

Université de Bourgogne
U.F.R. Sciences & Techniques

Thèse de Doctorat
Spécialité Informatique

**Intégration et Manipulation de Données Hétérogènes au Travers de Scènes 3D
Dynamiques, Evolutives et Interactives.
Application aux IFC pour la Gestion Collaborative de Projets de Génie Civil.**

Présentée par

Christophe CRUZ
Le 15 Décembre 2004

Devant le jury composé de :

Monsieur Henri Basson, Professeur à l'Université du Littoral Côte d'Opale, Président
Monsieur Kadi Bouatouch, Professeur à l'Université de Rennes I, Rapporteur
Monsieur Abder Koukam, Professeur à l'Université de Technologie de Belfort-Monbéliard, Rapporteur
Monsieur Marc Neveu, Professeur à l'Université de Bourgogne, Directeur de thèse
Monsieur Christophe Nicolle, Maître de conférences à l'Université de Bourgogne, co-Directeur de thèse
Monsieur Olivier Gaudard, Président Directeur Général du Groupe Archimen, Examinateur

« À Alice, José, Jennifer et Frédéric»

*« Todas as viagens são lindas, mesmo as que fizeres nas ruas do teu bairro.
O encanto dependerá do teu estado de alma. »*
Rui Ribeiro Couto, diplomata e escritor

Remerciements

C'est avec une certaine émotion, et beaucoup de sincérité que je voudrais remercier toutes les personnes ayant soutenu et apprécié mon travail, qui est le fruit d'une recherche menée le long de plusieurs années.

Je ne pourrais pas commencer mes remerciements sans évoquer mon directeur de thèse Monsieur Marc Neveu, Professeur à l'Université de Dijon. Je tiens à exprimer toute ma reconnaissance à Monsieur Christophe Nicolle qui a dirigé cette thèse dans la continuité de mon stage de D.E.A. Je voudrais le remercier pour le temps et la patience qu'il m'a accordés tout au long de ces années, d'avoir cru en mes capacités et de m'avoir fourni d'excellentes conditions logistiques et financières. De plus, les conseils qu'il m'a prodigués tout au long de la thèse, m'ont toujours été profitables.

J'adresse mes plus sincères remerciements à Monsieur Henri Basson président du jury, à Monsieur Kadi Bouatouch et Monsieur Abder Koukam pour avoir accepté d'être les rapporteurs de cette thèse. Je tiens à adresser des remerciements particuliers à Monsieur Olivier Gaudard membre du jury et directeur de la société Groupe Archimen, pour avoir cofinancé cette thèse et m'avoir accueilli dans les locaux d'ACTIVe3D pendant un an et demi.

La réalisation de ce travail s'appuie également sur un environnement qui est essentiel. À ce titre, je voudrais remercier toutes les personnes travaillant ou ayant travaillé sur le projet ACTIVe3D, pour leurs travaux, leurs encouragements, leurs aides. Je tiens aussi à exprimer toute ma gratitude aux membres de l'équipe Systèmes d'Informations et Images pour l'amabilité et la bonne humeur dont ils ont fait preuve. Je souhaite bonne chance et bon courage aux thésards qui commencent ou qui sont sur le point de terminer.

Enfin, j'aimerais remercier mes parents, ma famille et mes amis pour leurs soutiens durant ces trois années.

Table des Matières

Chapitre 1 – Introduction

1. Rappel de l'existant	2
2. Objectif de la thèse	4
3. Approche de la thèse	4
4. Apport de la thèse	5
5. Organisation de la thèse	8

Chapitre 2 – Travail Collaboratif et 3D

1. Réalité Virtuelle et 3D : Présentation	13
1.1. Domaines d'applications	13
1.2. Optimisation et applications	16
1.3. Structuration et stockage des scènes 3D	19
1.4. Partage et échange de l'information 3D	24
2. Exemple de stockage de scènes 3D	26
2.1. Création de la base de données	29
2.2. Application	32

Chapitre 3 – Intégration et Manipulation de Données Hétérogènes : Un Etat de l’Art

1. Intégration de données	38
1.1 Architectures d’intégration de données	39
1.2 Intégration de services	43
1.3 Conclusion	48
2. Approche sémantique de l’intégration	49
2.1 La notion d’ontologie	50
2.2 Typologie des ontologies	52
2.3 Les langages ontologiques	53
2.4 Intégration de données basée sur une ontologie	59
3. Conclusion.....	61

Chapitre 4 – Une Ontologie pour l’Intégration de Données XML

1. Formalisation des données XML	65
1.1. Les grammaires formelles et les documents XML	65
1.2. Les grammaires formelles et les grammaires XML.....	66
1.3. Notion de Facteur et de marque schématique	70
1.4. Conclusion	71
2. Définition sémantique des marques schématiques.....	71
2.1. Concepts et Relations.....	72
2.2. Extensibilité et Contexte	73
2.3. Conclusion	74
3. Formalisation schématique et sémantique des schémas XML.....	74
3.1. Formalisation schématique et règles d’intégration	75
3.2. Ontologie générique.....	78
4. Exemple.....	84
4.1. Intégration de schémas.....	84
4.2. Intégration de données	95
5. Conclusion.....	102

Chapitre 5 – Arbres contextuels et processus de manipulations

1. Modèle de représentation des données.....	106
2. La notion de contexte	109
2.1. Définition du contexte.....	109
2.2. Valeur d’un élément sémantique dans un contexte.....	110
3. Les arbres contextuels	111
3.1. Définition d’un arbre contextuel.....	111
3.2. Exemple de définitions d’arbres contextuels	112
3.3. Extraction d’arbres contextuels.....	113
4. Bilan	118

Chapitre 6 – Implémentation

1.	Architecture générale	123
1.1.	Le module 3D	124
1.2.	Les Web Services	125
2.	Les IFC	127
2.1.	Description des fichiers IFC	128
2.1.1	Le modèle des IFC	128
2.1.2	Les instances IFC	129
2.2.	IFC contre XML	130
2.3.	La méthode ACTIVe3D	131
3.	Démonstration de l’interopérabilité dans le monde du bâtiment grâce aux IFC	133
3.1.	Phase 1 : Conception	133
3.1.1	Viz’all : Solution de relevés de bâtiments via pocket PC	133
3.1.2	ADT : Enrichissement de la maquette	134
3.1.3	ARCHICAD : Enrichissement de la maquette	134
3.1.4	ALLPLAN : Importation d’un fichier et finalisation du bâtiment	135
3.1.5	Bilan	135
3.2.	Phase 2 : Etudes techniques	135
3.2.1	Renseignement des éléments sémantiques	136
3.2.2	Calculs de structure	138
3.2.3	Calculs thermiques	138
3.2.4	Calculs de metré	139
3.3.	Phase 3 : Gestion technique de patrimoine	140
4.	Bilan	144

Chapitre 7 – Conclusion

1.	Contribution	145
1.1.	Dans le domaine de la 3D	146
1.2.	Dans le domaine l’intégration et manipulation de données	146
1.3.	Dans le domaine de la gestion des projets d’ingénierie civile	146
2.	Perspectives de recherches	147
2.1.	Les systèmes d’informations et la validation sémantique	147
2.2.	Application dans l’ingénierie civile	147

Bibliographie et Annexes

Chapitre 1

« Qui voit le ciel dans l'eau, voit les poissons sur les arbres »
(Proverbe chinois)

Introduction

Sommaire

1.	Rappel de l'existant	2
2.	Objectif de la thèse	4
3.	Approche de la thèse	4
4.	Apport de la thèse	5
5.	Organisation de la thèse	8

La vie d'un projet de construction d'un bâtiment débute lors de la création d'un dossier commercial, lors de l'établissement d'un devis ou lors d'une commande. Ensuite, le projet prend forme au travers d'une succession de phases : la phase APS (Avant Projet Sommaire), la phase APD (Avant Projet Détailé), la phase EXE (Exécution), la phase Projet, les appels d'offres auprès des entreprises de travaux, etc. La vie du projet s'arrête lorsque la vie du bâtiment commence, à la fin de la phase de réception des ouvrages. De nombreux acteurs participent à la vie d'un projet, chacun apportant « sa pierre à l'édifice ». Pour qu'un projet soit bien vécu par tous les acteurs du bâtiment, il faut qu'il soit compris et partagé tout au long de son cycle de vie. Le respect de ces deux conditions, à la charge du maître d'œuvre (bureau d'étude ou architecte), requiert un lourd investissement en temps, et en moyens. Il faut suivre le projet au jour le jour, constituer une abondante documentation, coordonner le travail et l'attention de tous les acteurs (client, entreprise de travaux, bureau de contrôle, etc.), rassurer le client, planifier et facturer les bons volumes aux bonnes dates, etc. Pour mener à bien ces missions, chaque acteur utilise à plein régime son traitement de texte, son tableur, son outil de CAO, son fax, le courrier et le mail. Apportant un confort non négligeable, ces outils ne permettent malheureusement pas une gestion coordonnée du projet. De plus, la grande variété d'outils entraîne des lourdeurs d'utilisation. Chacun possède son propre format de fichiers pour les plans et autres documents et chacun passe souvent plus de temps à convertir, retrouver, parfois même recréer les informations

du projet, qu'au projet lui-même. Aujourd'hui, le besoin fondamental de tous les acteurs du bâtiment concerne un outil simple qui permet une gestion coordonnée des actions menées dans un projet. Cet outil doit correspondre parfaitement aux attentes des métiers du bâtiment et offrir une qualité de services en terme d'ergonomie, de traçabilité, de partage et de sécurisation des données optimale. Cet outil doit permettre la gestion des données générées au cours du projet au travers de la visualisation 3D d'une maquette numérique, mise à disposition de tout un chacun dans une plateforme collaborative sur Internet.

Le développement d'une méthode d'intégration des données générées au cours du projet et de mécanismes de manipulations de ces données au travers une interface 3D sont les objectifs de cette thèse. Avant de décrire l'approche proposée par cette thèse, nous présentons un bref rappel des domaines de recherche qui sera nécessaire à l'aboutissement de ce travail. Une présentation plus complète de ces domaines est présentée dans les chapitres suivants.

1. Rappel de l'existant

L'infographie est un domaine de l'informatique qui est largement répandu dans les applications métiers. Après les images filaires générées par ordinateur dans les années soixante, après les représentations spatiales dans les années soixante-dix grâce à la modélisation B-rep (Bounding Representation), après les arbres CSG (Constructive Solid Geometry), après les surfaces de formes libres, le rendu réaliste apparu dans les années 90 permet une interaction dynamique et complexe entre les objets de la scène et les utilisateurs. Aujourd'hui, le rendu réaliste permet une interaction physique entre les lumières, rendant la scène 3D plus réaliste que jamais. Toutes ces améliorations tendent à transformer l'infographie en outil idéal pour la conception industrielle ou artistique de produits, car elle permet de modifier les objets par interactions et par simulations de phénomènes [FAU79, MOR85, FEI90].

Aujourd'hui Internet tend vers deux domaines qui semblent opposés. D'une part, l'aspect visuel où le texte qui composait initialement les pages des premiers sites WEB a été remplacé par des images et des animations. D'autre part, l'aspect informatif qui s'est considérablement développé. Les informations données par les sites deviennent intelligentes et adaptatives en fonction des comportements des internautes. Les nouvelles avancées en matière d'interconnexions de bases de données permettent la création de sites dynamiques. À ce jour, un site Web doit être animé, pour être attrayant et intelligent et par conséquent actif et interactif. Néanmoins, il existe de nombreuses limites. Dans le domaine de l'aspect visuel, la représentation 3D est en pleine croissance sur Internet. Néanmoins, elle est souvent limitée à de petites animations, car les ressources nécessaires pour utiliser la 3D sur le réseau sont trop importantes. Quant à l'aspect informatif, il est encore souvent limité à l'interfaçage d'une base de données avec du code

HTML. La construction de systèmes complexes interconnectant plusieurs bases de données n'est encore développée qu'au stade de la recherche.

Dans ce domaine, les méthodes d'archivage de grandes quantités d'informations proposent peu de solutions pour le stockage de scènes 3D. De plus, ces méthodes ne fournissent aucun moyen de manipuler les données et les objets 3D contenus dans ces scènes. En effet, la manipulation des données englobe le stockage, l'extraction et la modification des scènes elles-mêmes, mais également les objets géométriques constitutifs des scènes comme la géométrie et l'aspect visuel, les transformations géométriques permettant de les assembler et les éléments d'observation comme les caméras, les lumières, etc.

Pour résumer, bien que l'évolution des recherches dans les domaines de l'infographie et des bases de données soit constante, la difficulté d'utiliser des scènes 3D dans des sites Internet dynamiques réside moins dans la création de scènes 3D que dans la réutilisation interactive de ces scènes notamment par consultation de bases de données sur Internet. Gérer des scènes 3D comme interroger par le contenu une base de scènes architecturales, modifier à grande échelle certains paramètres ou réaliser des statistiques restent très délicat. Cela suppose la réutilisation des structures de données décrivant la scène 3D par le SGBD. Malheureusement, ces structures sont souvent très diverses, voire incompatibles. En effet, de nombreux formats « standards » sont utilisés en synthèse d'images. Citons par exemple 3dmf (Apple pour Quickdraw 3D), 3ds (Autodesk pour 3D-Studio), dxf (AutoDesk pour AutoCAD), flt (ModelGen pour Multigen) , iv (Silicon Graphics Inventor), obj (Wavefront/Alias), etc. Certains formats tentent de fournir un standard pivot pour l'échange de données tel qu'IGES (The Initial Graphics Exchange Specification) ou STEP (The Standard for the Exchange of Product). Une approche alternative s'est développée autour des accès sans cesse croissants à l'Internet. VRML (Virtual Reality Modeling Language) permet de créer et d'explorer des scènes 3D sur le Web. Par contre, il existe peu de tentatives pour constituer de réelles bases de données de scènes 3D. Toutefois, il est possible de constituer des objets 3D intégrables dans des bases de données par le biais de modeleurs, mais ces objets 3D ne sont utilisables qu'à partir d'un même modeleur ou d'un modeleur compatible. Aujourd'hui, il existe des accès à des scènes 3D qui sont décrites en VRML sur Internet, mais l'accès aux objets constitutifs des scènes n'est pas possible (par exemple certaines bases de données de structures atomiques¹, de structures chimiques dans le domaine médical², etc.).

Beaucoup de travaux tentent de réduire la multiplicité des formats. Le successeur du langage VRML est le langage X3D, basé sur le langage XML. Dans le domaine des bases de données et

¹ The Fermi Surface Database : <http://www.phys.ufl.edu/fermisurface>

² The NCI DIS 3D database : http://dtp.nci.nih.gov/docs/3d_database/dis3d.html

de l'Internet, le langage XML est devenu le standard pour l'échange de données. XML (eXtensible Markup Language) est un "langage de balisage extensible", développé par le W3C (<http://w3.org>). "Langage de balisage Extensible" signifie que le XML n'est pas limité (tout comme le HTML) à un balisage prédéfini extrêmement figé mais au contraire tout un chacun pourra définir ses propres balises dans des grammaires. Les grammaires, ou schémas XML, décrivent des types d'éléments liés à un domaine de description particulier. Par exemple, le schéma du langage X3D décrit des types d'éléments qui permettent la construction de scènes 3D. On parle de document pour désigner une instance d'un schéma XML. Une scène 3D particulière est un document X3D. Malheureusement, bien que la représentation des données par des schémas XML réduise l'hétérogénéité syntaxique et structurelle, l'association de ces données hétérogènes, ou leurs intégrations dans un système commun posent de nombreux problèmes liés à l'hétérogénéité sémantique.

2. Objectif de la thèse

Notre objectif est de définir une méthode, pour permettre l'intégration sémantique de données XML hétérogènes, des mécanismes, pour manipuler ces données au travers d'une scène 3D dynamique, évolutive et interactive. Cette méthode et ces mécanismes devront être implémentés dans une plateforme Web collaborative pour permettre la gestion collaborative de projets de génie civil basés sur la norme IFC.

Pour cela, nous devons définir tout d'abord une formalisation structurelle des schémas XML. Cette formalisation nous permettra de définir la sémantique des éléments composant la structure des schémas XML. Ensuite, nous définirons une méthode pour représenter cette sémantique. Cette représentation sémantique sera utilisée pour établir des correspondances entre les structures de différents schémas XML. Après, nous transposerons ces correspondances au niveau des documents XML instances des schémas XML sémantiquement intégrés. Cela nous permettra d'intégrer les documents XML dans un système homogène. Puis, nous chercherons à définir des mécanismes de manipulation de ces données intégrées. Ces mécanismes devront structurer dynamiquement les données intégrées selon une vue utilisateur ou un contexte d'utilisation. Nous chercherons en particulier à construire une vue géométrique des données intégrées sous la forme de scène 3D.

3. Approche de la thèse

L'approche employée dans nos recherches consiste à formaliser les schémas et les données XML. De cette formalisation découlera un ensemble de notions qui permettra de caractériser les éléments structurels composant un document XML au niveau schématique. Cette première étape

achevée, la suivante consiste à définir un sens à ces éléments structurels. Grâce à la définition d'un sens formel, nous pouvons faire la correspondance entre les éléments structurels de différents schémas XML possédant la même sémantique. Par conséquent, l'étude de la structure sémantique des grammaires XML, ainsi que des ontologies, a inspiré la construction de classes formant une ontologie d'intégration des schémas XML. Cette ontologie permet de définir et de stocker dans un système de gestion de données, les informations sur la structure sémantique des schémas XML, ainsi que les correspondances sémantiques entre ces schémas intégrés. De plus, les classes définies par l'ontologie permettent de stocker les informations concernant les documents XML engendrés par les schémas XML intégrés. Les correspondances sémantiques entre les différents schémas XML permettront de définir les correspondances structurelles entre les documents XML stockés. De cette manière, les données XML seront intégrées dans le système de gestion de données.

Une architecture de Web Services sera développée pour fournir un ensemble de mécanismes de manipulations de ces données hétérogènes. Toutes les données hétérogènes seront ainsi liées et toute modification d'une donnée sera dynamiquement répercutée sur toutes les autres données hétérogènes sémantiquement liées. Ainsi, la scène 3D sera dynamiquement modifiée si les éléments sémantiques liés à cette scène sont modifiés. De plus, les scènes générées correspondront aux besoins des utilisateurs, ainsi l'information 3D pertinente sera transmise sans le surplus de données 3D inutiles améliorant ainsi les temps de transferts réseau.

4. Appart de la thèse

L'environnement d'intégration que nous avons défini peut être appliqué à de nombreux domaines autres que la gestion collaborative de projets du génie civil. En effet, la méthode ACTIVE3D permet de générer dynamiquement l'ontologie du domaine d'intégration.

De plus, nous allons présenter deux niveaux de représentation des grammaires XML. Le premier niveau représente les règles d'écritures syntaxiques. Dans ce niveau, nous allons définir deux notions importantes qui sont la notion de facteur et la notion de marque schématique. Le deuxième niveau représente les règles d'écritures sémantiques permettant de décrire à l'aide des ontologies la sémantique des éléments d'un schéma XML. Nous allons montrer que la combinaison des deux niveaux permet d'intégrer des schémas XML. Nous allons voir que cette intégration peut être étendue aux documents XML. Une fois cette intégration réalisée, nous obtenons un ensemble sémantiquement cohérent de données hétérogènes (texte, graphique, multimédia...), structuré dans un document XML bien formé.

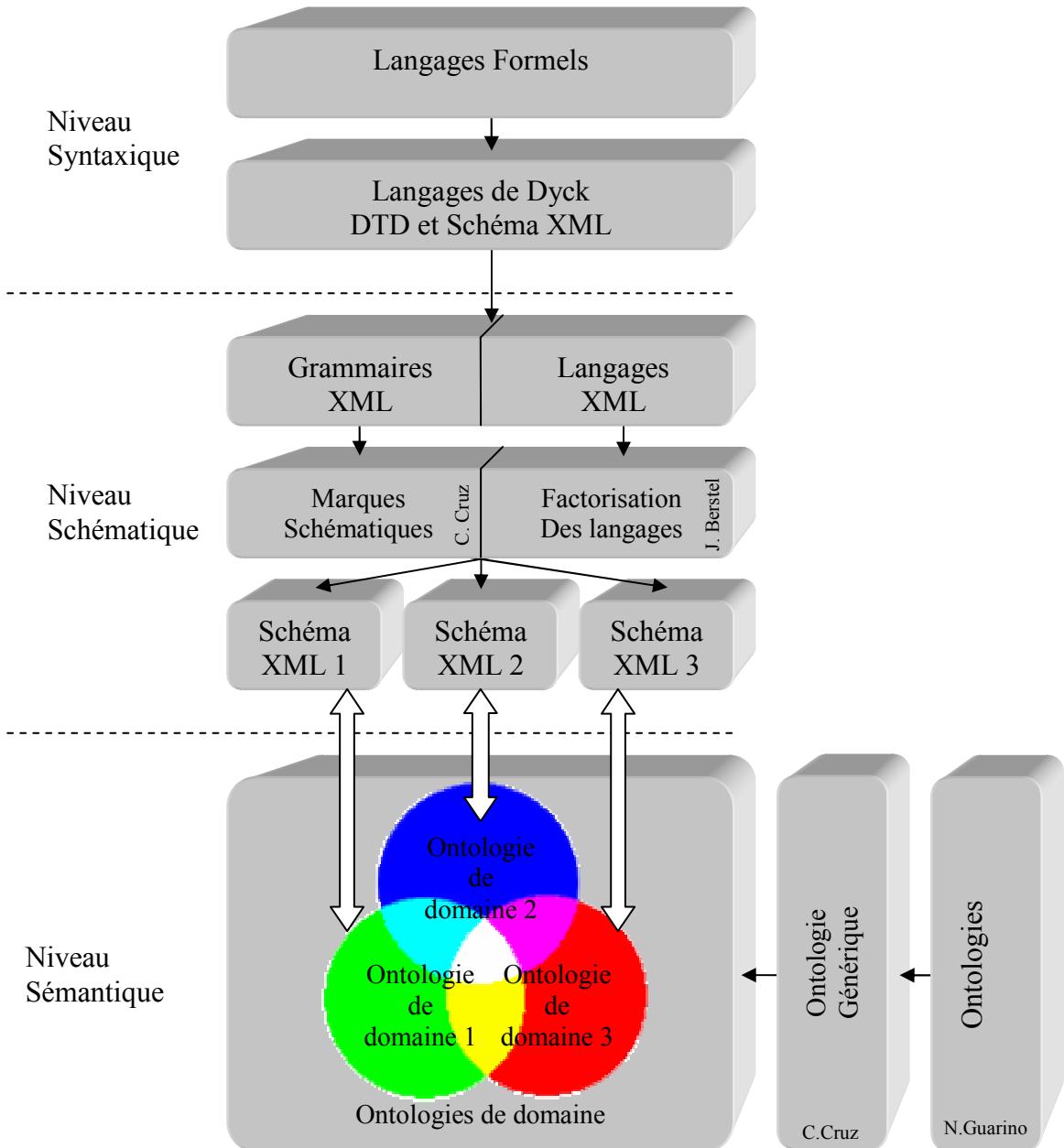


Figure 1.1. Étapes de définition de l'architecture d'intégration

La méthode d'intégration de schémas et de données XML hétérogènes que nous avons développée est résumée par la figure 1.1. Cette figure est constituée d'un ensemble de blocs représentant les étapes par lesquelles nous sommes passés pour définir notre architecture d'intégration. Les points de départ de notre méthode sont d'une part le bloc « Langages formels » en haut de la figure et le bloc « Ontologies » en bas à droite de la figure. L'étape d'arrivée est représentée par le bloc « Ontologies de domaines » qui constitue l'architecture d'intégration. Les travaux de N. Guarino sont la base de nos recherches sur les ontologies et ont inspiré notre ontologie générique constituée des éléments Concept, Relation, Attribut, Facteur conceptuel, Facteur relationnel, Facteur relationnel, Facteur d'attribut, Attribut simple. Ces

éléments permettent de définir les ontologies de domaines à partir des marques schématiques réalisées sur les schémas XML. En parallèle, nous avons étudié les travaux de J. Berstel sur les langages formels. Ces travaux ont débouché sur l'étude des langages de Dyck, permettant de définir les grammaires XML au niveau structurel (bloc « Langages de Dyck »). Dans ce bloc, nous nous sommes intéressés à la structure des emboîtements des balises XML plutôt qu'à la valeur de ces balises.

Au niveau XML, nous utiliserons le terme « grammaire XML » pour définir les règles d'écritures des documents XML associées à cette grammaire. Par exemple, la « grammaire X3D » est constituée de règles qui permettent l'écriture de documents X3D représentant des scènes 3D.

Au niveau des langages formels, nous utiliserons le terme « langage XML » pour définir tous les documents XML générés par la grammaire XML. Par exemple, le « langage X3D » est constitué de l'ensemble complet des documents X3D que la grammaire X3D peut générer.

La factorisation des langages XML est une définition de J. Berstel et forme le point de départ de ses travaux réalisés sur les grammaires XML. Cette notion de factorisation est à l'origine de la notion des marques schématiques (« bloc Marques Schématique »). Une marque schématique correspond à un facteur sur le langage et inversement. Cette marque permet de définir dans une grammaire XML un sous arbre commun aux différents documents XML générés.

Au niveau de l'ontologie, on peut définir une sémantique en associant un concept, une relation ou un attribut à une marque schématique (bloc « Ontologie Générique »). Si plusieurs marques schématiques de différents schémas sont associées à ce concept (ou relation ou attribut), alors chacune des marques schématiques va enrichir la définition du concept (ou relation ou attribut). Ce mécanisme permet de réaliser l'intégration sémantique des grammaires XML. Par instanciation, ce mécanisme permet de réaliser l'intégration sémantique de documents XML hétérogènes (bloc « Ontologies de Domaine »).

Nous avons également défini des mécanismes de manipulation du système intégré. Ces mécanismes sont basés sur la notion de contexte représenté par les arbres contextuels. Les arbres contextuels sont le résultat de l'extraction des données du système intégré dans un document XML indépendamment des structures d'origine.

Dans ce système, un objet qui possède plusieurs descriptions dans des contextes d'utilisations différents intégrera l'ensemble de ces descriptions dans le système intégré. Ainsi, l'utilisateur de cet objet dans un contexte spécifique pourra obtenir une définition de l'objet plus riche que celle initialement décrite dans ce contexte.

À l'inverse, l'utilisation des arbres contextuels permettra de réduire la définition d'un objet aux seules données pertinentes pour les domaines d'application correspondant au contexte. Dans

notre exemple, l'architecte peut vouloir extraire du système intégré seulement la description géométrique de la fenêtre. L'avantage de cette approche est de limiter la taille du flux d'information en augmentant la pertinence des données transmises.

Nous allons montrer que cette intégration peut être réalisée à partir de documents très hétérogènes comme ceux représentant des scènes 3D ou des maquettes numériques de bâtiment (IFC). Dans ce cas, les arbres contextuels peuvent générer dynamiquement des scènes 3D où chaque objet 3D est identifié et associé sémantiquement à d'autres données provenant de diverses sources. Dans le domaine de la synthèse d'image, notre approche peut être utilisée pour associer de la sémantique aux objets composants une scène 3D. Les données composant cette scène ne sont plus manipulées par rapport à leurs modèles de définitions géométriques, mais par rapport à la sémantique portée par ces définitions.

Cette méthode et ces mécanismes ont été implémentés dans la plateforme ACTIVe3D Build Serveur qui permet à tous les acteurs d'un projet de construction d'un bâtiment de partager et échanger des données au travers d'une maquette numérique en 3D. Cette plateforme a reçu la médaille d'or de l'innovation technologique au salon international du bâtiment en décembre 2003 à Paris.

5. Organisation de la thèse

Ce mémoire est composé de 7 chapitres. Le chapitre 2 présente les problématiques des domaines d'applications de la réalité virtuelle. Ces domaines d'applications reposent sur des recherches plus fondamentales visant l'optimisation du rendu des scènes 3D et le stockage des informations géométriques. La seconde partie de ce chapitre décrit une expérience de stockage et de manipulation des données géométriques que nous avons réalisée. L'objectif de cette expérience est de montrer qu'une approche syntaxique de l'information est insuffisante pour permettre la description, la manipulation et le stockage de scènes 3D.

Le chapitre 3 présente un ensemble de définitions et de méthodes qui offrent des solutions pour intégrer des données hétérogènes dans un ensemble homogène. La première partie de ce chapitre présente l'historique de l'intégration de données au travers de différentes approches allant de la traduction de schémas à l'utilisation de Web Services en passant par les approches distribuées ou fédérées. Dans cette partie nous verrons que l'utilisation du méta-langage XML permet de résoudre l'hétérogénéité structurelle et syntaxique des données. La seconde partie de ce chapitre concerne les travaux liés à la représentation sémantique des données et principalement aux solutions qui permettent de résoudre l'hétérogénéité sémantique. Les principaux travaux dans ce domaine concernent les ontologies. Nous verrons dans ce chapitre que de nombreuses ontologies sont construites à partir du méta-langage XML.

Le chapitre 4 présente une solution d'intégration qui met en relation différents niveaux d'abstraction sémantique et schématique. Cette solution est articulée en deux étapes. La première étape concerne la formalisation sémantique des règles d'écriture d'une grammaire XML en général. Cette formalisation nous permettra de définir les composants d'une ontologie générique. La seconde étape concerne la définition des mécanismes d'ontologisation de la sémantique des éléments d'une grammaire XML spécifique pour obtenir une ontologie de domaine. Les concepts et les relations de l'ontologie de domaine sont alors définis à partir des éléments des schémas XML. Dans ce chapitre, nous verrons que ces mécanismes permettent d'identifier certains concepts et relations communs à plusieurs schémas XML. Par conséquent, l'ontologie permettra de mettre en correspondance les concepts et relations en fusionnant les attributs des deux éléments communs. L'ontologie de domaine sera alors étendue et modifiée pour représenter la sémantique de plusieurs schémas XML relative à un domaine particulier. Nous verrons que cette intégration peut être étendue à l'intégration de documents XML.

Le chapitre 5 présente les mécanismes de manipulation de ces documents intégrés. Ces mécanismes sont basés sur la notion de contexte liée à des arbres-XML. Un contexte est une vue utilisateur du système en fonction d'un domaine d'application spécifique. Cette vue est un arbre-XML que nous appellerons arbre contextuel. Les arbres contextuels sont le résultat de l'extraction des données du document intégré dans un document XML indépendamment de leur structure originelle. Nous montrerons que ces arbres contextuels peuvent être utilisés pour construire dynamiquement une scène 3D à partir d'un ensemble d'objets sémantiques. Cette scène pourra alors être utilisée pour manipuler les données contenues dans le système intégré.

Le chapitre 6 présente le cadre des recherches en spécifiant la nature des projets d'ingénierie civile, ainsi que le résultat de ces recherches à travers les propriétés et une démonstration d'utilisation de la plateforme Web ACTIVe3D BUILD SERVEUR.

Le chapitre 7 résume les propriétés de la solution et les apports dans le domaine de l'ingénierie civile. Pour finir, ce chapitre présente les perspectives de recherche basées sur les méthodes, mécanismes et développement présentés dans ce rapport.

Chapitre 2

« Ce n'est pas le puits qui est profond, c'est la corde qui est trop courte. »
(Proverbe chinois)

Travail Collaboratif et 3D

Sommaire

1. Réalité Virtuelle et 3D : Présentation	13
1.1. Domaines d'applications	13
1.2. Optimisation et applications	16
1.3. Structuration et stockage des scènes 3D	19
1.4. Partage et échange de l'information 3D	24
2. Exemple de stockage de scènes 3D	26
2.1. Création de la base de données	29
2.2. Application	32

Une composante indissociable de la manipulation de l'information est sa représentation visuelle pour sa compréhension par l'humain. La représentation visuelle de l'information évolue conjointement avec l'amélioration des aspects matériels et des aspects logiciels de l'informatique. Au niveau du matériel, la puissance de calcul des ordinateurs ne cesse d'augmenter. De plus, la traditionnelle carte graphique se voit enrichie de son propre processeur et de sa propre mémoire vive. Au niveau logiciel, de nombreuses interfaces graphiques ont vu le jour, utilisant maintenant couramment la 3D pour la représentation et la manipulation de l'information.

Au niveau des interfaces homme-machine et plus particulièrement de la réalité virtuelle, il a fallu moins de trente ans pour passer des premiers concepts de l'immersion dans un monde simulé, et des interactions possibles avec ce monde virtuel [SUTH65] aux environnements virtuels 3D de type CAVE. Dans ces systèmes, l'illusion de l'immersion est créée par la projection stéréoscopique de scène 3D dans un cube composé d'écrans où les scènes sont projetées. Ces derniers systèmes sont couplés à des capteurs de mouvements de la tête, et de la main pour les interactions pour produire une perspective stéréo correcte.

Pour passer de l'étape de conceptualisation de la réalité virtuelle à l'étape d'utilisation dans un environnement de type CAVE, de nombreuses étapes ont du être franchies.

1. La première étape importante a été la généralisation de l'interface homme-machine permettant un usage et un apprentissage simplifié des ordinateurs. La plus connue des interfaces utilisateur a été « The Xerox Parc desktop metaphor » popularisé par Macintosh. Les interfaces utilisateur basées sur le principe du bureau virtuel sont une forme limitée d'environnement virtuel simplifiant l'interaction homme-machine. Cependant, bien que le bureau virtuel soit efficace pour l'interaction avec le système de fichier, certains fichiers quant à eux, requièrent une interface 3D pour leurs consultations et leurs manipulations. L'inconvénient majeur de la manipulation d'information en 2D est parfois le manque de corrélation entre les manipulations et les effets du au degré de séparation cognitive entre l'utilisateur et le modèle [CONN92, GOBB95]. En effet, l'inadéquation des interfaces utilisateur 2D est flagrante dans les applications de conceptions. Cette inadéquation oblige souvent les applications à utiliser des vues multiples pour apporter des informations supplémentaires, contraignant l'utilisateur à combiner différentes vues pour former un modèle mental de l'objet, compliquant sa tâche [HEND92] en le forçant à se concentrer sur comment la faire au lieu de la réaliser.
2. La deuxième étape importante a été la standardisation des bibliothèques de rendu de scène 3D (OpenGL, DirectX) permettant aux différentes communautés scientifiques de se consacrer à des problématiques autres que la visualisation 3D des informations graphiques. Ces problématiques sont nombreuses et font partie du sujet de la première partie de ce chapitre, dont l'objectif consiste à dresser une liste des domaines d'applications de la réalité virtuelle. Ces domaines d'applications reposent sur des recherches plus fondamentales visant l'optimisation du rendu des scènes 3D et le stockage des informations géométriques. Ces deux limitations sont encore fortes pour certaines problématiques demandant un effort supplémentaire d'optimisation, et une amélioration des outils applicatifs. La deuxième partie est un exemple de stockage et de manipulation des données géométriques qui rencontrent les mêmes limitations en terme de modélisation et de stockage, mais qui renforcent l'idée qui était jusque-là intuitive d'une nouvelle approche pour la gestion et la manipulation des scènes 3D.

La 3D offre la possibilité de créer une image numérique d'une représentation mental d'un produit. C'est pour cela qu'elle est souvent utilisé comme support pour la collaboration entre acteurs d'un projet. L'un des domaines de la représentation graphique 3D est la réalité virtuelle. Nous verrons que ce domaine est fertile en application pour la collaboration.

1. Réalité Virtuelle et 3D : Présentation

Dans cette section, nous souhaitons montrer l'importance du domaine de la réalité virtuelle et de la 3D pour la collaboration. Pour cela, nous présenterons tout d'abord les principaux domaines d'applications de ces techniques. Ensuite, nous focaliserons notre attention sur les enjeux actuels de recherches et de développements dans ces domaines. Enfin, nous montrerons l'intérêt d'utiliser des solutions de stockages de ces données 3D.

1.1. Domaines d'applications

Les domaines d'applications divers et variés montrent la fertilité du domaine de la réalité virtuelle et la 3D. Ces domaines suscitent de nouvelles idées comme la maquette numérique et le besoin d'interopérabilité de celle-ci. Elles introduisent aussi de nouveaux outils pour l'entraînement et l'aide à la prise de décisions. Ces domaines sont présentés dans les sections suivantes.

Prototypage virtuel : Les décisions prises lors des phases de conceptions dans les grands projets en ingénierie sont souvent les plus délicates à prendre, car leurs effets possibles sur le résultat final peuvent être dramatique, en temps et en argent. Ces décisions sont facilitées par l'utilisation de maquettes servant de prototypes au projet. Le prototypage permet alors de tester et d'améliorer la conception. Aujourd'hui, l'apparition de sites collaboratifs pour le prototypage virtuel offre un cadre de travail plus souple pour les acteurs distants d'un projet [BALA96, VIP01, KLI02].

Les maquettes numériques modélisent par définition les futurs éléments réels en 3D d'un projet. L'application de la maquette numérique connaît un succès flagrant dans le domaine de l'architecture. En effet, l'utilisation de la maquette dans des systèmes de réalité virtuelle permet aux architectes et aux ingénieurs de prototyper en itérant des modifications sur la conception du bâtiment en 3D [BROO86, AIRE90]. L'interaction avec la maquette numérique est primordiale, car elle permet de tester les objectifs finaux et d'avoir une idée précise de la faisabilité du projet ainsi que de son coût. Les professionnels en ingénierie développent aujourd'hui différents types d'outils spécialisés pour leur domaine comme le calcul de structure, de réseaux thermiques et fluidiques. Ces maquettes posent aujourd'hui problème car les différents logiciels métiers de prototypages et de calculs ont développé leur propre modèle et format de fichiers, les rendant difficilement interopérables. De plus, le partage de la maquette reste complexe, car elle se limite dans la majorité des cas à une série de fichiers qui sont en général une description géométrique de la conception sans information. Ces propriétés géométriques ne possèdent aucune information sur leur signification forçant l'intervention humaine pour leurs réutilisations.

Le centre national d'études spatiales françaises (CNES) et le CSSI de Toulouse se sont réunis pour lancer le projet de recherche PROVIS en 1995 pour le développement de nouvelles solutions pour la conception satellitaire permettant de créer, de manipuler et d'étudier des modèles utilisant les maquettes numériques. Ce projet s'appuie sur plusieurs concepts, une visualisation 3D interactive des prototypes, une interface de type "réalité virtuelle semi-immersive" avec spaceball, lunettes stéréoscopiques CrystalEyes VR, des données encapsulées dans des objets avec une notion de classe et de hiérarchie, et finalement des informations de type multimédia, du son, de la vidéo, des images, du texte, des schémas, des liens URL. Ces projets tentent de trouver des solutions aux problèmes qui apparaissent lors de l'utilisation de maquettes numériques sans apport d'information sur la nature des éléments géométriques qui composent le projet. Les notions d'objets, de classes et de hiérarchie permettent de pallier ce problème et ainsi d'intégrer à la géométrie aussi bien des informations sémantiques que des informations multimédias comme du son ou des schémas. Toutefois, il reste la problématique de l'interopérabilité du modèle utilisé dans ces projets pour les échanges de la maquette numérique avec les logiciels métiers. L'interopérabilité améliore l'efficacité de l'échange de l'information la rendant d'une part plus sûre et d'autre part plus rapide. L'information est plus sûr, car les erreurs d'interprétation humaine sont supprimées. L'information est plus rapide, car une étape de la modélisation est supprimée.

Simulation et entraînement : L'une des caractéristiques les plus importantes de la réalité virtuelle est dans la capacité des utilisateurs à exploiter les compétences cognitives et motrices de l'utilisateur pour interagir avec le monde à l'aide d'un ensemble de modalités sensorielles. L'expérience acquise dans le monde virtuel est directement utilisable dans le monde réel. Cet aspect a été mis au profit dans divers projets de simulations et de système d'entraînement [ADA01]. Il existe un large éventail d'exemples de recherches et d'applications industrielles, dont la simulation de vol [MORO91, MUEL96, PISA95], la simulation de conduit de véhicule [KUHL95, BAYA96] et la simulation médicale [YAGE96, ZAJT97, MEN03]. Le maître mot de ces systèmes est la sécurité, car toute tentative d'apprentissage sur des environnements virtuels ne met pas en jeu la vie des utilisateurs ou des patients. Les simulations permettent ainsi d'améliorer les compétences des opérateurs, limitant les risques d'erreurs qui dans des conditions réelles, ont une marge d'erreur restreinte.

Téléprésence et téléopération : La réalisation de tâches par l'humain dans des milieux hostiles est difficile voire impossible alors pour certain cas de figure, ces tâches sont réalisées par des robots. Cependant, ces robots n'ont pas suffisamment d'intelligence pour réaliser des tâches

complexes, donc la présence d'opérateur est nécessaire pour des tâches qui évoluent par rapport aux simulations. La téléprésence simule la présence d'un opérateur depuis un environnement sécurisé pour superviser et contrôler les tâches à réaliser. En mode de supervision, la réalité virtuelle fournit une interface à l'opérateur à l'aide de vues multiples. L'opérateur peut donc voir le site distant soit par l'environnement virtuel, soit par une caméra embarquée sur le robot. Pour réaliser les tâches que le robot ne peut faire seul, l'opérateur commute en contrôle interactif. Dans ce mode de téléprésence, l'opérateur a suffisamment de retours d'information pour estimer la présence sur le site. La NASA a réalisé des travaux de téléprésence où l'opérateur interagit avec les tâches de l'environnement simulé [FISH86]. La simulation de comportement est une partie des techniques de l'animation qui tente de reproduire le monde réel. Cette simulation de comportement aide les acteurs à interagir naturellement avec l'environnement en utilisant ses propres compétences [TERZ94, BURD96]. Le laboratoire IRIT de l'université de Toulouse a développé le projet EVIPRO (VIrtual Environment for Prototyping and Robotic) visant l'assistance de robots autonomes pendant la réalisation de missions de téléopération [HEGU01]. La réalité dans le cas de figure présent est une information visuelle supplémentaire pour l'opérateur, l'aidant à réaliser les tâches distantes. Ces tâches distantes sont réalisées par des robots manipulés par l'opérateur [MEN03]. Une simple vue comme une caméra embarquée n'est dans certains cas pas suffisante pour que l'opérateur puisse prendre des décisions et réussir la mission.

Réalité augmentée : Dans les systèmes de réalité augmentée, le monde virtuel est superposé sur le monde réel [GRA03]. L'intention de ces systèmes est de fournir des informations supplémentaires utiles dans l'encadrement de tâches dans le monde réel. Ce n'est qu'au cours des dernières années que les progrès dans les systèmes de vidéo temps réel et les systèmes d'infographie ont permis la convergence de l'affichage d'éléments virtuels dans un environnement vidéo réel. Les chercheurs dans ce domaine ont proposé beaucoup de solutions dans de nombreux domaines tels que des applications militaires [URBA96], les systèmes médicaux [STAT96, ROSE96, BBB03], la conception [AHLE95], la robotique [DRAS96], la manufacture et la maintenance [FEIN93]. La réalité augmentée comporte deux approches différentes, dont l'une a une portée temporelle et l'autre a une portée relationnelle. La notion de temps apparaît lorsqu'il est possible de faire apparaître dans une scène réelle des objets qui ne font pas partie du contexte temporel de la scène. Ces objets ajoutés sont aussi bien des objets passés ayant un intérêt archéologique pour la reconstitution de sites par rapport à des ruines, que des objets futurs pour un plan de mission simulée. La notion de relation apparaît quand la main de l'homme est guidée par la machine pour limiter les erreurs en condition réelles. Après

calibrage d'un système de guidage, le médecin peut opérer un patient à l'aide d'une vue. L'interface homme-machine permet alors un travail plus efficace cadrant les manipulations du médecin visualisable à l'aide des objets virtuels insérer dans la scène réelle.

1.2. Optimisation et applications

Toutes les applications décrites précédemment sont toutes bornées par les limites physiques de calcul de la machine. Chacune des applications a des besoins qui oscillent entre vitesse d'exécution et quantité de données en mémoire. Plus l'information en mémoire est grande et plus l'application est lente. Certaine application demande à la fois beaucoup de mémoire et une vitesse minimum d'exécution. Par exemple, la conception assistée par ordinateur produit souvent des modèles géométriques 3D complexes de très grandes tailles. Il n'est pas rare que les plans d'un bateau, d'un avion ou d'une structure architecturale dépassent la taille du gigabit. Le problème se situe donc au niveau de la visualisation des données, qui est récurrent dans de nombreuses applications graphiques. En effet, avec l'avènement des techniques de conception, la croissance du volume de données à traiter est largement supérieure à l'augmentation de la capacité du matériel graphique. Donc une phase d'optimisation est nécessaire. Elle se situe à la hauteur du modèle des données et des données elles-même. Beaucoup de techniques d'optimisations et d'accélérations pour l'affichage interactif ont été développées. Celle-ci inclut les calculs de visibilité, la simplification géométrique et la représentation basée sur les images.

Simplification géométrique : Les facettes triangulaires sont les primitives graphiques les plus populaires. Elles sont manipulées par la majorité des bibliothèques graphiques. Le maillage triangulaire d'un objet de conception est parfois très complexe, car il peut atteindre plusieurs centaines de milliers de facettes. Les outils de CAO génèrent couramment de tels objets, malheureusement l'accroissement de la taille des données dépasse toujours les capacités du matériel informatique. Toutefois, la représentation complète et précise du modèle n'est pas toujours nécessaire. Les algorithmes de simplification réduisent le nombre de facettes d'un objet selon des critères de qualité [PAI03]. Selon la perte de qualité de l'objet, il est possible de l'utiliser dans la scène 3D pour la navigation, voire de l'utiliser pour réaliser des calculs spécifiques sur le modèle simplifié en améliorant ainsi le temps de calcul. De plus, ces algorithmes permettent de définir différents niveaux de détails, selon des critères de distance définis par rapport au point de vue. La représentation du modèle sera plus ou moins simplifiée en fonction de la distance. Malheureusement, les niveaux de détails se confrontent au dilemme taille-mémoire/vitesse. En effet, les LOD (acronyme anglais de « levels-of-details ») optimisent la vitesse de navigateur de l'utilisateur à travers la scène 3D, mais augmentent aussi le nombre

d'objets en mémoire pour les différents niveaux de détails. La recherche sur la simplification de surface a connu un large engouement cette dernière décennie. [ROSS96, HECK94, ERIK96, PUPP97], mais ces techniques sont habituellement utilisées avec d'autres techniques d'optimisations car elles ne résolvent pas tous les problèmes.

Algorithmes de visibilité « Occlusion Culling » : La complexité des calculs d'une scène est liée au nombre d'objets présents dans celle-ci. Les calculs de visibilité déterminent si un objet est visible d'après le point de vue de l'observateur et le cas échéant l'objet est affiché ou non. Les temps de calcul et la place-mémoire dépendent du nombre d'objets constituant la scène 3D [PEA04]. Bien évidemment, le calcul du rendu d'une image d'un immeuble entier est bien supérieur à celui d'un petit appartement, car le nombre d'objets est bien plus important. Par contre, si les objets cachés par les objets visibles n'entrent pas dans le cycle de calcul de l'image alors le gain de temps est considérable, indépendamment du type de bâtiment. Les modèles architecturaux sont bien des milieux fortement occlusifs, alors pour ne pas prendre en compte les objets cachés de la scène en cours d'affichage, la scène doit être restructurée pour les algorithmes de visibilité. Les environnements architecturaux sont naturellement structurés. Par exemple [SETH91], les scènes sont décomposées en cellules et ouvertures. Une cellule est un volume spatial obtenu par un découpage de la scène, de préférence le long des murs. Une ouverture est une région 2D transparente située sur la frontière de deux cellules adjacentes. Le découpage de ses modèles s'effectue en faisant correspondre les cellules aux pièces de l'immeuble et les ouvertures aux portes et fenêtres. En déterminant les relations de visibilité des cellules entre elles, les algorithmes accélèrent fortement la recherche des objets visibles d'un observateur. Dans ce cas, le dilemme taille-mémoire/vitesse est résolu, mais ces algorithmes ne sont efficaces que dans les milieux fortement occlusifs, dans le cas contraire ils sont totalement inefficaces voire ils dégradent les performances.

Techniques basées sur les images : Le rendu basé sur les images est une application qui utilise des références de pré-rendu ou de pré-acquisition d'images dans une scène 3D. Les objets 3D sont remplacés par des avatars 2D. Ces techniques sont relativement peu coûteuses en ressource et ne requièrent pas de matériel graphique spécifique. Le temps de calcul d'une image est indépendant de la complexité géométrique et physique de la scène en cours de rendu. Il apparaît clairement que le rendu basé sur les images est extrêmement utile pour le rendu efficace des scènes complexes de la synthèse 3D [SHEN93, JONA96, STEV97, WILL97, CHR03]. Malheureusement, les méthodes les plus courantes de rendu basées sur les images fonctionnent dans l'hypothèse que toutes les surfaces visibles dans la scène sont des réflecteurs diffus idéaux

opaques, c'est-à-dire, que chaque point dans la scène semble avoir la même couleur quelque soit les différentes directions. Cependant, dans la pratique, les scènes réelles et synthétiques contiennent souvent diverses surfaces non diffuses, tels que les miroirs, les planchers brillants, les dessus de tables glacés, etc. Si une scène possède une de ces surfaces, alors des erreurs d'allucinations graves apparaîtraient. L'incapacité des techniques de rendu basé sur les images à manipuler les scènes non diffuses correctement limite fortement leurs utilités, applicabilités, et puissances.

Couplage des techniques : Toutes ces techniques ont été combinées avec succès pour le rendu de données spécifiques incluant des modèles architecturaux [FUNK96] et dans les modèles urbains [WONK01]. L'INRIA dans le projet iMAGIS [IMAG97] a réalisé un prototype logiciel permettant la navigation interactive dans une scène urbaine, en réalisant une segmentation de la base de donnée par rapport à la localisation du point de vue. Certaines données proches de l'observateur sont traitées en 3D, pour permettre par exemple de véritables tests de collision. Les données distantes sont simplifiées et regroupées dans des « imposteurs ». Un « imposteur » est une représentation simplifiée d'une partie de la scène 3D. Cette technique de construction des imposteurs à base d'images de synthèse permet de concilier le double impératif de la simplicité des primitives graphiques et de la fidélité du rendu. Le projet GigaWalk est un système de rendu permettant d'afficher des projets de CAO de plus de 10 millions de polygones. L'exemple le plus frappant est le projet de conception du tanker DoubleEagle constitué de plus de quatre gigaoctets de données, soit 82 millions de triangles et 127 mille objets. Le taux de calcul est de 11 à 50 images par seconde, ce qui permet de naviguer en temps réel à travers la maquette numérique après un temps de précalcul de 40 heures environ [WILL02].

Il n'existe pas de système idéal pour tous les cas de figure. Chaque technique a des avantages et des inconvénients, mais certaines combinaisons avec des conditions précises sont très efficaces. Ces techniques décrites précédemment ont fait leurs preuves dans un cadre statique. Par contre, les temps de précalcul avant que la scène puisse être accessible demande parfois plusieurs jours. Le temps de précalcul des scènes optimisées est le problème majeur de toutes ces techniques, car si les modèles 3D évoluent au court du temps, alors la gestion de la synchronisation des données doit être prise en compte. Ces synchronisations ne sont pas toujours possibles, car la structure complète de la scène peut parfois être remise en cause. D'autres pistes doivent être explorées pour permettre la visualisation d'une scène 3D évoluant au cours du temps. Les optimisations sont toujours réalisées par rapport à la géométrie ou par rapport à la topologie de la scène. La nature des objets géométriques n'a pas été prise en compte pour les calculs d'optimisation. La nature des objets s'avère donc une piste indéniable de recherche. Cette nature dépend de la

structuration de la scène, car si la structuration de la scène se limite à des informations géométriques alors seules les optimisations géométriques sont applicables. Par contre, si des informations sur la nature des objets sont indiquées alors ces informations ouvrent la voie à de nouvelles recherches sur la manipulation des informations géométriques et leurs stockages.

1.3. Structuration et stockage des scènes 3D

Les sections précédentes ont présenté quelques domaines de recherche ainsi que des optimisations possibles pour améliorer les systèmes avec interfaces graphiques 3D. La question qui se pose maintenant est la persistance des informations pour les réutiliser dans un cycle de conception ou de gestion ou de simulation. Ces informations sont stockées sous forme de fichiers ou dans des bases de données. Le stockage n'est possible que si l'information est structurée. La structuration d'une scène 3D est généralement réalisée en fonction des repères locaux des objets, procurant ainsi une structure hiérarchique à la scène 3D. L'origine de la scène est le repère global de la scène contenant un ensemble de repères locaux. Chaque repère local possède aussi un certain nombre de sous repères locaux. Tous les objets graphiques ou non composant la scène, comme la caméra ou les lumières, sont positionnés par rapport à un repère local ou non. Ces repères forment donc d'une certaine manière les liens entre les noeuds de l'arbre graphique.

Les fichiers : Les fichiers sont utilisés pour stocker sur le disque dur des données de tailles limitées. Les applications graphiques peuvent échanger l'information graphique par l'intermédiaire de ces fichiers. Les formats des fichiers pour décrire une scène géométrique sont très nombreux. Chaque application a sa propre spécialité, stockant l'information en fonction de ses besoins. Bien que l'information centrale de ces fichiers soit la représentation géométrique en 3D des objets, chaque type de fichier possède ses propres niveaux d'abstraction. Le format le plus élémentaire et se rapprochant le plus du modèle utilisé par les cartes vidéo est le modèle à facettes. Les objets sont décrits par des faces triangulaires ou rectangulaires, où les points sont fournis selon un sens précis pour que le vecteur de surface de la facette donne le côté intérieur ou extérieur de la facette. Le niveau de modélisation supérieur est la définition de l'objet par ses contours, où les faces sont décrites par un polygone extérieur et les trous par des polygones intérieurs. Les objets sont modélisables aussi par opérations de déformation par rapport à une ligne ou une courbe par exemple l'extrusion d'un polygone par rapport à une courbe de Bézier. Les déformations peuvent être des révolutions de courbes par rapport à un point ou un axe. Le niveau suivant consiste à définir un arbre d'opération booléenne de primitives comme des sphères et des cubes, comme l'union ou la différence de deux cubes. Ce modèle permet de réaliser des objets très complexes avec peu d'informations. Le modèle volumique quant à lui

permet de définir des objets à l'aide de volumes élémentaires nommés voxels. L'objectif n'est pas de fournir une liste complète de tous les modèles existants mais de donner un aperçu de la diversité. Cette diversité montre que plus le niveau du modèle est élevé et plus la taille des données est faible, mais que le temps de calcul pour descendre jusqu'au niveau le plus faible de modélisation pour le rendu (c'est-à-dire le modèle à facettes) est long. Par contre, plus le niveau d'abstraction de l'information est élevé et plus le fichier contient de la sémantique. En effet, si le fichier permet de définir des objets possédant un nom, un type et une géométrie, alors le modèle donne un sens à la géométrie. Cette sémantique est une connaissance supplémentaire sur les informations géométriques permettant de mieux réutiliser le fichier et ses définitions géométriques. En effet, si une scène est composée seulement de facettes alors comment distinguer les différents objets de la scène ? Par contre si les objets sont définis par leurs contours alors non seulement ils se distinguent les uns des autres dans les fichiers, mais ils portent en plus une information sur leurs ouvertures permettant de les catégoriser.

Les modèles pour structurer les objets et leurs syntaxes sont variés. Certaines applications privilégient la taille et la vitesse, dans ce cas le fichier sera binaire. D'autres applications préféreront un mode texte permettant d'écrire plus facilement le fichier comme les scripts VRML. Aujourd'hui dans le domaine du multimédia, les données sont de plus en plus structurées par XML par exemple X3D. La traduction de VRML sous forme de balises est un autre exemple flagrant de l'utilisation d'XML. XML est utilisé dans ce cas pour faciliter la réutilisation des fichiers par d'autres applications en réglant le problème de la syntaxe. Ce fichier XML est facilement interprétable par les analyseurs syntaxiques pour ensuite traduire les informations dans un format de fichier spécialisé d'une application. L'avantage majeur se situe au niveau des composants du langage. Un composant contient de nombreux noeuds comme par exemple le composant Nurbs contient tous les noeuds en relation avec les Nurbs. Chaque composant peut s'ajouter de nouvelles fonctionnalités par exemple le support d'un nouveau langage de script ou une interface utilisateur améliorée.

Les bases de données : Elles assurent la pérennité de grandes quantités d'informations. Celles-ci structurent et indexent les informations. Parfois, l'information est restructurée pour des impératifs applicatifs qui se traduisent par une réduction de temps d'accès à l'information. Restructurées ou non, les informations dans la base de données sont accessibles que par l'intermédiaire d'un langage de requête. Ce langage permet d'extraire une information partielle par rapport à des critères de sélection contrairement aux fichiers qui ne sont accessibles qu'en bloc. Grâce à ces langages de requêtes génériques, les systèmes de base de données stockent toutes les informations qu'un système d'information requiert. En général, les bases de données

portent la sémantique sous-jacente des informations qu'elles stockent. En effet, les structures qui reçoivent l'information modélisent les informations qu'elles doivent contenir, donc ces structures forment des méta-données sur l'information. La sémantique des méta-données spécifiée par le domaine d'application est utilisée pour interroger le système d'informations. Les systèmes d'informations géographiques sont l'exemple même des systèmes d'informations dont la sémantique joue un rôle central dans l'extraction d'informations pertinentes. Cette extraction est réalisée à l'aide de regroupements d'informations qui ne sont pas forcément en rapport initialement. La section suivante décrit les systèmes d'informations géographiques et les différents modèles d'informations manipulés montrant les relations de regroupement de l'information.

Les systèmes d'informations géographiques : Un système d'information géographique est un outil informatique qui permet de gérer différents types de données, que l'on a placées géographiquement sur un support carte. Ces systèmes permettent de faire des analyses statistiques, grâce à une visualisation synthétique et une analyse géographique propre aux cartes. Un système d'information géographique est avant tout un outil analytique ne stockant pas une carte, mais des données à partir desquelles on peut créer une ou des cartes selon le besoin. Ces systèmes stockent les informations concernant le monde sous la forme de couches thématiques reliées les unes aux autres par leurs coordonnées géographiques. L'information géographique contient soit une référence géographique explicite comme la latitude et la longitude ou une grille de coordonnées nationales, soit une référence géographique implicite comme l'adresse, le code postal, le nom de route, etc. Le processus automatique de géocodage est utilisé pour transformer les références implicites en références explicites et pour permettre la localisation d'objets et d'événements afin de les analyser. Les "données géographiques de références" permettent de localiser les informations, non pas directement par leurs coordonnées géographiques, mais indirectement par une adresse, par un numéro de parcelle ou d'îlot dont les coordonnées sont déjà connues. De plus, ils représentent les données de gestion des cartes permettant l'analyse de leur répartition spatiale. Pour manipuler toutes ces informations, les systèmes d'informations géographiques exploitent différents modèles géographiques.

- **Le modèle raster** : Les couvertures de type raster sont exclusivement constituées d'objets de type ponctuel, non modifiables et non sélectionnables dans un système d'information géographique classique. Ces objets ponctuels ne sont pas destinés à porter des informations, des données et des attributs d'objets mais permettent des localisations très précises à l'écran. Ils sont utilisés pour représenter des données continues. Les images raster sont des tableaux

de pixels. On traite ces objets ponctuels afin de distinguer des zones similaires et créer des cartes vecteurs.

- Le modèle vecteur : Les données du modèle vecteur sont le plus souvent issues de la digitalisation et du traitement de cartes papier, ou encore directement créées à partir de photos aériennes ou d'images satellitaires. Elles se composent d'objets géométriques de type ponctuel, linéaire, surfacique et/ou textuel auxquels des informations peuvent être rattachées. Dans le modèle vecteur, les informations sont regroupées sous la forme de coordonnées (x, y). Les objets ponctuels sont dans ce cas représentés par un simple point. Les objets linéaires (routes, fleuves...) sont eux représentés par une succession de coordonnées (x,y). Les objets polygonaux comme les territoires géographiques, les parcelles, etc, sont représentés par une succession de coordonnées délimitant une surface fermée. Le modèle vectoriel est utilisé pour représenter des données discrètes.
- Les bases de données alphanumériques : Les données alphanumériques contiennent des informations, numériques ou toponymiques, que l'on rattache aux objets des bases de données cartographiques de type vecteur. Certaines bases alphanumériques sont livrées avec des supports cartographiques, c'est le cas pour les données IRIS (Ilots Regroupés suivant des Indicateurs Socio-démographiques) de l'INSEE.
- Le Modèle Numérique de Terrain (MNT) : Un Modèle Numérique de Terrain est au départ un fichier altimétrique constitué par un réseau maillé régulier. Chaque maille est repérée par les coordonnées de son centre. Le MNT est une image matricielle du relief représentant l'altitude en chaque point dans un système de projection choisi. La qualité des MNT dépend de l'origine des données. Cette origine influe sur la précision et la valeur représentative. Un MNT peut recevoir plusieurs couches d'information. Par exemple, il est possible d'ajouter les contours des bassins versant et le réseau hydrographique. L'utilisation du MNT permet de corriger les déformations liées au mode prises de vue satellitaire, du à l'angle de visée du satellite et au relief de la zone couverte. Le MNT doit donc être appliqué pour chaque zone d'études très accidentée où l'on rencontre des falaises, des pics ou des vallées très encaissées. Les MNT sont les seules informations 3D que contiennent la majorité des systèmes d'informations utilisés dans l'industrie. De plus, ces informations sont des informations surfaciques, ne définissant pas à proprement parler d'objets 3D comme un arbre ou un immeuble. Parfois il est possible de trouver des éléments volumiques décrits à partir de volumes simples comme des cubes, des cylindres, des sphères, etc. Ces éléments sont assez rares en SIG, mais la tendance s'inverse, car les systèmes d'informations géographiques sont de plus en plus couplés à des données 3D [PFUN01, SHI02], ou les informations 2D sont converties en 3D [HUAN99].

Les techniques conventionnelles de bases de données ne suffisent pas pour la gestion de la complexité et du volume important des données et aux divers traitements géographiques. En effet, la composante spatiale de ces données nécessite la mise en œuvre d'un modèle intrinsèque décrivant la géométrie et/ou la topologie au sein du système. Les traitements sont de plus en plus complexes et font appel à des algorithmes géométriques, qu'il faut optimiser par des structures adaptées, en définissant par exemple des index spatiaux. Depuis plusieurs années, des recherches du laboratoire PriSM [PRIS02] dans ce domaine ont porté essentiellement sur la modélisation des données spatiales, leur exploitation à l'aide de langages et l'optimisation de leur traitement, d'abord pour des données en 2D puis en 3D et enfin en 4D (spatio-temporel). Des applications dans le domaine de la robotique mobile ou de la géologie ont nécessité des recherches plus spécifiques. Par ailleurs, la conception d'un modèle spatial ouvert et la gestion de métainformations figurent dans les derniers travaux. Le modèle spatial continue de susciter des travaux, spécialement de la part de groupes de normalisation tel que le CEN/TC287, ainsi que du comité international OpenGIS sur l'interopérabilité des systèmes d'informations géographiques. Le premier propose un modèle ouvert et paramétrable pour l'échange de n'importe quel format source de données géographiques. Quant au second, il établit un modèle géométrique pivot minimaliste pour l'accès à des serveurs hétérogènes. Concernant les données multi-dimensionnelles, sa gestion est nécessaire dans de nouvelles applications dans la gestion de l'environnement, l'urbanisme ou le data mining. Les recherches visent à modéliser aussi bien les aspects traditionnels de localisation et de morphologie des entités que les aspects de troisième dimension ou de dynamique (évolution dans le temps). Parmi ces problèmes, on peut citer la représentation d'entités complexes du monde géographique, l'accès efficace à des données volumineuses, l'indexation multi-dimensionnelle, l'hétérogénéité des dimensions. Ces problèmes deviennent plus aigus avec la troisième dimension géométrique.

L'utilisation d'informations 3D dans les systèmes d'informations géographiques sont encore récentes et restent problématiques, car la 3D n'a pas été prise en compte dès le départ pour la modélisation de ces systèmes. Par conséquent, la 3D se limite aujourd'hui à la visualisation de données 2D par la représentation graphique de concepts en 3D. La visualisation 3D interactive performante ne suffira probablement pas car le SIG est plus un outil d'aide à l'analyse qu'un moyen de communication. La question de l'utilité de la 3D dans ces conditions se pose. La 3D apporte une information supplémentaire, si elle est manipulable. Dans le sens où l'information que l'objet véhicule est non seulement la sémantique du concept sous-jacent, mais également l'information géométrique. C'est-à-dire que la géométrie de l'objet n'est pas seulement figurative, mais est aussi le résultat d'un travail de conception pour une réutilisation ultérieure. La question se pose donc en terme de services. Tout le problème est de poser le cadre de travail

des services d'échange de l'information de conception autour de la 3D, pour que la 3D ne soit plus seulement une information visuelle mais quelle puisse être réutilisé dans un cadre différent d'un système d'information géographique.

1.4. Partage et échange de l'information 3D.

Le travail collaboratif entre acteurs distants sur un même projet enrichit la conception du prototype en réduisant le temps entre chaque mise à jour. De nombreux outils de DAO ont été modifiés pour permettre le prototypage virtuel de maquettes numériques à partir de plans, et ce, indépendamment du besoin spécifique d'un projet. Malheureusement, la plupart de ces solutions sont limitées de par leurs aptitudes dans le domaine de la 3D, car elles ne réunissent pas les capacités d'interaction et de collaboration indispensables pour la réalisation d'un projet d'ingénierie. Cette problématique se retrouve dans les systèmes d'informations géographiques tentant de fournir une interface 3D, mais où le cadre de travail ne fournit pas de services pertinents pour la réutilisation de l'information 3D. Pour pallier ce problème, des projets ont vu le jour cherchant des solutions pour une meilleure interaction et collaboration.

Le projet CAVALCADE a pour objectif de faire avancer l'état de l'art dans le domaine des plateformes collaboratives 3D grâce à une architecture distribuée permettant à des équipes distantes de collaborer sur la conception, les tests, la validation et l'échange de documents. [CAVA98]. CAVALCADE fournit un système visuel de simulation 3D aux intervenants d'un projet. Contrairement à l'idée classique que l'on se fait des outils de simulation, la représentation du prototype virtuel est seulement l'aspect visible des attributs qui composent l'objet. Ces attributs invisibles sont des fonctions comme « être une partie d'un sous-système » et sont des documents comme une fiche technique ou un lien internet. L'interaction avec le prototype est une interface d'accès à des données. Le modèle 3D devient alors une interface visuelle de requêtes d'informations. CAVALCADE a pour objectif la gestion des données pour la conception.

Le CRS4 et le CERN ont développé conjointement le système I3D [BI3D95]. I3D est un système qui combine les possibilités de rendu des systèmes 3D et les possibilités d'un explorateur Internet. L'utilisateur se dirige à l'intérieur d'un scène virtuelle, tout en choisissant des objets 3D donnant accès à des documents distants de diverses natures comme du texte, des images, des animations ou même d'autres modèles 3D. Ce système est utilisé par le CERN pour la visualisation et la gestion du « Large Hadron Collider » dans le projet VENUS (Virtual Environment Navigation in the Underground Sites) et par le CRS4 pour le projet Sardinia. Le

projet Sardinia collecte un très large éventail de données hétérogènes sur l'île de Sardaigne et présente ces données pour rendre la plus facile d'accès possible à travers un voyage virtuel. Toutes ces données sont interconnectées à l'aide de liens hypermédias, accessibles depuis un navigateur Internet et le panel des informations contient aussi bien des données géographiques que des données archéologiques en passant par l'histoire et le tourisme.

Les constructeurs, pour échanger les modèles créés à l'aide des logiciels CAO, utilisent un format de fichier spécifique à leurs besoins. L'ensemble des fichiers formant les éléments de conceptions du projet constitue la maquette numérique. Cette maquette numérique a de nombreux avantages, car elle intègre généralement le modèle 3D de l'objet de conception, mais aussi un ensemble d'informations permettant la gestion du projet en lui-même. À partir des projets Saxo et 406, PSA Peugeot Citroën a vu l'augmentation du nombre de pièces conçues à l'aide des outils CAO s'accroître fortement conduisant à repenser l'organisation du travail du bureau d'études. Pour faciliter la mise en commun des données sur le projet 206, une maquette numérique a été mise en place. Grâce à cette maquette, tous les travaux de sélection pour extraire de la masse les données pertinentes envoyées par le constructeur sont quasiment réduits à zéro. Le travail de conception est tout de suite possible à partir de la maquette. L'accès aux dernières mises à jour élimine les erreurs coûteuses liées aux travaux réalisés sur des données déphasées. De plus, la mise à jour directement de la maquette numérique élimine des tâches sans valeurs ajoutées provoquant ainsi un gain de temps et de frais de déplacement. Le partage des données de conception est impératif pour accélérer les cycles de conception. Le standard STEP AP214¹ recherche la suppression des ambiguïtés de communication pour augmenter l'efficacité collective.

La notion de maquette numérique est avantageuse, car elle met tout le monde d'accord sur un ensemble de concepts que les projets du domaine ont besoin de manipuler. Pour cela, la maquette numérique doit reposer sur un langage qui unifie tous les concepts du domaine d'application. Cette démarche est employée par le groupe d'intérêts 3DIF « 3D Industry Forum » [3DIF04] associant éditeurs (Dassault Systèmes, Adobe, Ngrain...) et constructeurs ayant pour objectif la définition d'un format de données facilitant la diffusion et la consultation d'images 3D issues de la conception assistée par ordinateur. Le format de fichier U3D vient de naître sous l'impulsion d'Intel pour exploiter les masses de données 3D issues des logiciels CAO. Leurs formats sont trop complexes, trop fermés et trop lourds pour permettre aisément leur diffusion et leur consultation. La volonté des acteurs du marché de la 3D montre leurs besoins d'unification pour ne plus être limités par la modélisation, mais pour se concentrer sur les services pour

¹ STEP AP214 : <http://www.theorem.co.uk/docs/stepap214.htm>

l’interaction et la collaboration autour d’une maquette numérique exploitable par tous les corps de métiers utilisant la 3D comme base de travail.

La section suivante présente les travaux réalisés dans le cadre de cette thèse consistant à réaliser un système d’information à l’aide d’une base de données contenant des données 3D. Ce système se base sur le langage unique X3D comme langage unique d’échange de l’information. Cet exemple souligne l’importance de l’information sémantique des éléments de la scène pour l’échange d’informations géométriques d’une maquette numérique.

2. Exemple de stockage de scènes 3D

Archiver de grandes quantités d’informations n’est aujourd’hui encore qu’un moyen de stocker les scènes 3D plutôt qu’un réel moyen de les manipuler. En ce sens, les manipulations consistent à stocker, extraire mais aussi à modifier non seulement les scènes elles-mêmes mais également les objets géométriques constitutifs des scènes (géométrie et aspect), les transformations géométriques (permettant de les assembler) et les éléments d’observation (caméras, lumières, etc). Ces manipulations sont généralement réalisées à l’aide de langages spécialisés. Si les scènes 3D sont stockées dans une base de données relationnelle par le contenu alors il sera possible de manipuler chaque élément de la scène ainsi que ses propriétés à l’aide de SQL. Le stockage par le contenu signifie que les éléments de la scène et ses attributs sont séparés en conservant les liens pour pouvoir accéder directement aux informations voulu par rapport à son type, sans avoir à parcourir le fichier complet. La scène est simplement décomposée et cette décomposition est stockée dans la base de données. La difficulté ne réside donc pas dans la création de scènes 3D mais dans la réutilisation de ces scènes notamment par manipulations depuis un système de gestion de bases de données. Gérer des scènes 3D, comme par exemple interroger par le contenu une base de scènes architecturales ou modifier à grande échelle certains paramètres ou réaliser des statistiques, reste très délicat et cela suppose la réutilisation des structures de données décrivant la scène 3D par le SGBD.

Cette section présente une technique de stockage de scène 3D dans une base de données relationnelle. Son principal objectif est de créer un lien entre le stockage de l’information et sa représentation visuelle. Ainsi la scène 3D sera manipulable depuis la base de données à l’aide des outils d’interrogation relationnel. XML [XML02] est devenu rapidement un support universel de représentation de l’information sur Internet. Employé dans de nombreux domaines, il est largement utilisé pour véhiculer du multimédia et en particulier de l’image. La manipulation d’images avec XML se fait à plusieurs niveaux. Au niveau le plus simple, il est possible d’insérer une image dans un document XML à l’aide de balises contenant l’adresse du fichier image. A un niveau plus complexe, des langages permettent de modéliser des images 2D [SVG02] et 3D

[IAM98, XGL01, 3DM02, X3D02] dérivés d'XML. A ce niveau l'image est décrite en mode texte, sous forme de balises. Le choix s'est porté sur X3D car ce langage est libre et se base sur le langage VRML. De plus, le niveau de modélisation ne se limite pas à définir des facettes mais permet de définir une scène avec des primitives géométriques. L'information contenue dans ces fichiers est hiérarchisée grâce à XML, ce qui permet de manipuler des arbres. Tout d'abord, un passage à X3D (eXtensible 3D) est nécessaire pour convertir les fichiers VRML. En effet, de nombreux outils ont été développés pour manipuler des fichiers VRML tandis que les outils X3D sont rares ou propriétaires. L'insertion et l'extraction des scènes 3D dans la base de données sont réalisées à l'aide de feuilles de style. La méthodologie justifiant les choix de normes sur lesquelles s'appuie notre travail, est exposée dans la section qui suit.

Cette section présente le langage de modélisation 3D pour cette expérience. Ensuite suivra la présentation de la création de la base de données 3D à partir de X3D, qui s'appuie sur la définition d'un ensemble de règles de passage du schéma X3D au schéma Entité-Relation. La présentation de la création de la base de données inclut l'analyse des scènes X3D et les différents niveaux de création de la base de données (niveau conceptuel et niveau physique). Ensuite, nous présenterons l'architecture d'insertion et d'extraction de scènes 3D dans la base de données, pour finalement présenter un retour d'expérience sur la méthode et les améliorations à apporter.

Les langages de modélisation : Le langage de modélisation 3D le plus connu sur Internet est VRML. Largement répandu, il est possible de citer plus de trente programmes ou plug'ins utilisant VRML, tels que CosmoPlayer [COS02], Blaxxun3d [BLA02], etc. De nombreux modeleurs comme 3D Studio Max de Kinetix renferment des fonctions d'exportation pour la création de fichiers VRML. Malheureusement, un fichier VRML est un script monolithique difficilement exploitable. Pour résoudre ce problème, nous avons développé un outil Java de conversion VRML vers le langage X3D [X3D02]. Basé sur la norme XML, X3D fait l'objet d'un accord entre le W3C [W3C02] et Web3D Consortium. Le choix de X3D s'impose pour plusieurs raisons. XML est un méta-langage de description qui structure l'information à l'aide de balises organisées dans un arbre. En comparaison avec VRML, la modélisation de l'information est identique mais la forme est plus souple en terme d'utilisation. L'approche 3D avec XML apporte la flexibilité et la capacité d'extension que VRML ne peut pas apporter. Ceci permet l'ajout d'informations aux scènes, la création de nouvelles balises sans gêner la génération des scènes. Malgré sa création récente de nombreux outils ont été développés et sont disponibles pour manipuler des structures XML.

Une fois les scènes obtenues en X3D, nous devons insérer les éléments qui les composent dans la base de données. Pour cela, nous utilisons des feuilles de styles XSL [XSL02]. XSL est un

langage qui permet la manipulation d'un arbre XML. Le point fort de XSL réside dans sa capacité à projeter une source de données uniques sur de multiples cibles d'affichages et de formater les données sur chaque cible. Il existe deux types de formatage : le formatage des données et le formatage des résultats.

- Formatage des données : Le formatage des données est constitué de trois étapes. La première étape consiste à générer une structure de données sous forme d'un arbre source, en associant les éléments (nœuds de l'arbre) à divers motifs stylistiques (appelés patterns). La deuxième étape réalise une analyse de l'arbre pour produire un arbre résultat ou final. Celui-ci est basé sur les actions spécifiées dans les règles dites gabarits, patrons ou modèles. Dans la dernière étape, l'analyseur syntaxique prend le relais en modifiant les données XML à l'aide des instructions de formatage XSL. Il en résulte un arbre final sous forme de page HTML pour l'affichage sur le client Web. XSL est capable de réordonner les données de sorte que l'arbre final puisse être totalement différent de l'arbre source.
- Formatage du résultat : XSL est indépendant du langage d'affichage. Il est donc possible de développer un analyseur syntaxique XSL qui traduise l'arbre source en d'autres formats de sortie comme le RTF, le PDF ou encore créer des scripts comme PHP [PHP02] ou des scripts Oracle.

```
<schema>
  <element name="EspduTransform">...
  <element name="ReceiverPdu">...
  <element name="SignalPdu">...
  <element name="TransmitterPdu">...
  <element name="X3D">
    <type content="elementOnly">
      <group>
        <element minOccurs="0" ref="head"/>
        <element ref="Scene"/>
      </group>
    </element>
    <element name="head">...
    <element name="Scene">...
  </schema>
```

Exemple 2.1. Extrait de la grammaire de X3D

Un document X3D, comme tout document XML possède une grammaire qui définit la structure des balises qui composent le document. Cette grammaire appelée aussi schéma est décrite dans le format de balisages d'XML à l'aide du langage XML-Schéma. Un parseur teste la validité d'un document X3D en comparant sa structure avec cette grammaire. Un exemple de grammaire X3D est présenté dans le script 2.1.

Le schéma X3D est composé d'un ensemble de balises éléments. Ces balises éléments, par exemple X3D ou head, seront des nœuds de l'instance du schéma XML. Les nœuds de l'arbre peuvent avoir des fils qui seront des éléments déclarés dans le schéma. Ainsi dans le script 2.1,

l'élément X3D ne pourra avoir comme fils que l'élément head et l'élément Scene. Les règles associées au schéma sont les suivantes :

1. Un élément peut contenir une arborescence.
2. Une arborescence est toujours composée d'éléments.
3. L'utilisation conjointe de 1 et 2 permet de modéliser tous les nœuds de l'arborescence.
4. Les nœuds de l'arborescence sont ordonnés.

2.1. Création de la base de données

Pour créer la base de données, nous avons procédé en deux étapes. Tout d'abord, au niveau conceptuel, nous avons défini un ensemble de règles de passage du schéma X3D dans un schéma entité-association dont les deux plus importantes sont exposées dans cette section. Ensuite, nous avons traduit ce schéma dans un schéma relationnel. A ce niveau physique, nous avons optimisé la base de données.

Niveau conceptuel : Plusieurs règles ont été développées au niveau conceptuel. Pour illustrer cette partie, nous utiliserons l'exemple présenté dans le script 2.2.

```
<element name= "TextureTransform">
  <type content= "empty">
    <attribute default= "0 0" name= "center"/>
    <attribute default= "0" name= "rotation"/>
    <attribute default= "1 1" name= "scale"/>
    <attribute default= "0 0" name= "translation"/>
    <attribute name="DEF" type="ID"/>
    <attribute name="USE" type="IDREF"/>
  </type>
</element>
```

Exemple 2.2. Elément "TextureTransform"

Règle 1. Gestion des éléments "Un élément peut être composé d'une arborescence".

Cette règle implique une clé externe d'Arborescence pour chaque table élément. De plus, chaque élément du schéma aura une table composé d'un identifiant (**Nomtable_Id**) et d'un ensemble d'attributs. L'application de cette règle sur notre exemple donne l'entité présentée en figure 2.1. Cet exemple ne contient pas de clé externe car l'élément **TextureTransform** ne peut être composé d'un autre élément. Dans le cas présent, nous créons une table par élément X3D, ce qui a pour effet de générer un nombre non négligeable de tables.

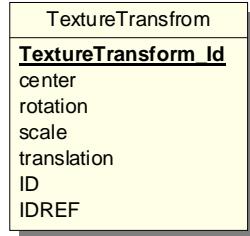


Figure 2.1. Entité TextureTransform

Règle 2. *Gestion des Nœuds* "Une arborescence est toujours composée d'éléments"

Pour traduire cette règle, nous avons créé une entité Arborescence et une entité **Nœud** modélisant la structure de l'arborescence. La modélisation de cette table nécessite l'utilisation de contraintes d'exclusion. Elles sont nécessaires pour modéliser le caractère optionnel de certaines relations entre les éléments. Par exemple l'élément **X3D** est formalisé sous la forme suivante :

$$X3D = (\text{head} \mid \lambda).Scene$$

Cette formalisation signifie que le noeud **X3D** est composé éventuellement d'un élément **head** et d'un élément **Scene** (l'élément λ étant l'élément vide). Dans un schéma XML le « et » et le « ou » logique sont spécifiés par l'attribut *order* dans la balise **<group>**. Si l'attribut *order* est égal à 'choice' cela signifie que la balise **<group>** est de type « ou », dans le cas contraire si elle est égale à 'seq' alors la balise **<group>** est de type « et ». La balise **<group>** est par défaut de type 'seq' (correspondant au « et » logique). La balise **<group>** a deux autres attributs. L'attribut **minOccurs** correspond au nombre minimum d'occurrences, par défaut 1 et l'attribut **maxOccurs** correspond au nombre maximum d'occurrences.

Les contraintes d'exclusion totale vont donc permettre de modéliser ces règles de grammaire. A partir de règles, nous avons réalisé le schéma entité-association de notre base de données. Ce schéma se compose de 103 entités dérivées de la règle 1, de 2 entités dérivées de la règle 2 (Entité **Nœud** et Entité Arborescence), de 104 relations et de 104 contraintes d'exclusions. L'ensemble des contraintes XML liées à la grammaire X3D est pris en compte grâce aux tables de contraintes d'exclusions.

Exemple : l'élément **TextureCoordinate** présenté dans le script 2.3. se formalise sous la forme:

$$\begin{aligned} \text{TextureCoordinate} &= (\text{Color}.ColorNode.(GeoCoordinate \mid \text{Coordinate}). \\ &\quad (\text{USE} \mid \text{ProtoInstance}) \mid \lambda)^* \end{aligned}$$

```

<element ref="TextureCoordinate"/>
<group minOccurs="0" order="seq">
  <element ref="Color"/>
  <element ref="ColorNode"/>
  <group order="choice">
    <element ref="GeoCoordinate"/>
    <element ref="Coordinate"/>
  </group>
  <group order="choice">
    <element ref="USE"/>
    <element ref="ProtoInstance"/>
  </group>
</group>
</element>

```

Exemple 2.3. Exemple d'élément complexe et multivalué

A ce niveau, le schéma entité-association fait place au schéma physique de la base de données. Ce passage a fait apparaître deux problèmes principaux dans notre phase de conception. Le premier problème réside dans les contraintes d'exclusion portées par les relations Nœud vers chacun des éléments. Leur traduction directe en relationnel surcharge de manière importante la taille de la base. Le second problème réside dans la table arborescence qui ne contient qu'un seul attribut clé primaire. Pour résoudre ces problèmes, nous avons considéré que tout document X3D manipulé était valide. Cette validation sera effectuée par un parseur, en amont de la base de données. Ainsi, le schéma relationnel peut être optimisé en fonction de son utilisation et non plus en fonction de la grammaire du langage X3D. En résultat, nous avons supprimé la table arborescence, les contraintes d'exclusion et nous avons modifié la table Nœud.

Nœud (Noed_Id, Elt_Id, Ordre, Type_Elt)

L'apport de l'attribut Type_Elt va permettre d'identifier la table élément liée. La clé primaire a aussi été modifiée, elle est composée de l'identifiant de Nœud et de l'Ordre d'apparition du nœud dans l'arbre. Les autres tables restent identiques. Le schéma final contient 104 tables. Nous avons un nombre conséquent de tables par rapport à l'approche d'Oracle mais notre méthode apporte deux avantages. Premièrement, la génération du script de création de base de données X3D est créée à partir du schéma XML, ce qui permet de suivre automatiquement les évolutions du langage X3D. La seconde, il faut autant d'interrogation de la base de données que d'attribut constituant un élément. Le fait de faire une table par élément supprime cette contrainte en faisant une simple sélection d'un tuple. En ce qui concerne la structure de l'arborescence, le nombre de requêtes pour l'extraction ou l'insertion est identique.

2.2. Application

Après avoir décrit la création du schéma de la base, nous allons maintenant présenter les processus d'insertion et d'extraction des scènes 3D.

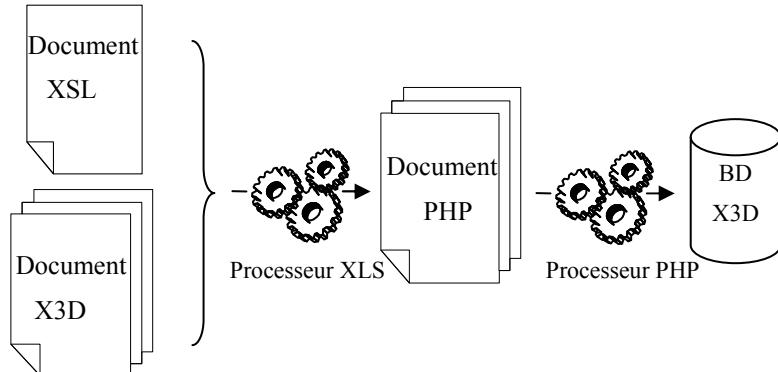


Figure 2.2. Architecture d'insertion

Processus d'insertion : Ce processus est basé sur l'utilisation de feuilles de style XSL. Les feuilles de style XSL ne fonctionnent pas avec la même logique que les autres langages de programmation, car les feuilles de styles permettent de manipuler des arbres. Nous avons fait deux modèles (un modèle peut être interprété comme une fonction dans un langage procédural) qui s'appellent récursivement. Ce procédé nous permet de parcourir n'importe quel type d'arborescence, quelle que soit sa taille. Le script 4 présente un extrait de la feuille de style XSL correspondante.

Pour compléter cette approche, nous avons géré dynamiquement la construction de scripts PHP qui seront créés en fonction du document X3D manipulé. L'architecture d'insertion mise en place est modélisée par le schéma présenté en figure 2.2. Chacune des flèches représente une action conjointement utilisée par les objets correspondant à l'origine de chacune des flèches. La feuille de style XSL qui va générer le script PHP d'insertion contient les éléments qui permettront d'établir la connexion avec la base de données.

```
...
<!-- Variable globale pour manipuler une valeur -->
<xsl:variable name="Noeud_courant" />

<!-- Modèle de départ -->
//La feuille de style se place à la racine
<xsl:template name="root" match="/">
  //Initialisation de la variable globale
  <xsl:variable name="Noeud_courant" value="0"/>
  // Appel de la boucle
  <xsl:call-template name="brother">
    <xsl:with-param name="PERE_ID" select="$Noeud_courant"/>
  </xsl:call-template>
</xsl:template>

<!-- Modèle brother pour passer en revue tous les frères -->
```

```

<xsl:template name="brother" match="/*">
//Pour chaque elt Passage en parametre id du pere
<xsl:param name="PERE_ID"/>
<xsl:for-each select="*">
<xsl:value-of select="$PERE_ID"/>
<xsl:value-of select="name()"/>
<xsl:number level="any" count="*" format="1.0 "/>
//Calcul du numero du noeud courant
<xsl:variable name="Noeud_courant">
  <xsl:number level="any" count="*" format="1.0 "/>
</xsl:variable>
<xsl:if test="count(>0)">
//Si le nœud courant a des fils alors on continue
//Alors appel du modèle pour le traitement récursif
<xsl:call-template name="child">
<xsl:with-paramname="PERE_ID" elect="$Noeud_courant"/>
</xsl:call-template>
</xsl:if>
</xsl:for-each>
</xsl:template>
...

```

Script 2.4. Exemple de feuille de style XSL récursive

Elle va tester chaque nœud et déterminer son type. Une fois le type déterminé, elle crée le script PHP d'insertion correspondant. Ce script PHP réalise l'insertion de l'élément courant dans la bonne table en exécutant des requêtes SQL. Il crée une variable globale portant le nom du numéro de nœud courant dans l'arbre. Cette variable a pour valeur l'identifiant du nœud père. A la fin, cette feuille crée un script de fermeture de la base de données. Une fois les scènes stockées dans la base de données, il est très simple de les manipuler à l'aide du SQL. Les valeurs de tous les attributs peuvent être modifiées. Des insertions, modifications, suppressions d'objets 3D dans plusieurs scènes peuvent être effectuées simultanément. Il est possible de faire évoluer dynamiquement des attributs de la scène pour avoir un rendu plus réaliste. Par exemple, positionner une source lumineuse dans une scène représentant le soleil et faire changer sa position au cours du temps.

Processus d'extraction : Le processus d'extraction des scènes 3D est basé sur le même procédé que le processus d'insertion. Un script PHP extrait les données de la base à l'aide d'une requête SQL qui va reconstituer les données de l'arborescence X3D. Ce script formate ensuite ces données sous forme de texte et de balises pour la restitution de la scène dans un format X3D. Une feuille de style XSL est appliquée pour traduire directement le document X3D en script VRML. Il est possible alors de visualiser à l'aide d'un navigateur la scène 3D correspondante. Ce processus d'extraction n'est pas limité uniquement à l'affichage des scènes 3D. La requête SQL d'extraction peut aussi porter sur des objets qui composent les scènes. A l'intérieur de la base de données la scène n'existe plus comme entité individuelle comme elle peut l'être dans un modeleur. C'est le contenu de la scène qui est stocké. Un des avantages de cette organisation est la possibilité d'extraire un objet particulier de plusieurs scènes à la fois. Au lieu d'interroger les

différentes scènes une par une, il suffit d'interroger la table contenant les instances de l'objet pour toutes les scènes.

Modifications des données : Une scène est modélisée sous la forme d'un arbre. Il en est de même pour les objets. Ainsi un objet regroupe un ensemble de composantes. Par exemple, une table est constituée d'un plateau et de quatre pieds. Il est possible de définir des attributs d'apparence pour chacun des éléments ou bien d'une façon générale pour tous les éléments. La plupart du temps les objets sont définis par une balise Shape contenant la description physique et visuelle de l'objet comme ses dimensions et sa couleur, voir figure 2.3. Il est possible de définir un nom pour chaque balise Shape donc lorsque l'on connaît le nom de l'objet il est possible d'avoir accès aux éléments de l'objet et de leurs caractéristiques. La modification des attributs s'opère à l'aide de requêtes SQL.

```
select SHAPE_ID, DEF from SHAPE where DEF is not NULL;
```

Cette requête permet de sélectionner l'ensemble des balises Shape portant un nom et de récupérer l'identifiant correspondant. Ensuite à l'aide de requêtes récursives on peut extraire les balises Material qui correspondent aux caractéristiques de la couleur.

```
update MATERIAL set TRANSPARENCY = '0.3' where MATERIAL_ID ='10';
```

Cette requête permet de modifier la valeur de la transparence de la balise Material dont l'identifiant est 10. L'avantage majeur de cette méthode réside dans le fait que cette modification s'opère pour l'ensemble des scènes contenant l'objet modifié. L'exécution automatique de la modification génère un gain non négligeable en terme de temps.

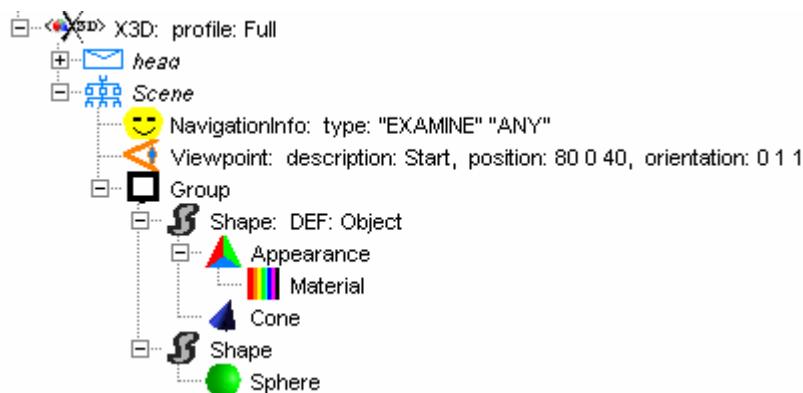


Figure 2.3. Arbre X3D

Retours d'expériences : La structure de la base de données X3D est calquée sur la structure des fichiers X3D. Par conséquent, cette base de données contient des arbres représentant les scènes 3D. Ceux-ci sont maintenant manipulables à l'aide du langage SQL et par rapport aux relations entre les éléments constitutifs de la scène. Grâce à cette nouvelle structuration des fichiers X3D, de grandes quantités d'informations sont manipulables de manière automatique, par exemple la modification d'information lors de l'insertion et de l'extraction de données. Mais les manipulations automatiques sont difficiles à mettre en place sans la connaissance de la nature même des objets. En effet, si l'on désire modifier la couleur d'une géométrie, celle-ci ne peut être modifiée qu'en fonction de critères géométriques ou de critères relationnels entre les éléments. Par exemple, un traitement courant consistant à créer des boîtes englobantes de géométrie, comme le remplacement de sphères en cubes, permet de simplifier la scène. Ces modifications sont rapides et réalisées directement sur les géométries stockées dans la base de données. Par contre, il faut connaître la nature de l'objet si l'on désire réaliser des modifications sur des critères sémantiques propres aux objets de la scène. En effet, comment modifier automatiquement la géométrie ou l'apparence d'une géométrie sans savoir ce qu'elle représente et sans savoir la fonction même de cet objet dans la scène ? Par conséquent, bien qu'il existe une sémantique propre aux objets de la scène (caméra, lumière, objet géométrique, etc.) les modifications ne peuvent être apportées sans une intervention humaine ayant une connaissance sur la nature des objets.

Tout naturellement, nous sommes venus à la conclusion, que le sens des objets dans une scène est nécessaire pour lui appliquer un traitement viable et intelligible. D'ailleurs, cette démarche est réalisée de manière manuelle lorsque les processus sont lancés par l'utilisateur pour apporter des modifications aux éléments constituants la scène 3D. On ne modifie la couleur d'un objet que si l'on connaît sa nature et que l'on veut qu'elle apparaisse d'une manière bien précise, car implicitement on connaît la fonction et la raison d'être de l'élément dans la scène. Par conséquent, le traitement des propriétés géométriques et visuelles ne peut être dissocié de la nature de l'objet pour que la scène garde une cohérence pour l'interprétation humaine après modification due à des processus automatiques. La nature des objets permet d'ailleurs de les extraire en fonction de leur sémantique, ce qui à terme permettra de générer des scènes 3D en fonction d'un contexte [BARB97, LIPK99]. De plus, l'ensemble des natures possibles des objets constituant les scènes est à la base de la création de filtres d'objets pour l'extraction d'information depuis la base de données. À dire vrai, les scènes générées dynamiquement, dues à l'extraction filtrée des objets, fournissent une vue partielle de la scène initiale. De plus, cette vue partielle contient les informations les plus pertinentes. Donc, l'extraction guidée par un filtre formant un contexte d'utilisation des scènes 3D stockées dans la base de données a l'avantage de

réduire le nombre d'objets de la scène et donc sa taille. Par conséquent, le temps de transfert des données sur le réseau est réduit pour un meilleur confort de l'utilisateur dans le cas de scènes de très grandes tailles.

L'utilisation de la sémantique des objets a d'autres avantages, car cette sémantique forme un lien entre diverses informations hétérogènes autour de la géométrie. En effet, l'objet géométrique sert d'index à des informations textuelles, notamment des fiches techniques concernant l'objet ou tout autre type de text. Cette géométrie peut aussi être une vue simplifiée d'un modèle trop complexe pour être affiché tel quel dans la scène. Alors, il est possible d'interroger le modèle complexe à partir de sa représentation simplifiée. La scène 3D se transforme alors en système d'indexation d'informations. La recherche est intuitive, car elle est réalisée en navigant à travers la scène 3D. Une fois l'objet de la recherche trouvé, celui-ci donne accès à un index secondaire sur les informations hétérogènes de l'objet.

Chapitre 3

« Il y a des choses que l'on ne peut savoir, faire ou expliquer. »
« Savoir s'arrêter devant l'incompréhensible est la suprême sagesse. »
(Tchouang Tseu - philosophe chinois)

Intégration et Manipulation de Données Hétérogènes : Un Etat de l'Art

Sommaire

1. Intégration de données.....	38
1.1 Architectures d'intégration de données.....	39
1.2 Intégration de services.....	43
1.3 Conclusion.....	48
2. Approche sémantique de l'intégration.....	49
2.1 La notion d'ontologie.....	50
2.2 Typologie des ontologies	52
2.3 Les langages ontologiques	53
2.4 Intégration de données basée sur une ontologie.....	59
3. Conclusion.....	61

Dans le chapitre précédent, nous avons montré qu'il existait de nombreux domaines liés à l'image et plus particulièrement à l'image de synthèse. Nous avons montré que les points clés des recherches menées dans ces domaines sont : une plus grande rapidité d'exécution des algorithmes de calcul, une modélisation et représentation toujours plus réaliste et enfin une meilleure interaction avec le monde réel. Nous avons vu qu'il était possible de coupler ces points clés en utilisant une base de données qui nous permettait de travailler sur une partie de l'image pour optimiser les calculs, en association avec de la sémantique pour que l'interaction avec le monde réel soit facilitée.

Néanmoins, l'association des concepts définissant une image avec des concepts définissant de la sémantique métier n'est pas encore résolue. Dans ce chapitre, nous allons présenter un ensemble de définitions et de méthodes qui proposent des solutions pour intégrer des données hétérogènes dans un ensemble homogène. Dans le domaine des bases de données, ce problème a été largement étudié depuis plus de 20 ans. La première partie de ce chapitre présente l'historique de

l'intégration de données au travers de différentes approches allant de la traduction de schémas à l'utilisation de Web Services en passant par les approches distribuées ou fédérées. De cette présentation, nous conclurons que le principal besoin lors de l'intégration est l'utilisation d'un modèle pivot ou d'un langage commun qui permet à la fois d'assurer l'homogénéité de représentation de l'information (résoudre l'hétérogénéité structurelle et syntaxique) et de préserver la sémantique des données lors du processus d'intégration (résoudre l'hétérogénéité sémantique). Dans cette partie nous verrons que l'utilisation du méta-langage XML permet de résoudre l'hétérogénéité structurelle et syntaxique des données. La seconde partie de ce chapitre concerne les travaux liés à la représentation sémantique des données et principalement aux solutions qui permettent de résoudre l'hétérogénéité sémantique. Les principaux travaux dans ce domaine concernent les ontologies. Nous verrons dans ce chapitre que de nombreuses ontologies sont construites à partir du méta-langage XML.

1. Intégration de données

L'intégration de données est l'action de faire entrer des données dans un ensemble. Cet ensemble est de nature très variée. Dans le cadre du Web, l'intégration d'un document dans un autre consiste à créer des liens hypermédia entre les documents à l'aide d'URL. Les documents associés sont des documents multimédias comme du texte, des images, de la vidéo, du son ou tout autre format de fichier. L'intégration de bases de données est aussi une forme d'intégration de données. Dans le cadre des bases de données, l'intégration de données provenant de différents systèmes d'information rend les informations plus complètes et plus pertinentes dans l'optique d'une application plus large. Par exemple, les sources de données biomédicales ont la particularité d'être hyper-liées, car les descriptions des objets proposent des hyperliens permettant à l'utilisateur de naviguer d'un objet à l'autre dans des sources de données multiples. En effet, il existe plus de cent bases de données génétiques, deux cent vingt-six sources de données de biologies moléculaires, etc. [BAX00, BEN00]. Un second exemple d'activité qui repose de plus en plus sur l'intégration de sources de données complexes est l'ensemble des métiers de gestion de l'information géographique. De même que les données biologiques et médicales, les données géographiques sont hétérogènes et distribuées. Ces données sont par exemple météorologiques¹ ou cartographiques². L'intégration de données de diverses sources permettra donc de réaliser des requêtes plus complexes et précises, donnant une valeur ajoutée aux différents systèmes d'informations disponibles. L'objectif est de tirer profit de la diversité des informations disponibles et de l'essor des nouvelles technologies de l'information et de la

¹ Météo France : <http://www.meteofrance.com/FR/mameteo/prevPays.jsp?LIEUID=FRANCE>

² Cartes de France : <http://www.viamichelin.com/>

communication. De plus, la réutilisation des données et des services permet d'optimiser les coûts à l'acquisition et la maintenance des informations. Finalement, la gestion et le coût des ressources informatiques sont répartis sur l'ensemble des acteurs fournisseurs de données.

Depuis 20 ans, de nombreuses architectures d'intégration de données ont été proposées. Dans cette section nous présentons d'une part, les architectures classiques telles que l'approche distribuée, l'approche fédérée et l'approche par médiation et d'autre part les architectures orientées services basées sur XML et les Web Services.

1.1 Architectures d'intégration de données

Dans cette section, les différentes approches d'intégration de systèmes d'informations seront présentées. Ces approches ont été proposées au cours du temps en réponse à des problèmes de coopérations de systèmes d'information. La première de ces approches concerne la traduction de schémas. Dans cette approche, chacun des systèmes d'informations souhaitant coopérer envoie une vue de son schéma local aux autres systèmes. Cette vue doit être traduite autant de fois que de systèmes distants hétérogènes. Cette approche qui nécessite la définition de $N^*(N-1)$ traducteurs pour N sources hétérogènes est très lourde à mettre en place et à maintenir [NIC05]. Nous n'avons retenu que les approches qui utilisent un modèle global pour intégrer les données (figure 3.1). Ces approches sont présentées brièvement pour que le lecteur se familiarise avec les définitions et les concepts qui seront utilisés dans la suite de ce mémoire.

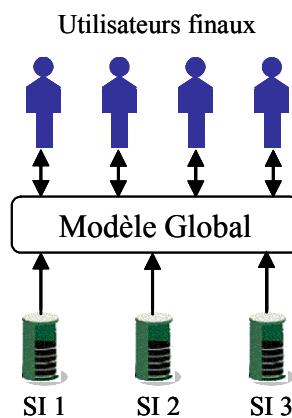


Figure 3.1. Architecture à base de Modèle Global

Approche distribuée : Cette approche est l'une des premières solutions opérationnelles pour résoudre le problème de partage de l'information [BAT83]. Ce partage est réalisé à travers plusieurs sites homogènes ou hétérogènes distants. Les systèmes locaux n'ont aucune autonomie lorsqu'ils sont intégrés à un système distribué. L'utilisateur passe donc par une interface globale pour toutes manipulations des données du système distribué.

Les systèmes distribués possèdent une architecture sur cinq niveaux (Figure 3.2). Le premier niveau concerne les données physiques locales. Le deuxième niveau représente le schéma interne de chaque site. Les schémas internes sont hétérogènes lorsque les sites locaux possèdent des modèles différents. Le troisième niveau contient le schéma de répartition qui fournit les informations de fragmentation et de répartition. Le quatrième niveau est le niveau conceptuel qui regroupe toutes les informations des différents sites dans un seul schéma global. Le dernier niveau représente les vues des utilisateurs sur le schéma conceptuel. Ce niveau est hétérogène, car chaque vue est représentée selon le modèle de chaque site.

La traduction de modèle dans les bases de données distribuées hétérogènes intervient d'une part entre les schémas internes et le schéma conceptuel. Pour constituer le schéma global conceptuel, correspondant au niveau quatre, il est nécessaire de rendre homogène la représentation des schémas internes. Il faut donc rajouter un niveau qui à partir des schémas internes hétérogènes construit des schémas internes homogènes. D'autre part, la traduction intervient entre le schéma conceptuel et les différents schémas externes. Ceci permet aux utilisateurs de chaque site d'interroger la base de données distribuée et de récupérer les informations dans leurs propres modèles.

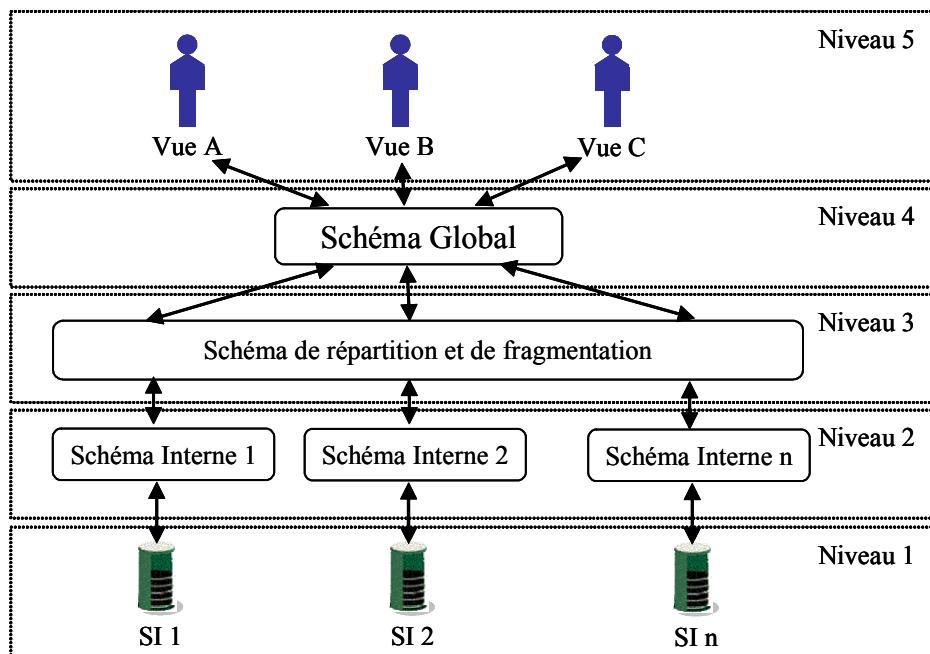


Figure 3.2. Architecture Distribuée

Approche fédérée : Dans les systèmes fédérés, plusieurs systèmes de bases de données préexistantes hétérogènes et autonomes coopèrent pour partager une partie de leurs données. Contrairement aux systèmes distribués, les utilisateurs locaux continuent à accéder aux données locales sans affecter la coopération. L'intégration des structures de données est réalisée soit à

l'aide d'un schéma fédéré, on parle de couplage fort, soit à l'aide de vues fédérant plusieurs systèmes d'informations, on parle de couplage faible. L'approche fédérée est caractérisée par la traduction des schémas locaux en schéma composant dans le modèle commun de la fédération. Sheth dans [SHE90] définit une architecture en cinq niveaux (Figure 3.3). Le premier niveau représente le schéma local de la base de données. Le deuxième niveau représente le schéma composant qui est le résultat de la traduction des schémas locaux dans le modèle commun de la fédération. Le troisième niveau concerne le schéma d'export qui représente le sous-ensemble du schéma composant qui sera utilisé dans la fédération. Le quatrième niveau concerne le schéma fédéré qui est une intégration des différents schémas d'export. Le dernier niveau concerne le schéma externe qui définit un schéma pour un groupe d'utilisateurs. Ce dernier schéma est une vue du schéma fédéré.

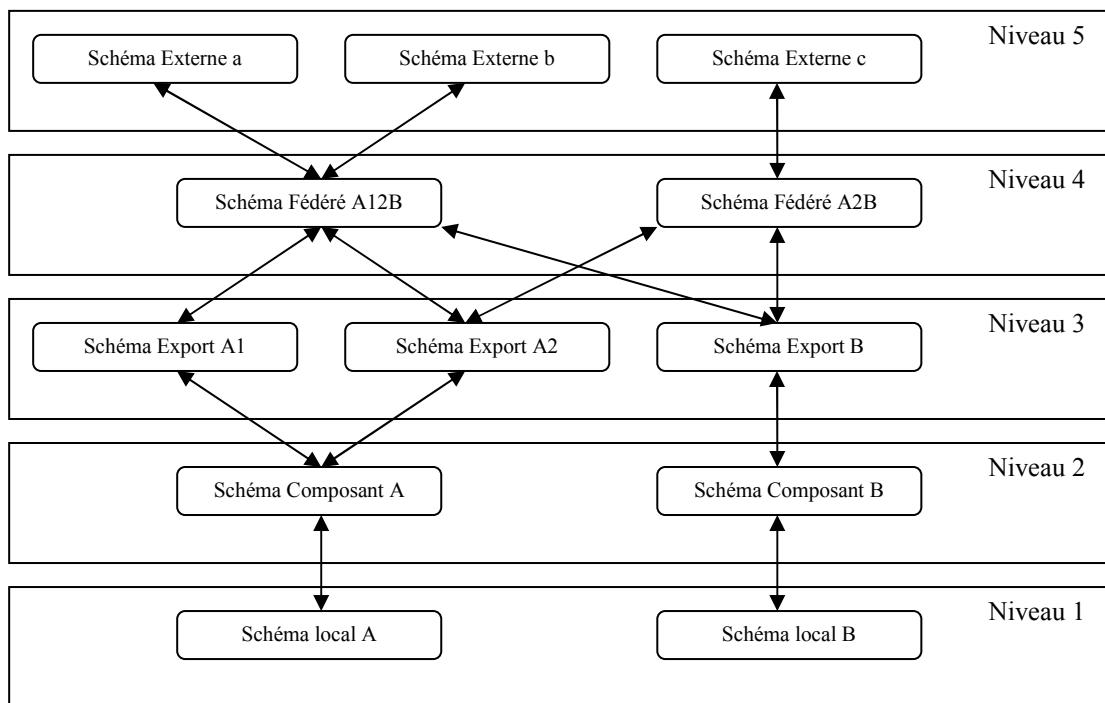


Figure 3.3. Architecture Fédérée

Deux types d'approches distinctes sont utilisés dans les systèmes fédérés :

- L'approche fédération fortement couplée qui consiste à construire un schéma fédéré qui intègre les différents schémas d'export. Dans ce type de systèmes, l'intégration des données est faite par un administrateur de la fédération qui définit un schéma unique. Toutes les manipulations réalisées par les utilisateurs se font sur ce schéma fédéré. La traduction intervient dans ce système pour homogénéiser les schémas conceptuels locaux avant de les intégrer. La traduction intervient aussi pour traduire les schémas externes pour qu'ils soient compréhensibles par un utilisateur n'utilisant pas le modèle qui a servi à la première phase de traduction.

-
- L'approche fédération faiblement couplée intègre les données des différents systèmes au coup par coup en fonction des besoins. Dans cette architecture, chaque administrateur ou utilisateur intègre les schémas externes des autres bases de données. Le système multi-bases est un cas particulier de ce type d'architecture. Dans ce système, les utilisateurs utilisent un langage de manipulation commun pour accéder aux bases de données locales. Un langage multi-base permet d'interroger plusieurs systèmes à la fois, ou de définir des vues sur plusieurs systèmes.

Approche par médiation : L'approche par médiation est une extension des approches fédérées. Cette approche est bien mieux adaptée pour un environnement dynamique où le nombre de systèmes d'informations est sans cesse changeant. Un système de médiation fournit un point d'accès unique à un ensemble de sources de données. Il est constitué d'un système global de gestion, nommé médiateur et d'un ensemble de sources dites locales. Les sources locales sont intégrées dans un seul schéma global qui est construit à partir des schémas exportés par les sources locales. Des requêtes globales sont ainsi posées sur le schéma global et traité par le médiateur. Le médiateur est chargé de la localisation des systèmes d'informations et des données pertinentes, il résout de manière transparente les conflits de données. Le médiateur sert d'interface entre les utilisateurs d'une coopération et les informations partagées. Le médiateur est un administrateur de bases de données qui résout les conflits entre les différents schémas et traduit le langage uniforme en sous-langages adaptés aux bases de données. Le wrapper est un outil qui permet à un ou plusieurs médiateurs d'accéder aux informations. Le wrapper fait le lien entre la représentation locale des informations et leur représentation dans le modèle de médiation. Ces éléments transforment les sous-questions dans le langage de la base à laquelle il est rattaché et reformule la réponse pour le médiateur.

Dans les systèmes de médiation, l'utilisateur n'a pas connaissance a priori des schémas des différentes sources, et pose sa requête sur le schéma global. Le rôle du système de médiation consiste à reformuler la requête en termes de schémas locaux à l'aide des wrappers, puis à la décomposer en sous-requêtes, qui seront envoyées aux différentes sources. Les résultats sont ensuite renvoyés au médiateur, qui intègre avant de les envoyer à l'utilisateur.

L'approche par médiation est composée de trois niveaux (Figure 3.4). Le premier niveau est constitué des sources d'informations locales. Le deuxième niveau englobe tous les outils nécessaires à l'interopérabilité des sources (médiateur et wrapper). Le dernier niveau est consacré aux applications, constituant l'interface homme machine qui constitue un point d'entrée uniforme avec un seul langage.

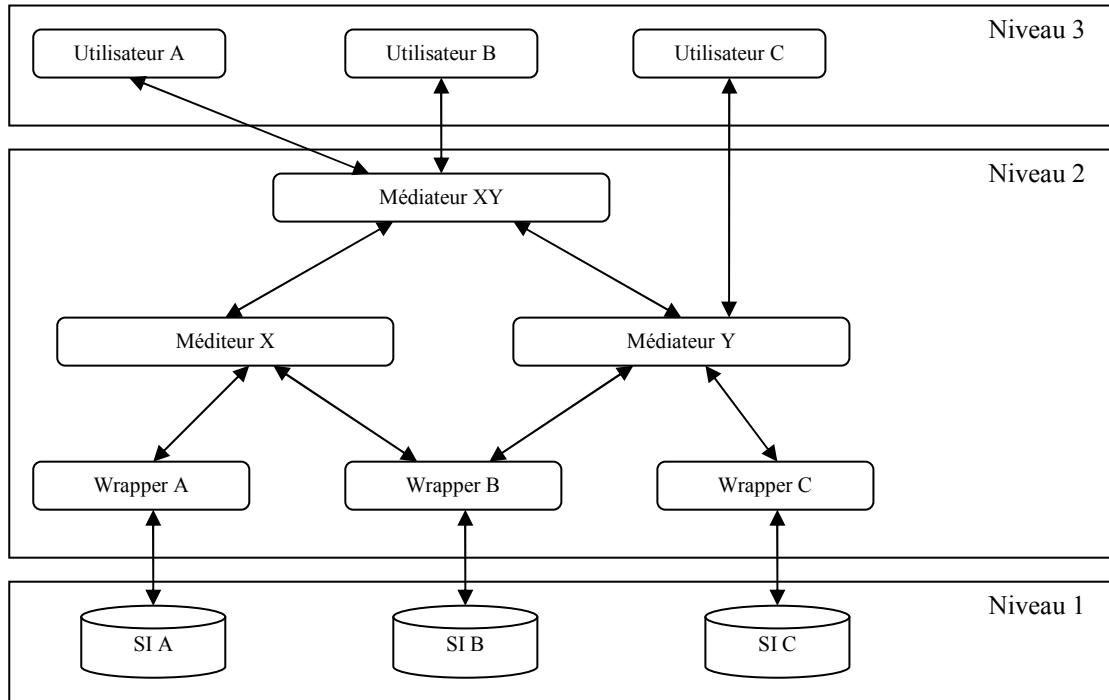


Figure 3.4. Architecture de Médiation

Nous venons de présenter les principales approches pour construire une intégration de systèmes d'informations orientées sur les données. Ces approches utilisent toutes un modèle commun pour représenter la connaissance des systèmes d'informations locaux. Nous avons vu que dans l'approche par médiation, le médiateur offre beaucoup plus de fonctionnalités que le modèle commun (localisation des données, résolution des conflits...). Cette évolution correspond à un réel besoin en matière d'interopérabilité, besoin concernant non seulement les données mais aussi les processus de localisation et d'accès aux données. Nous allons maintenant présenter les architectures orientées services qui proposent des solutions d'intégration de données en définissant des normes de localisation et d'accès aux données.

1.2 Intégration de services

Les architectures orientées services sont essentiellement des collections de services qui communiquent entre eux. La communication peut être une requête d'accès aux données ou une coordination de plusieurs services pour des activités plus complexe. Les architectures orientées services ne sont pas nouvelles. Pour beaucoup, la première architecture a été DCOM « Object Request Brokers(ORBs) » basé sur les spécifications CORBA. La figure 3.5 illustre le principe de l'architecture orientée services. Cette figure montre à droite un consommateur de service qui interroge un fournisseur de services à gauche. Le fournisseur de services retourne un message au consommateur de service.

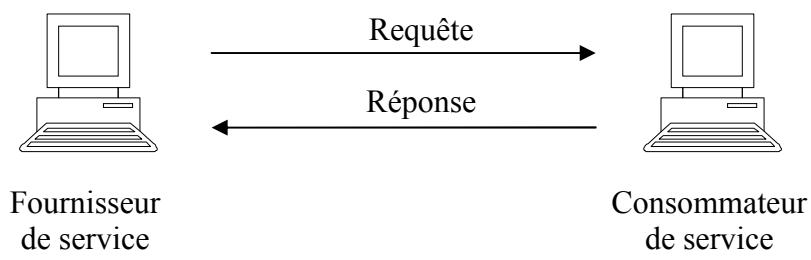


Figure 3.5. Architecture orientée services

La combinaison de services interne et externe à une organisation crée une architecture orientée services. Un service Web est une interface qui décrit une collection d'opérations qui est accessible sur le réseau à l'aide de la standardisation XML des messages échangés. Un service Web est décrit à l'aide de XML et inclut tous les détails pour l'interaction entre les services. Cela recouvre le format des messages, le format de transport et le format de localisation. Cette interface cache tous les détails de développement des services, ce qui permet d'utiliser les services indépendamment du matériel ou des plateformes logicielles sur lesquelles les services sont implémentés, ainsi que du langage de développement dans lequel les services sont développés. Il existe plusieurs manières d'implémenter un service Web. La plus connue utilise trois normes que sont SOAP, UDDI et WSDL.

Le concept des Web Services s'articule actuellement autour des trois acronymes suivants :

- SOAP (Simple Object Access Protocol) est un protocole d'échange inter-application indépendant de toutes plate-formes, basé sur le langage XML. Un appel de service SOAP est un flux ASCII encadré dans des balises XML et transporté dans le protocole HTTP par exemple.
- WSDL (Web Services Description Language) donne la description au format XML des Services Web en précisant les méthodes pouvant être invoquées, leurs signatures et le point d'accès (URL, port, etc.). C'est, en quelque sorte, l'équivalent du langage IDL pour la programmation distribuée CORBA.
- UDDI (Universal Description, Discovery and Integration) normalise une solution d'annuaire distribué de Web Services, permettant à la fois la publication et l'exploration. UDDI se comporte lui-même comme un service Web dont les méthodes sont appelées via le protocole SOAP.

Le format de description d'un service Web est en XML. XML est un langage à balises pour la structuration de l'information, mais il est aussi un méta-langage permettant de créer de nouveaux

langages. Développé par le W3C avec une première version datant du 10 février 1998, XML hérite du langage SGML (Standard Generalized Markup Language) réduit au strict minimum utilisant une syntaxe simple et flexible. Les objectifs fixés à l'origine par le W3C étaient de permettre la conception rapide d'application et surtout plus simple que SGML. Par conséquent, les documents XML sont faciles à créer, gérables par un langage de requêtes, lisibles, claires et facilement traitables par des programmes. XML n'est pas un langage de programmation, mais est un langage d'échange de documents structurés. Son rôle premier est de structurer l'information, à l'aide de balises en marquant les éléments composant les documents et les relations entre ces éléments. Ces balises sont redéfinissables pour obtenir de nouveaux modèles de documents. Sur le plan conceptuel, un document XML se décompose en deux parties. La première concerne le document descriptif de l'information proprement dite et le deuxième concerne la description structurelle du document descriptif. Les descriptions structurelles optionnelles des documents sont les DTD (Document Type Description). Une DTD contient les règles syntaxiques que doit respecter le document pour qu'il soit valide. Un document est « valide » s'il respecte la ou les DTD et s'il est « bien formé » en respectant les règles d'écritures générales d'un document XML comme l'imbrication correcte des balises. Une DTD définit les règles de grammaire pour écrire un document valide, il contient donc une liste d'éléments correspondant aux noms des balises. En effet, chaque élément est représenté par un mot réservé et l'ensemble des mots constitue le vocabulaire autorisé nommé aussi « espace de noms ou NameSpaces ». Pour pallier le manque d'expressivité et de contraintes typologiques sur les éléments définis dans une DTD, le W3C propose de définir les règles de grammaire à l'aide d'un schéma XML. Le but d'un schéma est de définir une classe de documents XML, par conséquent le terme « instance » est souvent employé pour nommer un document XML qui se conforme à un schéma particulier. Le traitement d'un document implique la lecture complète du document. Ce processus consiste à interpréter la structure hiérarchique des éléments et analyser chaque élément de cette structure.

L'architecture orientée service résout les nombreux problèmes syntaxiques et structurels (par l'utilisation d'XML) et les problèmes architecturaux de l'échange de l'information. Des « Interfaces de Programmation d'Applications » API ont été développées pour formater, manipuler et distribuer l'information sur le réseau. Ces API incluent des mécanismes d'échange de l'information entre services qui est une sur-couche du réseau Internet actuel. Les seuls composants à développer sont les services eux-mêmes. On gagne ainsi une grande souplesse de développement. Les services sont développés plus rapidement avec une plus grande robustesse. Toutes les approches d'intégration de l'information décrite dans la section précédente peuvent utiliser les Web Services. La seule obligation est le format de formatage des données orientées

XML. Actuellement, l'intégration des sources de données à l'aide de services web est un domaine de recherche en pleine expansion. Pour exemple, nous citons quelques projets en cours de développement dans ce domaine.

- Le projet e-XML [GAR02] cherche à résoudre les problèmes d'hétérogénéité des documents sur le Web à l'aide d'échange XML associé à la technologie de processus XQuery³. L'architecture d'e-XML est en quatre couches dont la première est constituée d'une interface graphique Java⁴. Cette interface cliente utilise XForm⁵ pour la gestion des formulaires et SOAP/WSDL pour l'échange de données. La deuxième couche est composée d'un ensemble de fonctionnalités partagées à l'aide de médiateur. Un parseur XQuery génère des arbres de requêtes et une bibliothèque commune de gestion de schémas XML est employée pour assurer la cohérence sémantique des produits. Quand les multiples réponses des différents médiateurs sont retournées, le Composer combine les réponses pour en donner une seule. La troisième couche est un ensemble de fonctionnalités de bases génériques pour la gestion de la persistance des données. Cette couche est composée de deux modules, dont l'un gère le stockage et l'autre les processus de requêtes. La dernière couche est une collection de wrappers, dont chacun à une spécialité. Le wrapper SQL accède aux sources d'informations contenant une base de données relationnelle. Le wrapper personnalisable permet d'accéder à des applications. Le wrapper HTML accède aux sources d'informations Web. De plus, un dépôt de documents XML gère les documents à l'aide d'un SGBD objet/relationnel.
- Le projet Enosys [PAP02] est basé sur l'approche par médiation. Les wrappers sont nommés XMLizers accédant à différentes sources d'information hétérogènes et distribuées, et exportent des vues XML de ces sources. Le XMediator exporte la base de données « Virtuel Integrated XML » (VIX) qui contient toutes les vues individuelles exportées par les wrappers. La transformation et l'intégration sont rapides et définies avec « XML Catalog Query Language » (XCQL). XCQL est un langage de haut niveau pour la définition et le requêtage de vue sur XML. XMLLizers convertit les données structurées et semi-structurées dans une vue XML virtuelle. Aujourd'hui, Enosys fournit XMLLizers pour les bases de données relationnelles à l'aide de JDBC, un dépôt de documents HTML, utilise les services SOAP et offre une gestion de fichiers. Les outils XSDesign développés par Enosys permettent un développement rapide et personnalisé d'interface Web qui donne une vue intégrée des données pour un domaine d'expertise précis. Cette approche est très avantageuse, car elle

³ Xquery : <http://www.w3.org/TR/xquery/>

⁴ Java, Sun Microsystems : <http://java.sun.com/>

prend en compte tous les processus de manipulations des données allant du stockage jusqu'à sa visualisation, en passant par le partage de l'information. De cette manière, l'administrateur gère l'information à tous les niveaux.

- Le projet Xperanto [XPE00] d'IBM fournit des moyens de publier et de requêter des vues XML sur des sources de données relationnelles. Les utilisateurs utilisent le même langage de requêtes XQuery aussi bien pour réaliser des requêtes sur les vues que pour les créer. Xperanto crée automatiquement des vues XML par défaut. Ces vues XML sont des vues de bas niveau décrivant les données des bases de données relationnelles. L'utilisateur définit alors ses propres vues sur ces vues par défaut à l'aide de XQuery. L'avantage de cette approche est qu'aucun langage propriétaire n'est utilisé pour créer les vues et les requêtes. Le problème majeur est qu'il ne fournit pas de cadre de travail pour l'améliorer la construction de vues personnalisées, ainsi l'administrateur doit définir manuellement ses propres vues. Xperanto définit un wrapper pour une source de données relationnelles. Celui-ci s'intègre dans un système plus large alliant une approche par médiation.
- Le projet Nimble [DRA01] a été développé pendant deux ans pour être transformé en produit commercial d'intégration de données XML. Ce produit fournit une interface pour la collaboration d'un ensemble de sources de données variées. Le langage de requête supporté par le système Nimble est XML-QL⁶. Les utilisateurs et les applications interagissent en utilisant un ensemble de schémas de médiation. Ces schémas sont essentiels, car ils définissent une vue au dessus de chaque schéma local des sources d'informations. Cette approche est similaire à l'approche Global-As-View [CAL01]. Ces schémas peuvent être construits de manière hiérarchique. Chacun des schémas peut être défini comme une vue d'un ensemble de schémas représentant aussi une vue. Cette caractéristique apporte une grande flexibilité quand il faut définir beaucoup de classes d'utilisateurs et d'applications utilisant le système. Cette approche facilite l'intégration, car elle peut être entreprise de manière incrémentale. Quand une requête est posée au moteur d'intégration, celle-ci est parsée puis divisée en fragment pour chaque source de données cibles. Le « Compiler » traduit chaque fragment dans le langage approprié de requête pour chaque source de données. Le serveur de métadonnées contient la correspondance entre les différents langages pour permettre le découpage et la traduction correcte des requêtes XML-QL. Cette correspondance est réalisée à l'aide des outils de gestion.

⁵ XForm : <http://www.w3.org/MarkUp/Forms/>

⁶ XML-QL : <http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/>

-
- Le projet DENODO [PAN02] est construit sur une architecture médiation. Ce système a été utilisé dans diverses applications industrielles pour l'intégration de données. Ces applications intègrent plus de 700 sources d'informations comme des sites Web, des bases de données et des documents semi-structurés XML. La couche physique est composée de wrappers pour chaque source d'informations. Les wrappers sont générés semi automatiquement à l'aide d'outils. La couche logique nommée motrice d'agrégation Denodo est le noyau de la plateforme et contient le générateur de plan de requête, l'optimiseur de requête et le moteur d'exécution. Un outil d'administration est utilisé pour la gestion du dictionnaire données. Dans la plateforme d'intégration de données Denodo, chaque source exporte une combinaison de relations, nommées les relations de base. Ces relations de base suivent le modèle relationnel. Chaque relation dans une source représente les possibilités de requêtage à l'aide des méthodes de recherche. Chaque wrapper fournit un accès à une source de telle manière qu'une fois face au médiateur, il se comporte comme une table dans une base de données relationnelle. Une fois la base de relations définie et que les wrappers sont construits, chaque relation du schéma global est définie par une requête comportant les relations de base. Celles-ci sont définies par analogie aux vues des bases de données conventionnelles. Cette approche est connue comme une approche Global-As-View. Les requêtes sont réalisées dans un langage proche du SQL.

Tous ces systèmes utilisant XML comme base d'intégration de données ont une approche par médiation. Les services Web forment des ensembles de fonctions externes et des collections virtuelles interrogeables. Les fonctions permettent de faire des traitements comme la conversion d'une devise dans une autre. Pour l'interrogation, ces systèmes utilisent tous un langage de requête unique pour interroger les sources de données. L'intégration d'information nécessite un modèle commun qui fournit les concepts pour la définition d'un schéma global. Ce schéma global intègre tous les éléments des schémas locaux

1.3 Conclusion

L'évolution des nouvelles technologies de l'information et de la communication a changé le problème de l'intégration de l'information. L'apport d'XML pour définir non seulement des schémas d'intégrations mais aussi les langages de définition des modèles correspondants a réduit considérablement les problèmes liés à l'hétérogénéité structurelle et syntaxique. L'apport des technologies de communications liées aux architectures orientées services ont résolu les problèmes de localisation et d'accès aux données, permettant la conception d'architectures

d'interopérabilité sur une plus grande échelle [ABE02]. Néanmoins, lors du processus d'intégration des données et des services, il reste de nombreux problèmes liés à l'hétérogénéité sémantique.

La connaissance implicite du sens des éléments qui compose un schéma XML est une information primordiale pour l'intégration des données. En effet, si cette connaissance implicite est définie formellement dans une architecture, elle permet alors d'être la base de la mise en correspondance des types d'éléments entre schémas. Grâce à cette mise en correspondance du sens des types d'éléments entre schémas, les éléments des différents documents XML peuvent aussi être mis en correspondance dans une architecture d'intégration. Par exemple, il serait possible de définir un élément sémantique « Mur » à l'aide d'un schéma XML, de lui associer dynamiquement une représentation graphique 2D définie en SVG, une représentation graphique 3D en X3D et d'une fiche technique à l'aide d'une description XML définie par un fournisseur à partir d'un catalogue.

Le W3C développe des technologies dans le cadre du projet « Semantic Web »⁷. « Le Web Sémantique est une extension du Web actuel dans lequel les informations seront données dans un sens bien défini, plus compréhensible par les machines et l'homme pour un meilleur travail collaboratif » Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, Scientific American, May 2001.

Pour le bon fonctionnement du Web Sémantique, les ordinateurs doivent avoir accès aux collections structurées de l'information et l'ensemble des règles d'inférences qui permettent la conduite de résonnements automatiques. Le défi du Web Sémantique consiste à fournir un langage qui exprime des données et des règles pour raisonner sur les données et permettant à des règles de n'importe quel système existant de représentation de la connaissance d'être exportée sur le Web. Dans la suite de ce chapitre, nous allons présenter les concepts qui sont utilisés pour définir la sémantique et qui permettent l'intégration de données à l'aide de la sémantique.

2. Approche sémantique de l'intégration

Pour modéliser la sémantique des connaissances, les connaissances doivent être placées au niveau conceptuel, ainsi que leur structure cognitive. La représentation de la connaissance est indépendante de ce que l'on veut en faire. Par conséquent, la représentation des connaissances et les mécanismes inférentiels sont dissociés [GUA94a]. Par contre, la conceptualisation d'un domaine de connaissance ne peut se faire, de manière non ambiguë, sans un contexte d'usage précis. Un terme peut désigner deux concepts différents dans deux contextes d'usage différents [BAC00]. La sémantique des connaissances est fortement contrainte par les représentations

symboliques utilisées par les machines. Pour cette raison, N. Guarino [GUA94b] introduit un niveau ontologique entre le niveau conceptuel et le niveau épistémologique. Le niveau ontologique apparaît donc comme un pont entre la sémantique interprétative au travers de laquelle l'utilisateur interprète les termes fournis par la machine et la sémantique opérationnelle au travers de laquelle la machine manipule les symboles [DEC00].

La suite de cette section est articulée en quatre parties. La première partie définit ce qu'est une ontologie. La seconde partie présente les différents types d'ontologies. La troisième partie présente les principaux langages d'ontologie. La dernière partie présente des travaux d'intégration de données XML basée sur une ontologie.

2.1 La notion d'ontologie

Les ontologies ne sont définies que par rapport au processus général de représentation des connaissances. Un consensus général s'accorde à définir le rôle des ontologies par la formule suivante : « Une ontologie est une spécification explicite d'une conceptualisation » [GRU93]. La définition d'une ontologie intervient après la phase de conceptualisation et elle consiste à identifier les connaissances spécifiques au domaine de connaissances à représenter. «A conceptualisation is an abstract, simplified view of the world that we wish to represent for some purpose»⁸ [GRU93]. N. Guarino considère les ontologies comme des spécifications partielles et formelles d'une conceptualisation [GUA95]. Elles sont formelles, car elles sont exprimées sous forme logique. Elles sont partielles, car toute conceptualisation ne peut pas être formalisée entièrement soit à cause de certaines ambiguïtés de la conception, soit à cause du langage de représentation des ontologies qui ne peut les représenter entièrement. Les formalismes opérationnels ont une faible tolérance d'interprétation. En effet, les ambiguïtés posent problème aux traitements, ce qui oblige de passer d'une ontologie informelle à une ontologie totalement formelle donc non ambiguë. Le processus général de représentation des connaissances est caractérisé par trois étapes : La conceptualisation, l'ontologisation et l'opérationnalisation.

- La conceptualisation consiste à identifier les connaissances du domaine. Elle se divise en deux étapes. Tout d'abord, il faut trier les connaissances spécifiques au domaine et celles qui ne sont que l'expression des connaissances de ce même domaine. Ensuite, des choix doivent être exprimé sur la nature même des éléments conceptuels des connaissances extraites (concepts, relations, propriétés, règles, contraintes, etc.). L'usage de questions de compétences, c'est-à-dire de questions auxquelles le système est censé répondre, est préconisé

⁷ Semantic Web : <http://www.w3.org/2001/sw/>

⁸ « Une conceptualisation est une vue abstraite et simplifiée du monde dans un objectif précis »

[GRU95]. Ensuite une normalisation sémantique est nécessaire [BAC00]. La problématique de la conceptualisation est la mise en évidence des connaissances implicites et ne peut se faire que lors de la phase d'utilisation de l'ontologie. En effet, lors de la conceptualisation des connaissances, elles ne sont jamais exprimées, car elles vont de soi pour tous [LEC02]. Une fois les concepts et relations identifiés par leurs termes, il faut en décrire la sémantique en indiquant les instances connues, les liens qu'ils entretiennent et leurs propriétés. Il faut ensuite formaliser le modèle conceptuel obtenu au cours de la phase d'ontologisation.

- Une formalisation partielle va permettre de construire une ontologie. Afin de respecter les objectifs généraux des ontologies, T. Gruber propose cinq critères permettant de guider le processus d'ontologisation [GRU93]. La clarté et l'objectivité doivent être indépendantes de tout choix d'implémentation. Les axiomes doivent être cohérents, consistants du point de vue logique. L'ontologie doit être extensible, c'est-à-dire qu'il doit être possible de l'étendre sans modification. Les postulats d'encodage doivent être minimes, pour assurer une bonne portabilité. Le vocabulaire doit être minimum pour que l'expressivité de chaque terme soit maximum. Le respect de la sémantique du domaine doit être assuré par un engagement ontologique, notion proposée initialement par [GRU93] comme critère pour utiliser une spécification partagée d'un vocabulaire. Un engagement ontologique est une garantie de cohérence entre une ontologie et un domaine, mais pas une garantie de complétude de l'ontologie. N. Guarino définit l'engagement ontologique comme une relation entre un langage logique et un ensemble de structures sémantiques. Le sens du concept est donné par son extension dans le contexte d'interprétation du langage. Respecter l'engagement ontologique revient à donner à chaque concept son extension et à manipuler ce concept conformément au sens prescrit par cette extension. Ces engagements sémantiques et ontologiques doivent être garantis par une structuration sémantique des connaissances. Cette structuration est de plus nécessaire pour combler le fossé formel entre les connaissances conceptualisées et le formalisme pour les représenter en machine. L'ontologisation mène à la construction de hiérarchie de concepts, de relations et d'attributs. Une fois le modèle structuré, il faut le traduire dans un langage semi-formel de représentation d'ontologies. Parmi les langages de représentation développés au niveau conceptuel, trois grands modèles sont distingués : les langages à base de frames⁹, les logiques de description et le modèle des graphes conceptuels (section suivante). Quelques-uns de ces langages ou des langages utilisant ces modèles sont déjà opérationnels et les ontologies exprimées dans ces formalismes peuvent être directement utilisées en machine. Dans les autres cas, une opérationnalisation est nécessaire.

-
- L'opérationnalisation consiste à implémenter une ontologie pour permettre à une machine de manipuler des connaissances du domaine. Or, si beaucoup de langages utilisant les modèles cités précédemment autorisent l'expression de connaissances inférentielles, peu sont implémentés pour rendre possible la manipulation de ces connaissances. Le modèle des graphes conceptuels fait exception, car la représentation des connaissances sous forme de graphes permet de mettre en oeuvre des raisonnements par des opérations formelles sur les graphes (comparaisons, fusions, etc.). Dans le cas où le langage n'est pas opérationnel, il est nécessaire soit d'implémenter ce langage, soit de transcrire l'ontologie dans un langage opérationnel. Finalement, l'ontologie opérationnalisée est intégrée en machine au sein d'un système manipulant.

2.2 Typologie des ontologies

Les ontologies ont été définies par rapport à un processus général qui vise la représentation des connaissances. Il existe une très grande variété d'ontologies résultant de conceptualisations de problématiques pouvant être catégorisées (Figure 3.5). Les ontologies de domaine appartiennent à la catégorie la plus vaste. Ces ontologies décrivent un ensemble de vocabulaires et de concepts modélisant un domaine de connaissances. L'objectif de ces ontologies est de créer des modèles d'objets du domaine. Ces ontologies sont en sorte un méta-modèle de connaissances dont les concepts et les propriétés sont utilisés pour la déclaration des types d'objets. Les ontologies de tâches forment la deuxième catégorie des ontologies utilisées pour conceptualiser des tâches spécifiques à un domaine. Ces tâches sont des tâches de diagnostic, des tâches de planification des tâches de conception, des tâches de configuration, etc. Ces ontologies régissent un ensemble de vocabulaires et de concepts qui décrit une structure de résolution des problèmes inhérents aux tâches et indépendants du domaine.

Les ontologies de domaine et les ontologies de tâches sont placées sur le même niveau conceptuel. Les ontologies génériques sont sur un niveau d'abstraction supérieur tandis que les ontologies d'applications sont plus spécifiques que les ontologies de domaine et de tâche. Ceci signifie que les ontologies de niveau supérieur sont utilisables par les ontologies de niveau inférieur. Par conséquent, les ontologies génériques véhiculent des connaissances visant à résoudre des problèmes génériques, ou des connaissances génériques utilisables par différentes ontologies de domaine. Alors que les ontologies d'application sont utilisées dans une problématique applicative de la connaissance mêlant ontologies de tâche et de domaine. Les

⁹ Frame languages : <http://hopl.murdoch.edu.au/findlanguages.prx?NodeID=2231240&which=ByMyCat>

concepts spécifiés par les ontologies d’application correspondent souvent aux rôles joués par les entités du domaine tout en exécutant une certaine activité.

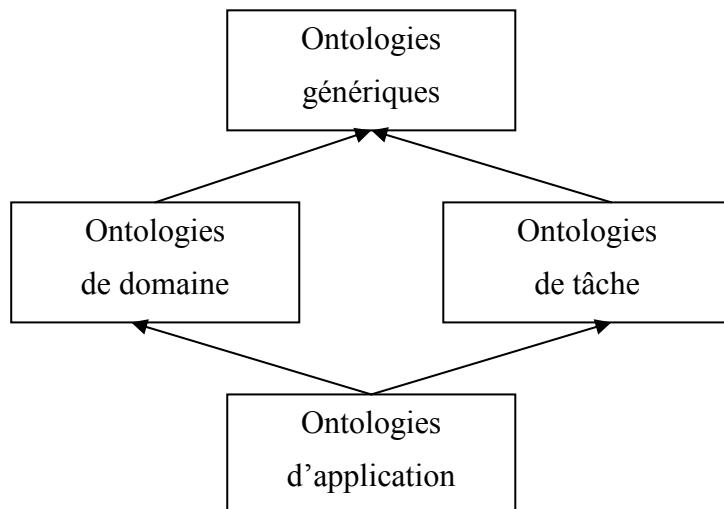


Figure 3.6. Typologie des ontologies

2.3 Les langages ontologiques

Parmi les langages de représentation développés au niveau conceptuel, trois grands modèles sont distingués : les langages à base de frames, les logiques de description et le modèle des graphes conceptuels. Introduit dans les années 70 en IA, le modèle à base de frames a depuis été adapté à d’autres problématiques puisqu'il a donné naissance au modèle objet. Une frame représente n’importe quelle primitive conceptuelle et est doté d’attributs (slot), qui peuvent porter différentes valeurs, et instances [KIF95]. Les logiques de description permettent de représenter les connaissances sous forme de concepts, de rôles et d’individus [KAY97]. Les rôles sont des relations binaires entre concepts et les individus sont les instances des concepts. Les propriétés des concepts, rôles et individus sont exprimées en logique des prédictats. Ce langage est utilisé dans le langage de représentation de connaissance OIL développé pour le Web. Introduit par Sowa au début des années 80 [SOW84], le modèle des graphes conceptuels se décompose en deux niveaux. Le niveau terminologique est le niveau qui décrit les concepts, les relations et les instances de concepts. Le niveau assertional est le niveau qui représente les faits, les règles et les contraintes sous forme de graphes où les sommets sont des instances de concepts et les relations des arcs. La figure 3.6 montre la superposition des langages ontologiques. Les couches supérieures utilisent les couches inférieures pour leurs définitions.

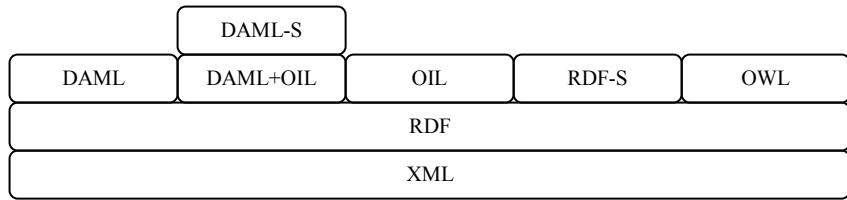


Figure 3.7. Langages ontologiques basés sur XML

Dans cette partie, nous présentons une liste non exhaustive des principaux langages ontologiques qui existent dans le domaine des bases de données et qui sont liés à XML.

Méta-données et Dublin Core: Dès la fin des années 1990, le besoin de donner des informations sur les données du Web s'est fait sentir. Pour créer de nouveaux processus et services, l'association de méta-données aux pages Web est nécessaire. Indépendamment d'autres organismes, le W3C s'est fixé comme objectif de définir un vocabulaire de méta-données pour la description des pages Web. Ce projet a donné naissance au Dublin Core. Grâce au vocabulaire de méta-données défini par le Dublin Core, il est désormais possible de définir des méta-données dans les pages Web comme le titre, l'auteur ou le sujet du document. De plus, le Dublin Core permet également de définir des relations entre documents.

RDF: Parallèlement au Dublin Core, le W3C a défini un format de méta-données plus générale permettant de définir des méta-données. Il s'agit du RDF¹⁰ (Resource Description Framework), langage de description d'informations sur le Web. Le modèle RDF définit trois types d'objets.

- Les ressources sont tous les objets décrits par RDF. Généralement, ces ressources peuvent être aussi bien des pages Web que tout objet ou personne du monde réel. Les ressources sont alors identifiées par leur URI (Uniform Resource Identifier).
- Les propriétés sont des attributs, des aspects, des caractéristiques qui s'appliquent à une ressource. Il peut également s'agir d'une mise en relation avec une autre ressource.
- Les valeurs sont les données particulières que prennent les propriétés.
- Ces trois types d'objets sont mis en relation par des assertions. Une assertion est un triplet constitué d'une ressource, d'une propriété et d'une valeur. Les composantes de ce triplet sont parfois nommées, sujet, prédicat, objet. Une description RDF est par conséquent un ensemble d'assertions.

¹⁰ RDF, <http://www.w3.org/TR/rdf-primer/>

Une ressource : <http://vision.u-bourgogne.fr/Le2i/>.
 Une propriété : date-de-modification.
 Une valeur : 09-01-2004.

Exemple 3.1. Ressource RDF

On peut alors construire l'assertion : (<http://vision.u-bourgogne.fr/Le2i/>, 09-01-2004), qui signifie la date de modification de la page Web du Le2i date du 9 janvier 2004. Il est possible de représenter les descriptions RDF par des graphes. Dans un graphe RDF, on représente par des ellipses les ressources nommées (c.-à-d. les objets qui possèdent des URI), et par des rectangles les littéraux (c.-à-d. les constantes, qui ne possèdent pas d'URI).

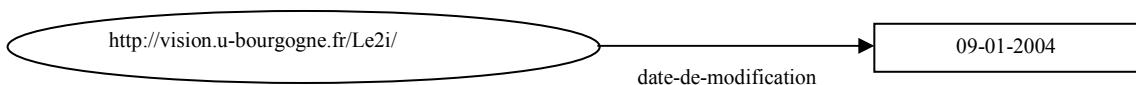


Figure 3.8 : Exemple de graphe RDF

RDF étant issu du W3C, les ressources RDF sont représentées à l'aide de XML. Ainsi, l'espace de noms XML RDF a été réservé pour l'inclusion de descriptions RDF dans des documents XML.

```

<rdf:RDF>
  <rdf:Description about="http://www.limsi.fr/">
    <s:date-de-modification>2003-02-11</s:date-de-modification>
  </rdf:Description>
</rdf:RDF>
  
```

Exemple 3.2. Ressource RDF en XML

On a vu qu'il est possible de représenter graphiquement des assertions RDF. En fait, une description RDF sous-tend un graphe. Les éléments noeuds de l'arbre XML, c'est-à-dire les ressources, correspondent aux noeuds du graphe. Les éléments propriétés correspondent aux arcs entre les noeuds. Dans l'exemple précédent, on utilise deux espaces de noms : rdf et s. On l'a vu, l'espace de nom rdf correspond aux balises utilisées pour décrire les descriptions RDF elles-mêmes. Quant à l'espace de noms s, il fait référence au schéma RDF utilisé. Jusqu'à présent, il n'a été spécifié nulle part quels étaient les attributs autorisés, à quelles ressources ils s'appliquaient, quelles étaient leurs valeurs admises. Tout comme un schéma XML spécifie la structure d'un document XML, le schéma RDF précise les structures possibles. Le schéma donne la sémantique à la description RDF. Le schéma définit le vocabulaire utilisé dans une description RDF. Les schémas sont eux-mêmes en RDF, en utilisant des balises de l'espace de nom du langage RDF Schema souvent désigné par le préfixe rdfs.

DAML+OIL : Les langages de communication entre agents tel que KQML¹¹, permettent aux agents de communiquer à l'aide de langages standardisés. Par contre, rien ne garantit que les données échangées entre les agents soient compréhensibles par ceux-ci. Le langage DAML¹² (DARPA Agent Markup Language) permet aux agents de partager le sens des données échangées. DAML est associé à OIL¹³ (Ontology Inference Layer), qui est un autre langage de description d'ontologies. Le couple DAML+OIL repose sur RDF. OIL est un langage de description et d'inférence sur les ontologies, basé sur RDF. Il prend appui sur les logiques de description. Il est composé de plusieurs couches. Le noyau correspond à peu de détail près au RDF Schema qui permet de décrire des vocabulaires. Le standard OIL permet de définir la sémantique de façon plus précise et de donner la possibilité d'utiliser les mécanismes d'inférence. L'instance OIL introduit les fonctionnalités des bases de données et le Heavy OIL est une extension à venir. DAML a pour but de favoriser le développement du Web sémantique. Tout comme OIL ou RDF Schema, il s'agit d'un langage de description d'ontologies. En ce sens, on peut définir des classes et des propriétés, et les mettre en relation.

DAML+OIL cherche à combiner toutes les caractéristiques de DAML, d'OIL, et de RDF Schema. Construit sur RDF, il permet de modéliser les aspects suivants :

- Définition de classes de propriétés.
- Définition de classes de ressources.
- Relations logiques entre classes (disjonction, union, équivalence, etc.).
- Relations d'héritage entre classes.
- Restriction de propriétés (cardinalité, etc.) et typage.
- Prise en charge des collections (listes).
- Instanciation de classes de propriétés et de ressources.

```
<daml:Class rdf:ID="Homme">
  <rdfs:subClassOf> rdf:resource="#Humain"
</daml:Class>

<daml:Class rdf:ID="Femme">
  <rdfs:subClassOf rdf:resource="#Humain">
    <daml:disjointWith rdf:resource="#Homme">
  </daml:Class>
```

Exemple 3.3. La classe Humain a deux sous-classes disjointes : Homme et Femme.

```
<daml:Class rdf:ID="Père">
  <rdfs:subClassOf rdf:resource="#Humain">
    <daml:minCardinality="1">
```

¹¹ KQML : <http://www.cs.umbc.edu/kqml/>

¹² DAML : <http://www.daml.org/>

¹³ OIL : <http://www.ontoknowledge.org/oil/>

```

<daml:onProperty rdf:resource="#possèdeEnfants"/>
<daml:toClass rdf:resource="#Homme"/>
</daml:Restriction>
</rdfs:subClassOf>
</daml:Class>

```

Exemple 3.4. La classe des Pères est la classe des Hommes pour lesquels la cardinalité de l'attribut possèdeEnfants vaut au moins 1.

DAML-S : DAML-S¹⁴ signifiant DAML-Services est un langage développé pour la description des capacités et des propriétés des services Web. La recherche, la découverte, l'utilisation et l'interconnexion des services Web sont automatisées grâce à ce langage. DAML-S est construit au-dessus de DAML+OIL ou de ses descendants. DAML-S permet de décrire ce qu'un service Web peut faire, et non pas comment il le fait. DAML-S décrit le sens des données transmises. La méthode de transmission est définie par WSDL et des langages de programmation. DAML-S doit pouvoir assister les services Web dans les tâches suivantes :

- Découverte de services Web, c'est-à-dire déterminer les services qui répondent à un cahier des charges spécifiées.
- Invoquer, exécuter un service Web donné.
- Interopérer entre services Web, ce qui peut impliquer des processus de traduction qui préservent la sémantique.
- Vérifier les propriétés des services.
- Surveiller l'exécution d'une tâche complexe par un service ou un ensemble de services, de façon à détecter et expliquer les défaillances.

DAML-S est une ontologie DAML+OIL, dont la classe parente s'appelle Service. Une instance de cette classe est décrite par deux aspects. Le premier consiste à définir ce que le service fournit aux agents et le deuxième consiste à définir ce qu'il attend des agents. Pour cela, un service présente son ServiceProfile, c'est-à-dire qu'il définit comment il fonctionne et quel est son modèle d'exécution. Le ServiceProfile permet donc aux agents de découvrir et d'identifier un service. Il donne entre autre, le nom du service, son niveau de qualité, le type de service rendu, etc. De plus, le ServiceModel permet aux agents de composer plusieurs services afin de résoudre un problème complexe, ou encore de surveiller le fonctionnement d'un service et d'établir des diagnostics en cas de défaillance. Le modèle d'exécution est décrit à l'aide de la classe ProcessModel qui fournit une ontologie des processus, qui décrit les services dans un modèle de machine à états. Il existe aussi une ontologie des ressources et du temps. Cependant, les descriptions en termes de ServiceProfile et ServiceModel sont situées à un niveau abstrait,

¹⁴ DAML-S : <http://www.daml.org/services/>

indépendant des protocoles de communication concrets qui permettent en pratique d'accéder aux services. C'est pourquoi, le ServiceGrounding est adjoint à cette description. Le grounding fait la liaison entre le modèle et le profil et les descriptions de protocoles en langage WSDL. Par conséquent, DAML-S et WSDL sont deux langages complémentaires pour la description de Services Web. DAML-S en décrit la sémantique, et WSDL les protocoles de communication concrets et effectifs.

OWL : OWL¹⁵ signifie Ontology Web Language et a été défini par le W3C. Comme les langages précédemment évoqués, OWL est construit au-dessus de RDF. Un document OWL est donc composé de triplets RDF, qui peut être écrit dans la syntaxe RDF de son choix. Au sein d'un document, les triplets RDF non définis dans la spécification OWL sont ignorés. Tout comme DAML+OIL, OWL permet de faire des raisonnements, des inférences sur l'ontologie.

Il existe trois sortes d'OWL :

- OWL Lite est une version d'OWL aux fonctionnalités réduites, mais qui restent quand même suffisantes pour bien des usages, comme la constitution de taxonomies ou de thesaurus.
- OWL DL correspond exactement aux logiques de description. Ce langage est expressif, et les procédures d'inférences sont complètes, et effectuables en un temps fini.
- OWL Full donne à l'utilisateur une expressivité maximale, mais on n'a aucune garantie quant à la complétude et à la terminaison des procédures d'inférence.

Une ontologie OWL se compose des éléments suivants :

- En-têtes optionnels : commentaire, version, importation d'ontologie.
- Éléments de classes.
- Éléments de propriétés.
- Instances.

OWL sépare en deux, d'une part le domaine des types de données (types de données de schémas XML), et d'autre part les objets membres de classes OWL ou RDF. Les classes sont soit anonymes, soit désignées par leur URI. Il existe deux classes particulières, owl:Thing et owl:Nothing. Tous les objets définis sont des instances d'owl:Thing. Au contraire, la classe owl:Nothing n'a aucune instance. Pour définir une classe, on peut utiliser des opérateurs de relations et de subsomptions entre classes (subClassOf, disjointWith, disjointUnionOf, sameClassAs, equivalentTo), combiner des attributs grâce à des opérateurs booléens (owl:intersectionOf, owl:unionOf, owl:complementOf), procéder à l'énumération exhaustive de

ses instances et définir la classe par restriction sur une propriété d'une classe préexistante (owl:allValuesFrom, owl:hasValue, owl:someValuesFrom, owl:cardinality, owl:maxCardinality, owl:minCardinality)

La définition des propriétés est faite en tant que sous-classes de ObjectProperty ou DatatypeProperty grâce à l'attribut rdfs:subPropertyOf, par restriction du domaine d'application (rdfs:domain, c.-à-d. l'ensemble des classes à qui cette propriété peut s'appliquer) ou du domaine de variation (rdfs:range, c.-à-d. l'ensemble des valeurs que peut prendre la propriété), par des relations entre propriétés (owl:samePropertyAs, owl:equivalentTo, owl:inverseOf), en tant que sous-classe d'une des sous-classes particulières prédéfinies (owl:TransitiveProperty, owl:SymmetricProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty). Ainsi, il est facile de doter une propriété de caractéristiques bien particulières.

```

<owl:ObjectProperty rdf:ID="situéDans">
  <rdf:type rdf:resource="owl:TransitiveProperty" />
  <rdfs:domain rdf:resource="owl:Thing" />
  <rdfs:range rdf:resource="#Lieu" />
</owl:ObjectProperty>

<Region rdf:ID="Fenêtre01">
  <locatedIn rdf:resource="#Ouverture" />
</Region>

<Region rdf:ID="Ouverture01">
  <locatedIn rdf:resource="#Mur" />
</Region>
```

Exemple 3.5. On indique tout d'abord que la propriété situéDans est transitive. Après les deux assertions suivantes (« Fenêtre01 est situé dans une ouverture » et « l'Ouverture01 est située dans un Mur »), on peut conclure sur « la Fenêtre01 est située dans un mur ».

2.4 Intégration de données basée sur une ontologie

Dans les sections précédentes, nous avons défini le concept d'ontologie. Dans cette section, nous allons présenter les travaux qui concernent l'intégration de données basée sur une ontologie. Nous montrerons que l'utilisation d'une ontologie nécessite une étape de mise en correspondance des éléments du système et de son « correspondant ontologique ». Lorsque cette correspondance est faite, la représentation dans l'ontologie des éléments du système est considérée comme un schéma de méta-données. Le rôle d'un schéma de méta-données est double [AMA03]. D'une part, il représente une ontologie des connaissances partagées sur un domaine, d'autre part il joue le rôle d'un schéma de bases de données qui est utilisé pour la formulation de requêtes structurées sur des méta-données ou pour constituer des vues. Ce principe est appliqué pour l'intégration de données à l'aide d'une ontologie de domaine pour

¹⁵ OWL, <http://www.w3.org/TR/owl-ref/>

fournir des structures d'intégration et des processus de requêtage sur ces structures. Nous allons maintenant présenter les travaux qui proposent l'intégration de données basée sur une ontologie.

- Isabel F. Cruz et coll. [CRU04] ont pour objectif l'interopération de deux documents XML à un niveau sémantique à l'aide d'une ontologie, tout en conservant la structure des fichiers initiaux. Dans cette approche GaV¹⁶, une ontologie définie à l'aide du langage RDF est générée pour chaque document XML. Une ontologie globale est le résultat de la fusion de chaque ontologie. L'ontologie globale unifie les requêtes d'accès et les connexions sémantiques entre les bases de données individuelles. Deux types de requêtes sont prises en compte. Celles qui sont posées sur l'ontologie globale et celles qui sont posées sur les documents XML indépendamment l'un de l'autre. Dans cette approche, une requête posée à l'ontologie globale peut être traduite en sous requêtes pour chaque source de données. Inversement, une requête posée à une source de données peut être traduite pour l'ontologie globale. Ce système d'intégration de données utilise donc un algorithme de traduction de requête bidirectionnel. Cette architecture de médiation basée sur une ontologie globale étend RDFS en définissant un méta-schéma additionnel. Ce méta-schéma encode la structure du schéma XML en schéma RDF. L'algorithme traduit les requêtes XQuery en RDQL et inversement.
- M. Klein dans [KLE02] interprète les documents XML à l'aide d'un schéma XML. Les schémas XML qui décrivent la structure des documents XML, ne décrivent pas la signification des balises. Les schémas définissent une hiérarchie de types d'élément. Le modèle de données RDF est utilisé pour donner une interprétation non ambiguë des données. L'objectif est d'utiliser une ontologie de domaine pour associer une signification à un document XML ambiguë. Pour cela, un algorithme détermine comment l'ontologie spécifie l'interprétation des balises XML et des données. L'ontologie détermine quel concept est pertinent, comment ils sont reliés et quelles propriétés les concepts doivent avoir. Le principe de la procédure est d'utiliser l'ontologie pour spécifier quelles balises ont un intérêt et de définir quels rôles elles ont. Cela implique que toutes les structures syntaxiques ne sont pas prises en compte dans le modèle de données RDF. Les relations entre deux éléments dans un document XML ne sont pas interprétées comme un nom de propriété. Au lieu de cela, l'ontologie détermine si la balise doit être interprétée comme un nom de classe ou un nom de propriété. Cette procédure a l'avantage de ne pas altérer les documents XML.

- L'idée de V. S. Lakshmanan et coll. est que toutes les fois qu'une source de données veut se joindre à un effort d'interopérabilité, il doit faire correspondre ces données à une ontologie standard qui existe déjà. Dans [LAK03] est définie une infrastructure basée sur des déclarations de sémantique locale pour permettre l'interopérabilité entre différentes sources de données. Ces déclarations sont des règles de correspondances qui font le lien entre les éléments des sources et un vocabulaire commun. Une ontologie standard définit la liste des noms de prédicat, ainsi que les usages. Elle définit les arguments des prédicats, c'est-à-dire le type et le rôle de chaque argument. Elle doit fournir des informations sur les identifiants. Pour fournir un maximum de flexibilité, les prédicats doivent être aussi simples que possible. Chaque prédicat définit une relation entre un ensemble de concepts et un ensemble de prédicats. Une fois que l'ensemble des prédicats est défini pour la source, la correspondance entre les données XML doit être établie. Le langage utilisé pour définir les correspondances est XPath. Ce modèle simple de prédicats simplifie significativement la tâche de formulation des requêtes. La fédération est définie par l'ensemble des sources locales, où le contenu est composé de chaque source qui a été déclarée par un ensemble de prédicats et de correspondances.

Nous voyons dans ces différents travaux qu'il faut définir des règles de correspondance entre les sources d'informations et le niveau ontologique. Le principe consiste à étiqueter les éléments des sources, et ainsi fournir une sémantique à l'élément par rapport à une définition consensuelle du sens. Cette phase est forcément nécessaire, car cette information n'a pas été ajoutée au document lors de sa création. Le schéma XML ne définit que la structure des documents XML associées. Toutefois, celui-ci peut servir à définir l'ontologie en extrayant un ensemble d'éléments et de propriétés dont le sens sera défini pour un usage plus global.

3. Conclusion

Dans les sections précédentes, nous avons montré que l'intégration de données pose de nombreux problèmes d'hétérogénéité syntaxique, structurelle et sémantique. Nous avons vu que pour répondre à ces problèmes, la mise en place d'une architecture orientée services, la définition de langages dérivés du XML pour décrire un domaine particulier, la définition d'une ontologie décrivant les connaissances de ce domaine et la mise en correspondance des éléments sources avec le niveau ontologique étaient nécessaires.

¹⁶ Global-As-View

Le premier objectif de notre travail est de fournir un système permettant de définir des ontologies de domaine dynamiquement pour l'intégration sémantique de schémas XML hétérogènes et par extension, l'intégration de documents XML hétérogènes. Ce travail sera présenté dans le chapitre 4 de ce mémoire.

Le second objectif de notre travail est de fournir des processus pour la gestion des données. Nous avons vu que la problématique de la réalité virtuelle et de la modélisation 3D est de fournir des mécanismes pour réduire la taille des informations pour améliorer les performances. Le débit des réseaux et la performance des machines limitent la taille des données à transférer. Pour répondre à cette problématique, la sémantique des informations va permettre de construire des vues basées sur la sémantique des données et de limiter le volume de données. Par exemple, un élément comme un mur peut être vu comme un élément graphique ou comme une propriété thermique. L'information qui sera utilisée par l'intervenant dépendra de son contexte d'activité. Les vues seront personnalisables en fonction du contexte de l'intervenant à l'aide des informations contenues dans les ontologies de domaine générée. Par exemple, seules les informations thermiques sur les murs sont nécessaires, pour réaliser un calcul d'échanges thermiques entre les pièces d'un bâtiment, alors, seul le contexte thermique sera transmis (les éléments de type mur et leurs propriétés thermiques seront extraits). Ce travail sera présenté dans le chapitre 5 de ce mémoire.

L'objectif de cette thèse est de fournir une solution générique d'intégration de données au travers de scènes 3D. Les scènes 3D constituent les vues utilisateurs sur les données des différentes sources hétérogènes. Ces scènes 3D seront générées dynamiquement en fonction d'un modèle utilisateur défini par les droits et le contexte d'utilisation des données hétérogènes attribués à chaque utilisateur.

Pour atteindre cet objectif, nous avons développé la méthode ACTIVe3D qui permet de définir dynamiquement une ontologie de domaine pour la mise en correspondance des concepts communs de différents schémas XML. A l'aide de cette ontologie, les différentes sources de données pourront être intégrées et manipulées de manière transparente par l'utilisateur final. Une architecture de Web Services sera développée pour fournir un ensemble de mécanismes de manipulation de ces données hétérogènes. Toutes les données hétérogènes seront ainsi liées et toutes les modifications d'une donnée seront dynamiquement répercutées sur toutes les autres données hétérogènes sémantiquement liées. Ainsi, la scène 3D sera dynamiquement modifiée si les éléments sémantiques liés à cette scène sont modifiés et réciproquement. L'implémentation de cette architecture sera présentée dans le chapitre 6 de ce mémoire.

Chapitre 4

« Si j'avais le pouvoir, je commencerais par redonner leur sens aux mots. »
(Confucius : philosophe chinois)

Une Ontologie pour l'Intégration de Données XML

Sommaire

1. Formalisation des données XML.....	65
1.1. Les grammaires formelles et les documents XML.....	65
1.2. Les grammaires formelles et les grammaires XML	66
1.3. Notion de Facteur et de marque schématique	70
1.4. Conclusion.....	71
2. Définition sémantique des marques schématiques	71
2.1. Concepts et Relations	72
2.2. Extensibilité et Contexte	73
2.3. Conclusion.....	74
3. Formalisation schématique et sémantique des schémas XML	74
3.1. Formalisation schématique et règles d'intégration.....	75
3.2. Ontologie générique	78
4. Exemple	84
4.1. Intégration de schémas	84
4.2. Intégration de données	95
5. Conclusion.....	102

Dans les sections précédentes, nous avons montré que l'intégration de données pose de nombreux problèmes d'hétérogénéité syntaxique, structurelle et sémantique. Nous avons vu que pour répondre à ces problèmes, la mise en place d'une architecture orientée services, la définition de langages dérivés du XML pour décrire un domaine particulier, la définition d'une ontologie décrivant les connaissances de ce domaine et la mise en correspondance des éléments sources avec le niveau ontologique étaient nécessaires. Nous avons montré aussi, que les problèmes d'hétérogénéité syntaxiques et structurelles étaient résolus grâce au langage XML. Par contre, la problématique de l'hétérogénéité sémantique persiste. Pour résoudre ce problème, nous proposons dans ce chapitre une solution qui met en relation différents niveaux d'abstraction sémantique et schématique. Cette solution est articulée en deux étapes. La première étape concerne la formalisation sémantique des règles d'écriture d'une grammaire XML en général.

Cette formalisation nous permettra de définir les composants d'une ontologie générique. La seconde étape concerne la définition des mécanismes d'ontologisation de la sémantique des éléments d'une grammaire XML spécifique pour obtenir une ontologie de domaine. Les concepts et les relations de l'ontologie de domaine sont alors définis à partir des éléments des schémas XML.

Dans ce chapitre, nous verrons que ces mécanismes permettent d'identifier certains concepts et relations communs à plusieurs schémas XML. Par conséquent, les ontologies permettront de mettre en correspondance les concepts et relations en fusionnant les attributs des deux éléments communs (c.f. Figure 4.1). L'ontologie de domaine sera alors étendue et modifiée pour représenter la sémantique de plusieurs schémas XML relatif à un domaine particulier. Nous verrons que cette intégration peut être étendue à l'intégration de documents XML.

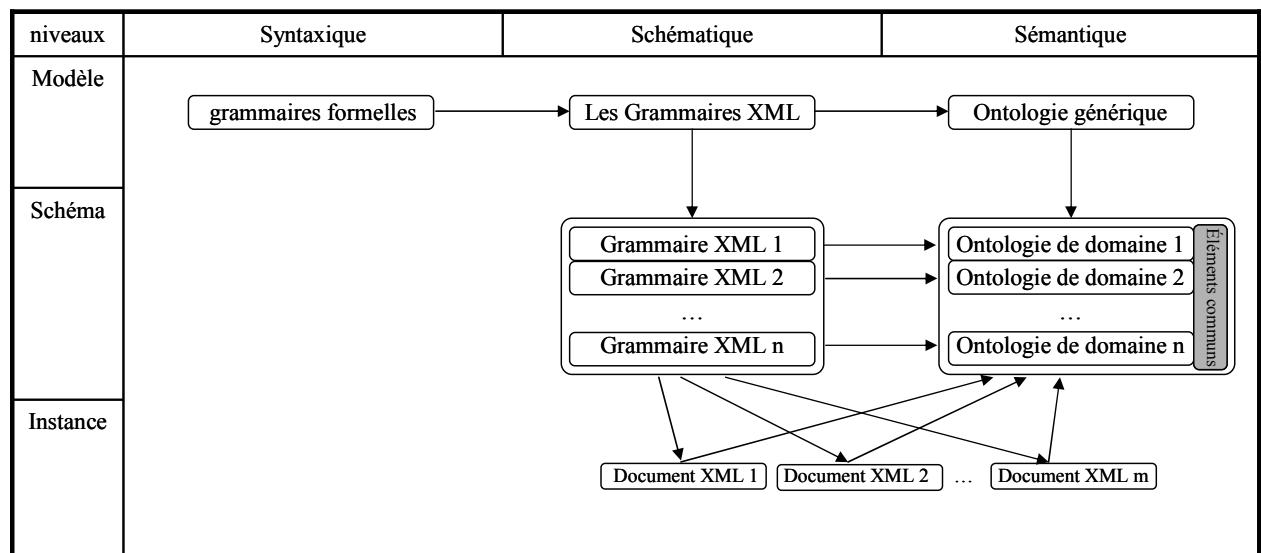


Figure 4.1. Éléments communs aux ontologies de domaine

Ce chapitre est composé de quatre sections. La première section présente notre formalisation des grammaires XML. Dans cette partie, nous définirons la notion de facteur qui sera utilisée pour construire des marques schématiques sur les éléments XML. La marque schématique est utilisée comme lien entre la structure d'un document XML et sa sémantique implicite. La deuxième section présente les composants d'une ontologie, puis explique comment cette ontologie permet d'intégrer les schémas XML et les documents XML. La troisième section présente l'ontologie générique utilisée à la fois pour définir les ontologies de domaine et pour l'intégration de données. La dernière section présente un exemple complet d'intégration de plusieurs schémas XML et de documents XML à l'aide des marques schématiques, des ontologies de domaine et de l'ontologie générique.

1. Formalisation des données XML

Pour spécifier la sémantique des éléments d'un schéma XML, il faut avant tout les identifier et les marquer à l'aide de marques schématiques. Ces marques seront utilisées pour établir des liens entre la structure du document XML et sa définition sémantique. Dans cette section, nous présenterons tout d'abord la formalisation des documents XML à l'aide des langages formels. Ensuite, nous présenterons une formalisation des grammaires XML toujours à l'aide des langages formels. Nous verrons que les grammaires XML génèrent des langages de Dyck. D'après certaines propriétés des langages de Dyck, nous définirons les notions de facteur et de marque schématique qui sont les notions de base du système d'intégration.

1.1. Les grammaires formelles et les documents XML

Un document XML est composé de texte et de balises ouvrantes associées à des balises fermantes. Certaines de ces balises sont à la fois ouvrantes et fermantes, ce sont des balises vides qui correspondent aux feuilles des arbres. Une des propriétés des fichiers XML canoniques est de ne posséder que des balises ouvrantes ou fermantes. Les balises vides peuvent être modifiées en balises ouvrantes et fermantes sans poser de problèmes à l'analyseur syntaxique de fichier XML. Par conséquent, tout fichier XML possédant des balises vides possède un équivalent sans balise vide. Cette propriété est syntaxique, car elle n'apparaît pas dans les règles de grammaire formalisant la structure du document. À partir de ces informations, un certain nombre de définitions sont exprimées.

Définition 1 : Un *alphabet* est un ensemble fini de symboles, noté Σ . Ses éléments sont appelés des *lettres*. On notera la plupart du temps dans la suite: $\Sigma = \{a, b, \dots\}$. La *taille* $|\Sigma|$ d'un alphabet Σ est le nombre de ses éléments.

Définition 2 : Un *mot* ou une *phrase* sur Σ est une suite de lettres de cet alphabet. Le mot « mur » est une suite de lettres de l'alphabet $\Sigma = \{a, b, \dots, z\}$. On définit par convention le *mot vide* comme le mot de longueur nulle. On le note : ε . L'ensemble de tous les mots que l'on peut écrire sur l'alphabet Σ se note : Σ^* . On note : $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Définition 3 : Un *langage formel* L est un ensemble de mots sur Σ^* . Un *langage* L sur un alphabet Σ est dit régulier¹ s'il est engendré sur l'alphabet Σ et s'il est décrit par une expression régulière. C'est-à-dire que l'ensemble des langages réguliers sur Σ est exactement l'ensemble des langages reconnaissables par automate d'états finis sur Σ . En d'autres termes, à tout automate

¹ Un langage régulier est langage de type 3 dans la hiérarchie de Chomsky

d'états finis on peut associer une expression régulière qui définit un langage égal à celui reconnu par l'automate, et réciproquement. Notons qu'il n'y a pas bijection car plusieurs automates différents peuvent reconnaître un même langage, et de même il existe plusieurs expressions régulières pour le définir.

Nous venons de voir qu'un langage L est formé de mots générés à partir d'un alphabet. Si dans un document XML, nous nous limitons à la structure arborescente sans prendre en compte les valeurs des attributs des balises, alors l'ensemble des documents XML qu'il est possible de générer à partir d'un schéma XML forme un langage. De plus, l'ensemble des balises des documents XML définies par le schéma XML forme une partie de l'alphabet du langage. Il forme seulement une partie, car l'alphabet peut contenir des lettres non utilisées par le langage.

Définition 4 : Une grammaire formelle peut être définie comme une entité mathématique à laquelle est associé un processus algorithmique permettant d'engendrer un langage. On le définit alors comme un quadruplet $G = \langle N, T, P, S \rangle$ dans lequel :

- T noté aussi Σ est l'alphabet terminal de G .
- N l'alphabet non terminal de G .
- $V = N \cup T$ est un alphabet composant l'ensemble des symboles de G .
- P est un ensemble de règles de production ou expression régulière.
- $S \in N$ est l'axiome ou symbole de départ de G .

$N = \{S, A\}$ $T = \{a, b\}$ $P = \{(S \rightarrow AA), (S \rightarrow \epsilon), (A \rightarrow aa), (A \rightarrow bb)\}$	s'écrit aussi : $S \rightarrow AA \mid \epsilon$ $A \rightarrow aa \mid bb$
--	--

Exemple 4.1. La grammaire $G_1 = \langle N, \Sigma, P, S \rangle$

L'exemple 4.1 montre une grammaire dont les mots engendrés sont « aaaa » « aabb » « bbaa » « bbbb » et « ». Le langage est composé alors de quatre mots et du mot vide.

1.2. Les grammaires formelles et les grammaires XML

Nous avons vu dans la section précédente la notion de langage régulier et de grammaire formelle, à présent nous allons faire le parallèle entre les grammaires formelles et les grammaires XML (c.f Figure 4.2). Ces grammaires ont la particularité d'avoir un vocabulaire terminal composé de balises ouvrantes et de balises fermantes.

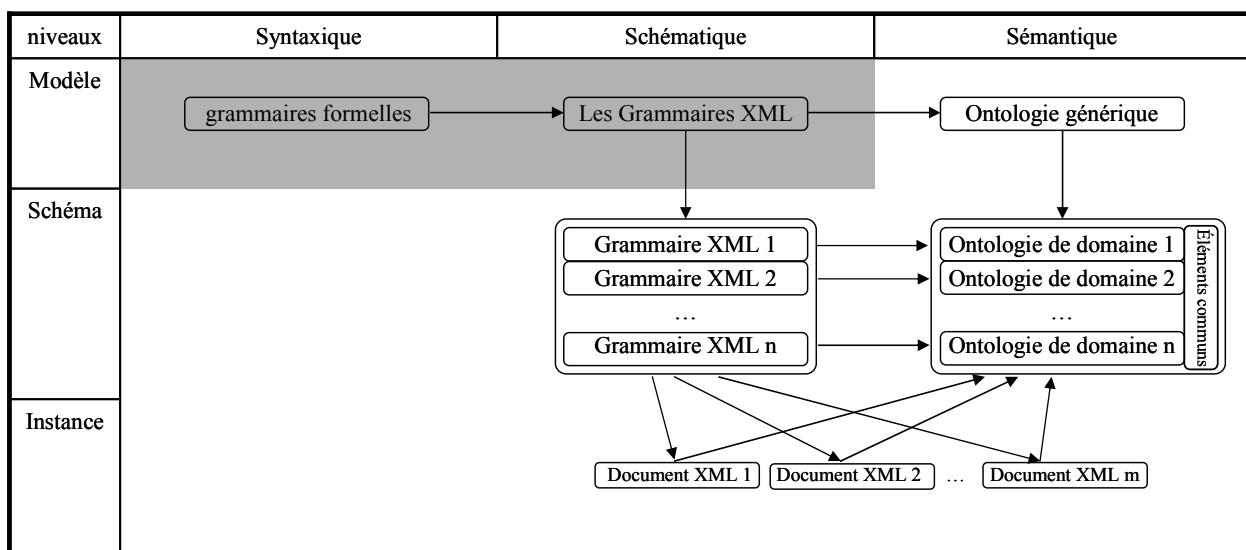


Figure 4.2. Parallèle entre les grammaires formelles et les grammaires XML

Définition 1 : Soit A un ensemble de balises ouvrantes et \bar{A} un ensemble de balises fermantes correspondantes, un document XML est un mot composé par l'alphabet $T = A \cup \bar{A}$. Pour l'instant, nous ne nous intéressons qu'à la structure syntaxique. Un document XML x est bien formé si une seule balise est une racine et si les balises sont correctement imbriquées.

Définition 2 : Un document x est bien formé si x est généré par des règles de production d'un langage de Dyck [LIE98] sur $T = A \cup \bar{A}$. Un langage de Dyck est un langage engendré par une grammaire hors contexte de la forme suivante où $a_n \in A$ et son équivalent $b_n \in \bar{A}$:

$$S \rightarrow SS \mid \epsilon \mid a_1 S b_1 \mid a_2 S b_2 \mid \dots \mid a_n S b_n \text{ avec } n \geq 1$$

Cette définition correspond à la terminologie et à la notation utilisée dans la communauté XML. En effet, un langage est un ensemble de documents XML qu'il est possible de générer à partir d'une grammaire. Ces grammaires de langage XML sont appelées « définition de type de document » (DTD). L'axiome des grammaires est qualifié de DOCTYPE, et l'ensemble des règles de productions est associé à une balise ELEMENT (cf. Exemple 4.2). La syntaxe des règles de production est construite par la correspondance terme à terme entre une paire de balises et un vocabulaire non terminal de la grammaire. Une balise ELEMENT est composée d'un type et d'un modèle. Le type est simplement le nom de la balise et le modèle est l'expression régulière pour l'ensemble qui est à droite de la règle de production de la balise.

$$P = \{(S \rightarrow a(S \mid T)(S \mid T)\bar{a}), (T \rightarrow b), (T \rightarrow Tb)\}$$

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT a ((a | b), (a | b))>
<!ELEMENT b (b*)>
```

Exemple 4.2. Correspondance entre une grammaire et une DTD

L'ensemble des règles de production correspondant à la grammaire d'un langage peut être représenté à l'aide d'un schéma XML pouvant être traduit à partir d'une DTD (cf. Exemple 4.3). À la différence des DTD, qui ne définissent que les relations entre les différents composants d'un document, les schémas définissent les types des données.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ROOT (A | (B, C))>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xss:element name="A"/>
  <xss:element name="B"/>
  <xss:element name="C"/>
  <xss:element name="ROOT">
    <xss:complexType>
      <xss:choice>
        <xss:element ref="A"/>
        <xss:sequence>
          <xss:element ref="B"/>
          <xss:element ref="C"/>
        </xss:sequence>
      </xss:choice>
    </xss:complexType>
  </xss:element>
</xss:schema>
```

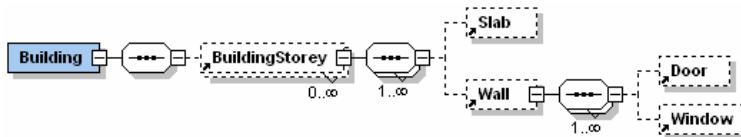
Exemple 4.3. Correspondance entre DTD et Schéma W3C

Un document XML est valide s'il suit la grammaire définie par une DTD ou un schéma XML. Cette grammaire est nécessaire si l'on désire effectuer une validation syntaxique du document. L'exemple 4.4 montre un schéma XML et l'exemple 4.5 montre un document valide correspondant à ce schéma. Pour simplifier la lecture, l'exemple 4.4 est aussi donné graphiquement.

```
<?xml version="1.0" encoding="UTF-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="qualified">
  <xss:element name="Building">
    <xss:complexType>
      <xss:sequence>
        <xss:element ref="BuildingStorey" minOccurs="0" maxOccurs="unbounded"/>
      </xss:sequence>
      <xss:attribute name="Label" type="xs:string" use="optional"/>
    </xss:complexType>
  </xss:element>
  <xss:element name="BuildingStorey">
    <xss:complexType>
      <xss:sequence maxOccurs="unbounded">
        <xss:element ref="Slab" minOccurs="0"/>
        <xss:element ref="Wall" minOccurs="0"/>
      </xss:sequence>
      <xss:attribute name="Label" type="xs:string" use="optional"/>
    </xss:complexType>
  </xss:element>
  <xss:element name="Slab">
    <xss:complexType>
      <xss:attribute name="Label" type="xs:string" use="optional"/>
    </xss:complexType>
  </xss:element>
  <xss:element name="Wall">
    <xss:complexType>
      <xss:sequence maxOccurs="unbounded">
        <xss:element ref="Door" minOccurs="0"/>
        <xss:element ref="Window" minOccurs="0"/>
      </xss:sequence>
      <xss:attribute name="Label" type="xs:string" use="optional"/>
    </xss:complexType>
  </xss:element>
</xss:schema>
```

```

</xs:complexType>
</xs:element>
<xs:element name="Door">
    <xs:complexType>
        <xs:attribute name="Label" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Window">
    <xs:complexType>
        <xs:attribute name="Label" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:element>
</xs:schema>
    
```



Exemple 4.4. Schéma XML Bâtiment

```

<?xml version="1.0" encoding="UTF-8"?>
<Building
    xsi:noNamespaceSchemaLocation="C:\Documents\batiment.xsd" Label="Ecole">
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <BuildingStorey Label="Etage 1">
        <Slab Label="Dalle Plafond"/>
        <Slab Label="Dalle Sol"/>
        <Wall Label="Mur 1">
            <Window Label="Fenêtre 1"/>
            <Window Label="Fenêtre 2"/>
            <Window Label="Fenêtre 3"/>
            <Door Label="Porte 1"/>
        </Wall>
        <Wall Label="Mur 2">
            <Window Label="Fenêtre 4"/>
        </Wall>
        <Wall Label="Mur 3">
            <Window Label="Fenêtre 5"/>
        </Wall>
    </BuildingStorey>
    <BuildingStorey Label="RDC">
        <Slab Label="Dalle Sol"/>
        <Wall Label="Mur 4">
            <Window Label="Fenêtre 6"/>
            <Window Label="Fenêtre 7"/>
            <Door Label="Porte 2"/>
            <Door Label="Porte 3"/>
        </Wall>
        <Wall Label="Mur 5">
            <Window Label="Fenêtre 8"/>
            <Window Label="Fenêtre 9"/>
        </Wall>
    </BuildingStorey>
</Building>
    
```

Exemple 4.5. Document XML respectant le schéma XML Bâtiment

Dans cette section nous avons vu des rappels sur les langages formels en faisant le parallèle entre une grammaire formelle et un schéma XML. Nous savons qu'un schéma XML est une grammaire formelle d'un langage de Dyck et qu'elle possède par conséquent les propriétés des grammaires de Dyck. Dans la section suivante, nous allons voir les propriétés des grammaires de Dyck et définir une nouvelle notion. Cette notion est la marque schématique.

1.3. Notion de Facteur et de marque schématique

Les propriétés des langages de Dyck ont fait l'objet d'études menées par J. Berstel [BER00]. En faisant le parallèle entre les grammaires XML et les langages de Dyck, J. Berstel définit la notion de facteur. D'après le lemme 3.3 de J. Berstel si G est une grammaire XML sur $T = A \cup \bar{A}$ générant un langage L , avec comme vocabulaire non terminal X_a et $a \in A$, alors pour chaque $a \in A$, le langage généré par X_a est un ensemble de facteurs de mots dans L qui sont des langages de Dyck démarrant par la lettre a : $L_G(X_a) = F_a(L)$

Soit le langage $L = \left\{ ca(b\bar{b})^{n_1} \bar{a}a(b\bar{b})^{n_2} \bar{a} \dots a(b\bar{b})^{n_k} \bar{a}c \mid n_{1,2,\dots,k} > 0 \right\}$

Alors $F_c(L) = L$, $F_b(L) = \{ b\bar{b} \}^*$, $F_a(L) = \{ a(b\bar{b})^* \bar{a} \}$

Exemple 4.6. Facteurs du langage L

Ceci signifie qu'un langage est factorisable en sous langage et un facteur d'un langage de Dyck est un langage d'un langage de Dyck. Par conséquent, un sous arbre d'un document XML peut être généré à partir d'un facteur du langage de Dyck auquel appartient le document XML.

D'après le corollaire 3.4 de J. Berstel, pour une grammaire XML, il existe un seul facteur $F_a(L) = L$. Sachant que a est la balise racine du document XML. Ceci signifie qu'il n'existe qu'une seule balise père de toutes les autres, soit une seule balise racine. Par conséquent, il n'existe qu'un seul facteur $F_a(L) = L$ où a est la racine.

Toujours d'après le corollaire 3.4 : Soit w est un mot du langage L , il existe une unique factorisation $w = au_{a_1}u_{a_2}\dots u_{a_n}\bar{a}$ avec $u_{a_i} \in D_{a_i}$ pour $i \in 1, \dots, n$. D_{a_i} est un langage de Dyck commençant par a_i et $D_{a_i} \subset D_A$. La *trace* d'un mot w est définie par le mot $a_1a_2\dots a_n$. La *surface* de $a \in A$ dans le langage L est l'ensemble $S_a(L)$ de toutes les traces des mots du facteur $F_a(L)$. La notion de surface est utilisée par Berstel pour démontrer la proposition suivante :

Proposition 1 : Pour chaque langage XML L , il existe une seule grammaire XML réduite générant L .

Ce qui signifie que tous les langages de la grammaire XML sont générés par la même grammaire XML réduite. Ceci indépendamment de la valeur des attributs des balises dans les documents et

en s'attachant seulement à la structure syntaxique des fichiers XML. Cette proposition implique que si un facteur a été défini sur le langage XML, alors ce facteur correspond à une règle de production de la grammaire XML réduite générant ce langage. Cette proposition permet d'introduire la notion de marque schématique. Une grammaire réduite ne possède pas de variable inutile. Sachant que dans une grammaire XML décrivant des informations, toutes les variables ont leurs places.

Définition 3 : Une marque schématique est une marque sur un schéma XML pour identifier une règle de production.

Cette marque schématique permet de faire correspondre un élément d'un schéma XML à un facteur sur le langage XML (ensemble des documents XML engendré par la grammaire). Les marques schématiques seront utilisées pour lier les composants de l'ontologie avec les éléments du schéma XML. Au niveau de l'ontologie, on parlera alors de la sémantique de la marque schématique qui permet d'identifier les éléments qui porte une sémantique implicite et non définie de manière formelle.

1.4. Conclusion

Nous avons vu dans cette section que le langage d'une grammaire XML est un langage de Dyck. Grâce à cette propriété, nous avons introduit de nouvelles notions qui sont les facteurs et les marques schématiques. La notion de marque schématique permet de faire le lien entre un élément portant une sémantique et un élément de document XML. Ceci signifie qu'une marque schématique correspond à un sous arbre de document XML, puisqu'une marque correspond à un facteur qui correspond à un langage de Dyck engendrant les sous arbres XML. Dans la suite, nous montrerons que plusieurs marques schématiques de différents schémas XML peuvent avoir la même sémantique définie par une ontologie. Dans ce cas, les propriétés des marques schématiques sont intégrées au sein du même concept défini dans l'ontologie. Nous parlerons alors d'intégration sémantique des schémas XML. Nous allons voir dans la section suivante l'approche sémantique des données XML pour spécifier la sémantique des marques schématiques.

2. Définition sémantique des marques schématiques

Nous avons vu dans la section précédente comment distinguer deux éléments d'un schéma XML grâce aux marques schématiques. À présent nous devons définir la sémantique des marques schématiques à l'aide d'une ontologie. Nous pourrons ainsi manipuler les éléments définis dans

les grammaires par leur sémantique. La section suivante présente les principales notions constituant une ontologie : les concepts et les relations. L'objectif de cette section est de familiariser le lecteur à la terminologie utilisée dans la suite de ce document. La dernière section, les avantages de l'approche ontologique seront discutés.

2.1. Concepts et Relations

Un concept est caractérisé par trois composantes qui sont un ou plusieurs termes, une notion et un ensemble d'objets [USC95]. La notion, également appelée intension du concept, contient la sémantique du concept. L'intension du concept correspond aux propriétés, aux attributs, aux règles et contraintes d'une classe d'objets. Par exemple, la largeur et la hauteur décrivent l'intension du concept « mur ». Analogiquement, les attributs d'une marque schématique caractérisent l'intension du concept définissant l'élément. De plus, l'intension d'une marque schématique intégrant plusieurs schémas est composée de tous les attributs des éléments dans les différents schémas XML. Ainsi, le concept modélise la sémantique des éléments dans les différents schémas XML.

Un concept peut être considéré comme une classe dont les objets instanciés forment l'extension du concept. Analogiquement, l'extension d'un concept correspondant à une marque schématique est un ensemble de sous arbres de documents XML validés par la grammaire. Si le concept intègre plusieurs marques schématiques, alors l'extension est composée des différents sous arbres provenant de documents XML produits par les différents schémas XML. De plus, si deux sous arbres XML provenant de deux documents XML de schémas XML différents correspondent à la même instance de concept alors ces sous arbres sont intégrés dans cette même instance. Nous notons ici deux niveaux d'intégration. Le niveau d'intégration des schémas où les attributs sont regroupés dans l'intension d'un même concept et le niveau d'intégration de données où les valeurs des attributs sont regroupées dans la même instance de concept.

Une relation comme un concept possède un ou plusieurs termes, une notion et un ensemble d'objets. Les relations sont aussi organisées de manière hiérarchisée à l'aide de propriétés de subsomption. Les propriétés intrinsèques à une relation sont les propriétés algébriques (symétrie, réflexivité, transitivité) et la cardinalité définissant le nombre de relations possibles entre les instances. De plus, la notion d'intension et d'extension définie pour les concepts est aussi définie pour les relations. Ainsi, les relations peuvent être marquées dans une grammaire XML. Néanmoins, un document XML définit deux types de relation. La première est définie par une balise, et la deuxième est définie par une relation directe entre une balise père et une balise fils. Dans ce cas, cette relation est une relation implicite que le système définira comme une relation, mais sans définition XML.

La structure cognitive devant stocker les informations doit non seulement permettre l'intégration des schémas, mais également l'intégration des données. Pour cela, la structure doit pouvoir stocker les nouveaux concepts réalisés à l'aide des marques schématiques, mais également les instances de concept contenant les valeurs des attributs. La notion d'intension et d'extension permet donc de définir la structure sémantique des données XML. La conceptualisation qui est réalisée pour décrire les informations sémantiques est d'un niveau d'abstraction supérieur. Par conséquent, l'ontologie qui en dérive est de type générique, car elle permet de définir de nouveaux concepts dynamiquement sans avoir à redéfinir la structure cognitive. Les nouveaux concepts stockés forment de cette manière une ontologie de domaine décrivant la sémantique des schémas XML. Cette approche possède des avantages qui seront discutés dans la section suivante.

2.2. Extensibilité et Contexte

Dans le cadre plus spécifique d'un langage orienté objet, la correspondance entre une intension et une extension est fixe. Si la modélisation de l'intension correspondant à la notion de classe est terminée alors, les objets générés à partir du modèle ont une structure qui ne peut plus évoluer. Il est alors possible de modifier seulement les valeurs des instances. L'approche ontologique rend envisageable l'ajout de nouvelles structures de données aux classes, ainsi que l'ajout de nouvelles données aux objets existants. Ceci est rendu possible par l'ajout de nouvelles intensions aux concepts. Cet apport ne remet pas en cause la structure initiale, mais permet simplement de l'enrichir au court de son cycle d'utilisation. Cette conception évolutive de la structure cognitive vise entre autres l'intégration de descriptions d'objets à l'aide de différents catalogues XML fournis par des services Web. En effet, chacun des services est décrit par une grammaire XML et l'information est transportée par un flux des données XML.

Jusqu'à présent, il n'a été question que de validation syntaxique des données par les grammaires XML. La question de la validation sémantique n'a pas encore été abordée. Dans la littérature, l'importance de certains attributs de l'intension pour la validité d'un concept n'a pas réellement été tranchée. En effet, certaines propriétés sont essentielles à la caractérisation d'un concept et leurs suppressions entraîneraient la suppression du concept. Par contre, l'intégration de plusieurs intensions au cœur du même concept permet d'aborder le concept de manière contextuelle. Si un contexte est caractérisé par le domaine de provenance de l'intension alors, la suppression d'une intension ne remet pas en cause le concept dans les autres contextes. Par conséquent, le système d'informations basé sur une ontologie intégrant les données XML peut faire évoluer sa structure cognitive en ajoutant ou en supprimant des intensions aux concepts. L'approche contextuelle est très avantageuse pour la mise en place des processus applicatifs permettant l'extraction des

données du système d'information [VAN04]. Cette notion de contexte est développée dans le chapitre 5 pour la manipulation de données.

2.3. Conclusion

Nous avons vu jusqu'à présent comment décrire un schéma XML à l'aide de marques schématiques pour définir des concepts et des relations. Nous avons vu deux niveaux d'intégration dont le premier est l'intégration de schémas et le deuxième est l'intégration de données. Pour réaliser le premier niveau d'intégration, nous avons besoin d'une ontologie générique pour définir les notions de concept, de relation et d'attribut. Cette ontologie nous permettra de définir dynamiquement des ontologies de domaine correspondant à l'intégration de schémas XML à l'aide des marques schématiques. La définition de cette ontologie générique fait l'objet de la section suivante.

3. Formalisation schématique et sémantique des schémas XML

Cette section est articulée en trois parties. La première section présente la formalisation des règles de construction des grammaires XML. Cette formalisation nous permettra de dégager un ensemble de termes qui seront utilisés pour construire notre ontologie générique. La seconde partie présente la définition de l'ontologie générique (c.f. Figure 4.3). Cette ontologie générique servira de modèle pour créer des ontologies de domaines. Dans cette partie, nous donnerons la structure formelle de l'ontologie sous la forme de classes. Cette structure formelle permet d'indexer les connaissances implicites contenues dans les schémas sous forme d'éléments. Cette connaissance est alors rendue explicite, car elle lève toute ambiguïté sur l'interprétation des termes d'un document XML généré à partir d'un schéma XML intégré.

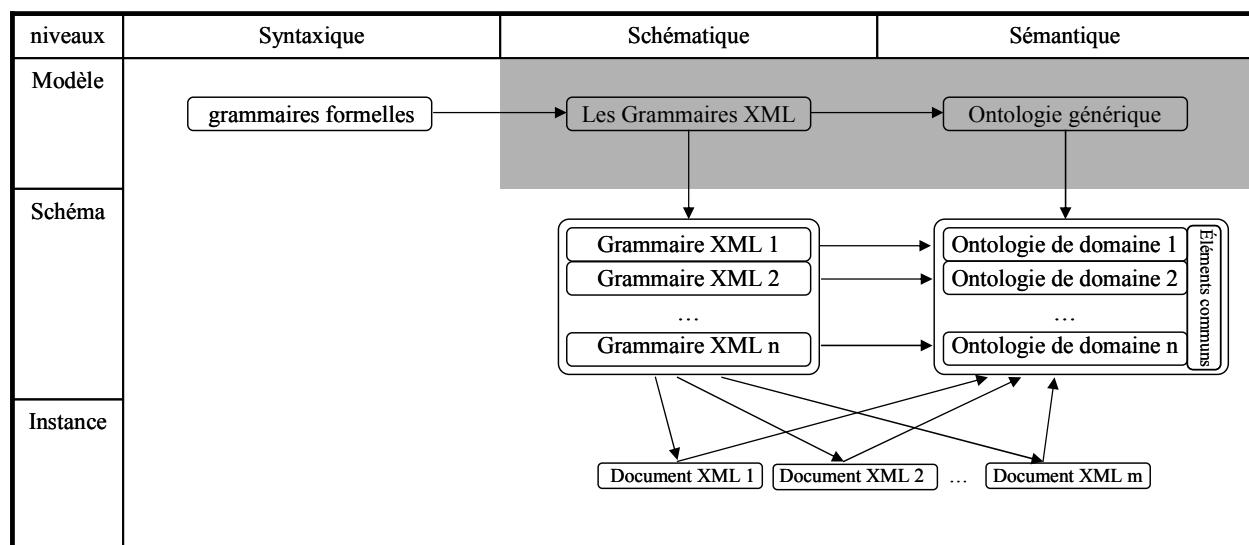


Figure 4.3. Définition de l'ontologie générique

3.1. Formalisation schématique et règles d'intégration

Dans cette section, nous présentons un ensemble de règles et de définitions qui permettront de formaliser la construction des grammaires XML et de définir les concepts, relations et attributs de notre ontologie générique.

Définition 1 : Le Facteur $F_a(L)$ d'un langage L est un *facteur conceptuel* s'il définit un concept. Par exemple, le facteur $F_{Bâtiment}(L)$ est un concept, car il définit le concept de Bâtiment.

Règle 1 : Si le facteur conceptuel $F_a(L_1)$ d'un langage L_1 et le facteur conceptuel $F_b(L_2)$ d'un langage L_2 portent la sémantique commune d'un concept, alors l'intension du concept est définie grâce aux facteurs conceptuels $F_a(L_1)$ du langage L_1 et de $F_b(L_2)$ du langage L_2 .

Définition 2 : Le Facteur $F_a(L)$ d'un langage L est un *facteur relationnel* s'il définit une relation. Par exemple, si un mur possède une porte, alors le facteur relationnel $F_{Possède}(L)$ est une relation, car il définit une relation entre un mur et une porte.

Règle 2 : Si le facteur $F_a(L_1)$ d'un langage L_1 et le facteur $F_b(L_2)$ d'un langage L_2 portent la sémantique commune d'une relation, alors l'intension de la relation est définie grâce aux facteurs relationnels $F_a(L_1)$ du langage L_1 et de $F_b(L_2)$ du langage L_2 .

Définition 3 : Le Facteur $F_a(L)$ d'un langage L est un *facteur attribut* s'il définit une ou des propriétés d'un concept ou d'une relation. Par exemple, si un mur possède une géométrie, alors le facteur attribut $F_{Géométrie}(L)$ est un attribut, car il définit la géométrie d'un mur.

Règle 3 : Si le facteur $F_a(L_1)$ d'un langage L_1 est un facteur attribut, alors il est intégré à l'intension d'un concept ou d'une relation existante. Par exemple, si un mur possède une propriété thermique, alors le facteur attribut géométrique $F_a(L_1)$ est intégré à l'intension du concept mur. Ce facteur est intégrable à plusieurs intensions de concept et relation.

Les règles et définitions 1, 2 et 3 permettent de définir les facteurs conceptuels, les facteurs relationnels et les facteurs attributs d'un langage. Ces facteurs correspondent à des marques schématiques réalisées sur les éléments de la grammaire XML. Une marque schématique sur une

grammaire XML correspond à un facteur sur le langage. En définissant ces facteurs comme conceptuels, relationnels ou attributs, nous leurs donnons une sémantique. Cette sémantique est déjà portée par les éléments mais ce marquage permet de la rendre explicite et non plus implicite. De plus, la définition de leurs attributs constitue l'intension du concept ou de la relation améliorant ainsi leur définition. En fusionnant les attributs des marques schématiques de différents schémas XML à travers le même concept ou la même relation, nous réalisons une intégration de concept ou de relation. Cette intégration constitue l'intégration de schéma XML. Les règles et définitions suivantes définissent les cas particuliers.

Règles 4 : Si $F_a(L)$ est un facteur relationnel d'un langage L alors $F_a(L)$ lie un facteur conceptuel père à un ensemble de facteurs conceptuels fils.

Règles 5 : Si la trace d'un facteur conceptuel $F_a(L)$ est composée d'un facteur conceptuel $F_b(L)$ alors le lien entre les deux facteurs conceptuels $F_a(L)$ et $F_b(L)$ est un facteur relationnel $F_r(L)$. Par exemple, le facteur conceptuel « mur » possède un facteur conceptuel fils « porte » alors il existe un facteur relationnel entre le facteur conceptuel mur et le facteur conceptuel porte.

Règles 6 : Si la trace d'un facteur conceptuel $F_a(L)$ est composée d'un ensemble de facteurs conceptuels $F_\alpha(L)$ avec $\alpha \in A$ (alphabet $T = A \cup \bar{A}$) et si la sémantique de la relation est la même alors le lien entre $F_a(L)$ et l'ensemble $F_\alpha(L)$ est un facteur relationnel $F_r(L)$ ayant pour facteur conceptuel père $F_a(L)$ et pour ensemble de facteurs conceptuels fils $F_\alpha(L)$. Par exemple, le facteur conceptuel « étage » possède un facteur conceptuel « mur », un facteur conceptuel « colonne », un facteur conceptuel « poutre » et un facteur conceptuel « dalle ». Si la signification du lien entre un étage et les éléments « mur », « colonne », « poutre » et « dalle » est la même alors le facteur relationnel entre l'élément « mur » et les autres est de même type.

Définition 4 : L'intension d'un concept est composée d'un ensemble de marques schématiques sur les grammaires XML. Ces marques schématiques correspondent à des facteurs conceptuels du langage engendrés par les grammaires.

Définition 5 : L'extension d'un concept est composée d'un ensemble d'instances. Dans le cas présent, ces instances nommées éléments sémantiques possèdent une trace qui est un ensemble d'arbres XML.

Définition 6 : L'intension d'une relation est composée d'un ensemble de marques schématiques sur les grammaires XML. Ces marques schématiques correspondent à des facteurs relationnels du langage engendrés par les grammaires.

Définition 7 : L'extension d'une relation est composée d'un ensemble d'instances. Dans le cas présent, ces instances nommées éléments relationnels possèdent une trace qui est un ensemble d'arbres XML.

Définition 8 : Un facteur définit soit un concept, soit une relation, soit un attribut d'un élément d'un schéma XML. Par conséquent, une marque correspond soit à un facteur conceptuel, soit à un facteur relationnel, soit à un facteur attribut.

Règles 7 : Si deux instances d'un concept représentent le même objet ou la même relation alors elles peuvent être identifiées comme identiques. Par exemple, un objet mur possède un ensemble de propriétés thermiques et un autre mur possède une définition géométrique. Ces deux murs sont le même objet s'ils possèdent un identifiant permettant de les identifier comme étant le même objet.

Ces définitions et ces règles font partie intégrante de la conceptualisation du système. Elles fournissent un ensemble de vocabulaire formant la taxinomie de notre système. Ce vocabulaire est composé des mots *concept*, *relation*, *facteur conceptuel*, *facteur relationnel*, *attribut*, *facteur attribut*, *élément sémantique*, *élément relationnel*. Chacun de ces mots définit un concept de l'ontologie et correspondra à une classe.

- La classe *concept* est définie par des propriétés modélisées par son intension et qui est composée d'une liste de facteurs conceptuels.
- La classe *relation* est définie par des propriétés modélisées par son intension et qui sont composées d'une liste de facteurs relationnels.
- Les *éléments sémantiques* et *relationnels* sont les classes permettant d'instancier des objets de documents XML.
- L'ensemble des objets de la classe *élément sémantique* lié à une instance de la classe *concept* forme l'extension de l'instance de *concept*.
- L'ensemble des objets de la classe *élément relationnel* lié à une instance de la classe *relation* forme l'extension de l'instance de *relation*.
- Les instances des classes *facteurs conceptuelles* et *facteurs relationnelles* référencent les marques schématiques sur des grammaires XML. Ces marques sont des documents XML

extraits des grammaires XML. Les traces sont des documents XML extraits des documents XML à intégrer.

Nous avons vu jusqu'à présent des définitions et des règles pour l'intégration des schémas XML dans une ontologie générique. Ces règles permettent de mettre en commun les propriétés de différents schémas XML. Ce niveau d'intégration est nommé intégration de schémas, car il regroupe au cœur du même concept ou relation différentes propriétés définissant l'intension d'un concept ou d'une relation. À ce niveau d'intégration vient s'ajouter un deuxième niveau d'intégration. Celui-ci est nommé intégration de données. En effet, le premier niveau définit les concepts, relations et attributs qui seront instanciés dans le deuxième niveau d'intégration. La section suivante présente la définition de l'ontologie générique sous la forme de classes permettant les deux niveaux d'intégration. Un certain nombre de classes représentent les notions de concept, de relation et d'attribut. Ces classes seront instanciées pour permettre la définition d'ontologies de domaine. De plus, d'autres classes permettront la définition des éléments sémantiques et des éléments de relation. Cette deuxième section sera suivie d'un exemple d'utilisation de l'ontologie pour l'intégration de schémas et l'intégration de données.

3.2. Ontologie générique

Nous avons vu une formalisation partielle des grammaires XML qui permet de construire une ontologie générique. Afin de respecter les objectifs généraux des ontologies, T. Gruber propose cinq critères permettant de guider le processus d'ontologisation [GRU93]. La clarté et l'objectivité doivent être indépendantes de tout choix d'implémentation. L'ontologie doit être extensible, c'est-à-dire qu'il doit être possible de l'étendre sans modification. Les postulats d'encodage doivent être minimes, pour assurer une bonne portabilité. Le vocabulaire doit être minimum pour que l'expressivité de chaque terme soit maximum. Notre ontologie vérifie ses critères car elle permet d'étendre les ontologies de domaine sans avoir à modifier l'ontologie générique. L'ontologisation mène à la construction d'une hiérarchie de concepts, de relations et d'attributs. Une fois le modèle structuré, il faut le traduire dans un langage semi-formel de représentation d'ontologies.

Parmi les langages de représentation développés au niveau conceptuel, il existe trois grands types de modèles principaux : les langages à base de frame, les logiques de description et le modèle des graphes conceptuels. Introduit dans les années 70 en IA, le modèle à base de frames a depuis été adapté à d'autres problématiques puisqu'il a donné naissance au modèle objet. Une frame représente n'importe quelle primitive conceptuelle et est doté d'attributs (slot), qui peuvent porter différentes valeurs, et instances [KIF95]. Les logiques de description permettent de représenter les connaissances sous forme de concepts, de rôles et d'individus [KAY97]. Les rôles

sont des relations binaires entre concepts et les individus sont les instances des concepts. Les propriétés des concepts, rôles et individus sont exprimées en logique des prédictats. Ce langage est utilisé dans le langage de représentation de connaissance OIL développé pour le Web. Introduit par Sowa au début des années 80 [SOW84], le modèle des graphes conceptuels se décompose en deux niveaux. Le niveau terminologique est le niveau qui décrit les concepts, les relations et les instances de concepts. Le niveau assertional est le niveau qui représente les faits, les règles et les contraintes sous forme de graphes où les sommets sont des instances de concepts et les relations des arcs. Pour définir notre ontologie, nous avons choisi le langage à base de frame, car le développement lié aux services web qui seront utilisés dans notre architecture est réalisé dans le langage de programmation orienté objets JAVA.

À présent, nous allons décrire chaque frame ou classe de l'ontologie. Celles-ci possèdent un ensemble de slots ou attributs qui sont parfois définis pour plusieurs classes. Les instances des slots sont utilisables par plusieurs objets limitant ainsi la redondance des informations. Ces classes ont été définies à l'aide de l'outil de création d'ontologie Protégé 2000.

Class Concept²

Slot name	Type	Allowed Values/Classes	Cardinality
Nom	String		1:1
Marque	Instance	Facteur_concept	0:*
Intension	Instance	Facteur_attribut	0:*
Extension	Instance	Element_semantique	0:*

Un concept est caractérisé par son nom qui le rend unique. Il possède une notion qui est définie par la propriété Intension. La marque du concept contient les balises XML éléments extraits du schéma XML. Cette marque permet de distinguer les différents éléments d'un schéma XML pour reconnaître les balises dans document XML. L'intension de la classe est composée d'une liste de facteurs conceptuels définie à l'aide d'un ensemble de marques schématiques provenant de différents schémas XML. L'extension d'un concept possède un ensemble d'instances de la classe Element_semantique correspondant au concept.

Class Relation

Slot name	Type	Allowed Values/Classes	Cardinality
Nom	String		1:1
Marque	Instance	Facteur_relation	0:*
Concept_pere	Instance	Facteur_concept	0:*
Concept_fils	Instance	Facteur_concept	0:*
Intension	Instance	Facteur_attribut	0:*
Extension	Instance	Element_relation	0:*

² La cardinalité 1:1 signifie qu'une instance de classe possède obligatoirement une et une seule instance pour ce slot. La cardinalité 0:/* signifie qu'une instance de classe possède éventuellement une ou « n » instances pour ce slot.

Une relation est caractérisée par son nom qui le rend unique. Il possède une notion qui est définie par la propriété Intension. La marque de la relation contient les balises XML éléments extraits du schéma XML. Cette marque permet de distinguer les différents éléments d'un schéma XML pour reconnaître les balises dans un document XML. Les propriétés Concept_pere et Concept_fils contiennent les balises des éléments des schémas XML pour être père de la relation et pouvant être fils de la relation. L'intension de la classe est composée d'une liste de facteurs relationnelle définie à l'aide d'un ensemble de marques schématiques provenant de différents schémas XML. L'extension d'une relation possède un ensemble d'instances de la classe Element_relation correspondant à la Relation.

Class Attribut

Slot name	Type	Allowed Values/Classes	Cardinality
Nom	String		1:1
Marque	Instance	Facteur_attribut	0:*
Attribut	Instance	Attribut	0:*
Extension	Instance	Element_attribut	0:*

Un attribut est caractérisé par son nom qui le rend unique. Il possède une notion qui est définie par la propriété Intension. La marque d'attribut contient les balises XML éléments extraites du schéma XML. Cette marque permet de distinguer les différents éléments d'un schéma XML pour reconnaître les balises dans un document XML. Il n'existe pas d'intension pour cette classe, car l'attribut a pour fonction de définir l'intension d'un concept ou d'une relation. La propriété Attribut constitue une hiérarchie d'attributs. En effet, un élément attribut peut avoir un lien direct avec un autre élément attribut, donc pour modéliser cette relation contenue dans un schéma XML nous ajoutons cette propriété. L'extension d'un attribut possède un ensemble d'instances de la classe Element_attribut correspondant au concept.

Class Facteur_concept

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Document_XML	String		0:1
AttributSimple	String		0:*
Id_schema	String		1:1

La classe Facteur_concept correspond à la définition d'une marque schématique sur un schéma XML. Cette marque ou facteur conceptuel possède un identifiant pour le distinguer des autres. La propriété Document_XML contient la balise élément du schéma XML et la propriété Id_schema contient l'identifiant du schéma, car les marques sur deux schémas XML peuvent être identiques. La propriété AttributSimple représente les attributs que possède une balise XML dans un document XML. En effet, un concept possède des éléments attributs caractérisés par des

balises élément dans le schéma XML et possède des attributs caractérisés par les balises attributs dans le schéma XML.

Class Facteur_relation

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Document_XML	String		0:1
AttributSimple	String		0:*
Id_schema	String		1:1

La classe Facteur_relation correspond à la définition d'une marque schématique sur un schéma XML. Cette marque ou facteur relationnel possède un identifiant pour le distinguer des autres, car les facteurs relationnels sont uniques. La propriété Document_XML contient la balise élément du schéma XML et la propriété Id_schema contient l'identifiant du schéma, car les marques sur deux schémas XML peuvent être identiques. Comme pour la classe Facteur_concept, la propriété AttributSimple correspond aux attributs qu'un élément possède, elle représente les attributs que possède une balise XML dans un document XML. En effet, une relation possède des éléments attributs caractérisés par des balises élément dans le schéma XML et possède des attributs caractérisés par les balises attributs dans le schéma XML.

Class Facteur_attribut

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Document_XML	String		0:1
AttributSimple	String		0:*
Id_schema	String		1:1

La classe Facteur_attribut correspond à la définition d'une marque schématique sur un schéma XML. Cette marque ou facteur attribut possède un identifiant pour le distinguer des autres, car les facteurs attribut sont uniques. La propriété Document_XML contient la balise élément du schéma XML et la propriété Id_schema contient l'identifiant du schéma, car les marques sur deux schémas XML peuvent être identiques. Comme pour la classe Facteur_concept, la propriété AttributSimple correspond aux attributs qu'un élément possède, elle représente les attributs que possède une balise XML dans un document XML. En effet, un élément attribut possède des attributs caractérisés par les balises attributs toujours dans le schéma XML.

Class AttributSimple

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Document_XML	String		0:1
Id_schema	String		1:1

La classe Facteur_attribut correspond à la définition des attributs d'une marque schématique sur un schéma XML. Ces attributs simples possèdent un identifiant pour les distinguer des autres. La propriété Document_XML contient la balise élément du schéma XML et la propriété Id_schema contient l'identifiant du schéma, car les attributs simples sur deux schémas XML peuvent avoir le même nom.

Class Element_semantique

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Type	Instance	Concept	1:1
Nom	String		0:1
Liste_attribut	Instance	Element_attribut	0:*
Liste_attribut_simple	Instance	Attribut_simple	0:*

La classe Element_semantique correspond à la définition d'un élément sémantique. L'identifiant Id rend les objets de la classe unique. Les objets de cette classe possèdent un ensemble d'attributs. L'attribut Type correspond à un concept et référence donc une instance de la classe concept. Le nom est l'identifiant de l'élément sémantique permettant l'intégration de plusieurs éléments sémantiques provenant de différents documents XML. La liste d'attributs correspond à la liste des éléments attributs de l'élément sémantique. La liste d'attributs simples correspond à l'ensemble des balises des éléments sémantiques des documents XML intégrés.

Class Element_relation

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Type	Instance	Relation	1:1
Nom	String		0:1
Liste_attribut	Instance	Element_attribut	0:*
Liste_attribut_simple	Instance	Attribut_simple	0:*
Elt_Sem_Pere	Instance	Element_semantique	0:1
Elt_Sem_Fils	Instance	Element_semantique	0:*

La classe Element_relation correspond à la définition d'un élément relationnel. L'identifiant Id rend les objets de la classe unique. Les objets de cette classe possèdent un ensemble d'attributs. L'attribut Type correspond à une relation et référence donc une instance de la classe relation. Le nom est l'identifiant de l'élément relation permettant l'intégration de plusieurs éléments relation provenant de différents documents XML. La liste d'attributs correspond à la liste des éléments attributs de l'élément relation. La liste d'attributs simples correspond à l'ensemble des balises des éléments sémantiques des documents XML intégrés.

Class Element_attribut

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Type	Instance Attribut		1:1
Nom	String		0:1
Id_Element_Attribut	Instance Element Attribut		0:*
Document_XML	String		0:1

La classe Element_attribut correspond à la définition d'un élément attribut. L'identifiant Id rend les objets de la classe unique. L'attribut Type correspond à un élément de la classe attribut et référence donc une instance de cette classe. Le nom est l'identifiant de l'élément attribut permettant l'intégration de plusieurs éléments attributs provenant de différents documents XML. L'identifiant d'éléments d'attributs permet de faire le lien avec d'autres éléments d'attributs pour conserver la hiérarchie d'éléments d'attributs. La propriété Document_XML contient la balise du document XML.

Class Attribut_simple

Slot name	Type	Allowed Values/Classes	Cardinality
Id	String		1:1
Type	Concept/Relation		1:1
Id_instance	String		1:1
Id_schema	String		1:1
Document_XML	String		0:1

La classe Attribut_simple contient les attributs des balises d'un élément sémantique ou relation. L'identifiant Id rend les objets de la classe unique. L'attribut Type définit le type d'élément auquel correspond l'attribut simple, c'est-à-dire un concept ou une relation. L'Id_instance référence soit une instance d'élément sémantique, soit une instance d'élément relation. L'Id_schema permet d'identifier le schéma XML auquel correspond cet attribut. La propriété Document_XML contient la balise de l'élément attribut.

Nous avons vu dans cette section les classes nécessaires pour définir l'ontologie générique. Ces classes peuvent être considérées comme des méta-classes dont les instances sont des classes définissant les ontologies de domaine. La méta-classe concept permet de définir des classes comme une classe « mur » spécifiant les attributs du concept « mur » dans une ontologie du domaine du bâtiment par exemple. Ces méta-classes conservent le lien entre les classes et les instances permettant une recherche rapide dans le système d'information soit à l'aide des métadonnées, soit directement sur les données. À présent que nous avons vu les règles d'intégration dans la première section et les classes de l'ontologie générique dans la deuxième section, la

section suivante présente un exemple d'utilisation des classes pour l'intégration de schéma, ainsi que l'intégration de données.

4. Exemple

Pour éclaircir l'usage des classes de l'ontologie, nous allons voir dans cette section un exemple d'intégration. Cette section est composée de deux parties, dont la première expose l'intégration de schémas (c.f. Figure 4.5). Cette partie définit les concepts et les relations communes aux différents schémas XML, ainsi que les attributs des concepts. Ces concepts, ces relations et ces attributs seront définis à l'aide des classes de l'ontologie. La deuxième partie de cette section présente l'intégration de données de différents documents validés par les différents schémas XML précédemment intégrées.

4.1. Intégration de schémas

Nous allons commencer par voir quels sont les schémas qui seront intégrés. Ces schémas sont de trois types différents. En ce qui concerne la partie géométrique, nous utiliserons le schéma XML X3D pour définir la partie graphique de l'intégration. L'exemple 4.8 permettra de définir un arbre de contenance des éléments constituant un bâtiment. Un troisième schéma XML apportera des informations techniques sur les éléments du bâtiment.

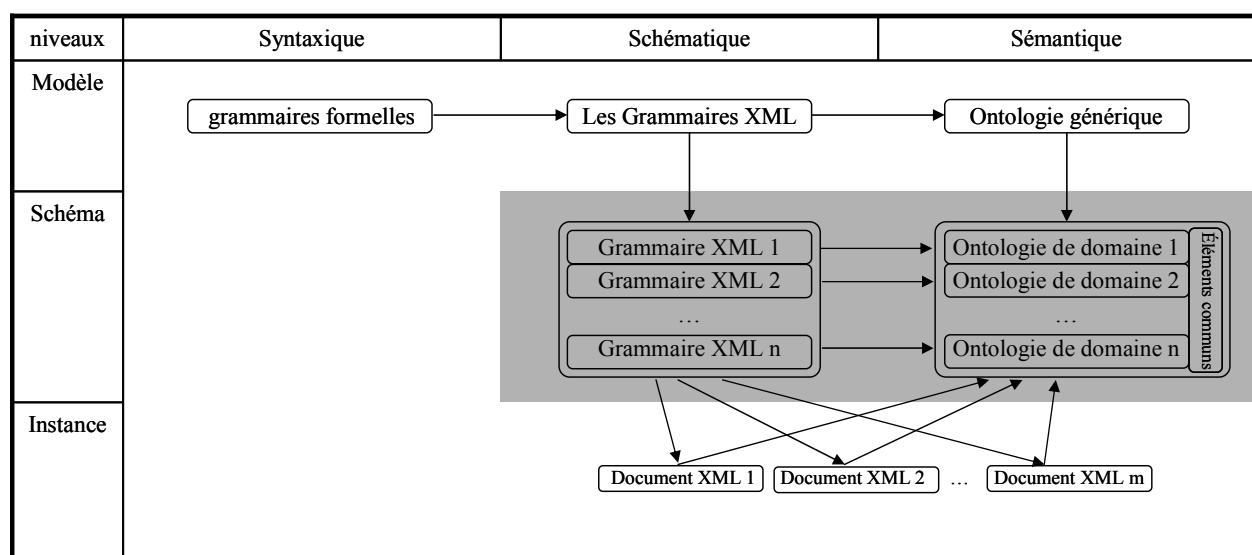


Figure 4.4. Intégration des schémas et définition des ontologies de domaine

L'intégration des schémas débute par le schéma X3D. Nous ne présentons ici qu'une partie des éléments du schéma X3D, car la présentation exhaustive des éléments du schéma n'a aucun intérêt pour la compréhension de la méthode. Pour cela, seuls les éléments X3D, head, Shape,

Scene, Transform ont été retenus, car se sont les éléments par minimum nécessaire pour définir une scène 3D. L'exemple 4.7 fournit un extrait du schéma X3D.

- **X3D** : L'élément X3D est la balise racine d'un document X3D, donc nous marquons cet élément pour le définir comme concept racine d'une scène graphique. Cet élément possède un ensemble d'éléments qui sont un ensemble éventuellement vide d'éléments head et un élément Scene. D'après la règle, si un facteur conceptuel est composé d'autres facteurs conceptuels, alors un facteur relation est défini entre ces deux éléments. Par conséquent, nous définissons un facteur relationnel **RelX3D** qui lie les éléments X3D aux éléments head et Scene.
- **head** : L'élément head fournit un ensemble d'informations sur le fichier et possède des éléments component et meta. Les éléments component et meta sont marqués comme des attributs, car ils définissent des attributs du concept head.
- **meta** : L'élément meta est un élément attribut qui sera utilisé pour définir l'intension du concept head.
- **component** : L'élément component est un élément attribut qui sera utilisé pour définir l'intension du concept head.
- **Scene** : L'élément Scene est marqué comme concept, car il définit la racine graphique de la scène. (c.f. Transform)
- **Shape** : L'élément shape définit les propriétés graphiques d'un objet de la scène, alors il représente un attribut et par conséquent une marque schématique attribut est réalisée sur l'élément shape.
- **Transform** : L'élément Transform est une relation qui lie deux éléments graphiques par une transformation géométrique. Cet élément est un cas particulier, car un élément Transform pour contenir un élément Transform (c.f. Figure 4.5). Malheureusement, il n'est pas possible de lier un élément de relation avec un autre élément de relation. Par contre, il est possible de le définir comme élément attribut Transform et de définir un élément de relation **RelPlacement** dont l'intension est composée de l'attribut Transform. De cette manière, les éléments sémantiques possédant un élément attribut Shape seront liés par un élément de relation RelPlacement (c.f. Figure 4.6). Donc contrairement au schéma X3D, à présent ce sont les éléments graphiques qui sont liés par une transformation et non les transformations qui possèdent des éléments graphiques. Par exemple, un élément Scene est lié à un autre élément sémantique possédant un élément attribut Transform par l'intermédiaire d'un élément de relation RelPlacement.

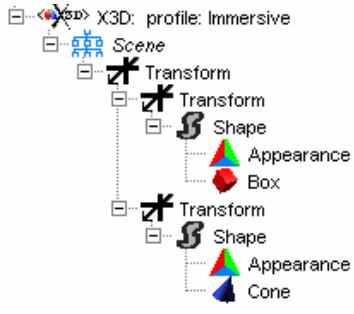


Figure 4.5. Avant modification

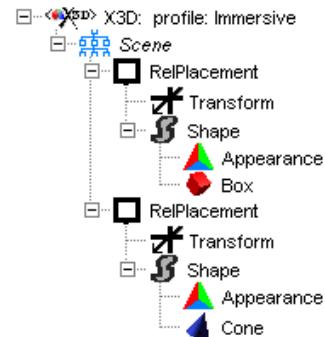


Figure 4.6. Après modification

Contrairement aux instances d'attribut et de concept, les instances de relation ne possèdent pas de marque. En effet, aucune relation n'a été définie à partir des éléments du schéma X3D, par conséquent ces nouvelles relations ne possèdent pas de balise dans les documents X3D.

```

<xsd:element name="X3D">
  <xsd:annotation>
    <xsd:appinfo/>
    <xsd:documentation source="http://www.web3d.org/specifications/ISO-IEC-19776/Part01/concepts.html#Header"/>
  </xsd:annotation>
  <xsd:complexType mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="SceneGraphStructureNodeType">
        <xsd:sequence>
          <xsd:element ref="head" minOccurs="0"/>
          <xsd:element ref="Scene"/>
        </xsd:sequence>
        <xsd:attribute name="version" type="SFString" use="optional" fixed="3.0"/>
        <xsd:attribute name="profile" type="profileNames" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
...
<xsd:element name="head">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="SceneGraphStructureNodeType">
        <xsd:sequence>
          <xsd:element ref="component" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="meta" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
...
<xsd:element name="Shape">
  <xsd:annotation>
    <xsd:appinfo/>
    <xsd:documentation source="http://www.web3d.org/specifications/ISO-IEC-19775/Part01/components/shape.html#Shape"/>
  </xsd:annotation>
  <xsd:complexType mixed="false">
    <xsd:complexContent mixed="false">
      <xsd:extension base="X3DShapeNode"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
...
<xsd:element name="Scene">

```

```

<xsd:annotation>
    <xsd:appinfo>
        <xsd:documentation source="http://www.web3d.org/specifications/ISO-IEC-19776/Part01/concepts.html#Header"/>
    </xsd:annotation>
    <xsd:complexType mixed="false">
        <xsd:complexContent mixed="false">
            <xsd:extension base="SceneGraphStructureNodeType">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:group ref="GroupingNodeChildContentModel"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
...
<xsd:element name="Transform">
    <xsd:annotation>
        <xsd:appinfo>
            <xsd:documentation source="http://www.web3d.org/specifications/ISO-IEC-19775/Part01/components/group.html#Transform"/>
        </xsd:annotation>
        <xsd:complexType mixed="false">
            <xsd:complexContent mixed="false">
                <xsd:extension base="X3DGroupingNode">
                    <xsd:attribute name="center" type="SFVec3f" default="0 0 0"/>
                    <xsd:attribute name="rotation" type="SFRotation" default="0 0 1 0"/>
                    <xsd:attribute name="scale" type="SFVec3f" default="1 1 1"/>
                    <xsd:attribute name="scaleOrientation" type="SFRotation" default="0 0 1 0"/>
                    <xsd:attribute name="translation" type="SFVec3f" default="0 0 0"/>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```

Exemple 4.7. Extrait du schéma X3D

Pour récapituler, nous avons défini les éléments suivants :

- Attributs : Shape, Transform, component, meta
- Concepts : X3D, head, Scene
- Relation : RelX3D, RelPlacement

A présent nous allons définir les instances des classes pour l'ontologie de domaine X3D. Les classes concept, relation et attribut vont être instanciés pour définir les nouvelles classes de l'ontologie.

Instance Attribut

Nom	Marque	Attribut	Extension
Shape	[Shape]	null	null
Transform	[Transform]	null	null
component	[component]	null	null
meta	[meta]	null	null

Instances Facteur_attribut

Id	Document XML	Id schema
Shape	[<xsd:element name="Shape">]	X3D
Transform	[<xsd:element name="Transform">]	X3D
component	[<xsd:element name="component">]	X3D
meta	[<xsd:element name="meta">]	X3D

Instances Concept

Nom	Marque	Intension	Extension
X3D	[X3D]	null	null
head	[head]	[component, meta]	null
Scene3D	[Scene]	null	null

Instances Facteur_concept

Id	Document XML	Id_schema
X3D	[<xsd:element name="X3D">]	X3D
head	[<xsd:element name="head">]	X3D
Scene	[<xsd:element name="Scene">]	X3D

Instances Relation

Nom	Marque	Intension	Extension	Concept_pere	Concept_fils
RelX3D	null	null	null	X3D	[head, Scene3D]
RelPlacement	null	Transform	null	[*]	[*]

[*] : En pratique tous les éléments sémantiques peuvent être liées par un élément relation RelPlacement, mais il est préférable de limiter cette liste aux éléments sémantiques possédant un élément attribut Shape.

Le schéma suivant que nous allons étudier et intégrer à notre système est un schéma décrivant les éléments composant un bâtiment. Nous commençons par définir les concepts suivants permettant de définir les éléments d'un bâtiment : Building, BuildingStorey, Slab, Wall, Door, Beam, Column, Window. Aucun élément de ce schéma XML ne définit un élément attribut, car les éléments définis dans le schéma XML représentent des concepts. D'après la règle 5, si un facteur conceptuel est composé d'autres facteurs conceptuels, alors un facteur relation est défini entre ces deux éléments. Par conséquent, nous définissons un facteur relationnel RelContents qui lie les éléments définis aux autres éléments.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified">
<xs:element name="Building">
<xs:complexType>
<xs:sequence>
<xs:element ref="BuildingStorey" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="BuildingStorey">
<xs:complexType>
<xs:sequence>
<xs:element ref="Slab" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="Wall" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="Beam" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="column" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

```

<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Slab">
<xs:complexType>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedTo" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedFrom" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Wall">
<xs:complexType>
<xs:sequence>
<xs:element ref="Door" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="Window" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedTo" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedFrom" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Door">
<xs:complexType>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedTo" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedFrom" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Beam">
<xs:complexType>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedTo" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedFrom" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="column">
<xs:complexType>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedTo" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedFrom" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Window">
<xs:complexType>
<xs:attribute name="Name" type="xs:string" use="required"/>
<xs:attribute name="Description" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedTo" type="xs:string" use="optional"/>
<xs:attribute name="ConnectedFrom" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:attribute name="Name" type="xs:string"/>
<xs:attribute name="Description" type="xs:string"/>
<xs:attribute name="ConnectedTo" type="xs:string"/>
<xs:attribute name="ConnectedFrom" type="xs:string"/>
</xs:schema>

```

Exemple 4.8. Schéma XML de définition d'un bâtiment

Pour récapituler, nous avons défini les éléments suivants :

- Concepts : Building, BuildingStorey, Slab, Wall, Door, Beam, Column, Window
- Relation : RelContents

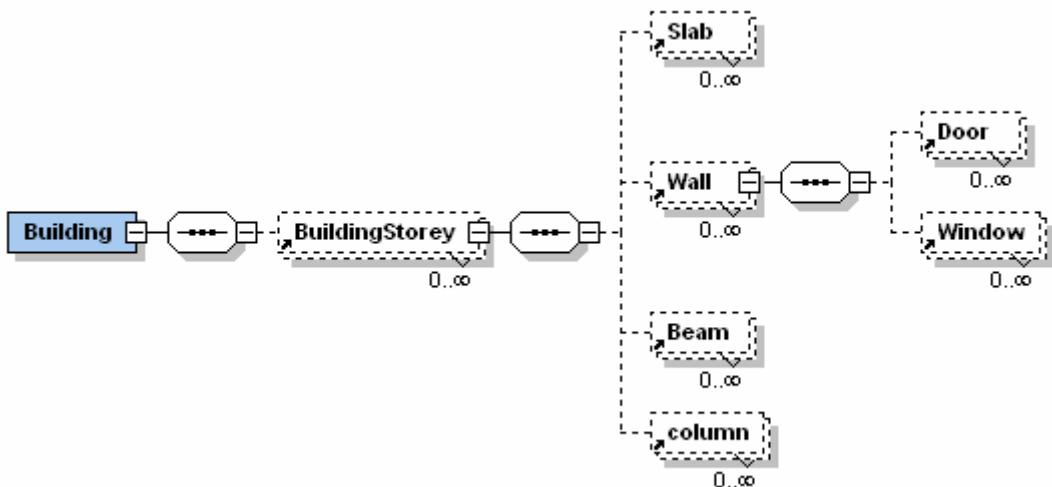


Figure 4.3. Graphe du Schéma XML de définition d'un bâtiment

A présent nous allons définir les instances des classes pour l'ontologie de domaine pour ce schéma. Les classes concept et relation vont être instanciés pour définir les nouvelles classes de l'ontologie.

Instance Attribut

Nom	Marque	Attribut	Extension
Shape	[Shape]	null	null
Transform	[Transform]	null	null
component	[component]	null	null
meta	[meta]	null	null

Instances Facteur_attribut

Id	Document XML	Id_schema
Shape	[<xsd:element name="Shape">]	X3D
Transform	[<xsd:element name="Transform">]	X3D
component	[<xsd:element name="component">]	X3D
meta	[<xsd:element name="meta">]	X3D

Instances Concept

Nom	Marque	Intension	Extension
X3D	[X3D]	null	null
head	[head]	[component, meta]	null
Scene3D	[Scene]	null	null
Building	[Building]	[Shape]	null
BuildingStorey	[BuildingStorey]	[Shape]	null
Slab	[Slab]	[Shape]	null
Wall	[Wall]	[Shape]	null
Door	[Door]	[Shape]	null
Beam	[Beam]	[Shape]	null
Column	[Column]	[Shape]	null
Window	[Window]	[Shape]	null

A présent, nous voulons intégrer les informations structurelles du schéma de définition d'un bâtiment avec les informations géométriques du schéma X3D. Pour cela, nous ajoutons aux concepts définis dans le schéma de définition de bâtiment, un attribut shape permettant de définir une propriété graphique aux éléments sémantiques. De cette manière, il est possible de définir une forme géométrique provenant d'un document X3D à un élément du bâtiment du schéma définissant un bâtiment.

Instances Facteur_concept

Id	Document XML	Id schema
X3D	[<xsd:element name="X3D">]	X3D
head	[<xsd:element name="head">]	X3D
Scene	[<xsd:element name="Scene">]	X3D
Building	[<xs:element name="Building">]	Batiment
BuildingStorey	[<xs:element name="BuildingStorey">]	Batiment
Slab	[<xs:element name="Slab">]	Batiment
Wall	[<xs:element name="Wall">]	Batiment
Door	[<xs:element name="Door">]	Batiment
Beam	[<xs:element name="Beam">]	Batiment
Column	[<xs:element name="Column">]	Batiment
Window	[<xs:element name="Window">]	Batiment

Instances Relation

Nom	Marque	Intension	Extension	Concept pere	Concept fils
RelX3D	null	null	null	[X3D]	[head, Scene3D]
RelPlacement	null	Transform	null	[*]	[*]
RelContents	null	null	null	listConcept**	listConcept**

**listConcept = [Building, BuildingStorey, Slab, Wall, Door, Beam, Column, Window]

Le dernier schéma que nous allons étudier et intégrer à notre système est un schéma décrivant les propriétés thermiques des murs et des dalles d'un bâtiment. Les concepts Building, Wall et Slab sont des concepts existants dans le schéma précédent, par conséquent l'intension de ces concepts sera intégrée à l'intension du concept existant. Deux nouveaux concepts sont définis dans ce schéma MaterialLayer et Space. MaterialLayer définit une couche de matériaux dans un mur ou une dalle et le concept Space définit un espace comme une cuisine ou se trouve un ensemble de murs et de dalles. Ces murs et ces dalles sont liés à ensemble de couches de matériaux par l'élément relation RelMaterialSet. Dans ce schéma aucun élément ne définit un attribut. D'après la règle 5, si un facteur conceptuel est composé d'autres facteurs conceptuels, alors un facteur relation est défini entre ces deux éléments. Par conséquent, nous définissons un facteur relationnel RelSpace qui lie les éléments Building avec les éléments Space. De plus, nous définissons une relation entre l'élément Space et les éléments Wall et Slab nommé RelSpaceContents.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="Space">
    <xs:complexType>
      <xs:sequence minOccurs="0">
        <xs:element ref="Slab" maxOccurs="unbounded"/>
        <xs:element ref="Wall" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Slab">
    <xs:complexType>
      <xs:sequence minOccurs="0">
        <xs:element ref="MaterialSet"/>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Wall">
    <xs:complexType>
      <xs:sequence minOccurs="0">
        <xs:element ref="MaterialSet"/>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Building">
    <xs:complexType>
      <xs:sequence minOccurs="0">
        <xs:element ref="Space" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="MaterialLayer">
    <xs:complexType>
      <xs:attribute name="Name" type="xs:string" use="required"/>
      <xs:attribute name="Material" type="xs:string" use="optional"/>
      <xs:attribute name="LayerThickness" type="xs:string" use="optional"/>
      <xs:attribute name="IsVentilated" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:attribute name="Name"/>
  <xs:attribute name="LayerThickness"/>
  <xs:attribute name="IsVentilated"/>
  <xs:attribute name="Material"/>
  <xs:element name="MaterialSet">
    <xs:complexType>
      <xs:sequence minOccurs="0">
        <xs:element ref="MaterialLayer" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Exemple 4.9. Schéma XML de définition thermique

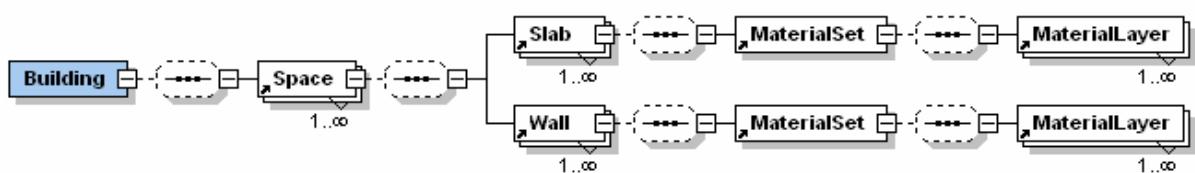


Figure 4.4. Graphe du Schéma XML de définition thermique

Pour récapituler, nous avons défini les éléments suivants :

- Concepts : Building, Space, Wall, Slab, MaterialLayer
- Relation : RelMaterialSet, RelSpace, RelSpaceContents

A présent nous allons définir les instances des classes pour l'ontologie de domaine thermique. Les classes concept et relation vont être instanciées pour définir les nouvelles classes de l'ontologie.

Instance Attribut

Nom	Marque	Attribut	Extension
Shape	[Shape]	null	null
Transform	[Transform]	null	null
component	[component]	null	null
meta	[meta]	null	null

Instances Facteur_attribut

Id	Document XML	Id_schema
Shape	[<xsd:element name="Shape">]	X3D
Transform	[<xsd:element name="Transform">]	X3D
component	[<xsd:element name="component">]	X3D
meta	[<xsd:element name="meta">]	X3D

Instances Concept

Nom	Marque	Intension	Extension
X3D	[X3D]	null	null
head	[head]	[component, meta]	null
Scene3D	[Scene]	null	null
Building	[Building]	[Shape]	null
BuildingStorey	[BuildingStorey]	[Shape]	null
Slab	[Slab]	[Shape]	null
Wall	[Wall]	[Shape]	null
Door	[Door]	[Shape]	null
Beam	[Beam]	[Shape]	null
Column	[Column]	[Shape]	null
Window	[Window]	[Shape]	null
Space	[Space]	[Shape]	null
MaterialLayer	[MaterialLayer]	[Shape]	null

A présent, nous voulons intégrer les informations concernant les couches matériaux du schéma de définition thermique avec les informations géométriques du schéma X3D. Pour cela, nous ajoutons aux concepts définis dans le schéma de définition de bâtiment, un attribut shape permettant de définir une propriété graphique aux éléments sémantiques, comme nous l'avons déjà fait pour les éléments sémantiques du schéma de définition d'un bâtiment. De cette manière, il est possible de définir une forme géométrique provenant d'un document X3D à un élément MaterialLayer du schéma de définition thermique.

Instances Facteur_concept

Id	Document XML	Id schema
X3D	[<xsd:element name="X3D">]	X3D
head	[<xsd:element name="head">]	X3D
Scene	[<xsd:element name="Scene">]	X3D
Building	[<xs:element name="Building">,<xs:element name="Building">]	Batiment, Thermique
BuildingStorey	[<xs:element name="BuildingStorey">]	Batiment
Slab	[<xs:element name="Slab">,<xs:element name="Slab">]	Batiment, Thermique
Wall	[<xs:element name="Wall">,<xs:element name="Wall">]	Batiment, Thermique
Door	[<xs:element name="Door">]	Batiment
Beam	[<xs:element name="Beam">]	Batiment
Column	[<xs:element name="Column">]	Batiment
Window	[<xs:element name="Window">]	Batiment
Space	[<xs:element name="Space">]	Thermique
MaterialLayer	[<xs:element name="MaterialLayer">]	Thermique

Instances Relation

Nom	Marque	Intension	Extension	Concept pere	Concept fils
RelX3D	null	null	null	X3D	[head, Scene3D]
RelPlacement	null	Transform	null	[*]	[*]
RelContents	null	null	null	listConcept**	listConcept**
RelMaterialSet	MaterialSet	null	null	[Wall, Slab]	[MaterialLayer]
RelSpace	null	null	null	[Building]	[Space]
RelSpaceContents	null	null	null	[Space]	[Wall, Slab]

**listConcept = [Building, BuildingStorey, Slab, Wall, Door, Beam, Column, Window]

Instances Facteur_relation

Id	Document XML	Id schema
MaterialSet	[<xsd:element name=" MaterialSet ">]	Thermique

A présent notre système a intégré trois schémas XML permettant d'intégrer des données graphiques, thermiques et structurelles d'un bâtiment. Certains concepts comme le concept mur est partagé par plusieurs schémas, alors ces éléments sont intégrés au sein du même concept. De même, des attributs comme l'attribut shape sont intégrés à l'intension de certains concepts ne faisant pas partie du schéma original X3D. De cette manière, il est possible d'intégrer des propriétés et des données d'un schéma à un autre schéma ne possédant pas cette propriété. Nous avons vu aussi le cas où il est nécessaire de définir un concept à la place d'une relation, car une relation ne peut lier une autre relation dans notre système. En effet, le fait d'autoriser cette règle pourrait engendrer des problèmes de cohérence de données, car s'il est possible de lier une transformation géométrique à une autre transformation géométrique, il est complètement incohérent de lier une transformation géométrique et un lien de contenance.

Nous allons voir maintenant comment utiliser ces données pour intégrer des données XML dans notre système. La section suivante présente un exemple d'intégration de trois documents XML,

dont le premier est un document X3D, le second est un document de description de bâtiment et le dernier est un document de description thermique d'éléments du bâtiment (c.f. Figure 4.7).

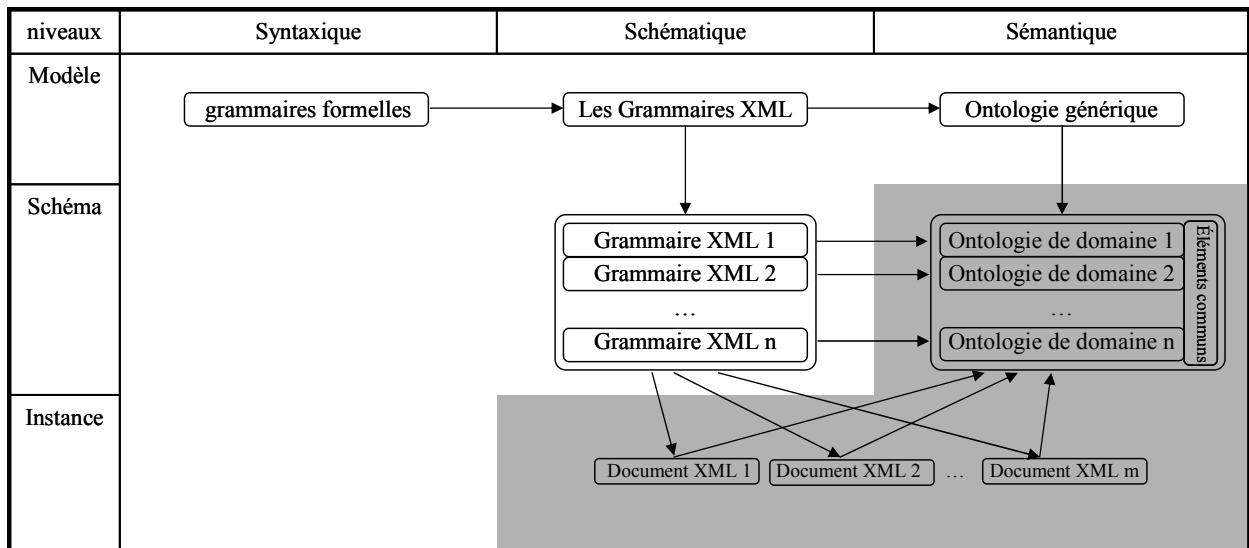


Figure 4.7. Intégration des données XML

4.2. Intégration de données

Pour décrire l'intégration de documents, nous allons commencer par un document de description de bâtiment. Ce document décrit les éléments d'un étage de bâtiment contenant un certain nombre de pièces composées de murs, de portes et de fenêtres.

```

<Building xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\ batiment.xsd" Name="BâtimentEssai"
Description="Ce fichier décrit la structure d'un bâtiment" >
  <BuildingStorey Name="RdC">
    <Slab Name="Sol"/>
    <Wall Name="Nord">
      <Door Name="Entrée"/>
    </Wall>
    <Wall Name="Sud"/>
    <Wall Name="Est">
      <Window Name="FenêtreEst"/>
    </Wall>
    <Wall Name="Ouest"/>
  </BuildingStorey>
</Building>
  
```

Exemple 4.10. Document XML de définition d'un bâtiment

Dans l'exemple 4.10, nous avons un fichier de description de bâtiment. Ce fichier contient un bâtiment, un étage, une dalle, quatre murs et une porte. Ces éléments sont des éléments sémantiques instanciés à partir de la définition des concepts. Les relations entre ces éléments seront aussi instanciées à l'aide de la relation RelContents. Nous commencerons par instancier les éléments sémantiques, puis les éléments de relations.

Instances Element_semantique

Id	Type	Nom	Liste attribut	Liste attribut simple
001	Building	BâtimentEssai	null	[001]
002	BuildingStorey	RdC	null	null
003	Slab	Sol	null	null
004	Wall	Nord	null	null
005	Wall	Sud	null	null
006	Wall	Est	null	null
007	Wall	Ouest	null	null
008	Door	Entrée	null	null
009	Window	FenêtreEst	null	null

Instances Element_relation

Id	Type	Nom	Liste_attribut	Liste_attribut_simple	Elt_Sem_Pere	Elt_Sem_Fils
001	RelContents	null	null	null	001	[002]
002	RelContents	null	null	null	002	[003,004,005,006,007]
003	RelContents	null	null	null	004	[008]
004	RelContents	null	null	null	006	[009]

Instance Attribut_Simple

Id	Type	Id_Instance	Id_schema	Document_XML
001	Concept	001	Batiment	<Building xmlns ... la structure d'un bâtiment" >

Cette première insertion d'éléments sémantiques et d'éléments de relation est simple, car elle ne possède pas d'élément attribut. Par contre l'élément building possède un ensemble d'attributs simples. Le document suivant ne possède pas non plus d'élément attribut par contre, il possède des concepts communs avec le document précédent. Nous commencerons par fusionner les éléments sémantiques à l'aide de l'identifiant commun Name. Dans le cas où les éléments ne possèdent pas d'attribut commun pour les identifier, la fusion des éléments est réalisée manuellement en mettant en correspondance les éléments identiques.

```

<?xml version="1.0" encoding="UTF-8"?>
<Building xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Christophe Cruz\Bureau\test XML
           Spy\thermique.xsd"
           Name="BâtimentEssai ">
    <Space Name="Hall">
        <Slab Name="Sol">
            <MaterialSet Name="Sol">
                <MaterialLayer Name="béton"/>
            </MaterialSet>
        </Slab>
        <Wall Name="Nord">
            <MaterialSet Name="">
                <MaterialLayer Name="béton"/>
                <MaterialLayer Name="pierre"/>
            </MaterialSet>
        </Wall>
        <Wall Name="Sud">
            <MaterialSet Name="">
                <MaterialLayer Name="béton"/>
                <MaterialLayer Name="pierre"/>
            </MaterialSet>
        </Wall>
        <Wall Name="Est">
    
```

```

<MaterialSet Name="">
    <MaterialLayer Name="béton"/>
    <MaterialLayer Name="pierre"/>
</MaterialSet>
</Wall>
<Wall Name="Ouest">
    <MaterialSet Name="">
        <MaterialLayer Name="béton"/>
        <MaterialLayer Name="pierre"/>
    </MaterialSet>
</Wall>
</Space>
</Building>
    
```

Exemple 4.11. Document XML de définition thermique

Instances Element_semantique

Id	Type	Nom	Liste_attribut	Liste_attribut_simple
001	Building	BâtimentEssai	null	[001,002]
002	BuildingStorey	Rdc	null	null
003	Slab	Sol	null	null
004	Wall	Nord	null	null
005	Wall	Sud	null	null
006	Wall	Est	null	null
007	Wall	Ouest	null	null
008	Door	Entrée	null	null
009	Window	FenêtreEst	null	null
010	Space	Hall	null	null
011	MaterialLayer	béton	null	null
012	MaterialLayer	béton	null	null
013	MaterialLayer	pierre	null	null
014	MaterialLayer	béton	null	null
015	MaterialLayer	pierre	null	null
016	MaterialLayer	béton	null	null
017	MaterialLayer	pierre	null	null
018	MaterialLayer	béton	null	null
019	MaterialLayer	pierre	null	null

Nous avons ajouté à la liste des éléments sémantiques les éléments Space et MaterialLayer. Les éléments Building, BuildinfStorey, Wall et Slab sont intégrés aux éléments existants, car il possède le même nom, soit le même identifiant. De plus, nous avons ajouté une référence à un attribut simple pour l'élément sémantique Building.

Instances Element_relation

Id	Type	Nom	Liste_attribut	Liste_attribut_simple	Elt_Sem_Pere	Elt_Sem_Fils
001	RelContents	null	null	null	001	[002]
002	RelContents	null	null	null	002	[003,004,005,006,007]
003	RelContents	null	null	null	004	[008]
004	RelContents	null	null	null	006	[009]
005	RelSpace	null	null	null	001	[010]
006	RelSpaceContents	null	null	null	010	[003,004,005,006,007]
007	RelMaterialSet	null	null	null	003	[011]
008	RelMaterialSet	null	null	null	004	[012,013]
009	RelMaterialSet	null	null	null	005	[014,015]
010	RelMaterialSet	null	null	null	006	[016,017]
011	RelMaterialSet	null	null	null	007	[018,019]

Nous avons ajouté à la liste des éléments relations les éléments RelSpace, RelSpaceContents et RelMaterialSet. De même, nous avons fait la correspondance entre les éléments sémantiques pères et les éléments sémantiques fils.

Instance Attribut Simple

Id	Type	Id_Instance	Id_schema	Document XML
001	Concept	001	Batiment	<Building xmlns ... la structure d'un bâtiment" >
002	Concept	001	Thermique	<Building xmlns ... Name="BâtimentEssai ">

L'intégration du dernier document est plus difficile, car les éléments non pas de concept en commun. L'intégration est réalisée à partir des attributs et il faut définir les relations de transformation entre les éléments sémantiques manuellement.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/x3d-3.0.dtd"
  "file:///www.web3d.org/TaskGroups/x3d/translation/x3d-3.0.dtd">
<X3D profile="Immersive"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.0.xsd">
  <head>
    <meta content="Xbâtimen.t3d*" name="filename"/>
  </head>
  <Scene>
    <Transform translation="-2.5 -2.50 0">
      <Shape DEF="Nord">
        <Appearance>
          <Material diffuseColor="0.5 0.5 0"/>
        </Appearance>
        <Box size="0.1 5 2"/>
      </Shape>
    </Transform>
    <Transform translation="0.1 0 0">
      <Shape DEF="Sud">
        <Appearance>
          <Material diffuseColor="0.5 0.5 0"/>
        </Appearance>
        <Box size="5 0.1 2"/>
      </Shape>
    </Transform>
    <Transform translation="2.5 -2.5 0">
      <Shape DEF="Est">
        <Appearance>
          <Material diffuseColor="0.5 0.5 0"/>
        </Appearance>
        <Box size="0.1 5 2"/>
      </Shape>
    </Transform>
    <Transform translation="0.1 0 0">
      <Shape DEF="Ouest">
        <Appearance>
          <Material diffuseColor="0.5 0.5 0"/>
        </Appearance>
        <Box size="5 0.1 2"/>
      </Shape>
    </Transform>
    <Transform translation="0 -2.5 -1">
      <Shape DEF="Sol">
        <Appearance>
          <Material diffuseColor="0.5 0.0 0.5"/>
        </Appearance>
        <Box size="5 5 0.1"/>
      </Shape>
    </Transform>
  </Scene>
</X3D>

```

</Transform>
</Scene>
</X3D>

Exemple 4.12. Document XML de définition thermique

Instances Element_semantique

Id	Type	Nom	Liste_attribut	Liste_attribut_simple
001	Building	BâtimentEssai	null	[001,002]
002	BuildingStorey	RdC	null	null
003	Slab	Sol	[006]	null
004	Wall	Nord	[002]	null
005	Wall	Sud	[003]	null
006	Wall	Est	[004]	null
007	Wall	Ouest	[005]	null
008	Door	Entrée	null	null
009	Window	FenêtreEst	null	null
010	Space	Hall	null	null
011	MaterialLayer	béton	null	null
012	MaterialLayer	béton	null	null
013	MaterialLayer	pierre	null	null
014	MaterialLayer	béton	null	null
015	MaterialLayer	pierre	null	null
016	MaterialLayer	béton	null	null
017	MaterialLayer	pierre	null	null
018	MaterialLayer	béton	null	null
019	MaterialLayer	pierre	null	null
020	X3D	null	null	[003]
021	head	null	[001]	null
022	Scene	null	null	null

Dans la liste des éléments sémantiques, nous avons ajouté les éléments X3D, head et Scene, ainsi que leurs attributs simples dans la table Attribut_Simple. Ensuite, après avoir ajouté les éléments attributs Shape dans la table appropriée, les quatre éléments sémantiques Wall et l'élément sémantique Slab reçoivent une référence d'un élément d'attribut correspondant à la définition géométrique que l'on souhaite lui donner. De cette manière, nous intégrons une instance de concept mur à une définition géométrique d'une instance Shape. Maintenant, il ne reste qu'à réaliser le lien entre les éléments sémantiques possédant une géométrie à l'aide d'éléments de relation Placement.

Instances Element_relation

Id	Type	Nom	Liste_attribut	Liste_attribut_simple	Elt_Sem_Pere	Elt_Sem_Fils
001	RelContents	null	null	null	001	[002]
002	RelContents	null	null	null	002	[003,004,005,006,007]
003	RelContents	null	null	null	004	[008]
004	RelContents	null	null	null	006	[009]
005	RelSpace	null	null	null	001	[010]
006	RelSpaceContents	null	null	null	010	[003,004,005,006,007]
007	RelMaterialSet	null	null	null	003	[011]
008	RelMaterialSet	null	null	null	004	[012,013]
009	RelMaterialSet	null	null	null	005	[014,015]

010	RelMaterialSet	null	null	null	006	[016,017]
011	RelX3D	null	null	null	020	[021,022]
012	RelPlacement	null	011	null	002	[003]
013	RelPlacement	null	007	null	002	[004]
014	RelPlacement	null	008	null	002	[005]
015	RelPlacement	null	009	null	002	[006]
016	RelPlacement	null	010	null	002	[007]
017	RelPlacement	null	null	null	022	[002]

Nous avons ajouté à la liste des éléments de relation la relation RelX3D. Celle-ci lit l'élément sémantique X3D (racine de la scène) aux éléments sémantiques head et Scene. Le problème apparaît lors de l'intégration de l'élément Scene. En effet, comment définir les éléments fils de l'élément Scene, puisqu'il n'existe pas d'élément sémantique pour les lier. La solution consiste à garder cette référence nulle et continuer l'intégration. Une fois tous les éléments de relation RelPlacement définis, il suffit de regarder les éléments sémantiques référencés comme père et non référencés comme fils. Nous avons le cas, après insertion des éléments de relation RelPlacement [020, 021, 022, 023, 024], ces éléments possèdent le père 002, mais celui-ci n'est pas référencé par un élément de relation RelPlacement. Par conséquent, nous faisons le lien entre l'élément sémantique Scene 022 et l'élément sémantique BuildingStorey 002 à l'aide d'un élément relation RelPlacement.

Instance Element_attribut

Id	Type	nom	Id_ElementAttribut	Document XML
001	meta	filename	null	<meta content="Xbâtiment.x3d*" name="filename"/>
002	Shape	null	null	<Shape DEF="Nord"> <Appearance> <Material diffuseColor="0.5 0.5 0"/> </Appearance> <Box size="0.1 5 2"/> </Shape>
003	Shape	null	null	<Shape DEF="Sud"> <Appearance> <Material diffuseColor="0.5 0.5 0"/> </Appearance> <Box size="0.1 5 2"/> </Shape>
004	Shape	null	null	<Shape DEF="Est"> <Appearance> <Material diffuseColor="0.5 0.5 0"/> </Appearance> <Box size="0.1 5 2"/> </Shape>
005	Shape	null	null	<Shape DEF="Ouest"> <Appearance> <Material diffuseColor="0.5 0.5 0"/> </Appearance> <Box size="5 0.1 2"/> </Shape>
006	Shape	null	null	<Shape DEF="Sol"> <Appearance> <Material diffuseColor="0.5 0.0 0.5"/> </Appearance> <Box size="5 5 0.1"/> </Shape>
007	Transform	null	null	<Transform translation="-2.5 -2.50 0"> </Transform>

008	Transform	null	null	<Transform translation="0.1 0 0"> </Transform>
009	Transform	null	null	<Transform translation="2.5 -2.5 0"> </Transform>
010	Transform	null	null	<Transform translation="0.1 0 0"> </Transform>
011	Transform	null	null	<Transform translation="0 -2.5 -1"> </Transform>

Dans la table Element_attribut, nous avons ajouté les définitions géométriques du document X3D comme éléments attributs pour les éléments sémantique. Nous avons ajouté de la même manière l'élément d'attribut meta définissant l'attribut filename de l'élément sémantiques head.

Instance Attribut_Simple

Id	Type	Id_Instance	Id_schema	Document_XML
001	Concept	001	Batiment	<Building xmlns ... la structure d'un bâtiment" >
002	Concept	001	Thermique	<Building xmlns ... Name="BâtimentEssai ">
003	Concept	020	X3D	<X3D profile="Im ... /specifications/x3d-3.0.xsd">

Nous constatons que l'intégration d'un document X3D n'est utilisée que s'il existe au préalable des éléments sémantiques dans le système d'information. Toutefois, il est envisageable de stocker un document X3D dans le système, puis ensuite de stocker des éléments sémantiques de manière à pouvoir associer par la suite les attributs géométriques aux éléments sémantiques.

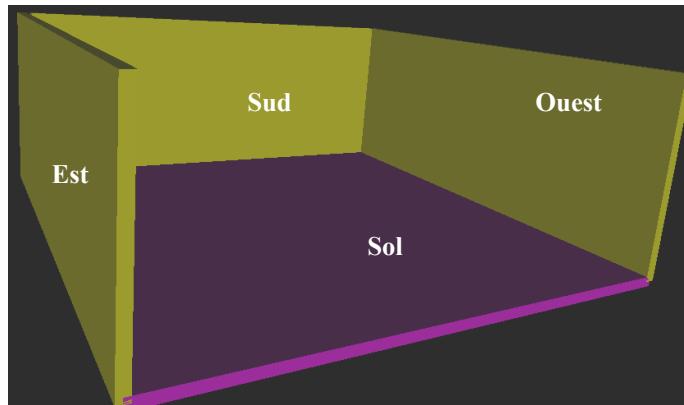


Figure 4.8. Fichier X3D de l'exemple 4.12

La dernière phase consiste à renseigner les extensions des instances de concept, de relation et d'attribut. Les tableaux suivants fournissent ces informations avec la valeur de l'attribut Extension mise à jour.

Instance Attribut

Nom	Marque	Attribut	Extension
Shape	[Shape]	null	[002, 003, 004, 005]
Transform	[Transform]	null	[007, 008, 009, 010, 011]
component	[component]	null	null
meta	[meta]	null	[001]

Instances Concept

Nom	Marque	Intension	Extension
X3D	[X3D]	null	[020]
head	[head]	[Component, meta]	[021]
Scene3D	[Scene]	null	[022]
Building	[Building]	[Shape]	[001]
BuildingStorey	[BuildingStorey]	[Shape]	[002]
Slab	[Slab]	[Shape]	[003]
Wall	[Wall]	[Shape]	[004, 005, 006, 007]
Door	[Door]	[Shape]	[008]
Beam	[Beam]	[Shape]	null
Column	[Column]	[Shape]	null
Window	[Window]	[Shape]	[009]
Space	[Space]	[Shape]	[010]
MaterialLayer	[MaterialLayer]	[Shape]	[011, 012, 013, 014, 015, 016, 017, 018, 019]

Instances Relation

Nom	Marque	Intension	Extension	Concept pere	Concept fils
RelX3D	null	null	[011]	X3D	[head, Scene3D]
RelPlacement	null	Transform	[012, 013, 014, 015, 016, 017]	[*]	[*]
RelContents	null	null	[001, 002, 003, 004]	listConcept**	listConcept**
RelMaterialSet	MaterialSet	null	[007, 008, 009, 010]	[Wall, Slab]	[MaterialLayer]
RelSpace	null	null	[009]	[Building]	[Space]
RelSpaceContents	null	null	[010]	[Space]	[Wall, Slab]

**listConcept = [Building, BuildingStorey, Slab, Wall, Door, Beam, Column, Window]

5. Conclusion

Après une approche syntaxique des données XML, ce chapitre présente une ontologie basée sur l'approche sémantique des données XML. L'étude de la structure sémantique des grammaires XML et des ontologies a inspiré la construction de classes formant une ontologie d'intégration des données engendrée par des grammaires XML. Cette structure cognitive à de nombreux avantages, dont la principale, est d'être évolutive aussi bien au niveau des données que de la définition des structures de données. Le domaine des ontologies a permis d'atteindre l'objectif fixé qui est une structure d'intégration de données XML. Néanmoins, le traitement des grammaires XML pour l'extraction de connaissance sémantique, réalisé par l'utilisateur de manière manuelle grâce à ces connaissances de la sémantique de la grammaire, n'est pas sans défauts. Cette extraction pose un certain nombre de problèmes dont le premier réside dans les choix de la décomposition des schémas XML. En effet, comment choisir le niveau de granularité en limitant le problème de la redondance d'information. Ce cas est assez courant dans les schémas XML complexes, où des balises font référence à des balises possédant un identifiant. De cette manière, une balise peut avoir deux pères, et donc ces balises pères deviennent en concurrence [DEK03]. Dans ce cas, si ces liens ne sont pas factorisés alors les informations du lien sont dupliquées pour chaque élément sémantique référent. Ces cas de figure sont facilement détectables et il est alors aisné d'aider l'utilisateur à supprimer cette redondance d'information.

Cette section a défini une ontologie pour la réalisation d'un système d'information. L'étape suivante consiste à mettre en place des processus opérationnels pour définir les règles d'usage de la sémantique contenue dans le système d'information. La section suivante présente l'approche contextuel pour la gestion, le partage et la manipulation des données.

Chapitre 5

« Je ne cherche pas à connaître les réponses, je cherche à comprendre les questions. »
(Confucius : philosophe chinois)

Arbres Contextuels et Processus de Manipulations

Sommaire

1. Modèle de représentation des données	106
2. La notion de contexte	109
2.1. Définition du contexte.....	109
2.2. Valeur d'un élément sémantique dans un contexte	110
3. Les arbres contextuels	111
3.1. Définition d'un arbre contextuel	111
3.2. Exemple de définitions d'arbres contextuels	112
3.3. Extraction d'arbres contextuels.....	113
4. Bilan	118

Dans le chapitre précédent, nous avons vu deux niveaux de représentation des grammaires XML. Le premier niveau représente les règles d'écritures syntaxiques. Dans ce niveau, nous avons défini deux notions importantes qui sont la notion de facteur et la notion de marque schématique. Le deuxième niveau représente les règles d'écritures sémantiques permettant de décrire à l'aide des ontologies la sémantique des éléments d'un schéma XML. Nous avons montré que la combinaison des deux niveaux permettait d'intégrer des schémas XML. Nous avons vu que cette intégration pouvait être étendue aux documents XML. Néanmoins, nous avons montré que cette intégration ne se fait pas automatiquement. Il est nécessaire d'utiliser un identifiant général pour associer les différentes définitions schématique et sémantique d'un même objet physique, manipulé par des acteurs de différents domaines d'applications. Malheureusement, certains schémas XML ne possèdent pas dans leur définition d'élément permettant d'identifier leurs instances. Par exemple, les documents X3D ne possèdent pas d'attribut « Nom » permettant d'identifier les éléments graphiques. Dans ce cas, l'intégration des propriétés graphiques nécessite la spécification manuelle des relations pour la combinaison de schémas. Une fois cette intégration réalisée, nous obtenons un ensemble sémantiquement cohérent de données hétérogènes (texte, graphique, multimédia...), structuré dans un document XML bien formé.

Dans ce chapitre, nous présentons les mécanismes de manipulation de ces documents intégrés. Ces mécanismes sont basés sur la notion de contexte lié à des arbres XML. Un contexte est une vue utilisateur du système en fonction d'un domaine d'application spécifique. Cette vue est un arbre XML que nous appellerons arbre contextuel. Les arbres contextuels sont le résultat de l'extraction des données du document intégré dans un document XML indépendamment de leur structure originelle.

Dans ce système, un objet qui possède plusieurs descriptions dans des contextes d'utilisations différents intégrera l'ensemble de ces descriptions dans le document intégré. Ainsi, l'utilisateur de cet objet dans un contexte spécifique pourra obtenir une définition de l'objet plus riche que celle initialement décrite dans ce contexte.

Par exemple, soit un objet mur et un objet fenêtre décrit par deux documents XML. Le premier, fourni par l'architecte donne une description géométrique des deux objets. Le second, fourni par un thermicien, définit les propriétés thermiques des deux objets. L'intégration de ces documents permettra de construire une définition de l'objet mur et de l'objet fenêtre intégrant les données géométrique et thermique. Si le thermicien souhaite modifier les données thermiques des ces objets, il pourra obtenir une visualisation graphique de ces objets.

A l'inverse, l'utilisation des arbres contextuels permettra de réduire la définition d'un objet aux seules données pertinentes pour les domaines d'application correspondant au contexte. Dans notre exemple, l'architecte peut vouloir extraire du système intégré seulement la description géométrique de la fenêtre.

L'avantage de cette approche est de limiter la taille du flux d'informations en augmentant la pertinence des données transmises.

Ce chapitre est composé de cinq sections. La première section présente le modèle de représentation des données pour la définition des processus de manipulation. La deuxième section définit la notion de contexte et la notion d'arbre contextuel. La troisième section fournit des exemples de manipulations de données en reprenant l'exemple décrit dans le chapitre 4. La dernière section donne un bilan sur la méthode que nous employons pour manipuler les données.

1. Modèle de représentation des données

Nous avons vu dans les chapitres précédents la définition des éléments sémantiques, des éléments de relations, des éléments attributs et des attributs simples. Les classes de l'ontologie générique permettent de créer des instances de ces éléments, dont les définitions sont contenues dans les instances des classes concepts, relation et attributs. Pour une manipulation de ces données, nous allons définir dans cette section un ensemble des règles d'écriture des instances d'éléments sémantiques, de relations et d'attributs.

Elément Sémantique : Dans le chapitre précédent, nous avons vu la définition d'un élément sémantique sous forme de classe. La définition suivante est une définition pour la manipulation des données des éléments sémantiques. La définition d'un élément sémantique est :

$$ES = (ts, Att_n[], EA[])$$

où

- ts est le type de l'élément sémantique correspondant à un concept.
- $Att_n[]$ une liste de références d'attributs simples.
- $EA[]$ une liste de références d'éléments attributs.

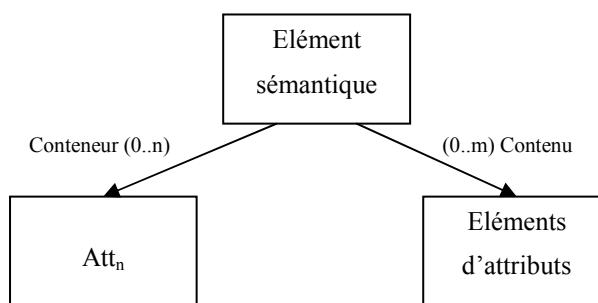


Figure 5.1. Relation entre éléments du modèle

Elément de relations : Les éléments de relations décrivent un lien entre plusieurs éléments sémantiques. Les cardinalités associées à ces relations sont de la forme (0..1) et (0..n). La cardinalité (0..1) est utilisée pour définir un lien avec l'élément père. La cardinalité (0..n) est utilisée pour définir les liens avec les éléments fils. La définition formelle d'un élément relation est :

$$ER = (tr, ES_{\text{père}}, ES_n[], Att_m[], EA[])$$

où

- tr est le type de l'élément qui correspond à la relation.
- $ES_{\text{père}}$ est une référence vers un élément sémantique père.
- $ES_n[]$ une liste de référence d'éléments sémantiques fils.
- $Att_m[]$ une liste de références d'attributs simples.
- $EA[]$ une liste de références d'éléments attributs.

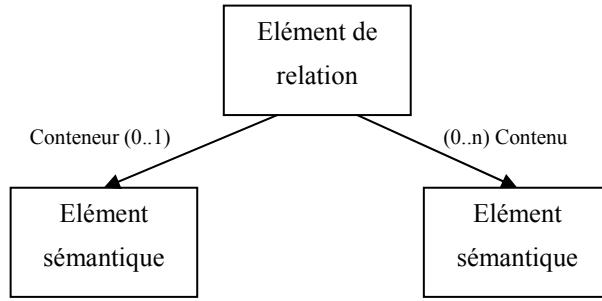


Fig. 5.2. Relation entre éléments sémantiques

Elément Attribut : Les éléments sémantiques et les éléments de relations possèdent des éléments attributs. Ces éléments attributs peuvent être composés d'autres éléments attributs. Les éléments attributs possèdent un ensemble d'attributs simples. Les liens entre éléments attributs forment une arborescence d'éléments attributs qui hiérarchisent l'information. La définition d'un élément attribut est :

$$\begin{aligned} EA &= (ta, v, Att_m[], EA_n[]) \\ EA_n[] &= (EA_1, EA_2, EA_3, \dots, EA_n) \end{aligned}$$

- ta est le type d'élément attribut EA.
- v est l'instance de l'élément attribut.
- Att_m[] une liste de références d'attributs simples.
- EA_n[] représente une liste de références d'éléments attributs.

A titre d'exemple, l'élément attribut *Représentation* d'un élément sémantique est constituée d'un élément attribut *Transform* composé de trois éléments attributs (*Origine*, *VecteurX* et *VecteurZ*), et d'un autre élément attribut *Shape*. Ce dernier peut être une boîte englobante, un modèle de surface, une représentation par les bords, etc.

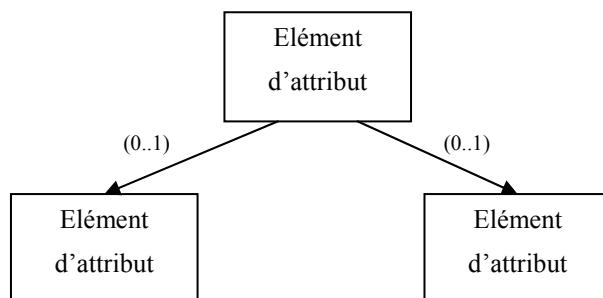


Figure 5.3. Relation entre éléments attributs

En reprenant l'exemple du chapitre précédent sur la table des instances d'attribut, nous pouvons voir qu'un attribut *Transform* possède trois éléments attributs qui sont *Origine*, *VecteurX* et *VecteurZ*.

Instance Attribut

Nom	Marque	Attribut	Extension
Shape	[Shape]	null	[002, 003, 004, 005]
Transform	[Transform]	null	[007, 008, 009, 010, 011]
component	[component]	null	null
meta	[meta]	null	[001]

2. La notion de contexte

Dans la section précédente, nous avons décrit notre modèle de représentation des données pour leur manipulation par les processus d'extraction. Nous allons voir à présent comment nous définissons un contexte pour le paramétrage des processus d'extraction de données. Cette notion de contexte est inspirée par la citation suivante de Recanati [REC93] : "La signification d'un mot comme le 'i' est une fonction qui nous porte d'un contexte d'expression à la valeur sémantique du mot dans ce contexte". Cette citation prendra tout son sens avec un petit exemple. L'exemple choisi est le mot coccinelle. Le mot ou la fonction coccinelle possède plusieurs contextes d'expression. Nous connaissons tous l'insecte, ainsi que la voiture inspirée du même insecte. La valeur sémantique, ou la signification du mot coccinelle dépend du contexte d'expression. Si le contexte choisi est voiture, alors le mot nous porte du contexte d'expression voiture à la valeur sémantique Volkswagen. Si le contexte choisi est insecte, alors le mot nous porte du contexte d'expression insecte à la valeur sémantique coléoptères.

De cette citation, nous définissons la fonction suivante :

$$Y = f(X)$$

- $f()$ est la signification du mot
- X est le contexte.
- Y la valeur sémantique portée par le mot dans le contexte X .

2.1. Définition du contexte

Par transposition aux éléments de l'ontologie générique que nous avons définis dans le chapitre précédent, le mot $f_{ts}()$ est l'élément sémantique de type 'ts'. Y est l'ensemble des valeurs de tous les attributs du mot pris en compte par le contexte. Le contexte joue le rôle de filtre sémantique, dans le sens où seul un certain nombre de valeurs d'attributs est conservé dans un élément sémantique. Dans un contexte thermique, seules les valeurs d'attributs des couches matériau sont conservés pour un mur.

Le contexte C est défini par :

$$C = (TER[], TA[], TEA[])$$

- $TER[] = (t_{er1}, t_{er3}, \dots, t_{erp})$ est un ensemble de types éléments de relations sélectionnés.
- $TA[] = (t_{a2}, t_{a7}, \dots, t_{aq})$ est un ensemble de types d'attributs simples sélectionnés.
- $TEA[] = (t_{ea3}, t_{ea6}, \dots, t_{ear})$ est un ensemble de types d'élément d'attributs sélectionnés.

La définition d'un contexte permet de sélectionner un ensemble d'éléments sémantiques par rapport à un ensemble de types de relation, par rapport à un ensemble de types d'éléments attributs et par rapport à un ensemble d'attributs simples. En ce qui concerne la sélection par rapport aux éléments de relations, seuls les éléments sémantiques référencés comme père ou fils par un élément de relation défini par le contexte sont sélectionnés.

2.2. Valeur d'un élément sémantique dans un contexte

Cette section présente comment, grâce à la définition d'un contexte, nous extrayons les valeurs sémantiques d'un élément sémantique. La fonction du contexte apparaît quand nous nous occupons de quantificateurs logiques. La gamme et l'interprétation des quantificateurs dépendent du contexte. Par exemple, le quantificateur « tout » généralement ne fait pas appel à tous les objets, mais seulement à ceux d'un certain type dans le domaine particulier, déterminé par le contexte.

Soit le contexte : $C = (TER[], TA[], TEA[]) = ((t_{er1}, t_{er3}, \dots, t_{erp}), (t_{a2}, t_{a7}, \dots, t_{aq}), (t_{ea3}, t_{ea6}, \dots, t_{ear}))$

Soit le mot fonction du contexte C: $Y = f_{ts}(C)$

$$Y = f_{ts}(C)$$

$$Y = f_{ts}((TER[], TA[], TEA[]))$$

$$Y = f_{ts}((t_{er1}, t_{er3}, \dots, t_{erp}), (t_{a2}, t_{a7}, \dots, t_{aq}), (t_{ea3}, t_{ea6}, \dots, t_{ear}))$$

alors

$$Y = (v_1, v_2, v_3, \dots, v_n)$$

Ou bien

$$Y = (\emptyset)$$

En langage naturel, Y est la liste des éléments d'attributs et des attributs simples de l'élément sémantique $f_{ts}()$ des attributs référencés par le contexte. La valeur sémantique du mot dans le contexte C peut éventuellement être vide, s'il ne possède pas d'attribut ou n'est pas référencé par

une relation spécifiée par le contexte. Pour l'instant la liste des types de relation n'est pas utilisée dans ce processus. Ces relations sont nécessaire pour créer des vues métier sur les données du système d'informations. A présent que nous savons extraire les propriétés d'un élément sémantique par rapport à un contexte, nous allons voir comment créer un arbre à partir de la définition d'un contexte. Cet arbre est créé par rapport aux éléments sémantiques sélectionnés et filtrés par le contexte. Sachant que la structure et les données de l'arbre dépendent du contexte, nous nommerons ces arbres : « arbres contextuels ».

3. Les arbres contextuels

Dans cette section nous allons voir les arbres contextuels. La définition d'un arbre contextuel est une requête sur le système d'information. Le résultat de cette requête est un arbre contextuel formant ainsi une vue métier sur les données. Cette requête est un filtre sémantique sur les informations, donnant ainsi une vue personnalisée des données. Grâce à un ensemble de contextes, l'intervenant d'un projet a accès à un ensemble de vues métiers sémantiquement différentes et dépendantes d'un contexte. Dans cette section, nous allons commencer par voir la définition générique d'un arbre contextuel. Ensuite nous allons voir un ensemble d'exemples d'arbres contextuels.

3.1. Définition d'un arbre contextuel

Pour définir un arbre contextuel, nous commençons tout d'abord par définir un contexte. Une fois ce contexte défini, l'intervenant doit définir un ensemble de types d'éléments sémantiques qu'il doit prendre en compte pour la création de l'arbre. Cette liste d'éléments sémantiques permet de réduire le nombre d'éléments sémantiques à prendre en compte, améliorant ainsi les performances du système.

Définition 1 : Soit $C = (TER[], TA[], TEA[])=((t_{er1}, t_{er3}, \dots, t_{erp}), (t_{a2}, t_{a7}, \dots, t_{aq}), (t_{ea3}, t_{ea6}, \dots, t_{ear}))$ un contexte et $ES[] = (ES_1, ES_2, \dots, ES_n)$ un ensemble d'éléments sémantiques. $Arb_{Contextuel} = (C, ES[])$ est la définition d'un arbre sémantique.

$$\begin{aligned}
 C &= ((t_{er1}, t_{er3}, \dots, t_{erp}), (t_{a2}, t_{a7}, \dots, t_{aq}), (t_{ea3}, t_{ea6}, \dots, t_{ear})) \\
 ES_1 &= (ts_1, Att_1[], EA_1[]) \\
 ES_2 &= (ts_2, Att_2[], EA_2[]) \\
 &\dots \\
 ES_n &= (ts_n, Att_n[], EA_n[])
 \end{aligned}$$

$$ES[] = (ES_1, ES_2, \dots, ES_n)$$

$$Arb_{Contextuel} = (C, ES[])$$

Règle 1 : Un élément sémantique est sélectionné si et seulement si cet élément est référencé par un élément de relations appartenant à la liste des éléments de relation du contexte.

Règle 2 : Soit $C = (TER[], TA[], TEA[])$ la définition d'un contexte. Si l'arbre contextuel résultant de sa définition est un arbre hiérarchique non cyclique, alors $C = (TER[], TA[], TEA[])$ est un contexte. C'est-à-dire que tout élément sémantique de l'arbre ne peut être ancêtre et descendant de tout autre élément sémantique.

3.2. Exemple de définitions d'arbres contextuels

Dans l'exemple suivant, nous voulons afficher les informations géométriques dans un navigateur 3D, alors pour générer le graphe de scène, l'arbre contextuel géométrique est nécessaire. Pour cela, d'après l'exemple du chapitre précédent, nous prenons en compte les éléments attributs *Shape*, l'attribut *name*, et pour les éléments de relation nous prenons en compte les éléments attributs *Placement*.

$$\begin{aligned}C_{Géo} &= (TR[\text{Placement}], TA[\text{name}], TEA[\text{Shape}]) \\ ES_{\text{tous}} &= (\bigcup ES_{\text{projet}}) \\ Arb_{Géo} &= (C_{Géo}, ES_{\text{tous}})\end{aligned}$$

$Arb_{Géo}$ est la définition de l'arbre contextuel géométrique et ES_{tous} est l'ensemble des éléments sémantiques du système d'information. Cet exemple a pour résultat lors de la création du graphe de scène la visualisation de tous les éléments géométriques de l'exemple du chapitre précédent. A présent, nous allons restreindre les éléments sémantiques et ne prendre en compte que les éléments de type *mur*, *tuyau* et *dalle*.

$$\begin{aligned}ES_{\text{mtd}} &= (ES_{\text{mur}}, ES_{\text{tuyau}}, ES_{\text{dalle}}) \\ Arb_{GéoMtd}(C_{Géo}, ES_{\text{mtd}}) &\end{aligned}$$

La définition de l'arbre contextuel $Arb_{GéoMtd}(C_{Géo}, ES_{\text{mtd}})$ a pour effet lors de la création du graphe de scène la visualisation de tous les éléments sémantiques de type *mur*, *tuyau* et *dalle*. Nous pouvons faire correspondre cet arbre contextuel à une vue métier plombier, car toutes les informations sur la tuyauterie sont extraites du système d'informations.

L'étape suivante, pour affiner l'interrogation du système d'information, consiste à sélectionner les types d'attributs et les types d'éléments attributs sur les éléments sémantiques. Ainsi les éléments sémantiques qui ne possèdent pas ces types, ne sont pas pris en compte par le processus

d'extraction et n'apparaissent donc pas dans l'arbre contextuel. À titre d'exemple, la recherche des éléments sémantiques de type *mur* doit se faire sur les murs possédant un enduit.

$$C_{GéoEnd} = (TR[\text{Placement}], TA[\text{name}], TEA[\text{Shape}, \text{enduit}])$$

$$ES_{murs}[] = (\bigcup ESmur)$$

$$\text{Arb}_{GéoEnd}(C_{GéoEnd}, ES_{murs}[])$$

Cet exemple montre comment extraire une information complexe de notre système. Ces informations sont des arbres et sont donc faciles à manipuler avec les outils existant pour la manipulation de données XML. Chaque arbre fournit une vue contextuelle des données du système d'information et ces vues peuvent être personnalisées en sélectionnant un certain nombre de types d'élément d'attribut. Nous avons vu dans cette section comment définir un arbre contextuel pour extraire des informations du système d'information. La section suivante présente comment extraire les informations du système d'information à partir de la définition d'un arbre sémantique.

3.3. Extraction d'arbres contextuels

Le processus d'extraction d'arbres prend en paramètre la définition d'un contexte. Ce contexte détermine le type de relation à prendre en compte ainsi que les types d'éléments sémantiques par rapport aux attributs simples et aux éléments d'attributs. Par contre, avant tout traitement il est nécessaire de déterminer la liste des types d'éléments sémantique que l'utilisateur veut utiliser. Dans le cas présent, nous désirons extraire les informations sur les murs et les éléments qui composent les murs comme les portes et les fenêtres. En effet, l'utilisateur cherche des renseignements sur les portes et les fenêtres installées dans le bâtiment. Pour la suite de l'exemple, nous reprendrons les données définies dans le chapitre précédent comprenant les données géométriques, structurelles et thermiques.

Instances Element_semantique

Id	Type	Nom	Liste_attribut	Liste_attribut_simple
001	Building	BâtimentEssai	null	[001,002]
002	BuildingStorey	RdC	null	null
003	Slab	Sol	[006]	null
004	Wall	Nord	[002]	null
005	Wall	Sud	[003]	null
006	Wall	Est	[004]	null
007	Wall	Ouest	[005]	null
008	Door	entré	null	null
009	Window	FenêtreEst	null	null
010	Space	Hall	null	null
011	MaterialLayer	béton	null	null

012	MaterialLayer	béton	null	null
013	MaterialLayer	pierre	null	null
014	MaterialLayer	béton	null	null
015	MaterialLayer	pierre	null	null
016	MaterialLayer	béton	null	null
017	MaterialLayer	pierre	null	null
018	MaterialLayer	béton	null	null
019	MaterialLayer	pierre	null	null
020	X3D	null	null	[003]
021	head	null	null	null
022	Scene	null	null	null

Dans le tableau précédent, nous sélectionnons donc l'ensemble des murs, des portes et des fenêtres. Ces éléments sémantiques portent les identifiants suivants : 004, 005, 006, 007, 008, 009. Ensuite, nous nous intéressons à la représentation graphique, car l'utilisateur veut une représentation graphique des données. Pour cela, seul le type d'élément de relation *Placement* est nécessaire. Dans la liste suivante nous sélectionnons les éléments de relation de type *Placement*, *RelX3D*, *RelScene* et les éléments de relation dont le père ou bien le fils possèdent un des identifiant suivant 004, 005, 006, 007, 008, 009, ceux-ci correspondant aux éléments sémantiques.

Instances Element_relation

Id	Type	Nom	Liste_attribut	Liste_attribut_simple	Elt_Sem_Pere	Elt_Sem_Fils
001	RelContents	null	null	null	001	[002]
002	RelContents	null	null	null	002	[003,004,005,006,007]
003	RelContents	null	null	null	004	[008]
004	RelContents	null	null	null	006	[009]
005	RelSpace	null	null	null	001	[010]
006	RelSpaceContents	null	null	null	010	[003,004,005,006,007]
007	RelMaterialSet	null	null	null	003	[011]
008	RelMaterialSet	null	null	null	004	[012,013]
009	RelMaterialSet	null	null	null	005	[014,015]
010	RelMaterialSet	null	null	null	006	[016,017]
011	RelX3D	null	null	null	020	[021,022]
012	RelPlacement	null	011	null	002	[003]
013	RelPlacement	null	007	null	002	[004]
014	RelPlacement	null	008	null	002	[005]
015	RelPlacement	null	009	null	002	[006]
016	RelPlacement	null	010	null	002	[007]
017	RelPlacement	null	null	null	022	[002]

La liste des éléments de relation retenue est 011, 012, 013, 014, 015, 016, 017. La liste des éléments sémantique retenue est 002, 003, 004, 005, 006, 007, 008, 009. Ainsi que l'élément 020 et 022, car ils font partie des éléments pères des éléments relations.

$$C_{G\acute{e}o} = (TR[\text{Placement}], TA[\text{name}], TEA[\text{Shape}])$$

$$ES_{\text{Wall}, \text{Door}, \text{Window}}[] = [004, 005, 006, 007]$$

$$Arb_{G\acute{e}o}(C_{G\acute{e}o}, ES_{\text{Wall}, \text{Door}, \text{Window}}[])$$

Pour la définition d'un arbre contextuel géométrique contenant les murs, les portes et les fenêtres, nous obtenons le fichier suivant (c.f. Exemple 1). Ce fichier ne possède pas les éléments sémantiques 008 et 009. En effet, les éléments de relations de possèdent pas de liens avec ces deux références. Dans ce cas, pour avoir accès aux informations des portes et des fenêtres, il faut passer par un autre contexte comme celui du contexte de contenant pour voir la structure du bâtiment. Grâce à ce contexte, l'utilisateur récupère une arborescence de tous les éléments composant le bâtiment ou seulement les éléments fenêtres et portes. Parfois, lors de la création de l'arbre contextuel, les éléments sémantiques intermédiaires dans la chaîne d'élément de relation, entre la racine et l'élément sélectionné, ne font pas partie de la définition de l'arbre contextuel. Dans ce cas, les éléments sémantiques sont extraits avec leur identifiant, mais tous les attributs simples et les éléments d'attributs sont automatiquement ignorés.

```

<?xml version="1.0" encoding="UTF-8"?>
<X3D id="020" name="">
  <RelX3D id="011">
    <Scene id="022" name="">
      < RelPlacement id="017">
        <BuildingStorey id="RdC">
          <RelPlacement id="013">
            <Transform id="007" translation="-2.5 -2.50 0"/>
            <Wall id="004" name="Nord">
              <Shape id="002" DEF="Nord">
                <Appearance>
                  <Material diffuseColor="0.5 0.5 0"/>
                </Appearance>
                <Box size="0.1 5 2"/>
              </Shape>
            </Wall>
          </ RelPlacement>
          < RelPlacement id="014">
            <Transform id="008" translation="0.1 0 0"/>
            <Wall id="005" name="Sud">
              <Shape id="003" DEF="Sud">
                <Appearance>
                  <Material diffuseColor="0.5 0.5 0"/>
                </Appearance>
                <Box size="5 0.1 2"/>
              </Shape>
            </Wall>
          </ RelPlacement>
          < RelPlacement id="015">
            <Transform id="009" translation="2.5 -2.5 0"/>
            <Wall id="006" name="Est">
              <Shape id="004" DEF="Est">
                <Appearance>
                  <Material diffuseColor="0.5 0.5 0"/>
                </Appearance>
                <Box size="5 0.1 2"/>
              </Shape>
            </Wall>
          </ RelPlacement>
          < RelPlacement id="016">
            <Transform id="010" translation="0.1 0 0"/>
            <Wall id="007" name="Ouest">
              <Shape id="005" DEF="Ouest">
                <Appearance>
                  <Material diffuseColor="0.5 0.5 0"/>
                </Appearance>
                <Box size="5 5 0.1"/>
              </Shape>
            </Wall>
          </ RelPlacement>
        </ RelPlacement>
      </ Scene>
    </RelX3D>
  </X3D>

```

```

</Shape>
</Wall>
</RelPlacement>
</BuildingStorey>
</RelPlacement>
</Scene>
</RelX3D>
</X3D>

```

Exemple 5.1. Arbre contextuel correspondant au contexte C_{Géo}

Nous pouvons constater que ce fichier n'est pas un fichier X3D valide. Par contre, celui-ci contient toutes les informations pour former un fichier X3D valide. Il existe deux solutions pour visualiser les informations graphiques. La première solution consiste à développer un interpréteur de contexte et de visualiser les informations graphiques dans un navigateur propriétaire. La deuxième solution consiste à réaliser une feuille de style XSL qui convertira cet arbre sémantique en fichier X3D valide. Dans l'application de ces travaux de recherche, nous avons développé un navigateur propriétaire. Ce développement sera présenté dans le chapitre 6.

L'exemple suivant montre la création d'un arbre sémantique dont les éléments font partie de deux schémas XML. Celui-ci permet de rechercher les informations thermiques par rapport aux relations de structure. C'est-à-dire que les propriétés thermiques des éléments sémantiques sont conservées, mais les éléments de relation pris en compte définissent un lien de contenance. Pour ce contexte, nous utiliserons les éléments sémantiques de type *Building*, *BuildingStorey*, *Wall* et *MaterialLayer*.

Instances Element_semantique

Id	Type	Nom	Liste_attribut	Liste_attribut_simple
001	Building	BâtimentEssai	null	[001,002]
002	BuildingStorey	RdC	null	null
003	Slab	Sol	[006]	null
004	Wall	Nord	[002]	null
005	Wall	Sud	[003]	null
006	Wall	Est	[004]	null
007	Wall	Ouest	[005]	null
008	Door	entré	null	null
009	Window	FenêtreEst	null	null
010	Space	Hall	null	null
011	MaterialLayer	béton	null	null
012	MaterialLayer	béton	null	null
013	MaterialLayer	pierre	null	null
014	MaterialLayer	béton	null	null
015	MaterialLayer	pierre	null	null
016	MaterialLayer	béton	null	null
017	MaterialLayer	pierre	null	null
018	MaterialLayer	béton	null	null
019	MaterialLayer	pierre	null	null
020	X3D	null	null	[003]
021	head	null	null	null
022	Scene	null	null	null

La liste des éléments sémantiques prise en compte par le processus est composée des éléments suivants : 001, 002, 004, 005, 006, 007, 011, 012, 013, 014, 015, 016, 017, 018, 019. En ce qui concerne les relations, nous prenons en compte les éléments de relations de type *RelContents* et *RelMaterialSet*. La liste des éléments de relations est : 001, 002, 007, 008, 009, 010.

$$C_{StrucTher} = (TR[RelContents, RelMaterialSet], TA[name], TEA[Shape])$$

$$ES_{Building, BuildingStorey, Wall} [] = [004, 005, 006, 007]$$

$$Arb_{StrucTher}(C_{StrucTher}, ES_{Building, BuildingStorey, Wall} [])$$

Instances Element_relation

Id	Type	Nom	Liste_attribut	Liste_attribut_simple	Elt_Sem_Pere	Elt_Sem_Fils
001	RelContents	null	null	null	001	[002]
002	RelContents	null	null	null	002	[003,004,005,006,007]
003	RelContents	null	null	null	004	[008]
004	RelContents	null	null	null	006	[009]
005	RelSpace	null	null	null	001	[010]
006	RelSpaceContents	null	null	null	010	[003,004,005,006,007]
007	RelMaterialSet	null	null	null	003	[011]
008	RelMaterialSet	null	null	null	004	[012,013]
009	RelMaterialSet	null	null	null	005	[014,015]
010	RelMaterialSet	null	null	null	006	[016,017]
011	RelX3D	null	null	null	020	[021,022]
012	RelPlacement	null	011	null	002	[003]
013	RelPlacement	null	007	null	002	[004]
014	RelPlacement	null	008	null	002	[005]
015	RelPlacement	null	009	null	002	[006]
016	RelPlacement	null	010	null	002	[007]
017	RelPlacement	null	null	null	022	[002]

```

<?xml version="1.0" encoding="UTF-8"?>
<Building id="001" name="BâtimentEssai">
    <RelContents id="001" name="">
        <BuildingStorey id="002" name="RdC">
            <RelContents id="002" name="">
                <Wall id="004" name="Nord">
                    <Shape id="002" DEF="Nord">
                        <Appearance>
                            <Material diffuseColor="0.5 0.5 0"/>
                        </Appearance>
                        <Box size="0.1 5 2"/>
                    </Shape>
                    <RelMaterialSet id="007" name="">
                        <MaterialLayer id="012" name="béton"/>
                        <MaterialLayer id="013" name="pierre"/>
                    </RelMaterialSet>
                </Wall>
                <Wall id="005" name="Sud">
                    <Shape id="003" DEF="Sud">
                        <Appearance>
                            <Material diffuseColor="0.5 0.5 0"/>
                        </Appearance>
                        <Box size="5 0.1 2"/>
                    </Shape>
                    <RelMaterialSet id="008" name="">
                        <MaterialLayer id="014" name="béton"/>
                        <MaterialLayer id="015" name="pierre"/>
                    </RelMaterialSet>
                </Wall>
            </RelContents>
        </BuildingStorey>
    </RelContents>
</Building>

```

```

</Wall>
<Wall id="006" name="Est">
    <Shape id="004" DEF="Est">
        <Appearance>
            <Material diffuseColor="0.5 0.5 0"/>
        </Appearance>
        <Box size="5 0.1 2"/>
    </Shape>
    <RelMaterialSet id="009" name="">
        <MaterialLayer id="016" name="béton"/>
        <MaterialLayer id="017" name="pierre"/>
    </RelMaterialSet>
</Wall>
<Wall id="007" name="Ouest">
    <Shape id="005" DEF="Ouest">
        <Appearance>
            <Material diffuseColor="0.5 0.5 0"/>
        </Appearance>
        <Box size="5 5 0.1"/>
    </Shape>
    <RelMaterialSet id="010" name="">
        <MaterialLayer id="018" name="béton"/>
        <MaterialLayer id="019" name="pierre"/>
    </RelMaterialSet>
</Wall>
</RelContents>
</BuildingStorey>
</RelContents>
</Building>

```

Exemple 5.2. Arbre contextuel correspondant au contexte C_{StrucTher}

Ce fichier correspond au résultat de la création d'un arbre contextuel avec les éléments de relations *RelContents* et *RelMaterial*. Dans le cas présent, nous savons que les éléments de relation ne référencent pas les mêmes éléments sémantiques. Dans le cas contraire, ce cas de figure est problématique, car les éléments sémantiques peuvent être référencés plusieurs fois. Cette référence multiple provoque des cycles dans la structure de données. Par conséquent, le document XML n'est plus un arbre, l'information est redondante et les processus de gestion des fichiers risquent d'entrer dans une boucle infinie. Lors de la création d'un arbre contextuel, le choix des types d'éléments de relations ne doit pas être fait au hasard et doit donc être cadre par rapport aux types d'éléments sémantiques que les éléments de relation peuvent référencer. Nous voyons ici la difficulté de créer des définitions d'arbres contextuels valides. En effet, une définition d'arbre contextuel n'est valide que si tous les documents générés par cette définition sont des arbres.

4. Bilan

Dans ce chapitre nous avons présenté une nouvelle approche de manipulation et d'extraction de données basée sur la notion de contexte. Nous avons montré que la valeur d'un élément sémantique dépend de son contexte d'utilisation. En définissant un contexte d'utilisation des éléments sémantiques, nous avons montré qu'il est possible de réduire le flux des informations en augmentant sa pertinence. La troisième partie de ce chapitre a présenté la notion d'arbre contextuel. Cette notion permet de définir à l'aide d'un contexte, un arbre contextuel qui sera

extrait du système cognitif. La dernière section présente deux exemples d’arbre contextuel extraits du système. Cette approche de la gestion des données à nombreux avantages. D’une part, elle permet de cibler l’information pertinente pour l’intervenant d’un projet et d’ainsi limiter la taille des données à déplacer sur le réseau. D’autre part, elle permet de coupler des informations qui ne l’étaient pas au départ. Par exemple, le système autorise la fusion des informations thermiques et la représentation graphique des données dans un seul document. Néanmoins, ce système possède des inconvénients. Tous les arbres contextuels extraits du système d’informations ne sont pas valides par rapport aux schémas XML intégrés précédents. Par conséquent, il n’est pas possible de valider les documents XML créés. De plus, si l’on désire créer une vue X3D d’un arbre contextuel, il est absolument nécessaire de créer une feuille de style XSL pour traduire l’arbre sémantique en arbre X3D ou de développer une interface graphique qui interprétera les arbres sémantiques de la définition d’un arbre contextuel.

Le travail fourni sur les arbres contextuels n’est qu’un point de départ sur le développement des processus de gestions du système cognitif. Dans le cas présent, une partie des éléments attributs sont pris en compte pour la création de l’arbre contextuel. La hiérarchie des éléments attributs est ignorée. De plus, les définitions d’arbres contextuels sont parfois fausses. Par conséquent, il est nécessaire de fournir un outil de validation d’arbres contextuels, basé sur les informations contenues dans le système cognitif. En effet, les informations sur les schémas XML intégrés renferment des informations précieuses qui permettent de cadrer la validation des définitions d’arbres contextuels.

La validation des définitions des arbres contextuels n’est que la première étape de la validation sémantique du système. L’étape suivante de la validation sémantique consiste à la réaliser sur les données. Les informations stockées dans le système cognitif sont des informations validées seulement de manière syntaxique et structurelle. En effet, les documents XML stockés sont bien formés et valides. Mais qu’en est-il de la sémantique ? Ce document est-il sémantiquement valide ? À titre d’exemple, les informations géométriques d’une scène contiennent deux parallélépipèdes qui se traversent. Dans le contexte géométrique, cela ne pose pas de problème. Dans le contexte structurel, le fichier contient une porte et une fenêtre. Si les deux documents sont intégrés dans le système, nous nous rendons compte que la fenêtre traverse la porte. Cette détection d’erreurs est réalisable par le système automatiquement, si nous lui fournissons les règles de validation sémantique des données. Cet exemple est simple, mais si le système possède mille règles de validation sémantique, celui-ci ira bien plus vite qu’un intervenant. Une voie future de recherche consiste à développer les règles et les processus de validation sémantique des données.

Chapitre 6

« Même avec neuf femmes, on ne peut pas faire un enfant en un mois ! »
(Proverbe asiatique pour OG)

Implémentation

Sommaire

1. Architecture générale.....	123
1.1. Le module 3D	124
1.2. Les Web Services	125
2. Les IFC	127
2.1. Description des fichiers IFC	128
2.1.1 Le modèle des IFC	128
2.1.2 Les instances IFC.....	129
2.2. IFC contre XML	130
2.3. La méthode ACTIVe3D	131
3. Démonstration de l'interopérabilité dans le monde du bâtiment grâce aux IFC.....	133
3.1. Phase 1 : Conception	133
3.1.1 Viz'all : Solution de relevés de bâtiments via pocket PC	133
3.1.2 ADT : Enrichissement de la maquette	134
3.1.3 ARCHICAD : Enrichissement de la maquette.....	134
3.1.4 ALLPLAN : Importation d'un fichier et finalisation du bâtiment.....	135
3.1.5 Bilan.....	135
3.2. Phase 2 : Etudes techniques	135
3.2.1 Renseignement des éléments sémantiques.....	136
3.2.2 Calculs de structure.....	138
3.2.3 Calculs thermiques.....	138
3.2.4 Calculs de méttré.....	139
3.3. Phase 3 : Gestion technique de patrimoine.....	140
4. Bilan	144

Les chapitres précédents présentent notre méthode d'intégration des schémas XML en vue d'une intégration de données XML. Une fois ces schémas et ces données intégrés, le système est interrogable à l'aide de définitions d'arbres contextuels. Dans cette partie, nous présentons l'implémentation de nos travaux dans le domaine des projets d'ingénierie civil.

La construction du bâtiment réunit un ensemble d'acteurs de professions diverses. Ces acteurs manipulent les informations à travers des logiciels de DAO qui sont spécialisés par profession (structure en métal, structure en béton, ingénierie thermique, plomberie...). Ces logiciels possèdent leurs propres modélisations des concepts métiers. L'échange de données entre les logiciels de DAO n'est alors pas possible, en raison de l'hétérogénéité des champs de modélisation. Les bâtiments prennent donc plus de temps et coûtent plus cher lors de la conception, de la construction et du fonctionnement.

Aujourd'hui, le besoin fondamental de tous les acteurs du bâtiment concerne un outil simple qui permet une gestion coordonnée des actions menées dans un projet. Cet outil doit permettre la gestion des données générées au cours du projet au travers de la visualisation 3D d'une maquette numérique, mise à disposition de tout un chacun dans une plateforme collaborative sur Internet.

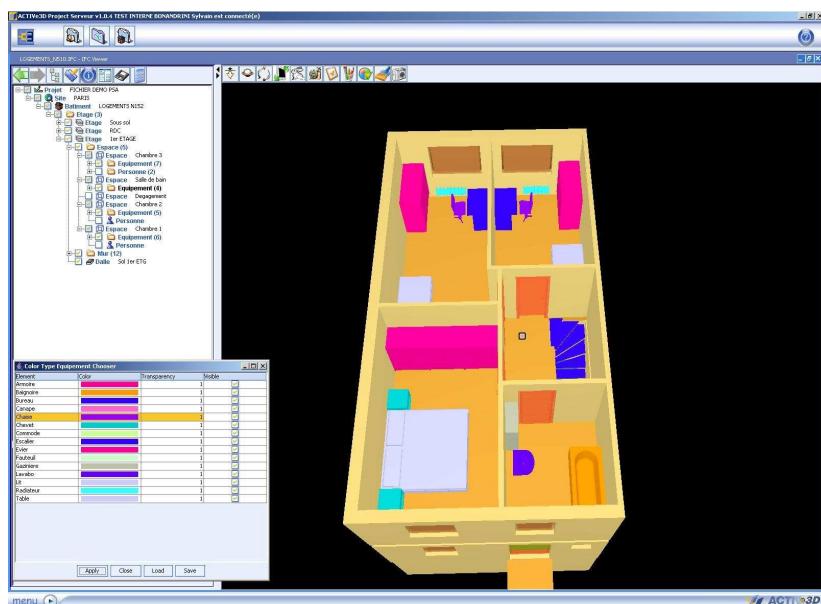


Figure 6.1. ACTIVe3D BUILD SERVEUR

Dans cette section, nous présentons la plateforme ACTIVe3D Build SERVEUR que nous avons développée à partir des travaux de recherche présentés dans ce document (Figure 6.1). Cette plateforme permet aux intervenants d'un projet, dispersés géographiquement – depuis l'architecte jusqu'au plombier – d'échanger des documents directement dans un environnement virtuel centralisé durant tout le cycle de vie d'un projet d'ingénierie civile. Une visualisation 3D permet aux intervenants de se déplacer autour du bâtiment en cours de conception et d'obtenir des informations sur les objets qui le composent.

Grâce aux processus de gestion et de manipulation basés sur les arbres contextuels, les intervenants d'un projet d'ingénierie civile ont accès aux informations intégrées à travers une vue métier. De plus, cette vue métier contient des informations réduites, mais pertinentes pour le

contexte d'utilisation. Cette vue métier, construite en fonction d'un contexte d'utilisation, possède une représentation graphique 3D qui se comporte comme un index sur le système d'information. En effet, la scène graphique est composée d'un ensemble d'éléments sémantiques possédant des informations complémentaires qui ne font pas forcement partie des contextes d'utilisation. Ces éléments sémantiques forment des ponts entre les informations de chaque vue métier et entre chaque métier de l'ingénierie civile.

Actuellement, 305 personnes utilisent de manière journalière l'application et gèrent 80 projets. Cette plateforme a été récompensé de la médaille d'or technologique de l'innovation à l'exposition internationale BATIMAT à Paris, novembre 2003.

Ce chapitre se compose de quatre sections. La première section présente le cadre des recherches en spécifiant la nature des projets d'ingénierie civile à travers les propriétés de la plateforme Web ACTIVe3D BUILD SERVEUR. La deuxième section présente les IFC qui sont le standard utilisé pour le partage et l'échange de données au cours du cycle de conception d'un bâtiment. La troisième section présente une démonstration de l'interopérabilité des IFC dans le monde de la construction et à travers la plateforme ACTIVe3D. La dernière section fournit un bilan général de l'implémentation.

1. Architecture générale

L'architecture générale de la plateforme est une architecture client-serveur. La particularité de cette architecture réside dans la modularité du client et du serveur. Chaque fonction de l'application est regroupée dans un module spécialisé. Chaque module communique à l'aide de messages. La méthode de communication entre les composants logiciels est un système de messagerie peer-to-peer. Un message client peut être envoyé à un module serveur et un message du serveur peut être reçu par un module du client. Grâce à cette architecture modulaire, le module qui envoie le message à un module et le module qui doit recevoir le message ne sont pas obligés d'être fonctionnels en même temps. En fait, le module qui envoie le message n'a pas besoin de connaître le destinataire, ce message ne fait appel qu'à un service. Cette notion de service est expliquée dans la section suivante.

Du côté serveur, le module BDD représente le module de base de données où toutes les données sont gérées et manipulées. La persistance des données est réalisée par ce module. Les autres modules sont les modules de traitement de l'information, dont chacun a ses spécialités. Le module IFC gère les informations IFC. Le module GED gère les informations de gestion de fichiers. Le module DQE gère les informations du cahier des charges réalisateur du bâtiment. Le module ADMIN gère les informations d'administration, c'est-à-dire les informations sur les intervenants, les projets, les droits, etc. Le module MAIL gère l'envoi automatique de courriers électroniques entre les intervenants des projets comme lors de la validation d'une phase du

projet, pour mettre au courant tous les acteurs. Le module de collaboration gère la synchronisation des évènements sur le projet.

Du côté client, le module 3D viewer manipule les informations 3D de la maquette numérique et gère les transferts d'informations graphiques avec le serveur. Le module IFC Viewer gère les données et la visualisation des données IFC d'un projet. Le module GED gère la visualisation des informations de gestion de fichiers. Dans la suite, nous allons présenter le module 3D et les services Web.

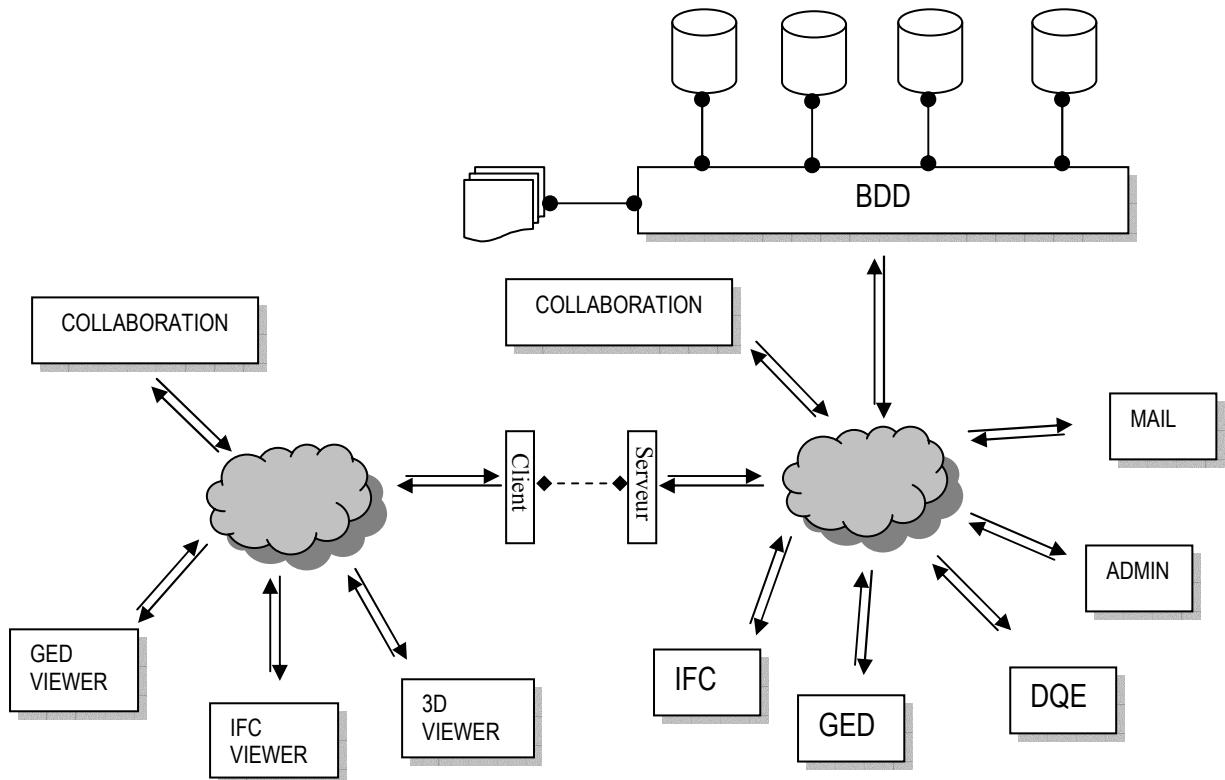


Figure 6.2. Architecture de la plateforme ACTIVe3D BUILD SERVEUR

1.1. Le module 3D

Les informations intégrées dans le système peuvent être restituées selon un arbre contextuel 3D. Cet arbre permet de construire une scène 3D représentant une vue de la maquette numérique en fonction des droits et des métiers de l'utilisateur. Les objets qui composent cette scène 3D sont sémantiquement reliés à d'autres éléments stockés dans les bases de données. Ainsi, il est possible de manipuler ces données au travers de la scène 3D. L'accès à la maquette 3D est possible par le parcours de l'arborescence des phases d'un projet (arbre de visualisation ou treeview). À chaque phase correspond une ou plusieurs maquettes 3D. L'administrateur de projets peut sélectionner le site qui l'intéresse par une navigation dans une scène 3D plus large. Les données intégrées dans le système peuvent être réorganisées pour être ensuite formatées dans des types de fichiers différents (fichier Word, fichier Excel, fichier PDF, fichier IFC). Lors de la

diffusion de plans au format IFC, un traitement automatique se déclenche afin d'analyser le plan et le stocker par le contenu dans la base de données. Les données ainsi stockées permettent la génération d'une maquette numérique 3D visible depuis la plateforme collaborative. Le module 3D permet aussi d'effectuer des rapprochements entre des plans d'une même phase de projet par fusion (selon des droits et le système de découpage : par bâtiment, par niveau...). Tout intervenant peut ainsi visualiser rapidement et simplement le projet dans son ensemble. Pour cela, l'utilisateur a à sa disposition un navigateur 3D (applet JAVA) lui permettant de se déplacer dans la scène. La navigation dans la scène peut aussi se faire par le parcours d'un treeview décrivant la composition de la maquette numérique et permettant à l'utilisateur d'aller directement à l'endroit souhaité (ce treeview correspond à l'implémentation d'un arbre contextuel). Par exemple, l'utilisateur peut accéder directement à la pièce X du bâtiment Y. Cette arborescence est modulable en fonction des préférences de l'intervenant, cependant une arborescence par défaut lui est proposée. On peut ainsi classer les objets 3D différemment, par exemple, par bâtiment puis par zone, puis par pièce ou par pièce puis par bâtiment. Le déplacement dans la scène interagit sur le treeview en déroulant l'arborescence en fonction de la position de l'utilisateur dans la scène. Ce treeview permet ainsi une interrogation de la scène 3D de manière simple et intuitive. Il aide l'utilisateur à effectuer des requêtes simples pour faire ressortir un élément de la scène ou pour extraire une information précise de la scène.

L'utilisateur peut interagir avec la scène et les objets qui la composent. Il peut ainsi, en cliquant simplement sur un objet, ajouter, supprimer, modifier des informations rattachées à l'objet, effectuer une action sur l'objet, extraire des informations pertinentes et isoler graphiquement des objets du reste de la scène. L'utilisateur a aussi la possibilité de sélectionner plusieurs éléments de la scène afin d'effectuer des requêtes ciblées sur ces éléments, d'effectuer une action groupée pour extraire ou modifier que les informations concernant ces objets.

1.2. Les Web Services

Le système de communication de la plateforme doit faciliter l'échange de l'information sur le réseau. Tous les arbres contextuels générés à partir de la plateforme sont des arbres-XML, donc les Web Services fournissent un cadre idéal pour le transport de l'information, car le flux de données de la plateforme et les Web Services utilisent tous deux XML. Généralement, les Web Services peuvent être développés en utilisant une technologie comme Java, Corba ou .NET. Le paquetage Java XML de Sun Microsystems contient une collection d'interfaces de programmation Java pour XML. Ces API supportent les standards SAOP, XMLP, WSDL et UDDI. JAXP (Java API for Parsing) couvre les technologies SAX (Simple API for XML), DOM (Document Object Model), and XSLT. JAXB (Java API for XML Binding) est une API qui permet la compilation de données XML en classes Java capables de lire un document XML et de

fournir les objets Java. Inversement, cette API écrit le document XML à partir des classes Java. JAXM (Java API for XML-based Messaging) est un protocole basé sur SOAP pour l'envoi de messages.

Pour construire un ensemble de Web Services, tout d'abord il est nécessaire de définir quels services l'entreprise veut vendre. Généralement, un Web Service est une représentation électronique de la spécification d'une compétence métier. Un Web Service peut être vu comme la représentation abstraite et comportementale d'un système d'informations. Pour cela, nous avons développé dans le cadre du projet ACTIVe3D, une méthode de construction automatique du programme de base des Web Services [SIM02]. Cette méthode s'articule en trois étapes : définition des compétences métiers, définition de bas niveau des Web Services, définition de haut niveau des Web Services. L'étape de définition des compétences métiers définit l'ensemble des compétences métiers qu'une compagnie veut vendre. Cette étape est subdivisée en trois niveaux : le niveau stratégique, le niveau analyse et le niveau normalisation.

- Le niveau stratégique concerne les orientations politiques et organisationnelles, concernant seulement les preneurs de décisions. Au terme de la discussion avec les preneurs de décisions, un ensemble d'objectifs en terme de Web Service est défini avec la planification et l'équipe correspondante de développement. Pour modéliser ces objectifs, nous utilisons un diagramme de « cas d'utilisation » UML.
- Le niveau d'analyse concerne tous les aspects statiques et dynamiques du système d'information. Ce niveau est une réorganisation conceptuelle et analytique suivant les objectifs des Web Services.
- Le niveau de normalisation concerne la définition de normes, de standard et du calendrier de développements des Web Services.

L'étape de définition de bas niveau des Web Services est une couche intermédiaire dans la construction des Web Services. Elle correspond à la traduction de la vue conceptuelle du système d'informations définie dans la première étape en un ensemble de classes, d'attributs, de liens, et de méthodes. Généralement, la vue du schéma, qui correspond à la compétence métier, peut être directement transformée en classes de Java. Les classes contiennent des attributs définissant des tables et des méthodes pour créer des objets par extraction d'informations de la base de données ou pour le stockage des objets comme instance de table. Ainsi, à la fin de cette étape, toutes les vues sont transformées en classes Java. Ces classes forment la base du programme pour le développement des Web Services indépendamment du système existant.

Dans la définition de haut niveau des Web Services, l'équipe de développement construit les Web Services. Pour cela, ils établissent un « diagramme de classe » UML. Ce diagramme contient les classes, qui sont dérivées de ceux définis dans l'étape précédente. Ainsi, le code pour l'accès aux bases de données est réutilisé. Le diagramme de classes résultant est directement

transformé en classes Java et organisé dans un applet Java. De plus, le schéma XML W3C du diagramme de classe est produit, ainsi quand les Web Services exportent leurs résultats sous forme de documents XML, ils pourront être validés par le schéma. Nous avons également des classes de niveau élevé, qui permettent des requêtes complexes sur les bases de données comme des opérations de jointure sur les tables, ou des opérations de choix multiples sur des tuples selon des valeurs d'attributs.

2. Les IFC

Nous avons vu dans la section précédente la présentation de la plateforme ACTIVe3D BUILD, ainsi que son architecture. À présent, nous allons voir, quelles sont les données sémantiques et géométriques manipulées par la plateforme ACTIVe3D BUILD SERVEUR.

Les logiciels de DAO utilisés dans les projets de génie civil modélisent chaque élément du bâtiment par un ensemble de vecteurs. Dans ce formalisme, aucune information sémantique sur les objets du bâtiment n'est modélisée. Pour résoudre cette problématique, l'alliance internationale pour l'interopérabilité a proposé une norme appelée IFC qui décrit la représentation des objets rencontrés dans les projets de construction. Le format IFC est un modèle qui associe la sémantique métier à la géométrie 2D/3D des éléments constituant le bâtiment. L'addition de la sémantique métier permet de limiter les redondances d'informations, car elle identifie instantanément chaque élément composant le bâtiment, pour qualifier plus rapidement le bâtiment. Les classes de base des IFC incluent la description des objets et fournissent une structure permettant l'interopérabilité des données entre les applications métiers. Par exemple, une porte IFC n'est pas simplement une collection de lignes et de primitives géométriques identifiés comme porte, mais cette porte est reconnue comme porte en tant que telle par la machine et possède des attributs correspondants à sa nature. L'adoption de ce format par tous les leaders des logiciels DAO permet une meilleure interopérabilité dans les échanges d'informations entre les diverses professions du génie civil. Chaque profession intervient dans le projet dans un contexte particulier, enrichissant le plan avec son propre vocabulaire, ses propres concepts et ses propres objets métiers. À la fin du projet, le fichier IFC correspondant au bâtiment contient l'ensemble des éléments du bâtiment, à l'aide d'une définition multicontexte. Les fichiers IFC sont des fichiers textuels dont la taille peut atteindre 100 mégaoctets. Plusieurs fichiers IFC peuvent coexister sur le même projet de génie civil. Par conséquent, l'objectif industriel fixé est d'appliquer notre méthode d'indexation sémantique aux IFC pour les intégrer à notre système avec d'autres données provenant des différents corps de métiers participants aux projets d'ingénierie civile.

2.1. Description des fichiers IFC

Dans cette section nous allons présenter les IFC sur un exemple concret et montrer comment notre méthode peut s'appliquer à ce type de fichier, bien qu'il ne soit pas écrit en XML. Un exemple de fichier IFC est donné dans le script 6.1. Ce fichier décrit un bâtiment contenant plus de 111000 objets métiers (un par ligne). Pour comprendre la complexité des IFC, la section suivante présente le modèle de représentation des IFC, ainsi que le modèle de représentation des données.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (('ACTIVE3D generated IFC file.),'2;1');
FILE_NAME ('Karlstr.IFC', '2002-06-19T15:48:48', ('Architect'), ('Building Designer Office'), 'PreProc',
'Windows System', 'The authorising person.');
FILE_SCHEMA ('IFC2X_FINAL');
ENDSEC;
DATA;
#1 = IFCORGANIZATION ('GS', 'Graphisoft', 'Graphisoft', $, $);
#3 = IFCPERSON ($, 'Undefined', $, $, $, $, $);
#4 = IFCORGANIZATION ($, 'OrganizationName', $, $, $);
#5 = IFCPERSONANDORGANIZATION (#3, #4, $);
#7 = IFCSIUNIT (*, .LENGTHUNIT., $, .METRE.);
.....
#111029 = IFCRELCONTAINEDINSPATIALSTRUCTURE ('25wKeWex98fQp5Pukf_Ilc', #6, 'BuildingStoryContainer',
'BuildingStoryContainer for Building Elements', (#111007), #110989);
#111030 = IFCRELAGGREGATES ('216Bv$yJj3tQjFeDohe6fQ', #6, 'BuildingContainer', 'BuildingContainer for
BuildigStories', #30, (#34, #16236, #29699, #56800, #62077, #67336, #72633, #91702, #110989));
#111031 = IFCRELAGGREGATES ('17XMUtNDr8FeFMtR6rOcy5', #6, 'SiteContainer', 'SiteContainer For Buildings',
#28, (#30));
#111032 = IFCRELAGGREGATES ('0pMN8yq8vDRfwN_tnJREKC', #6, 'ProjectContainer', 'ProjectContainer for Sites',
#26, (#28));
ENDSEC;
END-ISO-10303-21;
```

Script 6.1. Exemple de fichier IFC

2.1.1 Le modèle des IFC

Les fichiers IFC sont composés d'objets et de liens entre ces objets. Les attributs de ces objets décrivent la sémantique métier de ces objets. Les liens entre les objets sont représentés par des éléments de relations. Le modèle des IFC est un modèle objet modélisé à l'aide du langage EXPRESS¹ et possède approximativement 600 classes. Ces classes sont de trois types : les classes objets, les classes relations, les classes ressources.

1. Les classes objets sont constituées d'un triplet (GID, OS UF). Le GID définit l'identifiant global de l'objet IFC produit aléatoirement, dont les chances de redondances sont infimes. OS définit le propriétaire de l'objet (ownership features) et UF définit l'unité fonctionnelle. Les unités fonctionnelles définissent le contexte d'utilisation de ces classes. Dans le script 6.1, l'élément IfcPersonAndOrganisation #5 référence les éléments #3 et #4.

¹ Express Language: <http://www.iso.org/iso/fr/CatalogueDetailPage.CatalogueDetail?CSNUMBER=18348>

2. Les classes ressources constituent un ensemble d'attributs utilisé pour la description des unités fonctionnelles. Ces ressources sont organisées en graphes hiérarchiques (c.f. Figure 6.4).

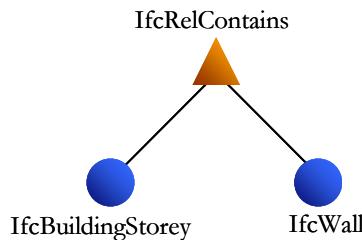


Figure 6.3. Un étage et un mur lié par un élément relation IfcRelContains

Les classes de relations représentent diverses relations (relation de capacité, relation d'agrégat, etc.) entre les classes objets (c.f. Figure 6.3). Ces classes relations possèdent également des unités fonctionnelles. Les éléments de liens sont préfixés par « IfcRel ». L'élément IfcRelAggregates d'après le script 6.1 possède l'identifiant #111030 et constitue un lien d'agrégat entre l'élément #30 et la liste suivante d'éléments (#34, #16236, #29699, #56800, #62077, #67336, #72633, #91702, #110989). L'élément #110989 est aussi référencé par l'élément qui est un lien nommé IfcRelContainedInSpatialStructure. Cela signifie que si un élément peut être référencé par plusieurs éléments, alors deux éléments peuvent mutuellement se référencer par l'intermédiaire d'une ou plusieurs relations.

2.1.2 Les instances IFC

L'étude des IFC montre la complexité des liens entre les instances de classes de relations et les instances de classes d'objets. À ce niveau, il existe deux types de liens entre les objets. Nous les nommons liens directs et liens indirects. Les liens indirects sont définis par les instances de relations. Les liens directs sont définis par les instances de classes ressources. Dans la Figure 6.3, les liens indirects sont caractérisés par les triangles oranges. Ces liens indirects sont des éléments de relations. Les instances d'objets dans notre architecture sont des éléments sémantiques. Dans la Figure 6.3, les éléments sémantiques sont représentés graphiquement à l'aide des disques bleus. Les instances ressources sont les éléments attributs dans notre architecture. Dans la Figure 6.4, les instances de ressources sont représentées graphiquement à l'aide des losanges rouges. La Figure 6.3 montre les liens indirects entre les éléments sémantiques. La Figure 6.4 montre les relations directes entre éléments sémantiques, notifiés en rouge. Nous dénombrons deux types de liens directs. Le premier type de lien direct définit un lien entre une ressource et un élément sémantique. Ces ressources sont structurées sous la forme d'arbres et correspondent aux éléments attributs. Le second type de relations directes définit une relation entre éléments

sémantiques. Le modèle IFC définit seulement un seul type de liens directs entre deux éléments sémantiques. Ce lien est le lien de placement utilisé pour la définition des repères entre les éléments graphiques de la scène 2D/3D. Dans le modèle IFC, ce lien est nommé IfcLocalPlacement. L'ensemble des liens de placement forme un arbre qui est la structure arborescente de la scène graphique.

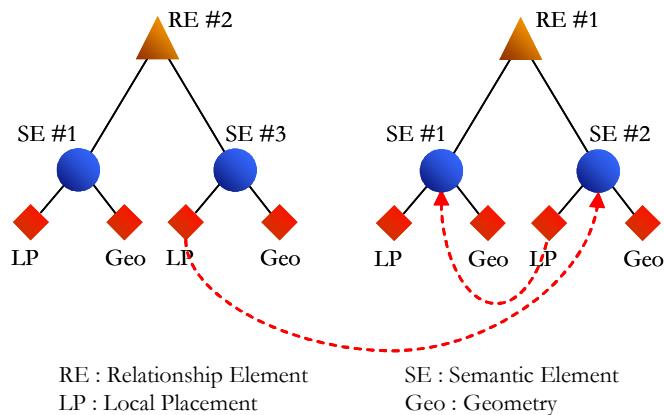


Figure 6.4. Exemple de liens directs entre plusieurs éléments sémantiques

2.2. IFC contre XML

Après étude des IFC, nous constatons qu'il est possible de marquer le schéma à l'aide de notre méthode de marquage schématique. Par conséquent, les structures définies dans les chapitres précédents sont utilisables pour les IFC. De plus, toutes les manipulations à l'aide des arbres contextuels sont applicables aux fichiers IFC de la même manière. Notre choix s'est porté sur les IFC, car nous n'avions pas à définir la grammaire des éléments d'un bâtiment, de plus son utilisation débouche sur une application industrielle réelle.

Nous devons souligner le fait que les fichiers IFC et les fichiers XML possèdent une différence majeure qui se situe au niveau de la syntaxe. En effet, les fichiers IFC ne possèdent pas le système de balisage XML. Par conséquent, la structure du fichier n'est pas hiérarchique au sens XML du terme. Par contre, elle définit une liste d'éléments et ces éléments comme décrits dans la partie précédente possèdent des attributs qui font le lien avec d'autres éléments. Ces liens forment un ensemble hiérarchique structurant les éléments. Par conséquent, bien que l'étude sur les langages de Dyck ne s'applique pas aux IFC, la méthode de marquage est entièrement fonctionnelle sur celle-ci. Par contre, l'étude et l'application de ce marquage ne sont pas décrites ici pour des raisons de confidentialité liées à la propriété industrielle du produit développé dans le cadre du contrat de collaboration réalisé pour le financement de cette thèse.

Nous avons orienté l'étude sur les documents XML pour plusieurs raisons. Tout d'abord, les processus d'entrée et de sortie, ainsi que les processus d'échange et les flux d'informations sont

basés sur XML. Par conséquent, à tous les stades de la gestion de l'information, XML est au cœur des processus. L'homogénéisation du langage des données offre la possibilité de développer des outils qui seront réutilisables pour d'autres langages possédant la même syntaxe. Par exemple, les interfaces de programmation d'application comme JAXP² (Java API for XML Processing) offre des outils pour parser tout type de documents XML. D'autre part, notre choix a été porté sur XML, car les domaines d'application sont bien plus vastes que les IFC. En effet, les IFC sont limitées au domaine du bâtiment. Tandis que les grammaires XML peuvent décrire un très vaste choix de domaine dont le nombre est croissant, car les langages développés aujourd'hui pour le Web sont basés sur XML.

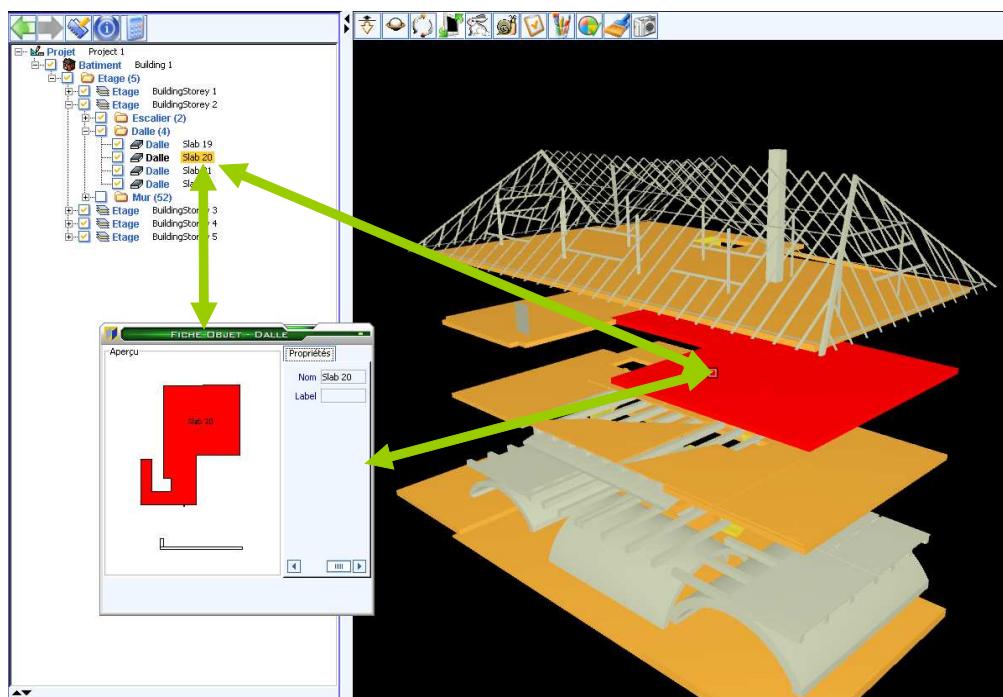


Figure 6.5. Capture d'écran du système de gestion de scène 3D

2.3. La méthode ACTIVe3D

Les mécanismes de gestion et de manipulation des fichiers IFC, comme la fusion de deux fichiers dans un seul, l'extraction partielle de données d'un seul fichier, la visualisation ou le stockage, doivent prendre en compte les multiples valeurs sémantiques des objets qui dépendent du contexte d'utilisation. Pour atteindre cet objectif, nous avons défini dans le chapitre précédent une structure hiérarchique de contexte appelé vue contextuelle. La solution consiste à réduire la complexité d'un graphe multicontexte cyclique en graphe acyclique monocontexte. La Figure 6.5 présente le système de gestion 3D qui construit une interface utilisateur spécifique constituée d'un arbre de contenance, d'une scène 3D et d'une fiche technique sur un élément sémantique de

² JAXP : <http://java.sun.com/xml/jaxp/index.jsp>

la scène. La navigation entre les éléments est réalisée à l'aide de liens hypermédia qui associent un ensemble d'éléments sémantiques à un objet métier. Dans le cas présent, l'objet métier est un élément sémantique Slab. L'arbre principal est l'arbre contextuel géométrique qui contient les relations topologiques des différents objets. La scène 3D résultante correspond à une vue métier particulière [BOO02].

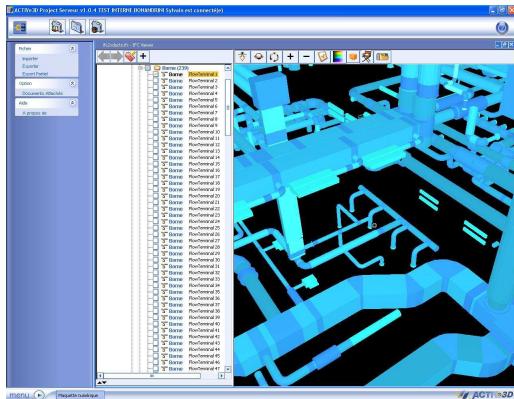


Figure 6.6. Une scène 3D correspondant au contexte plomberie

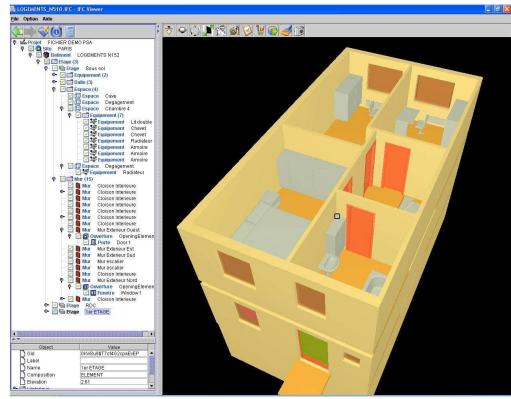


Figure 6.7. Une scène 3D correspondant au contexte architectural

La Figure 6.6 montre une représentation de la vue métier plomberie d'un fichier IFC. La Figure 6.7 montre une représentation de la vue métier architecturale. Depuis ces vues, l'intervenant a accès à toutes les unités fonctionnelles qui composent le fichier IFC. Ces vues métiers sont des informations textuelles, depuis lesquelles des documents spécifiques peuvent être générés ou associés (rapport technique, information de gestion, etc.). Dans la scène 3D, toutes les formes géométriques définies dans les arbres IFC sont converties dans le modèle de surface triangulaire [REM96]. Pendant la conversion, les objets 3D sont associés avec un GID. Ce GID est l'identifiant global utilisé pour identifier chaque objet métier d'un fichier IFC. Dans le script 6.1, le GID de l'élément IfcRelAggregates est #111032. Ce GID est utilisé pour lier la visualisation 3D avec des informations stockées dans la base de données. Toutes insertions de nouvelles données dans n'importe quelle base sont référencées par un GID correspondant à un objet IFC. Tous les arbres générés dans la plateforme sont des arbres-XML. Nous avons développé un schéma de base de données qui prend en compte la sémantique et l'aspect 3D des IFC. Les arbres et les éléments composants sont stockés dans une base de données relationnelle et sont manipulés à l'aide de SQL. Grâce à cette base de données et au GID, tous les types d'informations peuvent être attachés à un objet métier de la scène 3D.

3. Démonstration de l'interopérabilité dans le monde du bâtiment grâce aux IFC

Cette section montre l'utilisation de la plateforme, ainsi que des fichiers IFC dans le cadre d'un projet d'ingénierie civile. Dans la première partie de cette section, nous allons voir la phase de conception d'un bâtiment à travers l'intervention de plusieurs intervenants utilisant des logiciels de conception différents. La deuxième partie de cette section présente la phase de l'étude technique du bâtiment, pour la validation structurelle et thermique du bâtiment en cours de conception. La dernière partie présente la gestion technique du patrimoine. Cette phase consiste à enrichir la maquette numérique en ajoutant des fichiers techniques sur les éléments du bâtiment, ou en donnant une définition des éléments à partir d'un catalogue constructeur.

3.1. Phase 1 : Conception

La conception du bâtiment est réalisée en quatre phases. La première phase consiste à apporter à la plateforme un relevé de terrain d'un vieux bâtiment. La deuxième phase consiste à définir une extension à ce bâtiment. La troisième phase réalise l'extension et la dernière phase définit les étages du bâtiment relevés dans la première intervention.

3.1.1 Viz'all : Solution de relevés de bâtiments via pocket PC

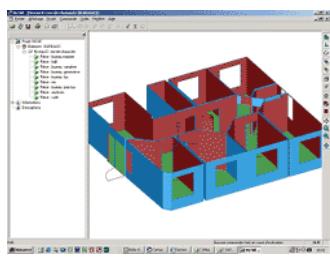


Viz'all® est une solution automatisée de relevé de bâtiment, associant l'utilisation d'un lasermètre, d'un pocket PC et d'un logiciel sur pocket PC.

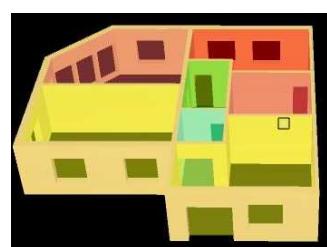
Le principe consiste à tracer à main levée le croquis de la pièce sur l'écran tactile du pocket PC. Après connexion et dépôt du relevé sur le terrain, la maquette du bâtiment est mise à jour et est disponible pour tous les autres intervenants du projet de rénovation du bâtiment. Les autres acteurs du projet peuvent à présent visualiser le bâtiment à partir de l'interface graphique 3D de la plateforme ACTIVe3D BUILD SERVEUR.



Bâtiment réel



Relevé sur Viz'all

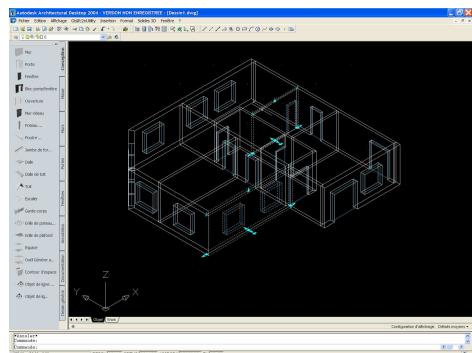


Plateforme ACTIVe3D

3.1.2 ADT : Enrichissement de la maquette



Après le dépôt du relevé de terrain, un autre acteur du projet se connecte à la plateforme pour récupérer ce relevé et enrichir la maquette. Pour cela, l'architecte définit les nouveaux espaces en utilisant ADT (Autodesk Architectural Desktop). Une fois que l'architecte a terminé ses mises à jour sur la maquette concernant une future extension du bâtiment, il ajoute ces nouvelles données à la plateforme.



Mises à jour sur ADT

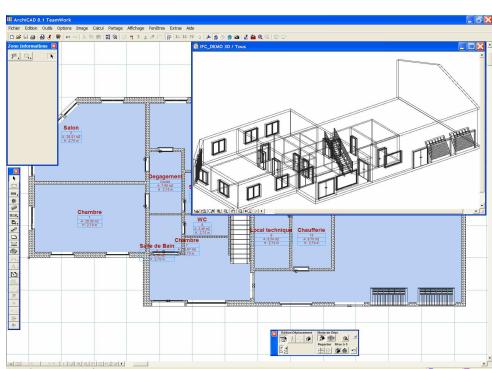


Plateforme ACTIVe3D

3.1.3 ARCHICAD : Enrichissement de la maquette



Après le dépôt par l'architecte des nouveaux espaces qu'il désire dans l'extension du bâtiment, les ingénieurs du génie civil se connectent à la plateforme pour recueillir les dernières informations. Ces ingénieurs travaillent sur ArchiCAD® de Graphisoft. Lorsqu'ils ont terminé leurs travaux de conception sur le bâtiment, ces nouvelles données sont ajoutées à la maquette numérique du bâtiment en cours de conception sur la plateforme ACTIVe3D BUILD SERVEUR.



Mises à jour sur ArchiCAD

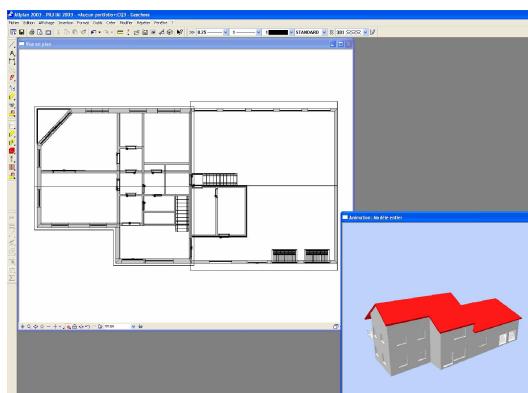


Plateforme ACTIVe3D

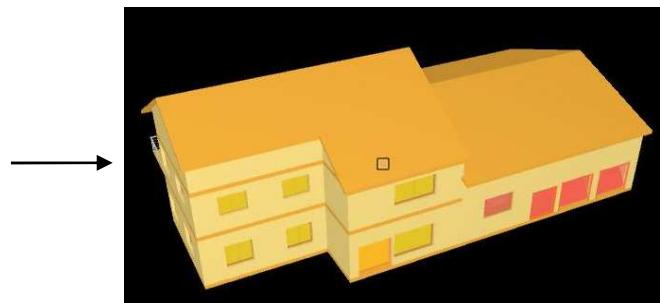
3.1.4 ALLPLAN : Importation d'un fichier et finalisation du bâtiment



La deuxième partie du projet sur ce bâtiment est la réhabilitation du bâtiment existant. Pour cela, une équipe d'ingénieurs gère la conception de cette partie. Tout comme les autres équipes, celle-ci se connecte à la plateforme pour extraire les informations concernant le bâtiment principal. Cette équipe travaille avec le logiciel AllPlan® de Nemetschek Systems Inc. Une fois que les mises à jour sont réalisées, les ingénieurs mettent ces nouvelles informations sur la plateforme.



Mises à jour sur AllPlan



Plateforme ACTIVe3D

3.1.5 Bilan

Grâce à la plateforme, tout un ensemble d'acteurs peut échanger les informations de conception du bâtiment entre différents logiciels CAO/DAO. La norme IFC 2.x est utilisée pour formater les données envoyées à chacun des acteurs. Tous les flux de données transitent à travers la plateforme ACTIVe3D BUILD SERVEUR, car elle offre la possibilité à chacun des acteurs d'avoir toutes les mises à jour en temps réel, une fois que celles-ci ont été placées sur le serveur. L'efficacité de ce processus d'échanges et de centralisation de l'information a pour résultat un très grand gain de temps, car les échanges de données sont quotidiens dans les projets de conception. De plus, l'attente de mises à jour de données peut bloquer le travail d'une autre équipe, donc l'accès à la maquette numérique à jour sur la plateforme permet de débloquer plus rapidement ces situations d'urgence. La plateforme ACTIVe3D BUILD SERVEUR est un outil de collaboration entre les acteurs du projet.

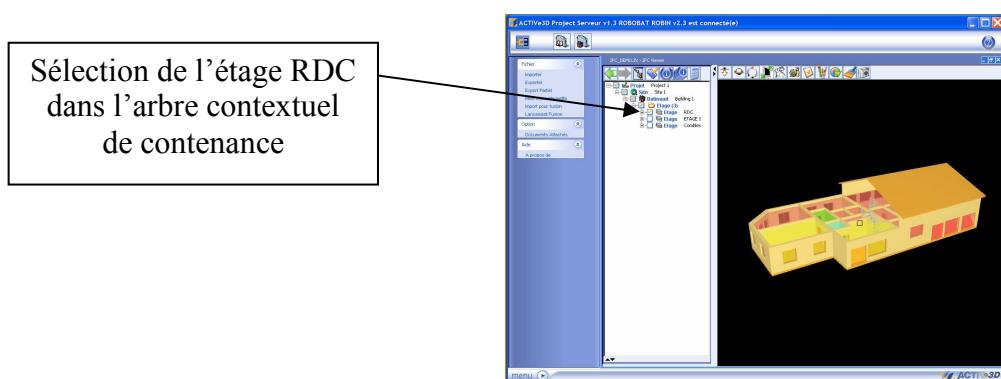
3.2. Phase 2 : Etudes techniques

Nous allons voir dans cette section qu'il est possible d'ajouter des informations aux éléments sémantiques de la maquette numérique du bâtiment. Ces informations sémantiques seront par la suite réutilisées dans les processus de création de nouvelles données sur le bâtiment à l'aide de calculs métiers.

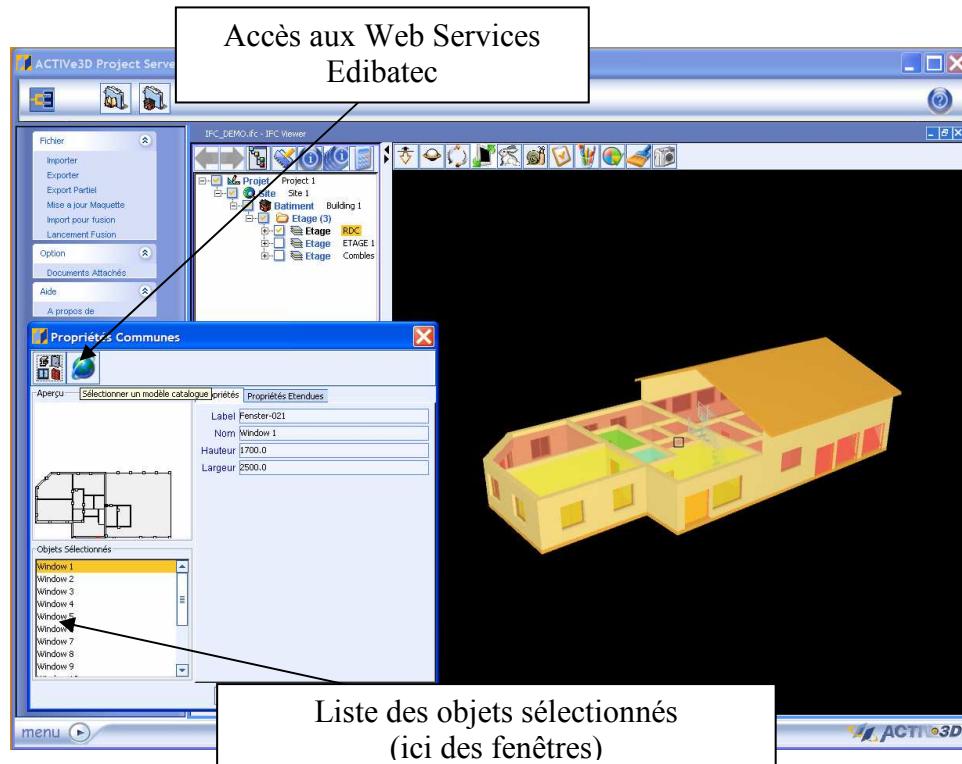
3.2.1 Renseignement des éléments sémantiques

Editbatec³ met au point, diffuse et maintient un format de base de données ouvert et public décrivant les produits et les équipements du bâtiment, utilisable par les professionnels du secteur. Aujourd’hui cette base de données est accessible à travers des Web Services. À présent, l’idée est de coupler la maquette numérique avec des informations techniques Edibatec. Nous allons voir comment coupler ces informations avec les informations graphiques stockées de la section précédente dans la plateforme ACTIVe3D.

- Sélection des objets à renseigner

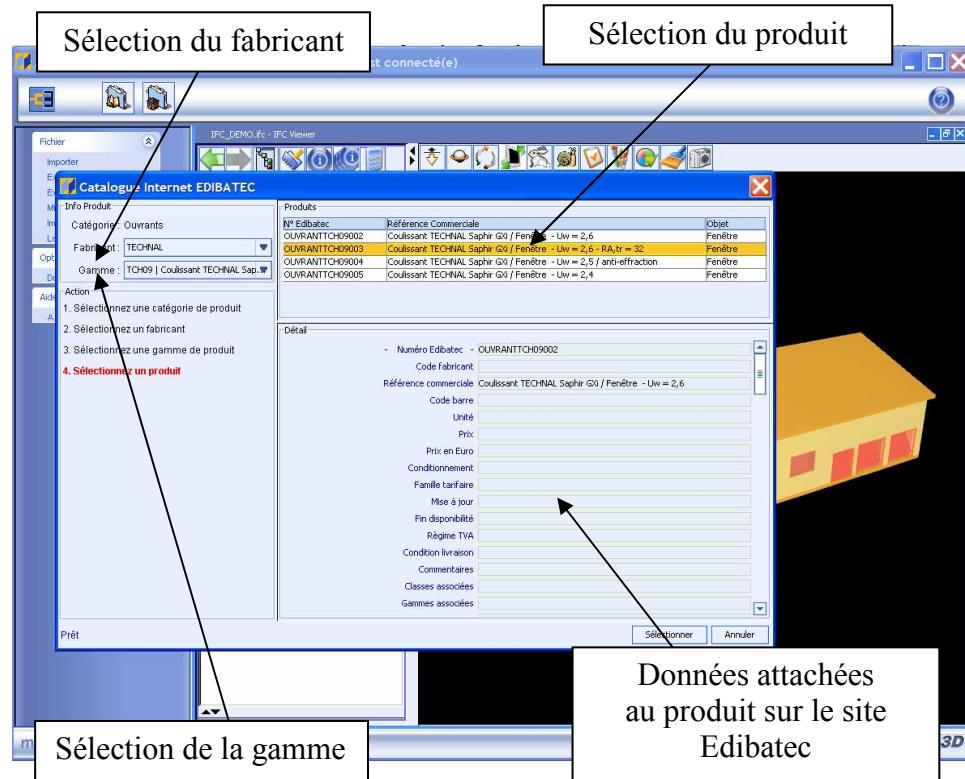


- Affectation multiple d’un équipement Edibatec à un ensemble d’éléments

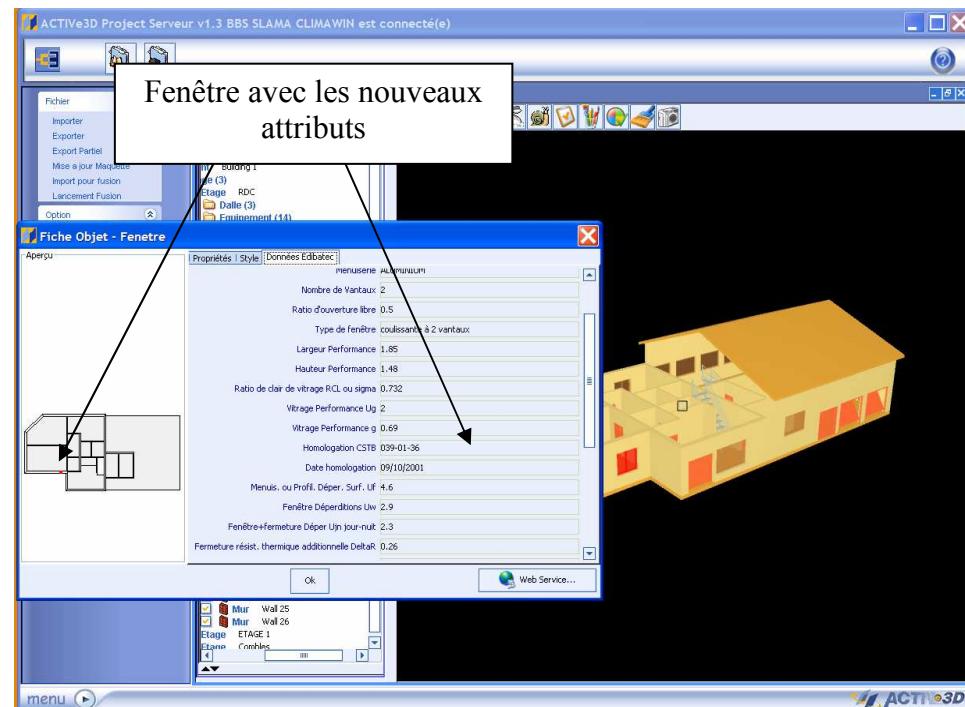


³Edibatec : <http://www.edibatec.org/>

- Sélection d'un produit sur le site Edibatec



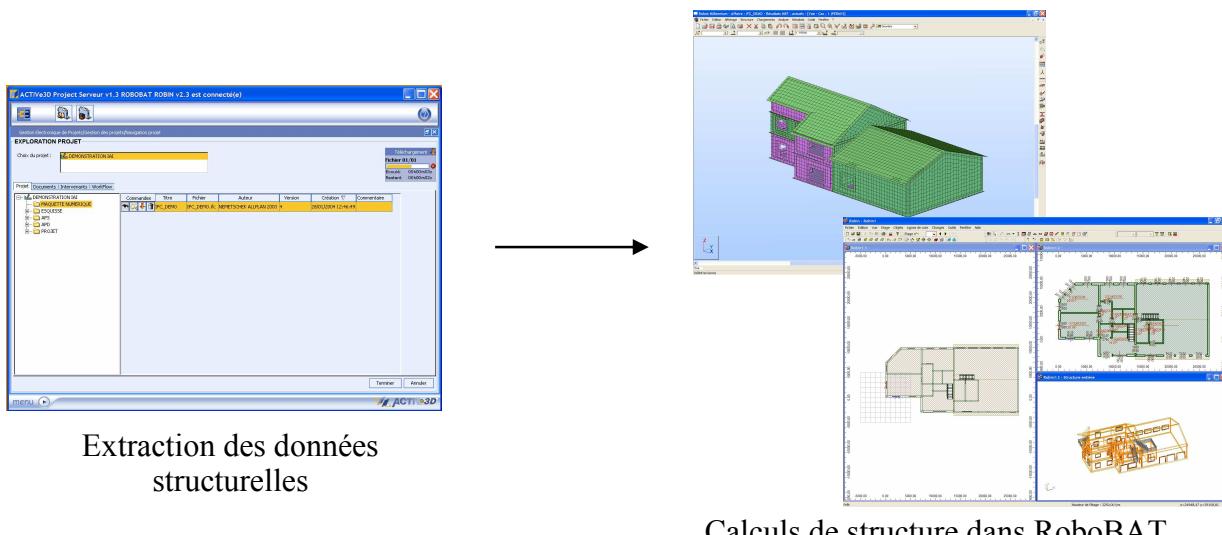
- Vérification des nouvelles données Edibatec



3.2.2 Calculs de structure



Au cours de la conception d'un bâtiment, les structures comme les charpentes doivent être validées. En effet, si les structures sont trop faibles et qu'elles ne vérifient pas les normes, alors les plans doivent être modifiés en conséquence. RoboBAT est un logiciel de calculs de structures. Ce logiciel permet d'optimiser et de valider les structures selon les normes nationales et européennes en Béton armé, bois, acier, aluminium, etc. Un des avantages de ce logiciel est de pouvoir importer des données IFC. Par conséquent, les bureaux d'études structurelles peuvent valider les informations de la maquette numérique se trouvant sur la plateforme ACTIVe3D BUILD SERVEUR.

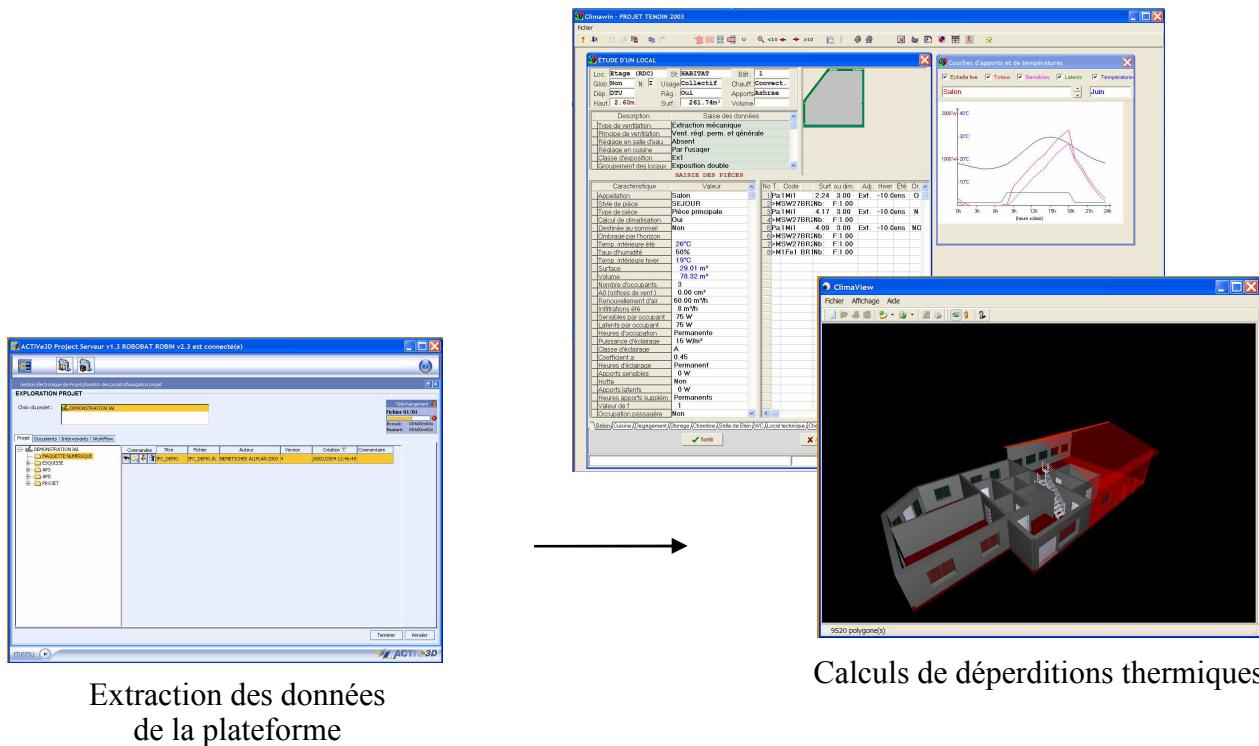


Pour réaliser cette étude, les ingénieurs doivent se connecter à la plateforme et sélectionner tous les éléments concernant la structure. Ces éléments sont les murs, les dalles, les poutres, les poteaux, etc. Pour cela, ils utilisent la définition de l'arbre contextuel « calculs de structure ».

3.2.3 Calculs thermiques



Les échanges thermiques entre les espaces d'un bâtiment possèdent des normes et ces échanges doivent être validés. Le module thermique du logiciel CLIMA-WIN® de la société « BBS Slama permet » de réaliser des calculs de déperditions thermiques Th-D 1991 ainsi que les coefficients réglementaires des bâtiments selon les règles ThBât/ThU de 2001. Ce logiciel importe et exporte des données IFC, ce qui permet aux ingénieurs de mettre à jour et de valider la maquette numérique sur la question des échanges thermiques.



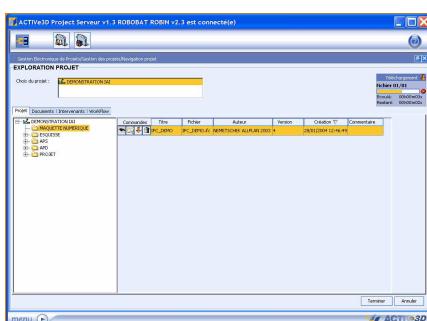
Extraction des données
de la plateforme

Calculs de déperditions thermiques

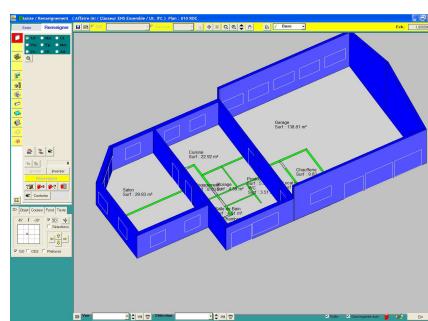
3.2.4 Calculs de méttré



Windesc® est un outil de méttré de la société ATTIC+ qui prend en import des IFC. Ce logiciel fournit des rapports et des devis à propos des surfaces de murs, de sols, etc. Cet outil est indispensable pour établir une réévaluation du coup de construction d'un bâtiment. Les rapports et les devis réalisés sur la maquette numérique sont ensuite ajoutés aux informations concernant le bâtiment et la phase de conception.



Extraction des données
de la plateforme



Calculs de surfaces des murs

3.3. Phase 3 : Gestion technique de patrimoine

La gestion technique de patrimoine consiste à gérer les informations concernant le patrimoine après sa construction et sa livraison au client. Cette gestion permet d'avoir aussi bien une connaissance des biens en terme de structures/bâtiments, que des connaissances sur tous les éléments composants le bâtiment avec toutes leurs informations annexes. La plateforme ACTIVe3D donne accès aux clients à leurs patrimoines immobiliers. De cette plateforme, ils sélectionnent le patrimoine qu'ils veulent gérer (cf. Figure 6.8).

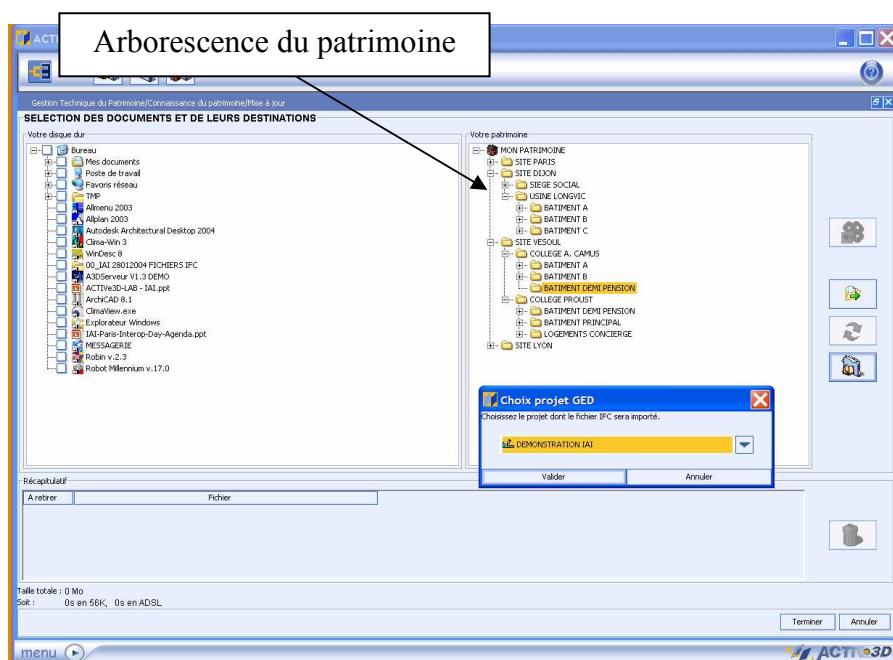


Figure 6.8. Sélection du patrimoine immobilier

Une fois que le patrimoine est sélectionné, le client a accès à un ensemble de fonctionnalités qui permettent d'extraire des données formatées et d'enrichir la maquette numérique. La plateforme offre par exemple une édition de rapports de surfaces (cf. Figure 6.9). Le navigateur 3D permet de mettre en évidence un certain nombre de types d'équipements choisis, comme le montre l'exemple de la Figure 6.10 où les chaises et les bureaux sont colorés de manière très visible. De plus, il est possible de mettre en transparence certains types d'éléments pour faciliter la recherche d'autres éléments (c.f. Figure 6.11).

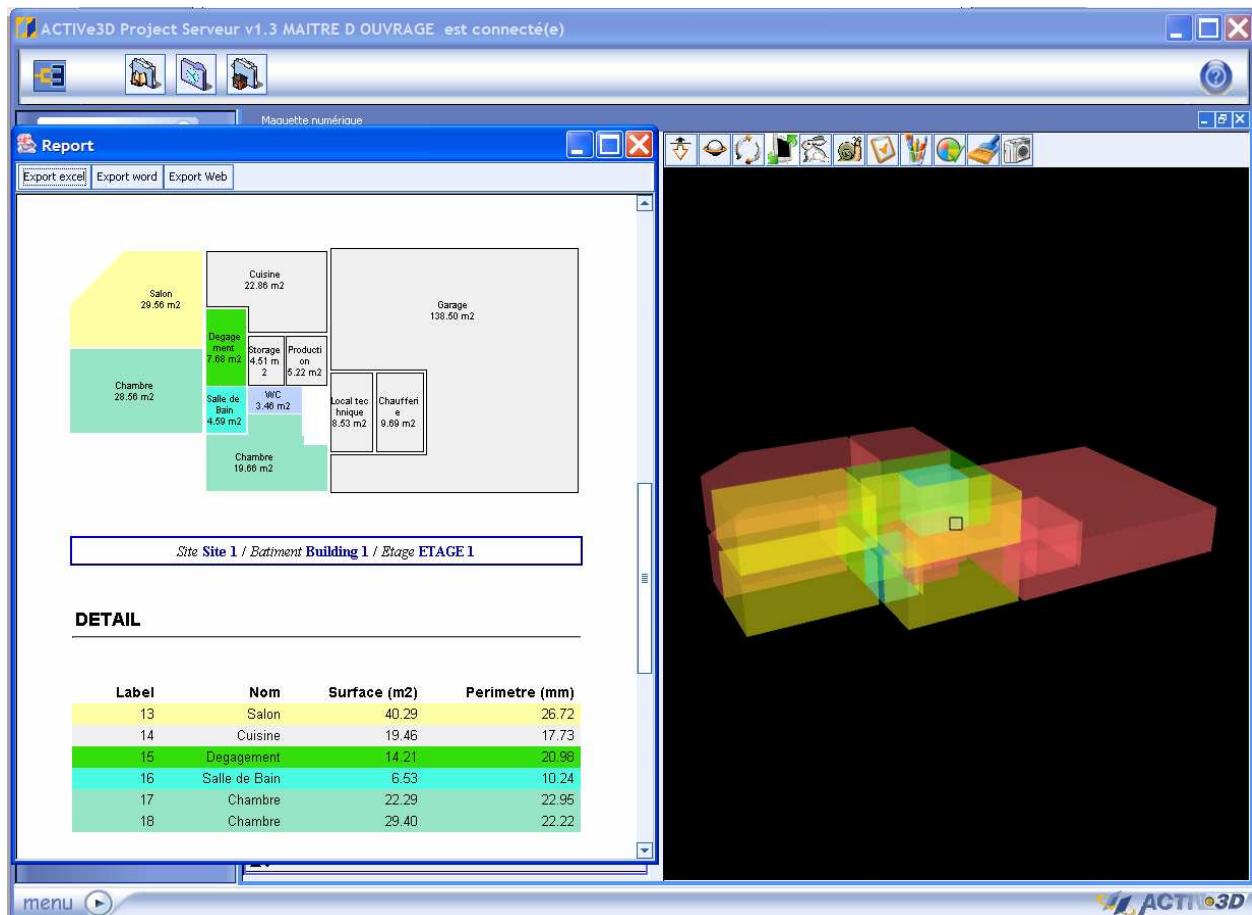


Figure 6.9. Edition de rapports de surfaces

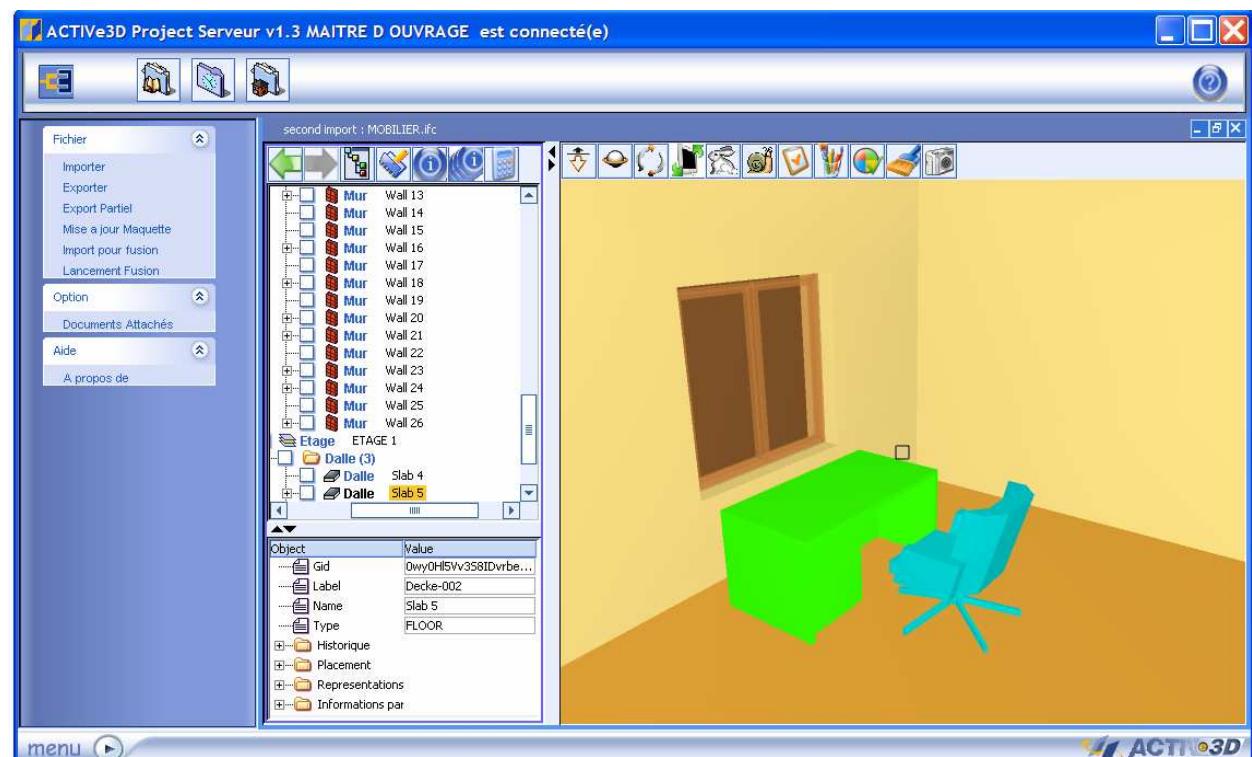


Figure 6.10. Mise en évidence de certains types d'équipements

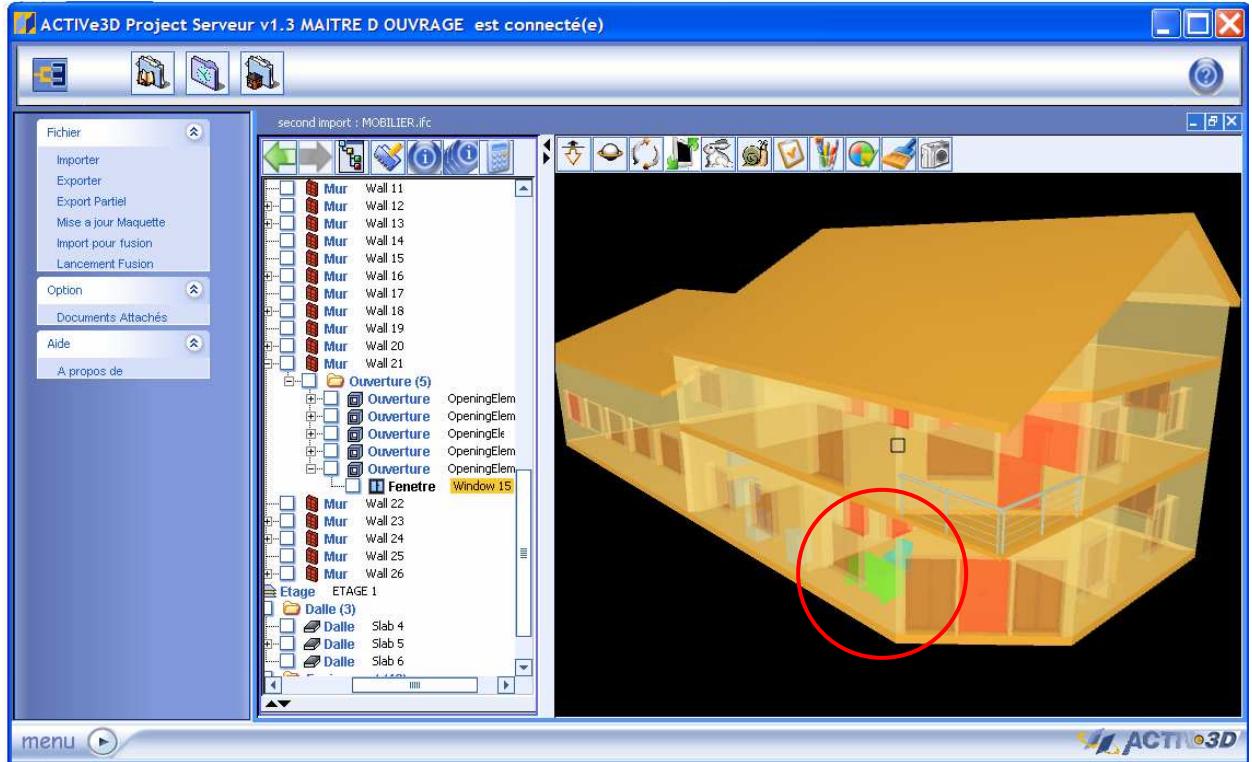


Figure 6.11. Mise en transparence de certains éléments pour faciliter la recherche d’éléments

La fonctionnalité suivante permet à chaque intervenant ou client de la plateforme d’associer un fichier directement sur un objet graphique de la scène (c.f. Figure 6.12). Ce fichier est sélectionné directement à partir du système d’exploitation, ou à partir du module de gestion électronique de documents de la plateforme. De cette manière, les documents sont référencés de plusieurs manières dans la plateforme, ainsi la maquette numérique 3D forme un deuxième index pour la recherche de fichiers électroniques.

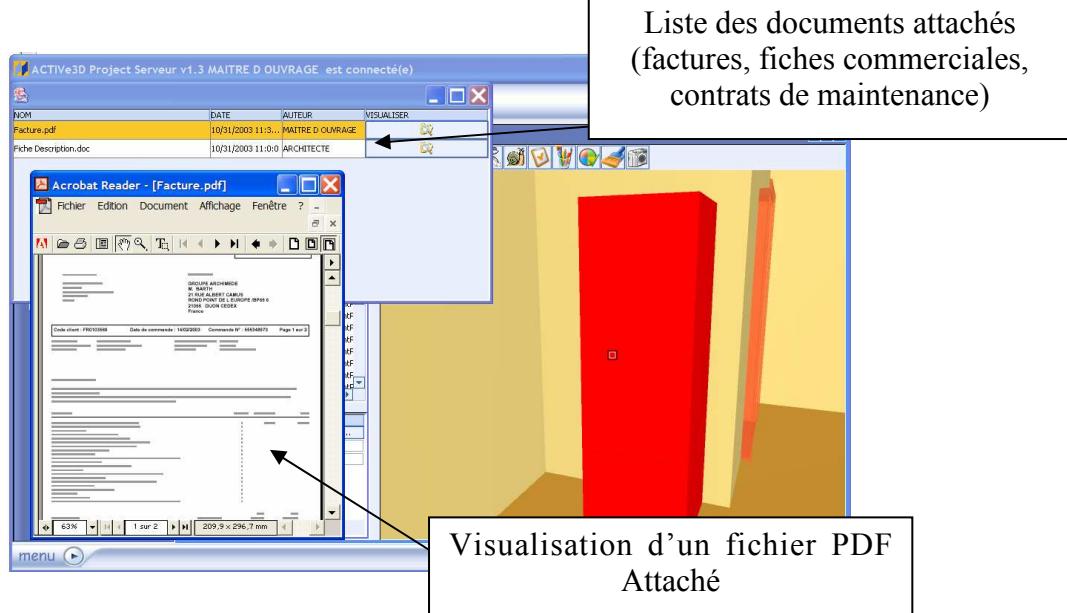


Figure 6.12. Indexation d’un fichier sur un élément graphique

La dernière fonction que nous allons exposer est la fonction de fusion de fichiers IFC dans la base de données contenant la maquette numérique. Cette fusion permet de mettre à jour la maquette numérique avec de nouveaux éléments. Si le fichier contient un catalogue fournisseur d'objets graphiques, alors l'intervenant peut mettre à jour directement sur la plateforme les éléments mobiliers. Cette fonction se passe en trois phases. La première phase consiste à importer sur la plateforme le fichier catalogue (c.f. Figure 6.13). La deuxième phase consiste à sélectionner les objets que l'intervenant désire insérer (c.f. Figure 6.14). La dernière phase consiste à sélectionner dans l'arbre contextuel de contenance l'objet qui servira de repère pour l'insertion graphique de l'objet (c.f. Figure 6.15).

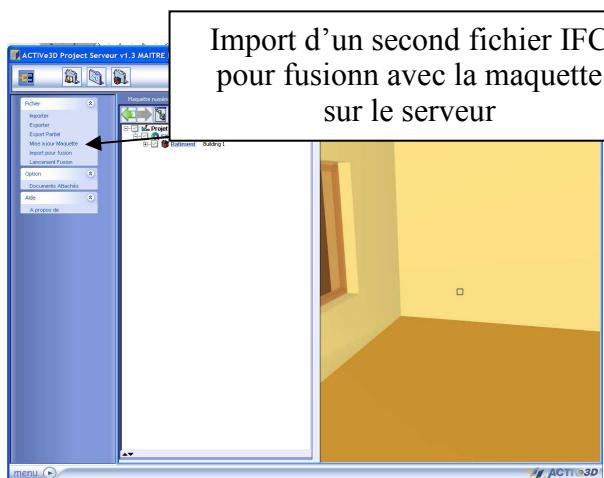


Figure 6.13. Phase n°1

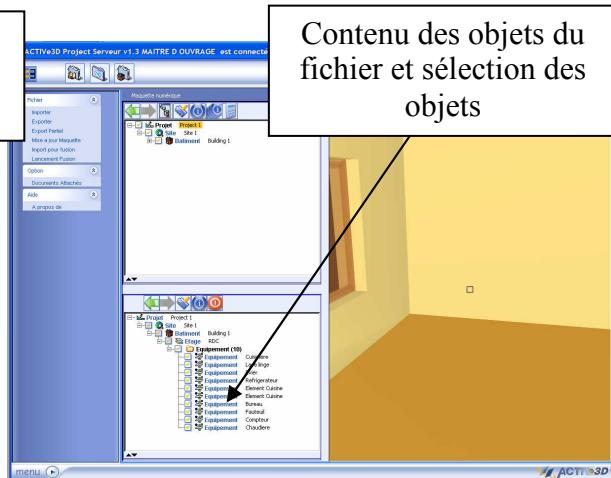


Figure 6.14. Phase n°2

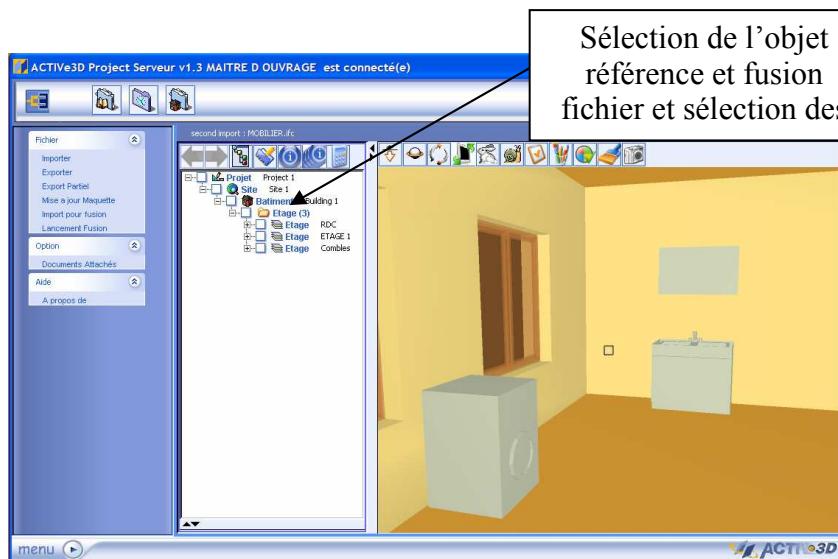


Figure 6.15. Phase n°3, Patrimoine enrichi d'équipements

4. Bilan

Ce chapitre a présenté les résultats obtenus sur le développement de la plateforme Web ACTIVe3D BUILD SERVEUR. Cette plateforme est au centre du travail collaboratif réalisé par les différentes ingénieries civiles. Basée sur une architecture client-serveur, notre plateforme utilise les Web Services pour l'interrogation d'informations et la réalisation de fonctions avancées. De plus, le flux de données XML facilite la manipulation des informations par les API JAVA. L'infrastructure de la plateforme est une infrastructure modulaire, où la communication est réalisée à l'aide de messages, rendant ainsi chaque module indépendant. Le standard utilisé pour échanger l'information est la norme IFC. Cette norme décrivant tous les éléments d'un bâtiment s'adapte bien à notre méthodologie d'intégration de schéma. Bien que ce schéma ne soit pas un schéma XML, nous avons défini des marques schématiques et extrait les concepts, les relations et les attributs nécessaires au bon fonctionnement du système.

Le serveur ACTIVe3D-Build fonctionne actuellement sur un serveur biprocesseur Dell (Pentium 4, 2.4 Gigahertz, 512 Mo de DDR, espace disque de 180 Go, OS Linux Red Hat 7.2). Un système de gestion de base de données relationnelle oracle 9i (SGBD) contrôle la couche de données. Les modules ont été développés en Java à l'aide du JDK 1.4.1 de Sun Microsystem). Nous employons Sax de Microstar et l'analyseur syntaxique Dom de Sherry pour tous les flux de XML. Le module de visualisation 3D appelé "IFC Viewer" a été développé à l'aide de GL4JAVA en utilisant l'environnement de développement Eclipse 3.0. L'IFC viewer a passé la certification ISO/PAS 16739 en mai 2003 sous la direction de l'IAI. L'IFC viewer permet à l'utilisateur de voir la description complète du bâtiment avec une vue d'arbre IFC. D'ailleurs, elle autorise les intervenants à se déplacer autour du bâtiment en cours de conception et ainsi d'obtenir des informations sur les objets qui composent la scène. Cette 3D est "sémantiquement" dynamique, car la structure de la scène est construite en temps réel depuis la base de données IFC selon les caractéristiques géométriques (distance du point de vue, objet caché, ...) et les caractéristiques sémantiques (droit des utilisateurs, type d'utilisateur, étape de la visualisation du projet...).

Chapitre 7

« Si l'on regarde dans la bonne direction, il n'y a plus qu'à avancer »
(Proverbe bouddhique)

Conclusion

Sommaire

1. Contribution.....	145
1.1. Dans le domaine de la 3D	146
1.2. Dans le domaine l'intégration et manipulation de données	146
1.3. Dans le domaine de la gestion des projets d'ingénierie civile	146
2. Perspectives de recherches	147
2.1. Les systèmes d'informations et la validation sémantique.....	147
2.2. Application dans l'ingénierie civile	147

Les travaux de recherche présentés dans ce mémoire dressent un état de l'art des domaines d'application de la synthèse d'images, ainsi que les problématiques de stockage et de l'échange à travers le réseau de l'information géométrique. De plus, ce mémoire dresse un état de l'art sur l'intégration de données au travers de différentes approches allant de la traduction de schémas à l'utilisation de services Web en passant par les approches distribuées ou fédérées. L'idée générale qui se dégage de cet état de l'art réside dans l'utilisation du méta-langage XML permettant de résoudre l'hétérogénéité structurelle et syntaxique des données. La résolution de la problématique de l'hétérogénéité sémantique ouvre la voie à une nouvelle approche pour la gestion et la manipulation des données graphiques d'une scène 3D. Ce chapitre présente les contributions apportées par ces travaux dans les différents domaines d'applications et expose les perspectives possibles de recherches.

1. Contribution

Les recherches réalisées, sur le projet ACTIVe3D, ont fourni des résultats dans plusieurs domaines. Notre méthode apporte une plus-value dans le domaine de la 3D, dans le domaine des systèmes d'informations et dans le domaine de la gestion des projets d'ingénierie civile.

1.1. Dans le domaine de la 3D

Nous avons présenté une nouvelle approche de manipulation et d'extraction de données basée sur la notion de contexte, montrant que la valeur d'un élément sémantique dépend de son contexte d'utilisation. La définition d'un arbre contextuel permet d'extraire des données graphiques du système d'informations. Cette approche de la gestion des données à de nombreux avantages, dont celui de cibler l'information pertinente et d'ainsi limiter la taille des données à déplacer sur le réseau. À présent, les scènes 3D sont générées dynamiquement à partir d'une base de données, et ces scènes 3D sont manipulables par rapport à la définition sémantique qui les compose. De cette manière, il est possible de réduire le nombre de triangles composant la scène 3D en fonction des besoins de l'utilisateur et du contexte de travail.

1.2. Dans le domaine de l'intégration et de la manipulation de données

Nous avons présenté des mécanismes permettant d'identifier certains concepts et relations communs à plusieurs schémas XML. Cette identification permet à travers notre ontologie de mettre en correspondance les concepts et relations en fusionnant les attributs des deux éléments communs. L'ontologie de domaine résultant sera alors étendue et modifiée pour représenter la sémantique de plusieurs schémas XML relative à un domaine particulier. Cette intégration est étendue à l'intégration de documents XML. L'étude de la structure sémantique des grammaires XML et des ontologies a inspiré la construction de classes formant une ontologie d'intégration des données engendrée par des grammaires XML. Cette structure cognitive a de nombreux avantages, dont la principale est d'être évolutive aussi bien au niveau des données que de la définition des structures de données. Le domaine des ontologies a permis d'atteindre l'objectif fixé qui est une structure d'intégration de schémas et de données XML.

1.3. Dans le domaine de la gestion des projets d'ingénierie civile

ACTIVe3D BUILD SERVEUR est une plateforme Web permettant aux acteurs des projets d'interopérer et de fournir un travail collaboratif. Médaille d'or au concours de l'innovation à Bâtimat 2003, ACTIVe3D est la nouvelle génération de plateforme de gestion de projets qui se différencie de ses petites sœurs par l'utilisation des IFC 2.x comme format d'échange de données. Les IFC s'affirment aujourd'hui comme seul candidat réellement capable de gérer l'ensemble du cycle de vie d'un projet : conception, construction, gestion du patrimoine. ACTIVe3D permet non seulement d'échanger les formats classiques, mais offre l'opportunité de partager ces données en IFC. Ainsi, architectes, ingénieurs, économistes peuvent tour à tour se connecter et récupérer la maquette numérique du bâtiment totalement ou partiellement en IFC directement dans leurs applications métiers, de la travailler et la réexporter dans ACTIVe3D. Le modèle IFC permet de visualiser cette maquette en 3D et de naviguer à l'intérieur en temps réel

directement dans ACTIVe3D. Cette interface 3D correspond à un index secondaire permettant l'accès à un ensemble d'informations textuelles et techniques. Ces informations proviennent de sources hétérogènes comme une base de données, un Service Web ou tout simplement d'un fichier. Les résultats commerciaux ont donné lieu à un certain nombre de communications et références dans la presse écrite. (c.f. annexe 1.). De plus, la société ACTIVe3D-Lab gérant du projet ACTIVe3D est membre de l'équipe pour la promotion et le développement des IFC dans le cadre de l'usage des technologies de l'information et de la communication par les professionnels de la construction.

2. Perspectives de recherches

Les travaux de thèse ont porté principales sur le développement d'une ontologie pour l'intégration de schéma XML et de données XML. La deuxième partie des recherches a porté sur le développement d'une méthodologie pour la manipulation des données intégrées. Des améliorations et des perspectives se dessinent néanmoins et font l'objet des sections suivantes.

2.1. Les systèmes d'informations et la validation sémantique

La validation des définitions des arbres contextuels n'est que la première étape de la validation sémantique du système. L'étape suivante de la validation sémantique consiste à la réaliser sur les données. Les informations stockées dans le système cognitif sont des informations validées seulement de manière syntaxique et structurelle. En effet, les documents XML stockés sont bien formés et valides. Mais qu'en est-il de la sémantique ? Ce document est-il sémantiquement valide ? À titre d'exemple, les informations géométriques d'une scène contiennent deux parallélépipèdes qui se traversent. Dans le contexte géométrique, cela ne pose pas de problème. Dans le contexte structurel, le fichier contient une porte et une fenêtre. Si les deux documents sont intégrés dans le système, nous rendons compte que la fenêtre traverse la porte. Cette détection d'erreurs est réalisable par le système automatiquement, si nous lui fournissons les règles de validations sémantiques des données. Cet exemple est simple, mais si le système possède mille règles de validations sémantiques, celui-ci ira bien plus vite qu'un intervenant. Une voie future de recherche consiste à développer les règles et les processus de validation sémantique des données.

2.2. Application dans l'ingénierie civile

L'approche sémantique permet aux utilisateurs de valider une information plus complexe qui combine des données textuelles et géométriques. En effet, la sémantique est un outil puissant pour la détection d'erreur assistée par ordinateur dans le cadre de projets d'ingénierie civile. Prenons l'exemple suivant d'une poutre en aluminium de type « x232 » traversant un mur

porteur. Le processus de détection consiste tout d'abord à extraire les informations graphiques de la scène en prenant en compte les boîtes englobantes des éléments sémantiques concernant les poutres et les fenêtres. Ce processus suit la règle métier : une poutre ne doit pas traverser une porte. De plus, la règle est affinée avec les propriétés des objets concernés, c'est-à-dire une poutre en aluminium de type « x232 » et un mur porteur. La deuxième partie du processus consiste à détecter si une boîte englobante de la première liste d'objets de type poutre traverse une boîte englobante de la deuxième liste d'éléments sémantiques porte. Dans le cas où elles se coupent, la détection est réalisée sur le modèle géométrique complet. Les futurs travaux consistent à créer d'un cadre formel permettant à des utilisateurs de définir des règles adaptées aux besoins du client pour des détections d'erreurs. Ces règles logiques seront définies à partir des informations de l'ontologie générique que nous avons définie, ainsi que les données intégrées.

Bibliographie

- [3DIF04] 3D Industrial Forum : <http://www.3dif.org/>
- [3DM02] 3DML, http://www.global-dev.com/ress_dev/3dml
- [ABE02] K. Aberer, P. Cudré-Mauroux, M. Hauswirth. *A framework for semantic gossiping*. SIGMOD Record, 31(4), 2002
- [ABE04] K. Aberer, P. Cudré-Mauroux, A. M. Ouksel, T. Catarci, M. S. Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, O. De Troyer, T. Risso, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, R. Studer, *Emergent Semantics Principles and Issues*, DASFAA, Jeju Island, Korea, 17-19 March 2004
- [ABR98] H. Abrams, K. Watsen, M. Zyda, *Three-tiered interest management for large-scale virtual environments*, VRST 1998, pp. 125-129
- [ADA01] R. J. Adams, D. Klowden, B. Hannaford, *Virtual Training for a Manual Assembly Task*, Journal Haptics-e, Vol. 2, Number 2, Oct 2001
- [AHLE95] K. Ahlers, A. Kramer, *Distributed augmented reality for collaborative design applications* Tech. rep., European Computer Industry Research Center, 1995.
- [AIRE90] J. M. Airey, J. H. Rohlf, JR. F. P. Brooks, *Towards image realism with interactive update rates in complex virtual building environments*, Computer Graphics (1990 Symposium on Interactive 3D Graphics) 24, 2 (Mar. 1990), 41–50.
- [AMA03] B. Amann, *Du Partage centralisé de ressources Web centralisées à l'échange de documents intensionnels*, Documents de Synthèse, 2003.
- [BAC00] B. Bachimont, *Engagement sémantique et engagement ontologique : conception et réalisation d'ontologie en ingénierie des connaissances*, in Charlet J., Zackland M., Kessel G. & Bourigault D., eds., Ingénierie des connaissances : évolution récentes et nouveaux défis, Eyrolles, pages 305-323, 2000.
- [BAC99] B. Bachimont, *L'intelligence artificielle comme écriture dynamique : de la raison graphique à la raison conceptuelle*, Grasset, Paris, 1999

- [BALA96] J. F. Balaguer, S. De Gennaro, *VENUS: A virtual reality project at CERN.* Computer Graphics 30, 4 (Nov. 1996), 40–48.
- [BARB97] A. Barbora Raposo, L. Pini Malgalhães, I. L. Marques Ricarte, Sate University of Campinas UNICAMP, School of Electrical and Computer Engineering FEEC, Dept. of Computer Engineering and Industrial Automation DCA, University of Waterloo - Computer Science Dept., *Working with Remote VRML Scenes Through Low-Bandwidth Connections*, 1997
- [BAT83] C. Batini, M. Lenzerni, *A Methodology for Data Schema Integration in the Entity-Relationship Model*, Proceedings of the 3rd International Conference on Entity-Relationship Approach, pages 413-420, North Holland, 1983.
- [BAX00] A. Baxevanis, *The Molecular Biology Database Collection*, Nucleic Acids Research, vol 28, n°1, 2000, p.1-7
<http://nar.oupjournals.org/cgi/content/full/27/1/1>
- [BAYA96] S. Bayarri, M. Fernandez, M. Perez, *Virtual reality for driving simulation*, Communications of the ACM 39, 5 (May 1996), 72–76.
- [BBB03] U. Bockholt, A. Bisler, M. Becker, *Augmented reality for enhancement of endoscopic interventions*, In Virtual Reality (VR), Los Angeles, USA, March 2003
- [BEN00] D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell, B. Rapp, D. Wheeler, *GenBank*, Nucleic Acids Res, vol. 1, n°28, 2000, p. 15-8
<http://www.ncbi.nih.gov/Genbank/>
- [BER00] J. Berstel, L. Boasson, *XML Grammars* MFCS 2000: 182-191
- [BI3D95] J.-F. Balaguer, E. Gobbi, *i3D: a high-speed 3D Web Browser*, In 1995 Symposium on the Virtual Reality Modeling Language (VRML '95) (Conference held in San Diego, CA, USA, Dec. 1995), ACM Press, pp. 69–76.
- [BLA02] Bluxxun, <http://www.bluxxun3d.com>
- [BOO02] Boon-Hee Kim, Jun Hwang, Young-Chan Kim, *The design of high-level database access method in a Web-based 3D object authoring tool*, The Fourth International Conference on Distributed Communities on the Web, 3-5 April 2002, Sydney, Australia.
- [BOW99] D. Bowman, E. Davis, A. Badre, L. Hodges, *Maintaining Spatial Orientation during Travel in an Immersive Virtual Environment*. Presence: Teleoperators and Virtual Environments, vol. 8, no.6, 1999, pp. 618-631
- [BROO86] JR. F. P. Brooks, *Walkthrough — A dynamic graphics system for simulating virtual buildings*. In Proceedings of 1986 Workshop on Interactive 3D Graphics (Oct. 1986), F. Crow and S. M. Pizer, Eds., pp. 9–21.
- [BURD96] G. C. Burdea, *Force and Touch Feedback for Virtual Reality*. John Wiley & Sons. 1996

-
- [CAL01] A. Cali, G. De Giacomo, M. Lenzerini, *Models for Information Integration: Turning Local-as-View into Global-as-View*, Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII 2001), 2001
- [CAVA98] Le projet Cavalcade : <http://vr.c-s.fr/cavalcade/index.html>
- [CHA98a] B. Chandrasekaran, J. Josephson, V. Benjamins, *The Ontology of Tasks and Methods*, in Proceedings of the 11th workshop on Knowledge Acquisition, Modeling and Management (KAW'98), 1998.
- [CHR03] D. Christopoulos, A. Gaitatzes, G. Papaioannou, *Image-Based Techniques for Enhancing Virtual Reality Environments*, 2nd International Workshop on ICT's, Arts and Cultural Heritage, November 2003, Athens, Greece
- [CONN92] D. B. Conner, S. S. Snibbe, K.P. Herndon, D. C. Robbins, R.C. Zeleznik, A. Van Dam, *Three-dimensional widgets*. Computer Graphics 25, 2 (Mar. 1992), 183–188.
- [COS02] Cosmo Player, <http://www.cai.com/cosmo>
- [CRU04] I. F. Cruz, H. Xiao & F. Hsu, *An Ontology-based Framework for Semantic Interoperability between XML Sources*, In Eighth International Database Engineering & Applications Symposium (IDEAS 2004), July 2004.
- [DEC00] T. Dechilly, B. Bachimont, *Une ontologie pour éditer des schémas de description audiovisuels, extension pour l'inférence sur les descriptions*, in Actes des journées francophones d'Ingénierie des Connaissances (IC'2000), 2000.
- [DEK03] A. Dekhtyar, I. E. Iacob, *A Framework for Management of Concurrent XML Markup*, International Conference on Conceptual Modeling, in ER 2003, pages 311-322, 2003.
- [DRA01] D. Draper, A. Y. HaLevy, D. S. Weld, *The Nimble XML Data Integration System*, IEEE International Conference on Data Engineering, April 02 - 06, 2001, Heidelberg, Germany
- [DRAS96] D. Drascic, *Stereoscopic vision and augmented reality*. ScientificComputing and Automation 9, 7 (June 1996), 31–34.
- [ERIK96] C. Erikson, *Polygonal simplification: An Overview*. Technical Report TR96-016, Department of Computer Science, University of North Carolina – Chapel Hill, February 16, 1996.
- [FAU79] I. D. Faux and Ellis Horwood Ltd Mr. J. Pratt. (1979) *Computational geometry for design and manufacture*.
- [FEI90] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes. *To compute graphics principles practice*. Addison-Wesley Pub. Co., Reading, MA, 1990.

- [FEIN93] S. Feiner, B. Macintyre, M. Haupt, E. Solomon, *Windows on the world: 2D windows for 3D augmented reality*. In Proceedings of the ACM Symposium on User Interface Software and Technology (1993), Virtual Reality, pp. 145–155.
- [FERM03] The Fermi Surface Database : <http://www.phys.ufl.edu/fermisurface>
- [FISH86] S. Fisher, M. McGreevy, J. Humphries, W. Robinett, *Virtual environment display system*. In Proc. 1986 ACM Workshop on Interactive 3D Graphics (Chapel Hill, NC, Oct. 1986), pp. 77–87.
- [FUNK96] T. Funkhouser, S. Teller, C. Sequin, D. Khorramabadi, *The UC Berkeley System for Interactive Visualization of Large Architectural Models*, Presence, the Journal of Virtual Reality and Teleoperators, vol5, nb1, MIT Press(1996), pp.13-44
- [GAR02] G. Gardarin, A. Mensch, A. Tomasic, *An Introduction to the e-XML Data Integration Suite*, EDBT 2002, 297-306
- [GEN87] M. R. Genesereth, N. J. Nilsson, *Logical Foundation of Artificial Intelligence*. Morgan Kaufmann, Los Altos, California, 1987.
- [GIA00] G. De Giacomo, D. Calvanese, M. Lenzirini. *Answering queries views in description logics*. In proceeding of AAAI, 2000.
- [GOBB95] E. Gobbetti, J. Balaguer, *An integrated environment to visually construct 3D animations*. In SIGGRAPH 95 Multimedia Conference Proceedings (Conference held in Los Angeles, CA, USA, Aug. 1995), R. Cook, Ed., Annual Conference Series, ACMSIGGRAPH, Addison-Wesley.
- [GOM99] A. Gomez-Pérez, *Ontological Engineering: A state of art*. Expert Update, 2(3), 33-43, 1999
- [GRA03] R. Grasset, J.D. Gascuel, *Réalité Augmentée et environnement collaboratif Un tour d'horizon*, Actes des 16èmes journées de l'AFIG 2003
- [GRU93] T. Gruber, *A translation approach to portable ontology specifications*, Knowledge Acquisition 5(2), pages 199-220, 1993.
- [GRU94] T. Gruber, G. Olsen, *An ontology for engineering mathematics*, in J. Doyle F. & Torano P., eds., Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning, Morgan-Kauffmann, 1994.
- [GRU95] M. Gruninber, M. S. Fox, *Methodology for the design and evaluation of ontologies*, in Proceedings of workshop on Basic Ontological Issues on Knowledge Sharing, IJCAI'95, 1995.
- [GRU95] T. Gruber, *Toward Principles for the Designs of Ontologies Used for Knowledge Sharing*. International Journal of Human And Computer Studies, 43(5/6):907-928, 1995.

-
- [GUA94] N. Guarino, C. Carrara, P. Giaretta, *An ontologie of meta-level categories*, in J. Doyle F. S & Torano P., eds., Principles of Knowledge representation and Reasonning, Morgan-Kauffman, pages 270-280, 1994.
- [GUA94a] N. Guarino, C. Carrara, P. Giaretta, *An ontologie of meta-level categories*, in J. Doyle F. S & Torano P., eds., Principles of Knowledge representation and Reasonning, Morgan-Kauffman, pages 270-280, 1994.
- [GUA94b] N. Guarino, *The ontological level*, in R. Casati B. S. & White G., eds, Philosophy and the cognitive sciences, Hölder-Pichler-Tempsky, 1994.
- [GUA95] N. Guarino, P. Giaretta, *Ontologies and knowledge based, towards a termonological clarification*, in Mars N., eds., Towards very large knowledge bases: knowledge building and knowledge sharing, IOS Press, pages 25-32, 1995.
- [GUA98] N. Guarino, *Formal Ontology and Information Systems*, Proceedings of FOIS'98, Trento, Italy, 6-7 June 1998. Amsterdam, IOS Press, pp. 3-15.
- [HAL01] A. Y. Halevy, *Answering queries using views: a survey*. VLDB Journal, 10(4), 2001.
- [HECK94] P. Heckbert, M. Garland, *Multiresolution Modeling for Fast Rendering*. In Graphics Interface '94 Proceedings, pages 43-50, 1994.
- [HEGU01] O. Heguy, N. Rodriguez, H. Luga, J.P. Jessel, Y. Duthen, *Vitrual Environment for Coopérative Assistance in Teleoperation*, WSCG 2001 Conference Proceedings, V. Skala, 2001
- [HEND92] K.P. Herndon, R.C. Zeleznik, R. C. Robbins, D. B. Conner, S. S. Snibbe, A. Van Dam, *Interactive shadows*. In ACM Symposium on User Interface Software and Technology, 1992, 1-6
- [HUAN99] Bo Huang, Hui Lin, *GeoVR: a web-based tool for virtual reality presentation from 2D GIS data*, Computers and Geosciences 25(1999), p1167-p1175, 1999
- [IAM98] Information Architecture Markup Language, IAML, <http://www.vizbang.com>
- [IMAG97] Rapport d'activité 1997, INRIA, Projet iMAGIS
- [KAY97] D. Kayser, *La représentation des connaissances*, Hermès, 1997.
- [KIF95] M. Kifer, G. Laussen, J. Wu, *Logical foundations of object-oriented and frame-based languages*, in journal of the ACM, 1995.
- [KIM02] B. Kim, J. Hwang, Y. Kim, *The design of high-level database access method in a Web-based 3D object authoring tool*, The Fourth International Conference on Distributed Communities on the Web, 3-5 April 2002, Sydney, Australia
- [KLE02] M. Klein, *Interpreting XML via an RDF schema*. In ECAI workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002), Lyon, France.

- [KLI02] G. Klinker, A. Dutoit, M. Bauer, J. Bayes, V. Novak, D. Matzke, *Fata morgana – a presentation system for product design*, In International Symposium on Augmented and Mixed Reality (ISMAR) 2002.
- [KUHL95] J. Kuhl, D. Evans, Y. Papelis, R. Romano, G. Watson, *The Iowa Driving Simulator: an immersive research environment*. Computer 28, 7 July 1995, 35–41.
- [LAK03] L. V. Lakshmannan, F. Sadri, *Interoperability on XML Data*, In Proceeding of the 2nd International Semantic Web Conference (ICSW'03), 2003.
- [LEC02] M. Leclere, F. Trichet, F. Fürst, *Operationalising domain ontologies : towards an ontological level for the SG family*, in Foundations and Applications of Conceptual Structure, contributions to the International Conference on Conceptual Structures (ICCS'02), Bulgarian Academy of Sciences, 2002.
- [LIE98] J. Liebehenschel, *Lexicographical Generation of a Generalized Dyck Language*, Technical Report: Interner Bericht 5/98, Fachbereich Informatik, Johann Wolfgang Goethe-Universität, Frankfurt am Main, Germany, submitted.
- [LIPK99] D. Lipkin, *Method and Apparatus for Implementating Dynamic VRML*, Belmont, Calif., Oracle Corporation, United States Patent, 1999
- [LXU04] L. Xu, D. W. Embley, *Combining the Best of Global-as-View and Local-as-View for Data Integration*, ISTA'04, July 2004
- [MEN03] S. Menon, B. Barnes, R. Mills, C. D. Bruyns, A. Twombly, J. Smith, K. Montgomery, R. Boyle, *Using Registration, Calibration, and Robotics to Build a More Accurate Virtual Reality Simulation for Astronaut Training and Telemedicine*, The 11-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2003, WSCG 2003
- [MOR85] M. E. Mortenson, *Geometric modeling*, Wiley, 1985
- [MORO91] W. F. Moroney, B.W Moroney, *Utilizing a microcomputer based flight simulation in teaching human factors in aviation*. In Proceedings of the Human Factors Society 35th Annual Meeting (1991), vol. 1 of Educators' Professional: Use of Microcomputers in Teaching Human Factors in Aviation, pp. 523–527.
- [MUEL96] C. Mueller, *Architectures of image generators for flight simulators*. Tech. Rep. TR95-015, Department of Computer Science, University of North Carolina - Chapel Hill, Apr. 24 1995. Wed, 26 Jun 1996
- [NCID94] The NCI DIS 3D database : http://dtp.nci.nih.gov/docs/3d_database/dis3d.html
- [NIC02] C. Nicolle, K. Yétongnon, *XML Based Toolkits for the Interoperability of Web Information Systems*, Chapter book, Web-Enabled Systems Integration, Challenges & Practice, 2002

-
- [NIC05] C. Nicolle, J. C. Simon, K. Yétonnon, Book Title: *Encyclopedia of Information Science and Technology*, Volume I-III, Chapter Title : *An Overview of issues for the interoperability of information systems*, Mehdi Khosrow-Pour Editor, IDEA Group Publishing, pp. 1 - 7, To appear Feb. 2005
- [PAI03] D. Paillot, F. Merienne, J.-P. Frachet, M. Neveu, *Triangulation et simplification de modèles surfaciques application à la visualisation temps réel*, GTMG, Aix en Provence, pp 131-138, 19-20 mars 2003.
- [PAN02] A. Pan, J. Raposo, M. Álvarez, P. Montoto, V. Orjales, J. Hidalgo, L. Ardao, A. Molano, Á. Viña, *The Denodo Data Integration Platform*, VLDB, Hong Kong, China, 2002
- [PAP02] Y. Papakonstantinou, V. Vassalos, *Architecture and Implementation of an XQuery-based Information Integration Platform*, IEEE Data Engineering Bulletin, Volume 25, Issue 1, pp 18-26, March 2002.
- [PEA04] D. Pearce Partial, A. M. Day, *Visibility for Virtual Reality Applications*, WSCG'2004, February 2-6, 2003, Plzen, Czech Republic.
- [PFUN01] M. Pfund, *Topologic data structure for a 3D GIS*, Proceedings of ISPRS, Vol.34, Part 2W2, 23-25 May, Bangkok, Thailand, pp. 233-237, 2001
- [PHP02] PHP, HTML embedded scripting language, <http://www.php.net>
- [PRIS02] <http://www.prism.uvsq.fr> : PriSM, Laboratoire de recherche en informatique CNRS UMR-8636, Université de Versailles St-Quentin en Yvelines
- [UPPP97] E. Puppo, R. Scopigno, *Simplification, LOD and Multiresolution – Principles and Applications*. Technical Report C97-12, CNUCE, C.N.R., Pisa (Italy), June 1997
- [RAB01] E. Rahm, P. A. Bernstein, *A survey of approaches to automatic schema matching*. The VLDB Journal, 10(4):334350, 2001
- [REC93] F. Recanati. *Direct Reference: From Language to Thought*. Blackwell Publishers, Oxford, UK, 1993
- [REM96] R. Ronfard, J. Rossignac, *Full-range Approximation of Triangulated Polyhedra*. Computer Graphics Forum 15(3): 67-76 (1996)
- [RON96] R. Ronfard, J. Rossignac, *Full-range Approximation of Triangulated Polyhedra*, Computer Graphics Forum 15(3): 67-76 (1996)
- [ROSE96] J. M. Rosen, D. LAUB, *Virtual reality and medicine: From training systems to performing machines*. In Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium (1996), pp. 5–13.
- [ROSS96] J. Rossignac, editor. *Geometric Simplification* (ACM SIGGRAPH Course Notes N°.35). ACM Press, 1996.

- [SANA99] J. El-Sana, A. Varshney, *Generalized view-dependent simplification*, Computer Graphics Forum, pp C83-C94, 1999
- [SDA02] Standard Data Access Interface, <http://www.perdis.esprit.ec.org/apf/sdai/>
- [SETH91] S. J. Teller, C. H. Sequin, *Visibility preprocessing for efficient walkthroughs of 3D scenes*, Graphics Interface '93, pp61-69, July 1991
- [SHE90] A. Sheth, J. Larson, *Federated Database Systems for Managing Distributed, Heterogenous and Autonomous Databases*. ACM Computing Surveys, Vol. 22, N°3, p. 183-236, September 1990.
- [SHEN93] E. C. Shenchang, L. Williams, *View interpolation for image synthesis*, In Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '93), pages 279–288, 1993.
- [SHI02] W. S. Shi, B.S Yang, QQ Li, 2002, *An Object-Oriented Data Model for Complex Objects in Three-Dimensional Geographic Information Systems*. International Journal of Geographic Information Science.
- [SIM02] J.-C. Simon, C. Cruz, C. Nicolle, *ACTIVe3D-WS : A Web-Services Based Multimedia Platform*, IEEE 4th International Conference on Multimedia Software Engineering (MSE'02), December 11-13 2002, NewPort Beach, USA, pp. 64-69.
- [SOW84] J. Sowa, *Conceptual structures: information processing in mind and machine*, Addison-Wesley, 1984.
- [STAT96] A. State, M. A. Livingston, W. F. Garrett, G. Hirota, *Technologies for augmented-reality systems: Realizing ultrasound-guided needle biopsies*. Computer Graphics 30, Annual Conference Series (1996), 439–446.
- [STE02] Standard for the Exchange of Product model data
<http://www.nist.gov/sc4/www/stepdocs.htm>
- [STEV97] S. J. Gortler, L.-W. He, M. F. Cohen, *Rendering layered depth images*, Technical Report MSTR-TR-97-09, Microsoft Research, Redmond, WA, March 1997.
- [SUTH65] I. E. Sutherland, *The ultimate display*. In Proceedings of IFIPS Congress (New York City, NY, May 1965), vol. 2, pp. 506–508.
- [SVG02] W3C Scalable Vector Graphics, <http://www.w3.org/Graphics/SVG/Overview.html>
- [SZA95] K. Szabo, P. Stucki, P. Aschwanden, T. Ohler, R. Pajarola, P. Widmayer, *A Virtual Reality Based System Environment for Intuitive Walk-Throughs and Exploration of Large-Scale*, Tourist Information. Proceedings of the Enter95 Conference, 1995, pp. 10-15
- [TERZ94] D. Terzopoulos, T. Xiaoyuan, R. Grzeszczuk, *Artificial Fishes with Autonomous Locomotion, Perception, and Learning in a Simulated Physical World*. Artificial Life IV, R. Brooks et P. Maes Editors, MIT Press, pp 17-27, 1994

-
- [ULL97] J. D. Ullman, *Information Integration using Logical Views*, In International Conference on Database Theory (ICDT), pages 19-40, 1997.
- [URBA96] E. URBAN, *The information warrior*. IEEE Spectrum 32, 11 (1996), 66–70.
- [USC95] M. Uschold, M. King, *Towards a methodology for building ontologies*, in Proceedings of the workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI'95, 1995.
- [USC96] M. Uschold, *Building ontologies : towards a unified methodology*, in Proceedings of the 16th conference of the British Computer Society Specialist Group on Expert Systems, 1996.
- [VAK98] A. Vakaloudis, B. Theodoulidis, Timelab, Dept. of Computation UMIST, *The storage and querying of 3D objects for the Dynamic Composotion of VRML Worlds*, 1998
- [VAN04] Vanlande R., Cruz C., Nicolle C., *Modélisation sémantique pour la génération dynamique de scènes 3D Application aux projets du bâtiment*, in SETIT 2004, Sousse, Tunisia, 15-19 mars.
- [VIP01] D. Aliakseyeu, J. B. Martens, S. Subramanian, M. Vroubel, W. Wesselink, *Visual Interation Platfrom*, In Interact 232-239, Tokyo, Japan, July 2001.
- [VRM97] Web3D Specifications VRML97 International Standard
http://www.vrml.org/fs_workinggroups.htm
- [W3C02] The World Wide Web Consortium (W3C) develops interoperable technologies
<http://www.w3c.org>
- [WILL02] W. V. Baxter III, A. Sud, N. K. Govindaraju, D. Manocha, *GigaWalk : Interactive Walkthrough of Complex Environments*, Eurographics Workshop on Rendering, 2002.
- [WILL97] W. R. Mark, L. McMillan, G. Bishop. *Post-rendering 3D warping*. In Proceedings of the 1997 Symposium on Interactive 3D Graphics. ACM SIGGRAPH, April 1997.
- [WONK01] P. Wonka, M. Wimmer, F. Sillion, *Instant Visibility*, EG 2001 Proceedings, vol20(3), Blackwell Publishing, A. Chalmers and T.-M. Rhyne, pp. 411-421
- [X3D02] Extensible 3D, X3D, <http://www.web3d.org/x3d.html>
- [XGL01] XGL File Format Specification, <http://www.xglspec.org>
- [XML02] Extensible Markup Language, <http://www.w3.org/XML/>
- [XPE00] M. J. Carey, J. Kiernan, J. Shanmugasundaram, E. J. Shekita, S. N. Subramanian, *XPERANTO : Middleware for Publishing Object-Relational Data as XML Documents*, The VLDB Journal, pp 646-648, 2000

- [XSL02] Extensible Stylesheet Language, <http://www.w3.org/Style/XSL/>
- [YAGE96] R. Yagel, D. Stredney, G. J. Wiet, P. Schmalbrock, L. Rosenberg, D. J. Sessanna, Y. Kurzion, S. King, *Multisensory platform for surgical simulation*, In IEEE Virtual Reality Annual International Symposium 1996 (VRAIS'96) (Mar. 1996), pp. 72–78.
- [ZAJT97] R. Zajtchuk, R. M. Satava, *Medical applications of virtual reality*, Communications of the ACM 40, 9 (Sept. 1997), 63–64.
- [ZWE99] P. Zweigenbaum, *Encoder l'information médical : des terminologies aux systèmes de représentation des connaissances*, in Innovation stratégique en informatique de santé (ISIS) (2-3), pages 27-47, 1999

Annexe 1

Articles de presse

- « Technologies Nouvelles » Innovation, le palmarès, janvier 2004.
- « L'entrepreneur » Build Server d'ACTIVE3D-Lab, janvier 2004.
- « BATIRAMA » Classe 6 Informatique Gestion, janvier 2004.
- « Le Moniteur » La construction s'essaye aux formats d'échanges universels, mars 2004.
- « Architecture intérieure » ACTIVE3D, Travail collaboratif et interopérabilité, n°: 313, mars/avril 2004.
- « Ingénierie & Conseil » L'interopérabilité des logiciels du BTP : enfin un vrai progrès depuis la Renaissance !, n° : 52, avril 2004.
- « MAG-I » Interopérabilité : des plate-formes font communiquer les logiciels, n° : 4, avril 2004.
- « Le journal du Palais de Bourgogne» Le groupe dijonnais Archimen lance « ACTIVE3D BUILD » en novembre, mai 2003.
- « Prêt à Poser » Plateforme Internet de gestion de projet, n°: 7, octobre 2003.
- « Le Moniteur » ACTIVE3D, n°: 5214, octobre 2003.
- « 01 Informatique » ACTIVE3D-Lab réunit les acteurs du bâtiment autour d'une maquette numérique, n°: 1746, novembre 2003.
- « Les cahiers techniques du bâtiment » Plate-forme Internet pour la gestion de projets, novembre 2003.
- « la lettre hebdomadaire industriel et technologies » L'innovation récompensée à Bâtimat, novembre 2003.
- « Le Moniteur » BUILD SERVEUR – Système de gestion de projets, n°: 5215, novembre 2003.

- « Le journal du Palais de Bourgogne» ACTIVe3D récompensé au salon Bâtimant, novembre 2003.
- « Le Géomètre » Internet et 3D pour le bâtiment, n°: 12, décembre 2003.
- « BTP Magazine » Les « Oscars » du bâtiment à Batimat 2003, décembre 2003.
- « Le Bien Public » NTIC : Une ministre étoilée pour un cybersalon, novembre 2002.
- « Impact » NTIC, page 7, décembre 2002.
- « Revue technique du bâtiment et des constructions industrielles » ACTIVE3D BUILD SERVEUR, n° :218, 2003.
- « Le Bien Public » Bourgogne Réseaux continu de tisser des liens, décembre 2001.
- « Le Monde Informatique» Le2I, un laboratoire de recherche qui trouve, n° : 914, novembre 2001.

Annexe 2

Publications des travaux

Chapitre de livre

Christophe Cruz, Christophe Nicolle, Marc Neveu, *Encyclopedia of Multimedia Technology and Networking*, Editor: Margherita Pagani Bocconi University, Italy, to appear january 2005.

Congrès internationaux avec actes et comité de lecture

Christophe Cruz, Renaud Vanlande, Christophe Nicolle, *Ontology-Based 3D Manager Systems*, International Conference of Adaptive Hypermedia and Adaptative Web-based System, pages 18-24, 23-26 August 2004 in Eindhoven.

Christophe Cruz, Renaud Vanlande, Christophe Nicolle, *ACTIVe3D: Semantic and 3D Databases for Civil Engineering Projects*, International Conference on Information and Knowledge Engineering (IKE 04) Monte Carlo Resort, Las Vegas, Nevada, USA. June 21-24, 2004.

Christophe Cruz, Renaud Vanlande, Christophe Nicolle, *Modélisation sémantique pour la génération dynamique de scènes 3D, Application aux projets du bâtiment*, Conférence International : Sciences Electroniques, Technologies de l'Information et des Télé-Communications, IEEE , Sousse, Tunisie, 15-20 Mars 2004.

Renaud Vanlande, Christophe Cruz, Christophe Nicolle, *Managing IFC files in civil engineering projects*, ACM CIKM International Conference on Information and Knowledge Management, New Orelans, Lousianne, USA, 3-8 November 2003.

Christophe Cruz, Christophe Nicolle, Marc Neveu, *Active3D : A Method for Storing 3D scenes into a RDBMS*, 2nd IEEE International Symposium on Signal Processing and Information Technology, (ISSPIT'02), Marrakesh, Morocco, December 18-21, 2002, pp. 163-168.

Jean-Claude Simon, Christophe Cruz , Christophe Nicolle, *ACTIVe3D-WS : A Web-Services Based Multimedia Platform*, IEEE 4th International Conférence on Multimadia Software Engineering (MSE'02), December 11-13 2002, NewPort Beach, USA, pp. 64-69.

Revues internationales avec comité de lecture

Christophe Cruz, Christophe Nicolle, Marc Neveu, *The Active3D-Build: A Web-Based Civil Engineering Platform*, IEEE Multimédia, Tiziana Catarci Editor, pp 87-90, Vol.9 No.4, October 2002.

Workshop & Posters internationaux avec actes et comité de lecture

Renaud Vanlande, Christophe Cruz, Christophe Nicolle, *Web Cooperative Application for Civil Engineering Project*, 22th International Conference on Conceptual Modeling, ER'2003, Chicago, Illinois, USA, 13 October 2003

Congrès nationaux avec actes et comité de lecture

Christophe Cruz, Christophe Nicolle, Marc Neveu, *ACTIVe3D : interrogation de scènes 3D en SQL*, Accès Intelligent aux documents multimédias sur l'internet, MediaNet 2002, Editions Hermés , Sousse, Tunisie, 17–21 juin 2002.

Colloques

Journée thématique : "Maquette numérique et Patrimoine", Institut Image-ENSA, Cluny, 7 mars 2002.

Intégration et Manipulation de Données Hétérogènes au Travers de scènes 3D Dynamiques, Evolutives et Interactives.
Application aux IFC pour la Gestion Collaborative de Projets de Génie Civil.

Mots clés : Base de données, Systèmes d'information, Intégration de données, XML, Ontologie, IFC.

Résumé : L'axe de recherche de cette thèse est la frontière de deux domaines qui sont la réalité virtuelle et les systèmes d'informations. L'objectif des recherches est de définir une méthode, pour permettre l'intégration sémantique de données XML hétérogènes et des mécanismes pour manipuler ces données au travers d'une scène 3D dynamique, évolutive et interactive.

La 3D offre la possibilité de créer une image numérique d'une représentation mentale d'un produit. C'est pour cela qu'elle est souvent utilisée comme support pour la collaboration entre acteurs d'un projet. L'utilisation du méta-langage XML permet de résoudre l'hétérogénéité structurelle et syntaxique des données pour la collaboration entre acteurs d'un projet. Par contre, la problématique de l'hétérogénéité sémantique persiste.

La solution développée dans cette thèse pour intégrer des données XML hétérogènes est articulée en deux étapes. La première étape concerne la formalisation sémantique des règles d'écriture d'une grammaire XML en général. Cette formalisation nous permettra de définir les composants d'une ontologie générique. La seconde étape concerne la définition des mécanismes d'ontologisation de la sémantique des éléments d'une grammaire XML spécifique pour obtenir une ontologie de domaine. Les concepts et les relations de l'ontologie de domaine sont alors définis à partir des éléments des schémas XML. La spécification explicite de la sémantique contenue dans les schémas XML permet alors d'intégrer les documents XML grâce aux concepts et aux relations communs.

Cette thèse traite également des mécanismes de manipulations des documents intégrés. Ces mécanismes sont basés sur la notion de contexte lié à des arbres-XML. Un contexte est une vue utilisateur du système en fonction d'un domaine d'application spécifique. Cette vue est un arbre-XML que nous appellerons arbre contextuel. Les arbres contextuels sont le résultat de l'extraction de données provenant de documents intégrés pour produire un document XML indépendamment de leur structure originelle.

Les principes développés dans cette thèse sont appliqués dans le cadre d'une plateforme Web collaborative pour les projets d'ingénieries civiles. Grâce aux processus de gestions et de manipulations basés sur les arbres contextuels, les intervenants d'un projet d'ingénierie civile ont accès aux informations intégrées à travers une vue métier. Cette vue métier possède deux avantages majeurs. D'une part, elle contient des informations réduites limitant la taille du flux réseau tout en étant pertinentes pour le contexte d'utilisation. D'autre part, elle possède une représentation graphique 3D construite en fonction d'un contexte d'utilisation. Cette représentation 3D se comporte alors comme un index sur le système d'informations.

Integration and Handling of Heterogeneous Data through Dynamic, Evolutionary and Interactive 3D Scenes.
Application to the IFC for Collaborative Management of Civil Engineering Projects.

Mots clés : Database, Information Systems, Data Integration, XML, Ontology, IFC.

Summary: The research orientation of this thesis is at the border of two fields which are virtual reality and information systems. The research objective consist to define a method, to allow semantic integration of heterogeneous XML data and mechanisms to handle these data through dynamic, evolutionary and interactive 3D scene.

3D makes it possible to create a digital picture from a mental representation of a product. For this reason, it is often used as support for collaboration between actors of a project. The use of the XML meta-language allows to solve structural and syntactic data heterogeneity for the collaboration between actors of a project. On the other hand, the problems of semantic heterogeneity persist.

The solution developed in this thesis to integrate heterogeneous XML data is articulated in two stages. The first stage relates to the semantic formalization of the XML grammar writing rules. This formalization will enable us to define the components of a generic ontology. The second stage relates to the semantics definition of the ontologisation mechanisms from elements of a specific XML grammar to obtain an ontology of field. The concepts and the relations of the domain ontology then defined starting from the elements of XML schema. The explicit specification of the semantics contained in XML schema then makes it possible to integrate XML documents thanks to common concepts and common relations.

This thesis also treats of handling mechanisms for the integration of documents. These mechanisms are based on the concept of context related to XML trees. A context is a user view of the system according to a specific domain of application. This view is a XML tree which we will call contextual tree. The contextual trees are resulted of the data extraction coming from integrated documents to produce a XML document independently of their original structure.

The principles developed in this thesis are applied within the framework of a collaborative Web platform for civil engineering projects. Thanks to the processes of management and handling based on the contextual trees, actors of a civil engineering project have access to the information integrated through a trade view. This trade view has two major advantages. On the one hand, it contains reduced information limiting the size of network flow while being relevant for the context of use. In addition, it has a 3D graphical representation built according to a context of use. Then 3D representation behaves like an index on the information system.

