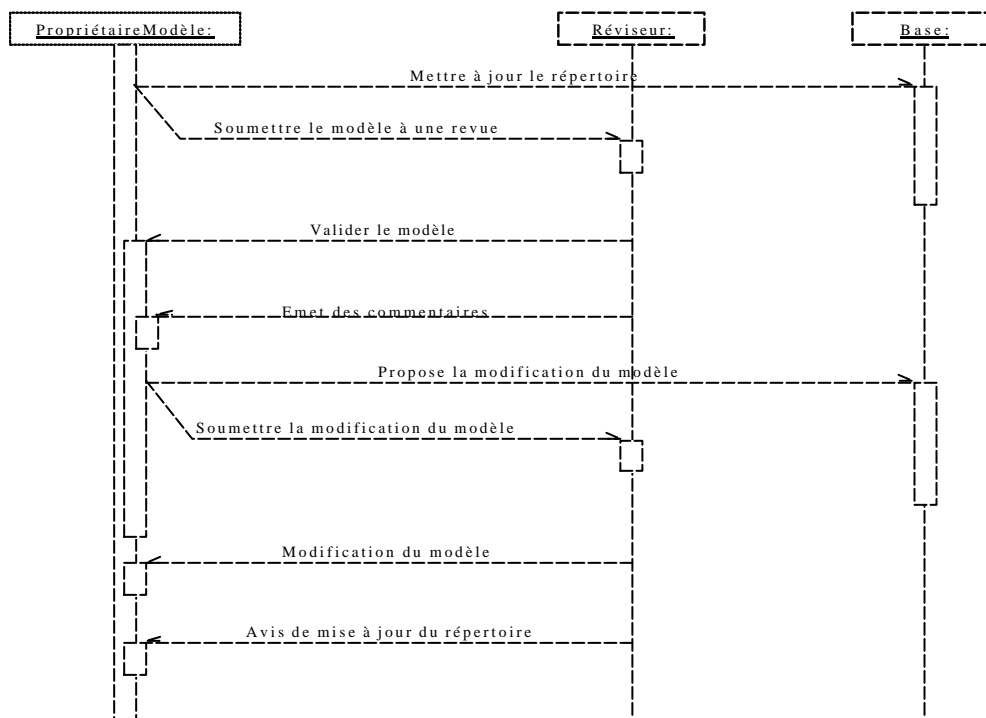
3-1 CONSULTATION DU PUBLIC



3-2 CONSULTATION DU PARTICIPANT AU GROUPE DU TRAVAIL



3-3 RÉVISION DU MODÈLE



ANNEXE 5 : FORMULAIRES DE DÉCLARATION DES MODÈLES

1 FORMULAIRE DE DÉCLARATION D'UN MODÈLE SIMPLE

MODÈLE SIMPLE

[Aide](#) pour remplir le formulaire

*Les champs précédés par des étoiles sont obligatoires à renseigner

* SECTEUR D'ACTIVITE

- | | | |
|------------------------------------|-----------------------------------|------------------------------------|
| <input type="checkbox"/> Santé | <input type="checkbox"/> Finances | <input type="checkbox"/> Économie |
| <input type="checkbox"/> Éducation | <input type="checkbox"/> Banques | <input type="checkbox"/> Industrie |
| <input type="checkbox"/> Presse | <input type="checkbox"/> Autres | |

* ROLE

- | | | |
|---------------------------------------|--------------------------------------|---------------------------------------|
| <input type="checkbox"/> Client | <input type="checkbox"/> Fournisseur | <input type="checkbox"/> Distributeur |
| <input type="checkbox"/> Transporteur | <input type="checkbox"/> Autres | |

* DOMAINE D'USAGE

- | | | |
|---|---|--|
| <input type="checkbox"/> Commerce électronique | <input type="checkbox"/> Procédure administrative | <input type="checkbox"/> Catalogue |
| <input type="checkbox"/> Production Information | <input type="checkbox"/> Gestion Information | <input type="checkbox"/> Édition Information |
| <input type="checkbox"/> Partage Information | <input type="checkbox"/> Échange Information | <input type="checkbox"/> Archivage Information |

* RECOMMANDATION W3C

- | | | |
|-------------------------------------|--------------------------------|--------------------------------|
| <input type="checkbox"/> SCHEMA XML | <input type="checkbox"/> DTD | <input type="checkbox"/> XPATH |
| <input type="checkbox"/> XSLT | <input type="checkbox"/> XLINK | <input type="checkbox"/> XSLFO |
| <input type="checkbox"/> AUTRES | | |

* MODE DE RESTITUTION DES DOCUMENTS

- | | | |
|---|-------------------------------|----------------------------------|
| <input type="checkbox"/> Édition papier | <input type="checkbox"/> HTML | <input type="checkbox"/> WAP |
| <input type="checkbox"/> SGML | <input type="checkbox"/> XML | <input type="checkbox"/> TELETEL |
| <input type="checkbox"/> AUTRES | | |

* TYPES DE DONNEES

- | | |
|---|---|
| <input type="checkbox"/> Données texte structurées | <input type="checkbox"/> Données texte non structurées |
| <input type="checkbox"/> Données multimédia structurées | <input type="checkbox"/> Données multimédia non structurées |
| <input type="checkbox"/> Données mixtes structurées | <input type="checkbox"/> Données mixtes non structurées |

***PROCESSUS**

Pour introduire le ou les processus d'affaires liés à ce modèle, veuillez [cliquer ici](#).

*** CLASSIFICATION DU PRODUIT**

Pour introduire la ou les classifications du produit liées à ce modèle, veuillez [cliquer ici](#).

***ORGANISATION**

Pour introduire l'organisation responsable de ce modèle, veuillez [cliquer ici](#).

***CONTACT**

Pour introduire un ou plusieurs contacts pour ce modèle, veuillez [cliquer ici](#).

PROJET

Pour introduire le projet lié à ce modèle, veuillez [cliquer ici](#).

***MODELE XML**

*Nom du modèle: *Version :

Commentaires sur la

Version :

N.B.: Veuillez introduire les dates sous forme : **aaaa/mm/jj** Ex: **2002/07/30**

*Date de Version 2002/07/30 * Date Début : 2002/07/30 Date Fin : 2005/07/30

N.B.: Veuillez entrer les mots clés séparés par des virgules.

*Mots clés :

***Principes du modèle**

Veuillez introduire les principes du modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

***Objectifs du modèle**

Veuillez introduire les objectifs du modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

***Résumé du modèle**

Veuillez introduire un résumé du modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

***Fonctionnalités du modèle**

Veuillez introduire les fonctionnalités du modèle (en texte, ou en URI (si le texte existe ailleurs))

Texte :

URI :

Autres informations concernant le modèle

Veuillez introduire toute autre information concernant le modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

CONTRAINTES LEGALES

Texte :

LOCALISATION GEOPOLITIQUE

Texte :

DECLARATION ALTERNATIVEPour introduire une ou plusieurs déclarations alternatives concernant ce modèle, veuillez [cliquer ici](#).***SCHEMA XML ou DTD**

Description

Veuillez introduire le schéma XML en texte et/ou en URI (s'il est sur le net).

Si votre schéma dépasse 65535 caractères, veuillez l'envoyer en fichier [attaché](#).

Texte :

URI :

FEUILLE DE STYLE

Pour introduire la ou les feuille(s) de style pour le document XML, veuillez [cliquer ici](#)

META-MODELE

Pour introduire le méta-modèle qui vous a permis de valider ce modèle, veuillez [cliquer ici](#)

UTILISATEURS

Pour choisir un utilisateur dans la liste ou ajouter un nouveau, veuillez [cliquer](#) [ici](#).

MODELES LIES

Pour introduire le(s) identifiant(s) et/ou le(s) URI(s) des modèles liés à votre modèle, veuillez [cliquer ici](#).

DOCUMENTS LIES

Pour Introduire tous les documents liés à votre modèle, veuillez [cliquer ici](#).

OUTILS ASSOCIES

Pour Introduire tous les outils associés à votre modèle, veuillez [cliquer ici](#).

URLs

Veillez introduire tous les [URLs](#) qui ont un lien avec votre modèle

Envoyer

Annuler

2 FORMULAIRE DE DÉCLARATION D'UN MODÈLE CORE COMPONENT**MODÈLE CORE COMPONENTS****Aide pour remplir le formulaire**

* Les champs précédés par des étoiles sont obligatoires à renseigner

*** SECTEUR D'ACTIVITE**

- | | | |
|------------------------------------|-----------------------------------|------------------------------------|
| <input type="checkbox"/> Santé | <input type="checkbox"/> Finances | <input type="checkbox"/> Économie |
| <input type="checkbox"/> Éducation | <input type="checkbox"/> Banques | <input type="checkbox"/> Industrie |
| <input type="checkbox"/> Presse | <input type="checkbox"/> Autres | |

*** ROLE**

- | | | |
|---------------------------------------|--------------------------------------|---------------------------------------|
| <input type="checkbox"/> Client | <input type="checkbox"/> Fournisseur | <input type="checkbox"/> Distributeur |
| <input type="checkbox"/> Transporteur | <input type="checkbox"/> Autres | |

*** DOMAINE D'USAGE**

- | | | |
|---|---|--|
| <input type="checkbox"/> Commerce électronique | <input type="checkbox"/> Procédure administrative | <input type="checkbox"/> Catalogue |
| <input type="checkbox"/> Production Information | <input type="checkbox"/> Gestion Information | <input type="checkbox"/> Édition Information |
| <input type="checkbox"/> Partage Information | <input type="checkbox"/> Échange Information | <input type="checkbox"/> Archivage Information |

*** RECOMMANDATION W3C**

- | | | |
|-------------------------------------|--------------------------------|--------------------------------|
| <input type="checkbox"/> SCHEMA XML | <input type="checkbox"/> DTD | <input type="checkbox"/> XPATH |
| <input type="checkbox"/> XSLT | <input type="checkbox"/> XLINK | <input type="checkbox"/> XSLFO |
| <input type="checkbox"/> AUTRES | | |

*** MODE DE RESTITUTION DES DOCUMENTS**

- | | | |
|---|-------------------------------|----------------------------------|
| <input type="checkbox"/> Édition papier | <input type="checkbox"/> HTML | <input type="checkbox"/> WAP |
| <input type="checkbox"/> SGML | <input type="checkbox"/> XML | <input type="checkbox"/> TELETEL |
| <input type="checkbox"/> AUTRES | | |

*** TYPES DE DONNEES**

- | | |
|---|---|
| <input type="checkbox"/> Données texte structurées | <input type="checkbox"/> Données texte non structurées |
| <input type="checkbox"/> Données multimédia structurées | <input type="checkbox"/> Données multimédia non structurées |

Données mixtes structurées

 Données mixtes non structurées

***PROCESSUS**

Pour introduire le ou les processus d'affaires liés à ce modèle, veuillez [cliquer ici](#).

*** CLASSIFICATION DU PRODUIT**

Pour introduire la ou les classifications du produit liées à ce modèle, veuillez [cliquer ici](#).

***ORGANISATION**

Pour introduire l'organisation responsable de ce modèle, veuillez [cliquer ici](#).

***CONTACT**

Pour introduire un ou plusieurs contacts pour ce modèle, veuillez [cliquer ici](#).

PROJET

Pour introduire le projet lié à ce modèle, veuillez [cliquer ici](#).

***MODELE XML**

*Nom du modèle: : *Version

Commentaires sur la Version :

N.B.: Veuillez introduire les dates sous forme : **aaaa/mm/jj** Ex: **2002/07/30**

*Date de Version : 2002/07/30 * Date Début : : 2002/07/30 Date Fin : : 2005/07/30

N.B.: Veuillez entrer les mots clés séparés par des virgules.

*Mots clés :

***Principes du modèle**

Veuillez introduire les principes du modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

***Objectifs du modèle**

Veuillez introduire les objectifs du modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

***Résumé du modèle**

Veillez introduire un résumé du modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

***Fonctionnalités du modèle**

Veillez introduire les fonctionnalités du modèle (en texte, ou en URI (si le texte existe ailleurs))

Texte :

URI :

Autres informations concernant le modèle

Veillez introduire toute autre information concernant le modèle (en texte, ou en URI (si le texte existe ailleurs)):

Texte :

URI :

CONTRAINTES LEGALES

Texte :

LOCALISATION GEOPOLITIQUE

Texte :

DECLARATION ALTERNATIVE

Pour introduire une ou plusieurs déclarations alternatives concernant ce modèle, veuillez [cliquer ici](#).

DESCRIPTION DU CORE COMPONENT

* Type du composant :

Classe d'objets

Qualifiant de la classe d'objet :

* Nom de la Classe d'origine:

Terme caractéristique

Qualifiant de la classe propriétaire

:

* Nom de la classe propriétaire

d'origine :

* Terme de Présentation :

Amount 

Autre Nom : Pour introduire un autre nom du core component, veuillez [cliquer ici](#)

* **DEFINITION :** Pour introduire une définition pour le présent core component, veuillez [cliquer ici](#)

REMARQUES : Pour introduire toute remarque concernant le présent core component, veuillez [cliquer ici](#)

DOMAINES D'APPLICATION :

Pour introduire les champs d'application du présent core component, veuillez [cliquer ici](#)

COMPOSANT REUTILISE :

Si vous avez réutilisé des composants qui existent dans la base, veuillez les [préciser](#)

Code Liste associé :

*Code liste :

* URI code liste :

Valeur du code: Pour introduire la ou les valeurs du code liste associé, veuillez [cliquer ici](#)

*SCHEMA XML ou DTD

Description

Veillez introduire le schéma XML en texte et/ou en URI (s'il est sur le net).
Si votre schéma dépasse 65535 caractères, veuillez l'envoyer en fichier [attaché](#).

Texte :

URI :

FEUILLE DE STYLE

Pour introduire la ou les feuille(s) de style pour le document XML, veuillez [cliquer ici](#)

META-MODELE

Pour introduire le méta-modèle qui vous a permis de valider ce modèle, veuillez [cliquer ici](#)

UTILISATEURS

Pour choisir un utilisateur dans la liste ou ajouter un nouveau, veuillez [cliquer ici](#).

MODELES LIES

Pour introduire le(s) identifiant(s) et/ou le(s) URI(s) des modèles liés à votre modèle, veuillez [cliquer ici](#).

DOCUMENTS LIES

Pour Introduire tous les documents liés à votre modèle, veuillez [cliquer ici](#) .

OUTILS ASSOCIES

Pour Introduire tous les outils associés à votre modèle, veuillez [cliquer ici](#) .

URLs

Veillez introduire tous les [URLs](#) qui ont un lien avec votre modèle

Envoyer

Annuler

3 FORMULAIRE DE DÉCLARATION D'UN CORE COMPONENT À PARTIR D'UN FICHIER EXCEL

Importer des core component

*Les champs précédés par des étoiles sont obligatoires à renseigner

DONNÉES COMMUNES

SECTEUR D'ACTIVITE

- | | | |
|------------------------------------|-----------------------------------|------------------------------------|
| <input type="checkbox"/> Santé | <input type="checkbox"/> Finances | <input type="checkbox"/> Économie |
| <input type="checkbox"/> Éducation | <input type="checkbox"/> Banques | <input type="checkbox"/> Industrie |
| <input type="checkbox"/> Presse | <input type="checkbox"/> Autres | |
-

ROLE

- | | | |
|---------------------------------------|--------------------------------------|---------------------------------------|
| <input type="checkbox"/> Client | <input type="checkbox"/> Fournisseur | <input type="checkbox"/> Distributeur |
| <input type="checkbox"/> Transporteur | <input type="checkbox"/> Autres | |
-

DOMAINE D'USAGE

- | | | |
|---|---|--|
| <input type="checkbox"/> Commerce électronique | <input type="checkbox"/> Procédure administrative | <input type="checkbox"/> Catalogue |
| <input type="checkbox"/> Production Information | <input type="checkbox"/> Gestion Information | <input type="checkbox"/> Édition Information |
| <input type="checkbox"/> Partage Information | <input type="checkbox"/> Échange Information | <input type="checkbox"/> Archivage Information |
-

RECOMMANDATION W3C

- | | | |
|-------------------------------------|--------------------------------|--------------------------------|
| <input type="checkbox"/> SCHEMA XML | <input type="checkbox"/> DTD | <input type="checkbox"/> XPATH |
| <input type="checkbox"/> XSLT | <input type="checkbox"/> XLINK | <input type="checkbox"/> XSLFO |
| <input type="checkbox"/> AUTRES | | |
-

MODE DE RESTITUTION DES DOCUMENTS

- | | | |
|---|-------------------------------|----------------------------------|
| <input type="checkbox"/> Édition papier | <input type="checkbox"/> HTML | <input type="checkbox"/> WAP |
| <input type="checkbox"/> SGML | <input type="checkbox"/> XML | <input type="checkbox"/> TELETEL |
| <input type="checkbox"/> AUTRES | | |
-

TYPES DE DONNEES

- | | |
|---|---|
| <input type="checkbox"/> Données texte structurées | <input type="checkbox"/> Données texte non structurées |
| <input type="checkbox"/> Données multimédia structurées | <input type="checkbox"/> Données multimédia non structurées |
| <input type="checkbox"/> Données mixtes structurées | <input type="checkbox"/> Données mixtes non structurées |
-

PROCESSUS

Pour introduire le ou les processus d'affaires liés à ce modèle, veuillez [cliquer ici](#).

CLASSIFICATION DU PRODUIT

Pour introduire la ou les classifications du produit liées à ce modèle, veuillez [cliquer ici](#).

*ORGANISATION

Pour introduire l'organisation responsable de ce modèle, veuillez [cliquer ici](#).

*CONTACT

Pour introduire un ou plusieurs contacts pour ce modèle, veuillez [cliquer ici](#).

PROJET

Pour introduire le projet lié à ce modèle, veuillez [cliquer ici](#).

META-MODELE

Pour introduire le méta-modèle qui vous a permis de valider ce modèle, veuillez [cliquer ici](#).

DONNÉES IMPORTÉES

Si vous avez un fichier Excel à importer, veuillez l'enregistrer d'abord en ".csv", en mettant un point virgule (;) comme séparateur entre les différents éléments. [Aide](#)

Joindre votre fichier :

