MASTER EN SCIENCES INFORMATIQUES

Les données spatiales sur le Web
représentation 3D de villes et de bâtiments

David, Mike

Award date:
2016

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2015–2016

Les données spatiales sur le Web : représentation 3D de villes et de bâtiments

Mike David

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Pr. Philippe Thiran

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Avant-propos

Je tiens à remercier toutes les personnes qui m'ont aidé dans la réalisation de ce mémoire.

Premièrement, je remercie mon directeur de mémoire, Monsieur Philippe Thiran, professeur à l'université de Namur, pour ses conseils et son orientation sans lesquels ce mémoire n'aurait pu aboutir.

Je remercie également ma mère et Elisa Firantello pour leur relecture et leurs conseils lors de la rédaction de ce mémoire. Enfin, je remercie toute ma famille et mes amis pour leur soutien et leurs encouragements.

Table des matières

Table des figures

Introduction

Spatial Data on the Web Working Group

Très souvent, un facteur commun à de nombreux ensembles de données est la notion de localisation spatiale, rendant très important l'utilisation de cartes pour la visualisation des corrélations entre elles. Malheureusement, les formats utilisés pour ces données spatiales ne sont pas encore suffisamment intégrés sur le Web.

C'est pourquoi le W3C (World Wide Web Consortium) qui s'occupe des standards pour le Web au niveau international, a lancé un groupe de travail sur la représentation de données spatiales sur le Web. La mission de ce groupe consiste à clarifier et formaliser les standards les plus adéquats. Plus particulièrement, ses objectifs sont [25] :

- déterminer comment les informations spatiales peuvent être intégrées au mieux avec les autres données du Web ;
- déterminer comment les machines et les personnes peuvent découvrir que différents faits d'ensembles de données variées concernent le même endroit, et plus particulièrement quand "un endroit" est exprimé de différentes façons, à des niveaux de granularité différents ;
- identifier et évaluer les différentes méthodes et les différents outils existants et ensuite créer un ensemble de bonnes pratiques pour leur utilisation ;
- lorsque c'est nécessaire, compléter la standardisation des technologies informelles déjà très répandues.

Ce groupe de travail (le *Spatial Data on the Web Working Group*) est chargé de travailler en collaboration avec l'Open Geospatial Consortium (OGC). Dans ce travail, nous évaluerons deux modèles de données spatiales en insistant sur leurs forces respectives et les correspondances qui peuvent être tirées entre eux. Ansi nous donnerons un début de réponse au deuxième objectif de ce groupe de travail.

Objectifs

De nombreuses villes utilisent des systèmes d'information géographique pour leur cartographie. Ces systèmes permettent de répertorier de nombreux aspects de celles-ci en termes d'utilisation du territoire mais ils permettent également l'analyse et la planification de nouvelles mesures. Parmi ces systèmes, CityGML se démarque particulièrement pour plusieurs raisons :

- il permet la représentation des différentes caractéristiques d'une ville en 3 dimensions ;
- il permet une représentation de la ville sur différentes échelles en allant de la carte en 2,5 dimensions pour les usages du territoire jusqu'à des cartes 3D affichant tous les détails des bâtiments ;
- enfin, il permet la représentation de l'intérieur des bâtiments, se rapprochant ainsi des outils de représentation (et de modélisation) que l'on peut retrouver en architecture et en ingénierie.

Servant de base au projet européen INSPIRE (Infrastructure for Spatial Information in the European Community) ayant pour but d'unifier les formats de données spatiales au niveau européen, son adoption augmente et c'est ainsi que l'on retrouve désormais des modèles pour de nombreuses villes d'europe. Néanmoins, les données CityGML sont souvent issues d'anciens systèmes d'information géographique en 2 dimensions et donc ne permettent pas une exploitation optimale de ce modèle. Bien qu'il existe des techniques de génération automatisée de modèles plus détaillés (comme l'usage d'images satellites et aériennes, de scan laser, etc.), celles-ci ne permettent pas d'obtenir des représentations des intérieurs des bâtiments. C'est pourquoi nous nous intéressons au modèle IFC qui trouve son origine dans le domaine de la construction (architecture, ingénierie, etc.). Ces modèles étant réalisés dès la conception d'un bâtiment dans le but de servir de référence durant sa construction ainsi que le reste de son cycle de vie, ceux-ci sont souvent plus détaillés que leur équivalent en CityGML. En effet, ils exposent des informations structurelles, et des données à une échelle plus fine de manière générale.

C'est ainsi que nous avons entrepris dans ce travail de montrer comment intégrer ces deux modèles aux domaines et aux applications très différents.

Démarche

Dans ce document, nous nous intéresserons donc tout particulièrement à CityGML, un standard de l'OGC pour la représentation 3D de villes et à son intégration avec IFC. Nous commencerons dans le premier chapitre par présenter le standard CityGML. Nous détaillerons tout d'abord ses origines avec les systèmes d'information géographique ainsi que sa création par l'OGC. Nous passerons également en revue les cas d'utilisation pour lesquels il a été conçu. Nous nous attarderons ensuite sur la structure de son modèle de représentation tel que défini dans la spécification en version 2.0.0. Pour ce faire, nous utiliserons le langage UML et plus particulièrement des diagrammes de classes pour en illustrer sa structure. Durant ce développement, nous insisterons sur les points qui font de CityGML un système de représentation géographique unique. Enfin, nous tâcherons de relever les forces et les faiblesses de ce modèle afin d'avoir une vue claire et globale de ce que nous pouvons mobiliser dans le cadre d'une intégration avec un autre.

Dans le deuxième chapitre, nous présenterons le modèle IFC. Une fois de plus, nous présenterons son domaine d'origine et les applications pour lesquelles il a été conçu. Nous verrons ensuite comment se structure son modèle de représentation des données en se basant sur la spécification pour la version 4 d'IFC. Celle-ci étant exprimée dans le langage EXPRESS (et EXPRESS-G pour sa version graphique), nous traduirons ces diagrammes en diagrammes de classes du langage UML afin d'avoir un langage commun pour la représentation des deux modèles. Pour terminer, nous tâcherons de mettre en évidence les forces et faiblesses de ce modèle en vue de son intégration avec CityGML.

Une fois ces deux standards présentés, nous examinerons dans le chapitre 3 les correspondances qui peuvent être établies entre les deux modèles, en se concentrant sur la représentation des bâtiments. En se basant sur des résultats déjà existants, nous passerons en revue les entités principales pour identifier celles qui peuvent être mises en relation et celles qui, en revanche, posent problème. Nous mettrons l'accent sur les points les plus difficiles à mettre en correspondance.

Dans le chapitre 4 nous passerons en revue les différentes méthodes d'intégrations que l'on peut trouver dans la littérature scientifique et nous tâcherons de les comparer en mettant en évidences les forces et faiblesses de chacune.

Enfin, le dernier chapitre montrera comment mobiliser cette intégration des modèles CityGML et IFC à travers une série de cas d'utilisation. Nous insisterons tout particulièrement sur les avantages que présente cette intégration à travers différents domaines d'application.

Chapitre 1

CityGML

Dans ce chapitre nous allons présenter les systèmes d'information géographique en nous attardant sur un modèle particulier de représentation 3D de villes : CityGML. Après une introduction à ce type de systèmes, nous parlerons des domaines visés par CityGML. Ensuite nous entrerons dans les détails du modèle en en présentant sa structure et son système de représentation. Nous verrons également ses possibilités d'extension à l'aide de "l'Application Domain Extension". Enfin nous terminerons ce chapitre par une mise en avant des forces et faiblesses de ce type de modèle.

1.1 Les systèmes d'information géographique

Un système d'information géographique (de l'anglais *Geographic Information Systems*, GIS) est [16] une base de données digitale pour laquelle un système de coordonnées spatiales commun constitue le moyen de référence principal. Un tel système doit permettre d'ajouter des données depuis des cartes topographiques, des photos aériennes, des images satellites, et d'autres sources. Ces données doivent alors être stockées soit sous la forme d'images matricielles ou de vecteurs. Ces systèmes doivent permettre d'accéder aux données sous leur forme brute mais également de pouvoir leur appliquer des transformations, de les analyser ou encore d'effectuer des statistiques. La visualisation des données peut se faire sous la forme de cartes, de rapports et de plans. L'intérêt principal des GIS réside donc dans l'agrégation de données de sources très variées qui, grâce à une référence spatio-temporelle partagée, permet d'établir des liens entre ces données pour en faire émerger de nouvelles.

Les GIS sont désormais largement utilisés par les gouvernements, les entreprises et la recherche dans des domaines très variés incluant la gestion des ressources naturelles, la gestion de l'utilisation du territoire, la gestion des installations, l'évaluation de l'impôt, l'analyse de l'immobilier, l'analyse démographique ou encore l'analyse archéologique.

Bien que ces systèmes enregistrent leurs données en 3 dimensions (ou même 4 si l'on tient compte de l'axe temporel), historiquement ces systèmes ne proposaient une visualisation qu'en 2 dimensions car ils visaient à représenter de larges régions, sans détails de petite échelle (comme les bâtiments par exemple). Parmi ces systèmes, nous pouvons notamment citer ArcGIS [23] et QGIS [24].

Désormais, les GIS sont également utilisés pour de la planification urbaine et nécessitent donc de pouvoir représenter des objets en 3D à une échelle de détails plus élevés.

1.1.1 L'Open Geospatial Consortium et GML

L'Open Geospatial Consortium (OGC) [17] est un consortium industriel international de plus de 521 entreprises, gouvernements, agences et universités ayant pour but de développer des standards publics pour les informations spatiales dans toutes sortes d'applications.

C'est dans ce cadre qu'a été développé GML (*Geography Markup Language*), une grammaire XML définie par l'OGC servant de format ouvert d'échange de données géographiques sur Internet. Un document GML est décrit par un schéma GML dont les concepts principaux sont tirés de la norme ISO 19100. Le modèle GML propose de nombreuses primitives parmi lesquels nous pouvons citer [18] :

- les éléments (*Feature* ou "une abstraction d'une phénomène du monde réel" (ISO 19101)) (une route, une personne, une zone administrative, etc.)
- la géométrie (points, lignes, polygones ainsi que leurs compositions)
- la topologie (qui renseigne les connexions entre les différents éléments géométriques)
- les informations temporelles et les caractéristiques dynamiques (évolution d'un élément au cours du temps)
- les systèmes de coordonnées de référence
- les unités de mesures, les mesures (longueur, angle, etc.) et les valeurs
- les directions (vecteurs de direction, points cardinaux)
- les observations (modélisation de l'acte d'observer)
- les couvertures (une correspondance entre un domaine spatio-temporel et des valeurs où le type des valeurs est identique pour toutes les positions géographiques au sein du domaine spatio-temporel)

GML est très modulaire puisqu'il permet l'utilisation de *profils*, chacun utilisant un sous-ensemble des primitives du modèle complet. En outre, GML prévoit la possibilité de définir des schémas d'application permettant d'étendre le modèle à d'autre domaines d'application.

1.2 Le modèle CityGML

CityGML est un modèle ouvert de représentation de données spatiales, ayant pour but le stockage et l'échange de modèles 3D de villes (bâtiments, ponts, végétation, tunnels, etc.). Il est produit par l'Open Geospatial Consortium et est implémenté comme un schéma d'application de GML. [19] Ainsi, il hérite du point de vue des GIS. Étant généralement destinés à répertorier l'existant, les modèles 3D de ce type de systèmes sont souvent issus de scans des surfaces observables (depuis le sol mais également depuis des prises de vues aériennes), les différents objets sont donc représentés en *surfaces*. Au-delà d'une représentation purement géométrique (comme dans GML), CityGML ajoute également une couche sémantique aux données.

Les domaines d'applications ciblés explicitement par CityGML sont la planification de villes, le design architectural, les activités touristiques et de loisirs, les simulations environnementales, les télécommunications mobiles, la gestion des désastres, la sécurité de l'intérieur, la gestion de l'immobilier, la navigation des véhicules et des piétons et les simulations d'entraînement. [1]

CityGML est désormais utilisé comme modèle 3D pour de nombreuses villes en Europe et en Asie, avec différents niveaux de détails. En effet, il existe des modèles CityGML 3D pour Berlin, Cologne, Dresden, Munich, Paris et les centres villes de Aix-en-Provence, Lille, Nantes et Marseille. CityGML est également utilisé comme format d'échange de données spatiales à Monaco, Genève, Zurich, Leewarden et dans certaines villes du Danemark. Pour l'Asie, il s'agit d'Istanbul, Doha, Katar et Yokohama. Il faut noter également que CityGML joue un rôle important pour l'infrastructure des données spatiales 3D en Malaisie et au Pays-Bas. (CityGML Standard section 0.2) [1] CityGML est donc un modèle de représentation très répandu, renforçant ainsi son importance en tant que standard.

CityGML étant conçu pour modéliser des villes entières, il permet de représenter la plupart des éléments qui les composent. Il permet ainsi de représenter :

- La topologie du terrain
- Les bâtiments
- Les ponts
- Les tunnels
- Les routes
- La végétation (aussi bien par zones, volumes, que des objets isolés)
- Les zones d'eau
- Les zones de transport
- Les utilisations de terrain (Une zone dédiée à une utilisation spécifique)
- Les ameublements que l'on peut retrouver dans une ville (aussi bien en intérieur qu'en extérieur)

Le modèle de données CityGML est décomposé en plusieurs modules : Le module

principal (core module) et les modules d'extension qui correspondent aux différents éléments cités précédemment. Ainsi, une implémentation se conformant au standard n'est contrainte qu'à implémenter le module principal. Les 13 modules d'extension pourront être employés en fonction du domaine de l'application. Notons que l'ensemble du module principal et d'une partie des modules d'extension constitue un *profil*. Les modules d'extensions sont :

- Appearance
- Bridge
- Building
- CityFurniture
- CityObjectGroup
- Generics
- LandUse
- Relief
- Transportation
- Tunnel
- Vegetation
- WaterBody
- TextureSurface (déprécié)

1.2.1 Les niveaux de détails

Le modèle CityGML utilise un système de représentation reposant sur 5 niveaux de détails (Level of details ou LoD), chacun apportant une visualisation ainsi qu'une sémantique de plus en plus détaillée. Cette découpe en niveaux de détails permet au modèle de s'adapter aux besoins d'une application. En effet, en plus de considérations purement géométriques (plus le niveau de détails est haut, plus il y a de polygones à afficher, et donc à calculer), les niveaux de détails existent également au niveau sémantique. Inutile par exemple pour une application de calcul de surface habitable de charger les données relatives aux meubles à l'intérieur des bâtiments. Les objets sont donc généralisés dans les niveaux inférieurs de détails : l'ensemble des éléments constituant l'extérieur d'un bâtiment seront représentés au niveau 3, mais seront généralisés en un parallélépipède rectangle au niveau 1 de détails.

Les niveaux de détails du standard sont :

- LoD 0 : Les bâtiments sont représentés par des surfaces au sol (2,5D)
- LoD 1 : Les bâtiments sont représentés par des parallélépipèdes rectangles
- LoD 2 : Les bâtiments sont représentés avec leur toit
- LoD 3 : Les bâtiments sont très détaillés avec la représentation des ouvertures (portes, fenêtres)
- LoD 4 : Les détails (meubles, installations intérieures) à l'intérieur des bâtiments sont représentées

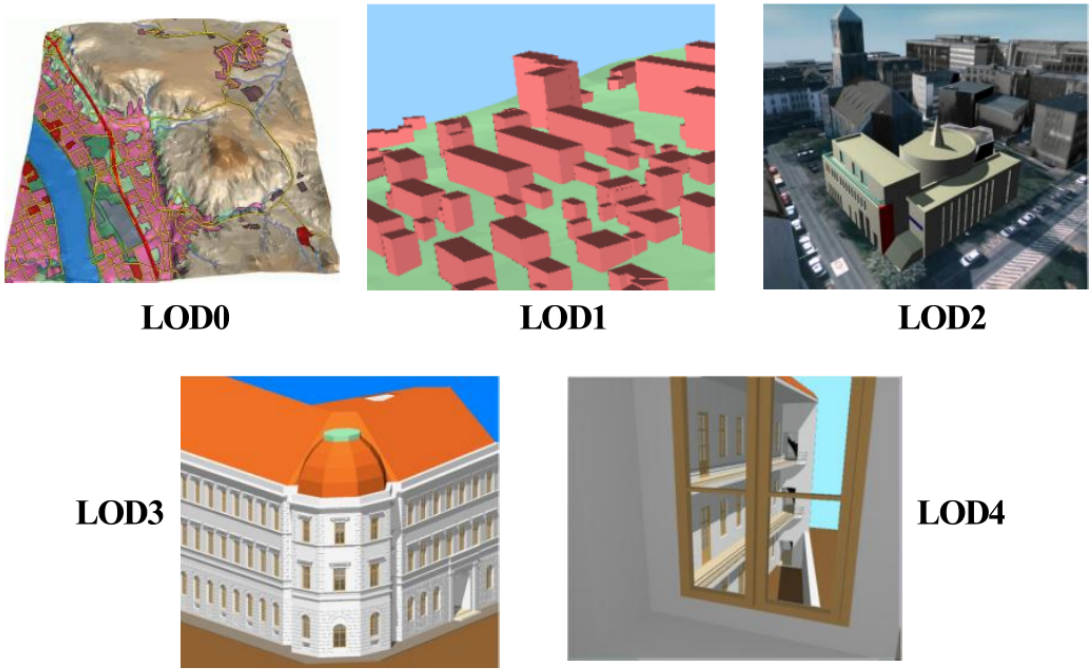

FIGURE 1.1 – Les différents niveaux de détails de CityGML (OGC [1])

1.2.2 Structure du modèle

Dans cette section, nous allons passer en revue la structure du modèle pour sa partie principale (module "core") ainsi que pour la partie concernant la représentation des bâtiments (module "building"). Pour illustrer les explications, nous aurons recours à un exemple fictif venant du document du standard [1] qui est repris dans son intégralité pour le LoD 4 à l'annexe B. Cet exemple consiste simplement en une maison 4 façade sur un terrain plat.

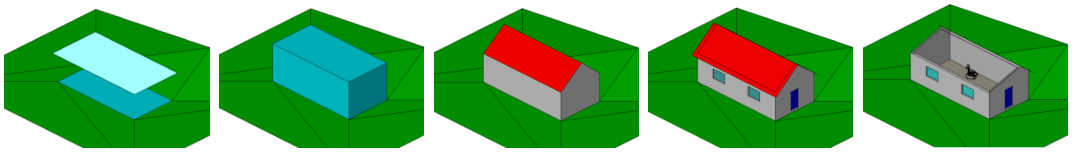

FIGURE 1.2 – Les différents niveaux de détails du bâtiment fictif

Module principal

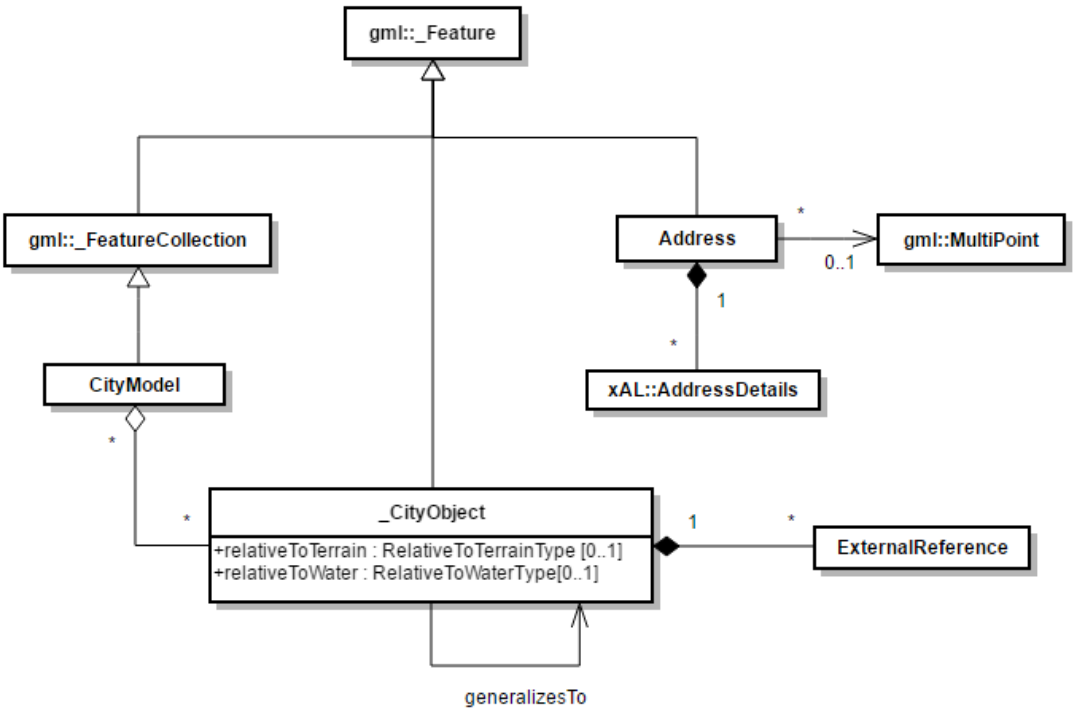

FIGURE 1.3 – Diagramme UML simplifié du module "core" de CityGML (extrait de OGC [1], section 10.1)

Le module *core* définit un *CityModel* comme étant une agrégation de *_CityObject*. Un *_CityObject* est donc un *_Feature* de GML et a pour attributs une date de création et une date de fin. Un *_CityObject* possède également deux attributs *relativeToTerrain* et *relativeToWater* spécifiant la position de l'objet par rapport

au terrain ou à un plan d'eau, ces deux attributs prenant respectivement une valeur dans les énumérations *RelativeToTerrainType* et *RelativeToWaterType*. Enfin, un *_CityObject* possède une association *generalizesTo* vers un autre *_CityObject* représentant sa version généralisée dans un niveau de détails inférieur.

Tout *Feature* possède des attributs *class*, *function* et *usage* représentant respectivement la classe de l'objet, l'usage prévu de l'objet et l'usage effectif de l'objet. La plupart des classes de CityGML utilisent un type *gml :codeType* pour représenter des énumérations pour ces champs. Cela permet de centraliser le sens de ces codes, et donc d'accorder un code unique à une information sémantique donnée. (cf. Annexe C.1 de la spécification [1] pour la liste des codes du module "Building").

Système spatial de référence

Les GIS venant du domaine de la cartographie géographique, ils ont également récupéré leurs systèmes de coordonnées. En effet, pour pouvoir représenter des espaces du monde réel sur une carte en deux dimensions, il est nécessaire de projeter la surface du globe sur un plan. Si désormais, il existe des représentations en 3 dimensions de la Terre, celles-ci se basent toujours sur des systèmes de coordonnées.

Pour pouvoir représenter un élément géographique sur un plan, il est tout d'abord nécessaire d'avoir un modèle simplifié de la Terre. En effet, la Terre n'est pas une sphère parfaite de par son relief mais peut être approximée par un ellipsoïde. Le centre et les dimensions de cet ellipsoïde est appelé un *système géodésique* ("geodetic datum" en anglais). Un datum définit en fait un ensemble de points sur le globe terrestre servant de référence pour définir cet ellipsoïde. Précisons que ces points sont susceptibles de changer à travers le temps (dû au mouvement des plaques tectoniques par exemple), c'est pourquoi un datum est daté. Citons par exemple les datum *ED50* (European Datum 1950), le *NAD83* (North American Datum 1983) ou encore le *WGS84* (World Geodetic System 1984).

Une fois le système géodésique défini, il est nécessaire de définir un système de coordonnées pour situer un point sur le globe. Nous pouvons distinguer trois systèmes de coordonnées principaux :

- Le système de coordonnées géocentrique qui utilise un système de coordonnées cartésien à 3 axes perpendiculaires centrés sur le centre du globe (*ECEF* pour earth-centered earth-fixed).
- Le système de coordonnées sphérique (ou géographique) qui utilise un angle relatif à un méridien (longitude), à l'équateur (latitude) et une hauteur (généralement par rapport au niveau de la mer ou de la surface du datum). Il s'agit là du système le plus connu.
- Le système de coordonnées cartésien centré sur un point à la surface de l'ellipsoïde et ayant ses axes x et y sur un plan tangeant à cette surface. (*ENU* pour local east north up coordinates ou *local north east down coordinates*).

— Le système de coordonnées cartésien qui aplatit le globe en une surface en deux dimensions et qui est donc utilisé depuis longtemps pour les représentations sur carte. Il s'agit ici de *projections*, pour lesquelles il existe plusieurs types différents. Chaque type de projection présente des déformations pour une propriété (aires, angles, distances) et ainsi sont utilisées à des fins différentes (représentation géopolitique, navigation, etc.).

Enfin, un autre élément important est l'utilisation de *projections* consistant à projeter la surface du globe sur une surface plane. Celles-ci sont utilisées depuis longtemps pour les représentations sur carte et il en existe plusieurs types différents. En effet, aucun type de projection n'est parfait et chacun présente des déformations pour une propriété donnée de la carte(aires, angles, distances). C'est pourquoi il existe différents types, chacun utilisé à des fins différentes (représentation géopolitique, navigation, etc.). Parmi les différents types de projections existants, citons par exemple les projections de *Mercator* qui préservent les angles, les projections de *Plate Carree* qui préservent les distances égales ou encore les projections de *Mollweide* qui préservent les aires.

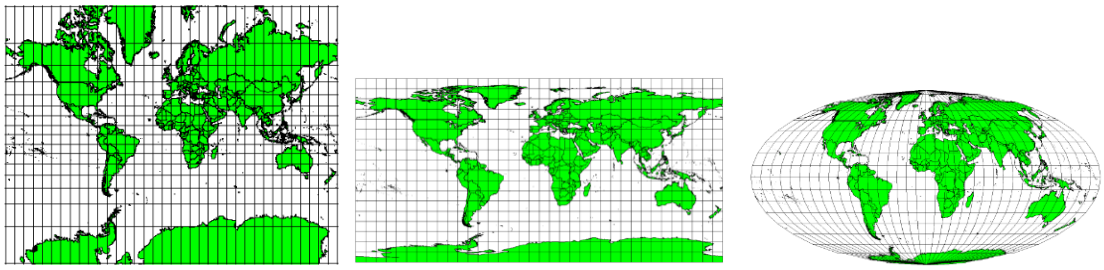

FIGURE 1.4 – Les projections de Mercator, Plate Carree et Mollweide (images extraites de [27])

Un système spatial de référence ou système de coordonnées de référence (*Coordinate Reference System*, CRS) rassemble tous ces éléments : un ellipsoïde, un système géodésique et un système de coordonnées. Notons qu'un tel système est identifié par un identifiant, le *Spatial Reference System Identifier* (SRID), permettant d'interpréter correctement des coordonnées quand il est joint aux données. Il s'agit d'un système primordial dans les GIS puisque ces coordonnées ont le rôle d'identifiant pour les données spatiales.

CityGML hérite de la gestion de CRS de GML3 pour définir une référence spatiale. [1] Tous les éléments d'un modèle sont donc associés à un CRS 3D qui peut être fourni par une organisation faisant autorité comme *l'European Petroleum Survey Group* ou encore être défini localement au sein même du document CityGML. Notons qu'il est également possible de faire référence vers un CRS composé (un CRS pour la référence planaire et un autre pour la référence de hauteur). GML3 permet en outre de définir des CRS d'ingénierie locaux (des CRS qui approximent la courbature de la surface de la Terre en une surface plane) qui sont souvent utilisés dans le domaine AEC/FM et donc utiles pour l'intégration de modèles BIM avec les modèles GIS.

Géométries implicites

Dans le but d'optimiser l'espace requis pour les données répétitives d'un modèle, le concept de géométrie implicite a été défini. Une géométrie implicite est un objet dont la forme n'est stockée qu'une fois mais réutilisée à plusieurs endroits. Pour ce faire, un *ImplicitGeometry* a une association vers la géométrie de l'objet en coordonnées locales et un attribut correspondant à une matrice de transformation vers des coordonnées absolues. Cet objet possède également une association vers un point de référence absolu, le CRS. Ainsi, pour obtenir une occurrence particulière d'un objet, il suffit d'appliquer la matrice de transformation propre à celle-ci à la géométrie "prototype" de l'objet ("prototypic objects"). Ce type d'objet est par exemple employé pour des arbres, des lampadaires, etc.

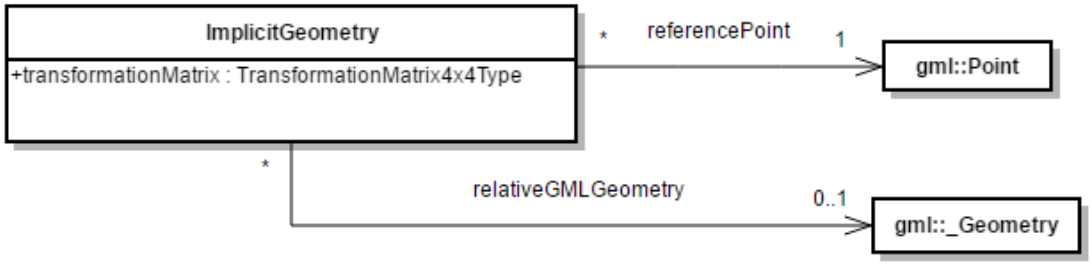

FIGURE 1.5 – Diagramme UML d'une géométrie implicite de CityGML (extrait de OGC [1], section 10.1)

Objets génériques

Le standard définit des "Generic objects" (dans le module "Generics") pour pouvoir représenter des objets qui ne sont repris dans aucun des modules. Tous les *CityObject* peuvent se voir ajouter des attributs génériques lorsque le module *Generics* est employé.

Représentation des objets

Les objets sont représentés comme des volumes fermés, ceci pour pouvoir calculer facilement leur volume (Notons qu'il est possible d'avoir des n-tores). En LoD 2 nous avons des *ClosureSurfaces* pour fermer des volumes qui ne le sont pas dans la réalité (comme un hangar par exemple). Ces surfaces ne sont pas affichées, mais utilisées juste pour les calculs de volume. Ces surfaces devraient également être utilisées en LoD 4 pour séparer des pièces aux fonctions différentes (mais qui n'ont pas de portes par exemple).

Deux volumes peuvent partager une même surface (deux maisons partageant un même mur par exemple). Deux pièces sont considérées adjacentes si elles partagent

des *_Openings* ou des *ClosureSurface*. Cela permet de déterminer un graphe d'adjacence des différentes pièces d'un bâtiment (pour calculer des chemins d'évacuation automatiquement par exemple).

Les installations intérieures non-mobiles d'un bâtiment sont représentées par la classe *IntBuildingInstallation*. Celle-ci peut donc représenter des escaliers, des radiateurs, des tuyaux, etc.

Notons enfin une notion sémantique qui manque au modèle CityGML, celle d'étage. En effet, il n'existe pas d'entité dans le standard permettant de les définir (et donc de les identifier facilement au sein d'un bâtiment). Pour les représenter, le standard recommande d'utiliser des *CityObjectGroups* pour agréger tous les objets (*Room*, *Door*, *Window*, *IntBuildingInstallation*, *BuildingFurniture*) d'une certaine hauteur. Cela impose de fragmenter les objets qui s'étendent sur plusieurs étages (comme les murs par exemple).

Références externes

Le standard CityGML prévoit la possibilité d'ajouter des références vers d'autres systèmes pour un objet donné (du type *_CityObject*). Ainsi, lorsqu'il y a un changement concernant un objet dans le modèle CityGML, nous pouvons répercuter ce changement vers l'équivalent dans l'autre système. Ces références sont exprimées sous la forme d'un URI (Uniform Resource Identifier)

Listing 1.1 – Définition d'une référence externe en CityGML

```
<xs:complexType name="ExternalReferenceType">
  <xs:sequence>
    <xs:element name="informationSystem" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="externalObject" type="ExternalObjectReferenceType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ExternalObjectReferenceType">
  <xs:choice>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="uri" type="xs:anyURI"/>
  </xs:choice>
</xs:complexType>
```

Adresse réelle

CityGML permet de représenter une adresse physique grâce à la classe *Address* de son module principal. Une adresse est une *Feature* composée d'un *AddressDetails* (à

travers l'association *xalAddress*) et d'une association vers un *multiPoint* optionnel spécifiant les positions des entrées correspondant à l'adresse. La modélisation d'une adresse comme une *Feature* permet de pouvoir stocker un lien vers l'information concrète plutôt que l'avoir directement dans le document.

Listing 1.2 – Exemple d'une adresse dans un document CityGML

```
<bldg:address>
   <Address>
      <xalAddress>
         <xAL:AddressDetails>
            <xAL:Country>
               <xAL:CountryName>Germany</xAL:CountryName>
               <xAL:Locality Type="Town">
                  <xAL:LocalityName>
                     Eggenstein-Leopoldshafen
                  </xAL:LocalityName>
                  <xAL:Thoroughfare Type="Street">
                     <xAL:ThoroughfareNumber>1</xAL:ThoroughfareNumber>
                     <xAL:ThoroughfareName>
                        Hermann-von-Helmholtz-Platz
                     </xAL:ThoroughfareName>
                  </xAL:Thoroughfare>
                  <xAL:PostalCode>
                     <xAL:PostalCodeNumber>76344</xAL:PostalCodeNumber>
                  </xAL:PostalCode>
               </xAL:Locality>
            </xAL:Country>
         </xAL:AddressDetails>
      </xalAddress>
      <multiPoint>
         <gml:MultiPoint>
            <gml:pointMember>
               <gml:Point>
                  <gml:pos srsDimension="3">458880.0 5438352.7 112.0
                      </gml:pos>
               </gml:Point>
            </gml:pointMember>
         </gml:MultiPoint>
      </multiPoint>
   </Address>
</bldg:address>
```

Le module Building

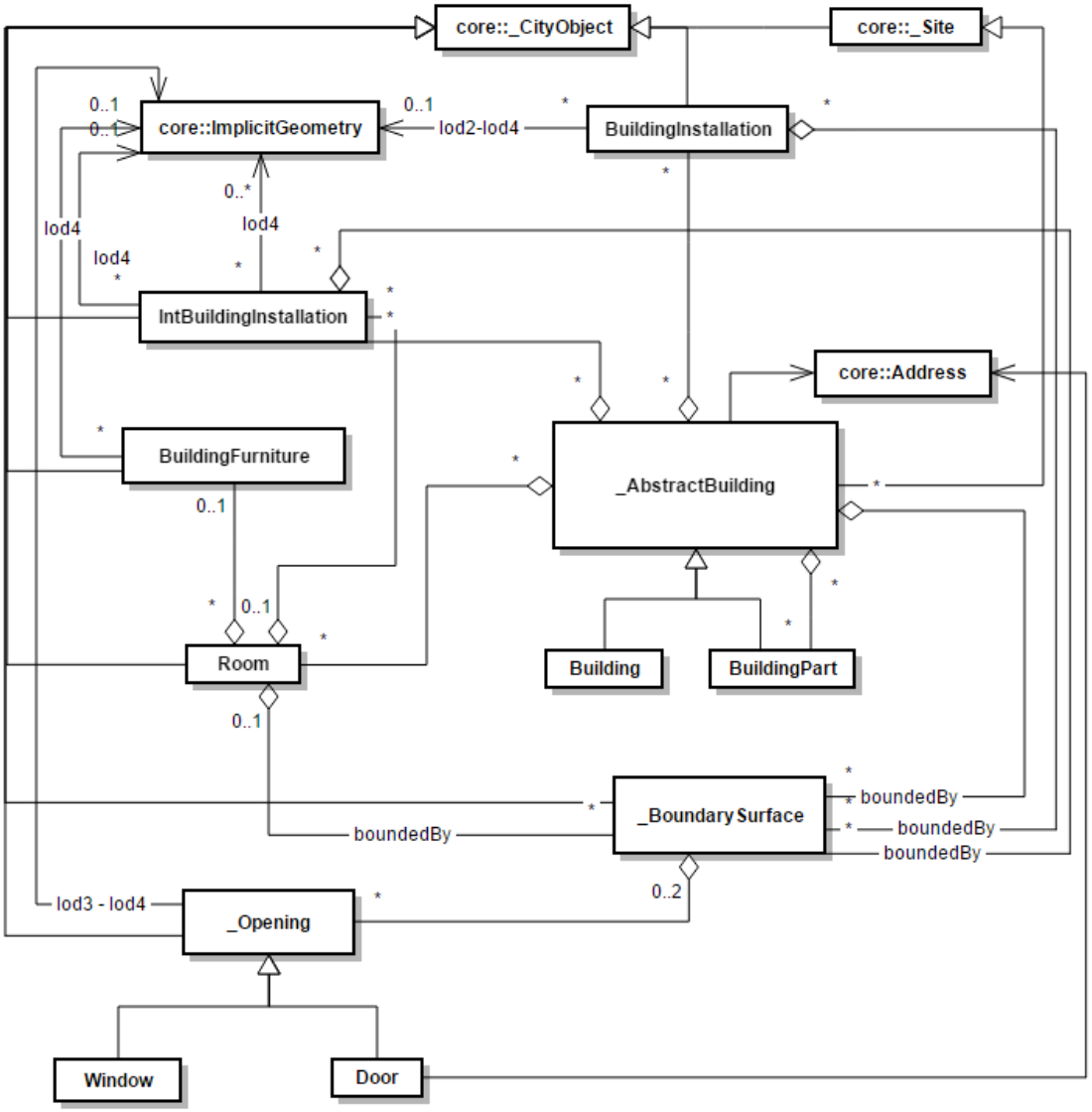

FIGURE 1.6 – Diagramme UML du module "building" de CityGML (OGC [1])

En CityGML, un bâtiment est représenté par la classe *Building* qui peut être constituée de plusieurs *BuildingPart*. Ces deux classes héritent de *_AbstractBuilding* qui est en fait une agrégation de *BuildingInstallation* (les installations extérieures), de *IntBuildingInstallation* (les installations intérieures), de *Room* et de *_BoundarySurface* (les surfaces constituant les limites du volume du bâtiment). Un *_AbstractBuilding* peut être représenté par un *gml : :_Solid* par niveau de détails (hormis LoD 0) et par un *gml : :MultiSurface* par niveau de détails (hormis LoD 0). En LoD 0, l'empreinte du bâtiment est représentée par une *gml : :MultiSurface*. L'intersection du bâtiment avec le terrain est représenté par un *gml : :MultiCurve* quel que soit le niveau de détail (sauf en niveau 0).

Une pièce (*Room*) du bâtiment, représentée par un *gml : :_Solid* en LoD 4, est composée de plusieurs *BuildingFurniture* et est elle aussi délimitée par des *_BoundarySurface*. Une *_BoundarySurface* est composée de plusieurs textit_Opening (portes et fenêtres) et a pour sous-classes :

- RoofSurface (un toit)
- WallSurface (un mur extérieur)
- GroundSurface (le sol au niveau du terrain)
- ClosureSurface (une surface virtuelle pour fermer un volume, cf. 1.2.2)
- CeilingSurface (un plafond)
- InteriorWallSurface (un mur intérieur)
- FloorSurface (le sol d'un étage)
- OuterCeilingSurface (un plafond extérieur)
- OuterFloorSurface (un sol extérieur)

Visuellement, une *_BoundarySurface* sera représentée, uniquement dans les niveaux de détails de 2 à 4, par une *gml : :MultiSurface*. Notons enfin que la notion d'adresse est associée à un *_AbstractBuilding* mais également à une *_Opening* de type *Door*.

Listing 1.3 – Exemple d'une pièce en CityGML

```
<bldg:Room>
   <bldg:lod4Solid>
      <gml:Solid>...</gml:Solid>
   </bldg:lod4Solid>
   <bldg:boundedBy>...(une _BoundarySurface)</bldg:boundedBy>
   ...(autres "bldg:boundedBy")
   <bldg:interiorFurniture>
      <bldg:BuildingFurniture>
         <gml:name>Rocking Chair</gml:name>
         <bldg:function codeSpace="http://www.sig3d.org/codelists/
             standard/building/2.0/BuildingFurniture_function.xml">
            1340
         </bldg:function>
         <bldg:lod4Geometry>
            <gml:MultiSurface>...</gml:MultiSurface>
         </bldg:lod4Geometry>
      </bldg:BuildingFurniture>
   </bldg:interiorFurniture>
</bldg:Room>
```

1.2.3 Application Domain Extensions

L'extension du modèle CityGML avec de nouvelles propriétés ou de nouveaux types d'objets est possible grâce aux Application Domain Extensions (ADE). Une extension doit être définie dans un schéma XML et fournie en annexe du modèle (elle doit être importée dans le fichier). Cela permet au receveur des données de s'assurer de l'exactitude du modèle par rapport à la définition de l'extension. Le gros intérêt des ADE est donc de pouvoir les formaliser et même de les standardiser.

La définition d'ADE est complètement orthogonale aux modules de CityGML, et peut donc être définie sur un ou plusieurs de ces modules simultanément. En outre, un document CityGML peut bien sûr utiliser plusieurs ADE simultanément.

1.2.4 Forces et Faiblesses

Pour terminer ce chapitre consacré à la présentation de CityGML, nous allons passer en revue ses forces et faiblesses afin de mettre en évidence ce qui peut être exploité ou non dans le cadre d'une intégration avec un autre type de modèle. Du côté de ses forces, CityGML est désormais un standard très répandu pour la modélisation de villes (cf. section 1.2) et nous avons vu qu'il peut être facilement étendu à d'autres domaines spécifiques grâce à son mécanisme d'extensions (ADE). Bien qu'il permette la représentation de données sur des échelles très différentes (des villes entières jusqu'aux meubles d'une pièce), son système de niveaux de détails permet d'adapter la quantité d'information d'un document par rapport à l'application visée.

Néanmoins, CityGML montre encore quelques faiblesses dans sa version actuelle. On constate en effet que le système de niveaux de détails pourrait être amélioré (notamment pour avoir des niveaux de détails également pour la représentation intérieure des bâtiments [10]). En outre, certains concepts sémantiques très utiles manquent, tel que la représentation d'étages par exemple. Enfin, la nature même de CityGML en tant que GIS rend difficile l'obtention de données pour l'intérieur des bâtiments. En effet, les données CityGML étant obtenues après leur construction, les méthodes de récupération automatiques se limitent aux aspects extérieurs de ceux-ci, reléguant l'insertion des données relatives à l'intérieur à une méthode manuelle.

Chapitre 2

IFC

Dans ce chapitre nous allons nous intéresser à un autre modèle de représentation de données spatiales 3D : IFC. Nous commencerons par présenter ce que sont les BIM, les problèmes d'interopérabilité qui existent et comment IFC se propose de les résoudre. Nous expliquerons la structure d'IFC et plus particulièrement des parties qui nous serons utiles en vue d'une intégration avec CityGML. Enfin nous conclurons ce chapitre en listant les avantages et inconvénients de ce type de modèle.

2.1 Modélisation des données du bâtiment

Le domaine de la construction nécessite la collaboration de nombreux métiers sur un même projet, et ce tout au long de son existence. La modélisation des données du bâtiment (*Building Information Modeling (BIM)*) est un processus de création et de gestion de la représentation des caractéristiques physiques et fonctionnelles d'un bâtiment. Ce processus donne lieu à des fichiers *BIM* (Building Information Models) qui sont échangés et partagés entre les différents métiers lors de l'évolution du projet (de sa conception à sa démolition).

Avec l'adoption de l'outil informatique dans l'industrie AEC/FM [1] s'en est suivi une multiplication des formats de données, souvent propriétaires, obligeant les différents intervenants d'un même projet à utiliser des logiciels d'un même vendeur, ou de vendeurs compatibles. C'est pourquoi un consortium de 12 entreprises américaines (de l'initiative d'Autodesk) à été créé en 1994 avec pour but de créer un ensemble de classes répondant aux besoins d'interopérabilité de l'industrie. Ce consortium d'abords appelé *IAI* (International Alliance for Interoperability), sera finalement renommé *buildingSMART* et donnera lieu aux *IFC*, les Industry Foundation Classes. [20]

1. Architecture/Engineering/Construction (AEC) and Facilities Management (FM)

2.2 Les Industry Foundation Classes

Les Industry Foundation Classes représentent une spécification ouverte pour l'échange et le partage d'information BIM (Building Information Modeling), et l'interopérabilité des applications. IFC est un modèle développé par *buildingSMART* dans le but de décrire les données de construction et de bâtiments, en définissant toute une série d'objets organisés sous forme d'une hiérarchie d'héritages. La spécification d'IFC est écrite à l'aide du langage de définition *EXPRESS*, qui est un modèle entité-relation. [2] Notons que venant du domaine AEC/FM [2], la géométrie est ici représentée par des volumes primitifs représentant les composants structurels d'un bâtiment.

Ce modèle se sérialise en trois formats de fichier :

- Un format texte clé-valeur (STEP-file) qui est le plus courant (d'extension ".ifc")
- Un format XML moins courant dû à son poids élevé (d'extension ".ifcXML)
- Un format zip (qui est en fait le format texte compressé, d'extension ".ifczip")

2.2.1 Structure du modèle

L'architecture du modèle IFC définit quatre couches conceptuelles :

- La couche de ressources : la couche la plus basse, contient les définitions des ressources, mais celles-ci ne disposent pas d'un identifiant global unique.
- La couche principale : contient les définitions des entités les plus générales.
- La couche d'interopérabilité : contient les définitions des entités spécifiques à un produit, un processus ou une spécialisation de ressource utilisé par plusieurs disciplines.
- La couche de domaine : contient les définitions des entités spécifiques à une discipline.

2. Architecture/Engineering/Construction (AEC) and Facilities Management (FM)

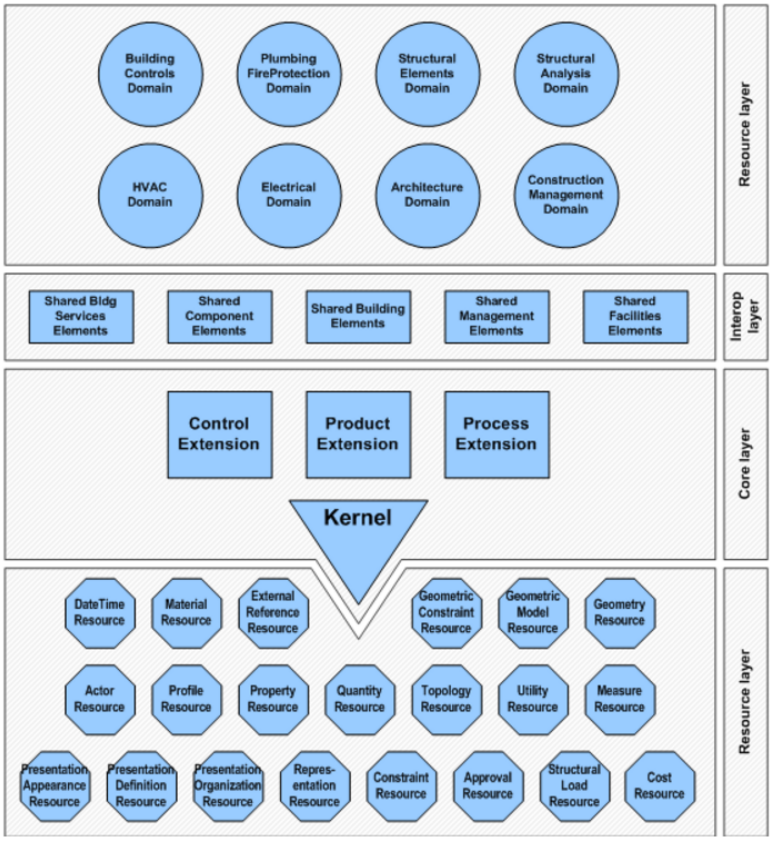

FIGURE 2.1 – Modèle global d'IFC (extrait de la spécification d'IFC 4 [2])

Kernel

Toutes les entités des couches au-dessus de celle des ressources héritent de l'entité "IfcRoot" qui fournit, entre autres, un GUID (Globally Unique Identifier). Tous les objets tangibles (murs, poutres, etc.), physiques (espaces), conceptuels (grilles, limites virtuelles), les tâches de travail, les contrôles (objets de coût), les ressources ou encore les acteurs sont représentés par le type abstrait "IfcObjectDefinition". "IfcObjectDefinition" est ensuite spécialisée en occurences d'objets ("IfcObject"), en types d'objets ("IfcTypeObject"), ou en "IfcContext".

On dénote six types fondamentaux d'entités :

- Les produits : les objets physiques à incorporer dans un projet
- Les processus : les actions prenant place dans le projet (placés en séquences dans le temps)
- Les contrôles : les concepts qui contrôlent ou contraignent les autres objets (spécifications, régulations, exigences, etc.)
- Les ressources : les concepts qui décrivent l'usage d'un objet à l'intérieur d'un processus
- Les acteurs : les personnes participant au projet
- Les groupes : groupes arbitraires d'objets

La plupart des relations entre entités sont représentées sous la forme d'un "IfcRelationship", permettant de découpler la sémantique d'une relation entre objets de ceux-ci. On en dénote six types : Assignation, Association (Réfère à des sources d'information extérieures et les associe aux définitions d'objets ou de propriétés), Décomposition, Définition, Connectivité, Déclaration. La spécification d'IFC étant exprimée au moyen du langage *EXPRESS*, nous avons traduit les associations du type *IfcRelAggregates* (une forme de décomposition) par des associations d'agrégation au sens UML lors du passage à un diagramme de classes.

Modèle des bâtiments

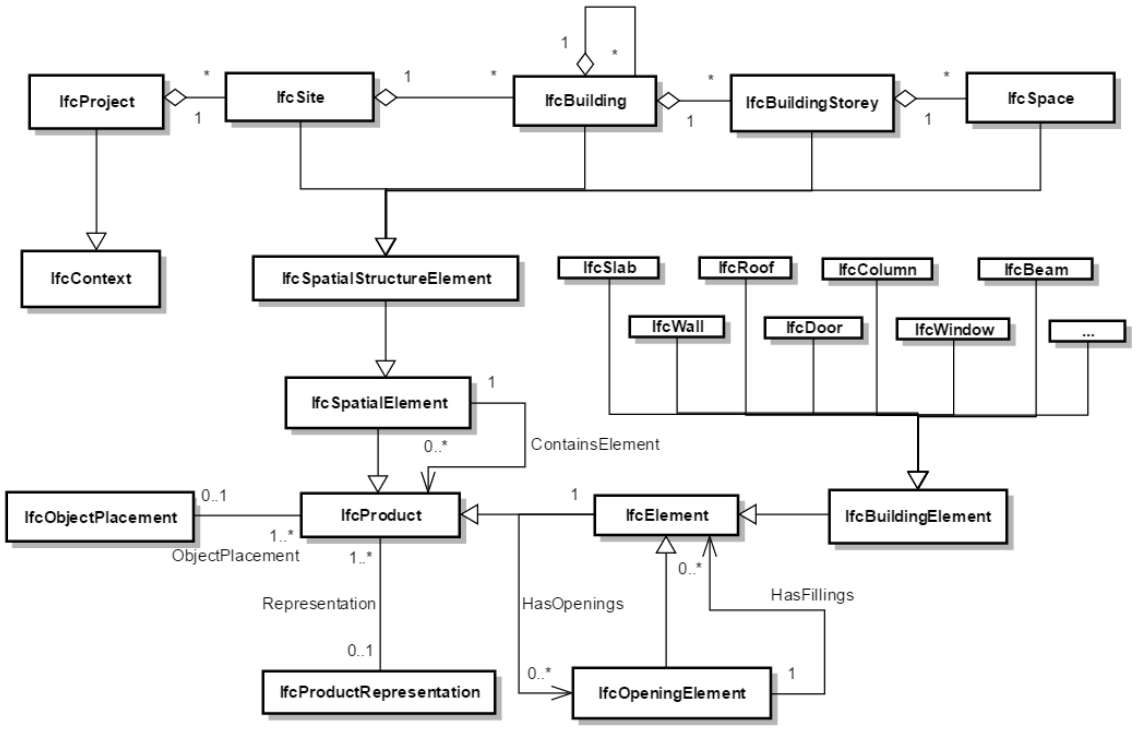

FIGURE 2.2 – Modèle des bâtiments IFC

La modélisation des bâtiments en IFC se fait dans le cadre d'un projet (*IfcProject* qui hérite de *IfcContext*). Un projet dispose d'une agrégation de *IfcSite*, chacun pouvant disposer de plusieurs bâtiments (*IfcBuilding*). Un *IfcBuilding* peut être composé de plusieurs *IfcBuilding* ainsi que de plusieurs étages (modélisés par la classe *IfcBuildingStorey*), chacun disposant de plusieurs pièces (*IfcSpace*).

Les classes *IfcBuilding*, *IfcBuildingStorey* et *IfcSpace* sont des *IfcSpatialStructureElement* qui est elle-même une sous-classe de *IfcSpatialElement*. Un *IfcSpatialElement* peut contenir plusieurs éléments de la classe très générale *IfcProduct* dont elle est elle-même héritière.

La classe *IfcProduct* a également pour sous-type la classe *IfcElement* qui se spécialise

en *IfcBuildingElement* (murs, fenêtres, toits, poutres, etc.) ainsi qu'en *IfcOpeningElement*. Un *IfcElement* peut posséder plusieurs ouvertures, et une ouverture (*IfcOpeningElement*) peut être comblée par plusieurs éléments (de type *IfcElement*, une fenêtre par exemple). Enfin, tout produit (*IfcProduct*) peut posséder une représentation (*IfcProductRepresentation*) ainsi qu'un emplacement (*IfcObjectPlacement*). Cet emplacement doit être fourni pour tout produit disposant d'une représentation et peut être donné de manière :

— absolue (par rapport au système de coordonnées du "monde")
— relative (par rapport à l'emplacement d'un autre produit)

Notons qu'un site est localisé selon le standard *WGS84*[3] à l'aide d'attributs *RefLatitude*, *RefLongitude*, *RefElevation*. Ainsi, tous les objets d'un site peuvent être positionnés à l'échelle géographique en se référant aux coordonnées globales de ce dernier.

2.3 Forces et faiblesses

Pour conclure ce chapitre consacré au modèle IFC, nous allons mettre en avant ses forces et faiblesses. Les Industry Foundation Classes venant du domaine de la construction, le niveau de détails des intérieurs sont très élevés non seulement du point de vue des possibilités offertes par le modèle (un grand nombre de types d'éléments et pour chacun d'eux, des informations sur leur fonctionnement et leur structure) mais également sur la quantité d'information déjà présente dans les documents. En effet, ces documents étant utilisés pendant tout le cycle de vie du projet par de nombreux métiers différents, les données sont à jour et très complètes. Notons que le modèle IFC permet la représentation de la notion d'étage.

En contrepartie, IFC ne permet pas de représenter des éléments comme des routes, des plans d'eau, etc. Bien qu'il puisse représenter des sites avec plusieurs bâtiments, la représentation de sites à l'échelle d'une ville ne fait pas partie de sa fonction.

3. Le "World Geodetic System 1984" est un standard utilisé en cartographie et en navigation (GPS) comprenant notamment un système de coordonnées pour la Terre

Chapitre 3

Correspondances entre les deux modèles

Dans ce chapitre, nous allons nous intéresser aux correspondances qui peuvent être définies entre le modèle CityGML et le modèle IFC. Nous mettrons en évidences les notions présentes dans l'un et pas dans l'autre, et nous montrerons quelles correspondances sont plus délicates à établir. Nous commencerons donc par parler de la structure principale d'un bâtiment et de ses ouvertures (portes et fenêtres). Nous aborderons ensuite de la structure intérieure d'une pièce et des différentes installations que l'on peut y retrouver. Enfin nous terminerons ce chapitre en montrant une différence importante de structure entre les deux modèles.

Notons que nous nous limiterons à l'étude des bâtiments physiques. Nous écarterons donc dans cette étude les entités qui n'y sont pas liées aussi bien dans le modèle CityGML (tunnels, ponts, routes, etc.) que dans le modèle IFC (Contrôle, Ressources, Acteurs, etc.)

Plus précisément les parties d'IFC qui vont nous intéresser pour l'intégration avec CityGML sont :

- La *Product Extension*
- La *Shared Building Elements* (les constituants d'un bâtiment : poutres, portes, colonnes, murs, toits, etc.)
- La *Shared Facilities Elements* (les entités décrivant des ameublements : tables, chaises, etc.)
- *IfcArchitectureDomain* (les informations sur les portes et les fenêtres : style, type d'ouverture, etc.)
- *IfcElectricalDomain* (les informations sur l'installation électrique : lampes, prises, câbles, etc.)
- *IfcHvacDomain* (tout ce qui concerne le chauffage et la climatisation d'un bâtiment)
- *IfcPlumbingFireProtectionDomain* (tout ce qui concerne la plomberie ainsi

que la protection contre les incendies)
— *IfcStructuralElementsDomain* (qui définit des éléments structurels supplémentaires par rapport à *IfcSharedBuildingElements*

3.1 Structure principale d'un bâtiment

Le point de départ de notre analyse des correspondances est la notion de bâtiment. En IFC, un bâtiment est représenté par un *IfcBuilding* qui est une agrégation (*IfcRelAggregates*) d'étages (*IfcBuildingStorey*), mais qui fait partie d'une agrégation de bâtiments pour former un site (*IfcSite*). Ici, un site désigne la zone de terrain sur laquelle un projet de construction est en cours. La localisation d'un étage fait référence à celle du bâtiment dont il fait partie, et de la même façon, la localisation du bâtiment se base sur celle du site.

Un étage quant à lui est une agrégation d'espaces *IfcSpace* partageant une même élévation dans le bâtiment. Notons néanmoins qu'un étage peut être découpé en plusieurs étages partiels (et donc d'élévations différentes), par exemple s'il s'agit de paliers.

Pour ce qui est de CityGML, un bâtiment est un *Building*, pouvant être composé de plusieurs *BuildingPart* chacun composé de plusieurs pièces (*Room*). CityGML ne définit pas de notion d'étage, mais recommande d'utiliser des *CityObjectGroups* pour y regrouper des *Rooms*, *Doors*, *Windows*, *IntBuildingInstallations*, *BuildingFurniture* ainsi que des fragments verticaux des surfaces extérieures du bâtiment. Ce groupe se voit assigner :
— la valeur "building separation" comme *class*
— la valeur "lodXStorey" comme *function* (où X est le numéro du niveau de détail pour lequel ce groupe représente un étage)
— la valeur "storeyNo_X" pour la propriété *gml :property* (où X désigner le numéro de l'étage)
Enfin, tous les étages d'un même bâtiment sont regroupés dans un *CityObjectGroup* qui est associé avec l'objet *Building* et avec une valeur "building storeys" comme *class*.

En utilisant ces recommandations du standard CityGML, il est possible d'établir une correspondance entre un étage au sens IFC et un "building separation" de CityGML.

CityGML	**IFC**
Building	IfcBuilding
CityObjectGroup (*class* = "building separation")	IfcBuildingStorey
Room	IfcSpace

3.2 Les ouvertures

En CityGML, les ouvertures sont représentées par la classe abstraite *_Opening* qui est étendue par les classes *Window* et *Door*. Les ouvertures ne sont gérées qu'à partir du troisième niveau de détails. Ces éléments sont associés à une géométrie de type *gml :MultiSurface*. Les fenêtres et portes se distinguent par la présence optionnelle d'une (ou plusieurs) adresse(s) pour une porte et le fait que celles-ci soient prévues, d'un point de vue sémantique, pour le passage de personnes ou de véhicules.

Du côté d'IFC, les portes sont représentées par la classe *IfcDoor* tandis que les fenêtres sont représentées par la classe *IfcWindow*. Toutes deux héritent directement du type *IfcBuildingElement* sans moyen de les distinguer du reste des éléments du bâtiments (colonnes, plafonds, etc.). Néanmoins, une fenêtre peut être associée à une ouverture *IfcOpeningElement* avec la relation *IfcRelFillsElement* (ou l'attribut *FillsVoids* de la fenêtre). Une fenêtre peut également être agrégée en un *IfcCurtainWall* (avec l'attribut *Decomposes*) ou encore être "autonome". Une fenêtre peut également être de type *IfcWindowStandardCase* (un sous type de *IfcWindow*) pour les fenêtres correspondant à un certain profil. Elles doivent notamment être en relation avec un *IfcOpeningElement*, avoir une localisation relative à cette ouverture et avoir une référence vers un *IfcWindowType*. Toutes les fenêtres spécifient leur hauteur et largeur, ainsi que la direction de leur ouverture. La classe *IfcWindowType* définissant une *IfcWindow* ajoute toute une série d'informations, dont notamment :

- Le partionnement de la fenêtre
- Le type de mécanisme d'ouverture (type de poignée, oscillo battant, etc.)
- La position des charnières
- Le type de matériaux utilisés dans sa conception

Les portes quant à elles sont définies comme étant prévues pour l'accès de personnes et de biens. Leur définition est parallèle à celle des fenêtres et nous retrouvons ainsi les mêmes possibilités de relations, à savoir une porte comblant un vide (*FillsVoids*), une porte faisant partie d'un agrégat (*Decomposes*), ou encore ni l'un ni l'autre en étant "autonome". Nous retrouvons également un sous-type pour les portes correspondant à un profil, *IfcDoorStandardCase*. Les portes sont définies par un *IfcDoorType* spécifiant notamment :

- Le type de mécanisme d'ouverture (type de poignée, oscillo battant, etc.)
- La position des charnières
- Le type de matériaux utilisés

Avec ces informations, nous voyons aisément que les deux modèles apportent bien des classes sémantiquement équivalentes pour représenter les portes et les fenêtres, mais pas forcément pour les ouvertures. En effet, les ouvertures de CityGML reprennent l'ensemble des portes et des fenêtres du bâtiment alors que certaines portes et/ou fenêtres d'IFC peuvent ne pas être en relation avec une ouverture. De plus, certaines ouvertures peuvent ne pas être "comblées" par une porte ou une fenêtre. Cette cor-

respondance est toutefois très intéressante car IFC apporte beaucoup d'informations sur la structure et le fonctionnement de ces entités là où CityGML se contente de les identifier.

3.3 Structure intérieure d'une pièce

Avec CityGML, une pièce est représentée par un objet de type *Room* dont le volume doit être fermé (à l'aide éventuellement de *ClosureSurfaces*) et est décrit soit par un solide *gml :_Solid*, soit par un ensemble de surfaces *gml :MultiSurface*. Les surfaces visibles d'une pièce peuvent être représentées par des *BoundarySurfaces*.

L'équivalent pour le modèle IFC est la classe *IfcSpace* qui agrège tous les éléments d'une pièce. Cette classe possède un attribut *BoundedBy* de type *IfcRelSpaceBoundary* définissant la frontière entre deux espaces. Cette frontière peut être physique (auquel cas l'élément créant cette frontière est référencé) ou virtuelle (et un élément virtuel est référencé). L'attribut *PhysicalOrVirtualBoundary* donne cette information. Notons également que chaque espace possède sa propre relation avec l'élément représentant la frontière, et ainsi une géométrie de cette frontière qui lui est propre (Deux espaces peuvent avoir une "vision" différente de leur frontière partagée)

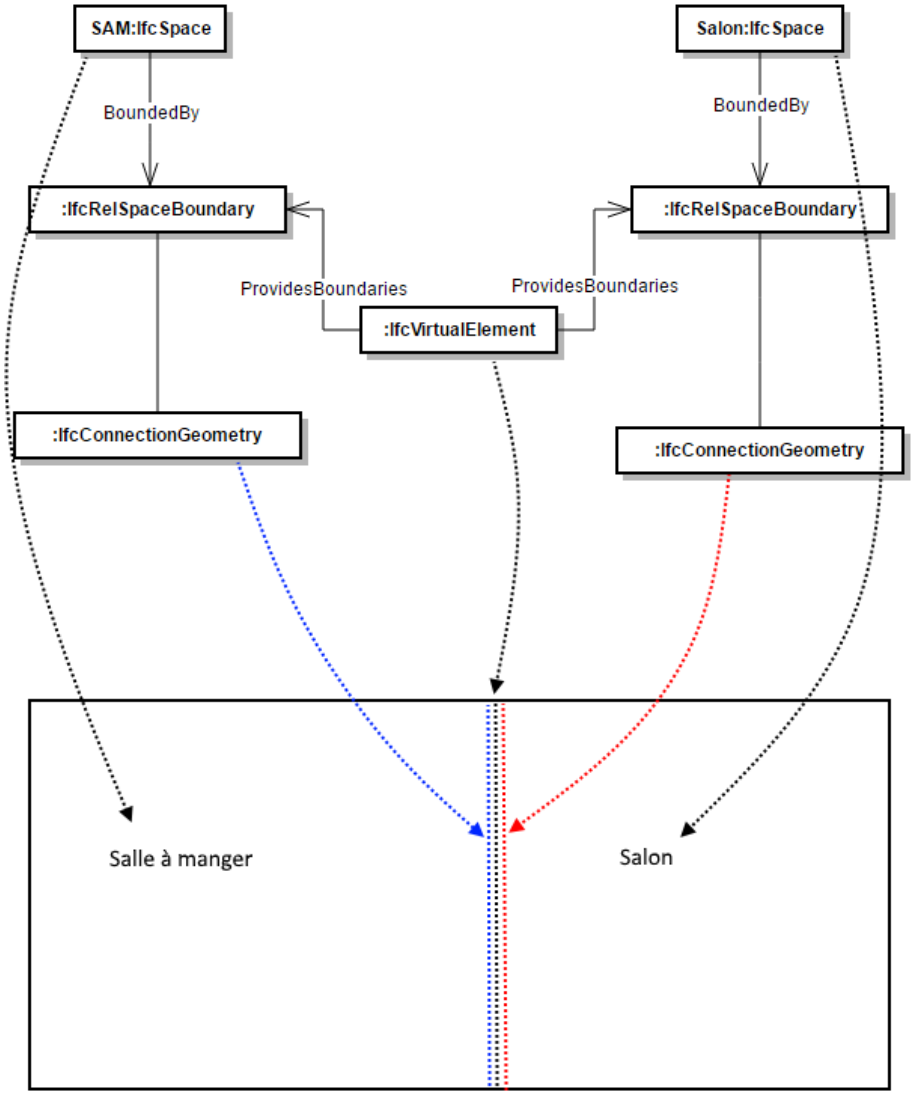

FIGURE 3.1 – Relations entre espaces en IFC (extrait de la spécification IFC [2])

Les murs en IFC sont représentés par la classe *IfcWall* qui peut être spécialisée en *IfcWallStandardCase* (pour les murs dits standards, c'est à dire dont l'épaisseur ne change pas) ou *IfcWallElementedCase* (pour les murs qui sont une agrégation d'éléments). Un *IfcWall* peut posséder des ouvertures *IfcOpeningElement*. Il s'agit donc de l'objet physique pouvant supporter une charge.

En CityGML, nous utiliserons les deux classes *WallSurface* (pour les murs extérieurs) et *InteriorWallSurface* (qui sera réservée pour le LOD4), toutes deux des *_BoundarySurface* pour représenter les murs. Comme pour IFC, les murs (toutes les *BoundarySurface* en fait) peuvent posséder des ouvertures *_Openings* (qui seront donc représentées comme un trou dans la surface).

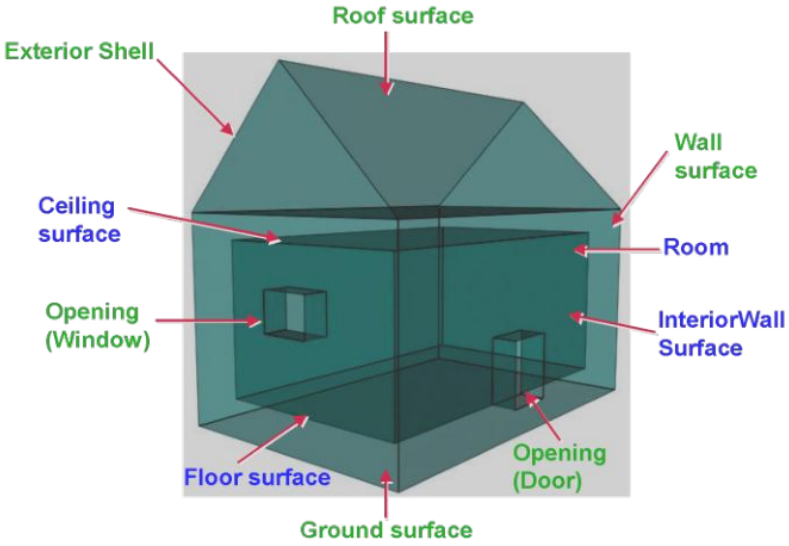

FIGURE 3.2 – Les différentes *BoundarySurfaces* en CityGML (extrait de la spécification CityGML [1])

En ce qui concerne les séparations entre étages, en CityGML nous aurons une *CeilingSurface* pour représenter le plafond depuis la pièce du dessous, et une *FloorSurface* pour représenter le sol de la pièce du dessus. CityGML définit également les classes *OuterCeilingSurface*, *OuterFloorSurface* pour des plafonds et sols extérieurs, et la classe *RoofSurface* pour le toit.

En IFC, nous utiliserons la classe *IfcSlab* pour représenter cette séparation. Celle-ci possède un attribut de type *IfcSlabTypeEnum* indiquant s'il s'agit d'une dalle utilisée pour le sol (*FLOOR*), pour le toit (*ROOF*), une rampe (*LANDING*), ou pour la dalle de base de la maison (*BASESLAB*). Tout comme les murs, les *IfcSlab* peuvent posséder des *IfcOpeningElement*. Notons que le toit du bâtiment est représenté par la classe *IfcRoof*, pouvant contenir des *IfcSlab*.

La représentation des murs et des séparations entre étages constitue un point majeur de divergence entre les deux types de modèles. Il est donc utile d'avoir une méthode de conversion entre la vision en termes de surfaces de CityGML, et d'objets physiques d'IFC. Pour transformer une pièce d'un modèle IFC vers une pièce exprimée en CityGML, nous pouvons utiliser la méthode développée dans l'article [8]. En effet, la section 5.2 de celui-ci présente l'algorithme suivant :

— Tout d'abord, nous réalisons l'intersection entre l'*IFCSpace* (le volume complet intérieur de la pièce) et les *IFCWall* de cette pièce. Nous obtenons ainsi les surfaces intérieures des murs de la pièce (*InteriorWallSurfaces*)
— Nous réalisons le même procédé avec les *IFCSlab* pour obtenir les *CeilingSurface* et *FloorSurface*
— Pour obtenir les surfaces extérieures des murs, il nous suffit de retirer les surfaces intérieures aux polygones des murs.

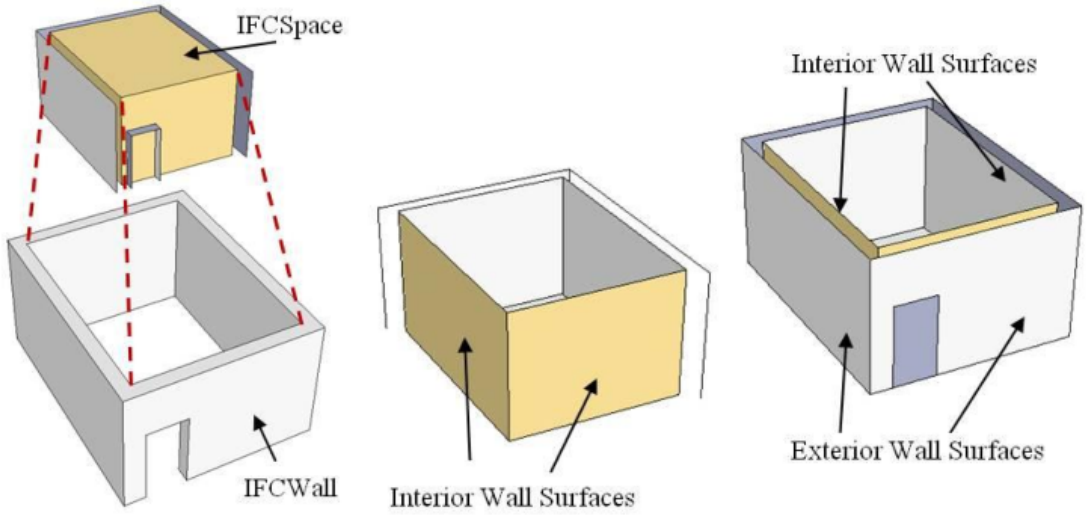

FIGURE 3.3 – Méthode pour extraire les surfaces hors des *IFCSpace* et *IFCWall* [8]

Dans cette même section, nous y trouvons également la procédure à suivre pour extraire des murs et dalles de type IFC à partir de surfaces CityGML. L'algorithme est le suivant :

- nous récupèrons les surfaces (*_BoundarySurface*) d'une pièce ;
- nous prenons une de ces surfaces, et nous créons un parallélépipède rectangle autour de celle-ci, avec une épaisseur plus grande que celle des murs, sols et plafonds ;
- nous réalisons l'intersection de ce volume avec les surfaces voisines. Les surfaces complètement comprises dans ce volume font partie du mur final ;
- nous combinons ces surfaces (6 idéalement) pour former le volume du mur ;
- il est également possible de classifier ce mur en extérieur ou en intérieur selon la présence ou non d'une surface extérieure en son sein.

3.4 Installations intérieures

Une part importante de l'intérêt à établir une correspondance entre des modèles IFC et CityGML est de pouvoir relier les informations relatives aux intérieurs des bâtiments. En effet, bien qu'il soit possible de les représenter en CityGML dans son quatrième niveau de détails, les méthodes de récupération automatiques d'informations sur des bâtiments déjà existants se concentrent sur leur extérieur (photo aériennes, scan lasers depuis la rue, etc.). Ces informations sont donc rarement disponibles. De plus, même quand l'intérieur d'un bâtiment est modélisé dans un document CityGML, le niveau de détail est limité par le modèle. Au contraire, en IFC les installations intérieures sont souvent modélisées car elles sont nécessaires dans la construction du bâtiment. Les modèles IFC servant à de nombreux domaines, les entités utilisées apportent une quantité d'informations non négligeable par rapport à un modèle CityGML.

Dans le reste de cette section, nous allons nous efforcer de détailler les correspondances (concernant les installations intérieures) qui ont été mises en évidences dans l'article [4] comme ayant le plus d'utilité dans un contexte GIS. Les correspondances établies sont les suivantes :

IFC	**CityGML**
IfcBuilding	Building
BuildingAddress	Address
IfcWall	InteriorWallSurface ou Wall-Surface (en fonction de boundaryType)
IfcWindow	Window
IfcDoor	Door
IfcSlab	RoofSurface ou FloorSurface (en fonction de IfcSlabTypeEnum)
IfcRoof	RoofSurface
IfcColumn	Column
IfcFurnishingElement	BuildingFurniture
IfcFlowTerminal	FlowTerminal
IfcSpace	Room
IfcStair	Stair
IfcRailing	Railing
IfcAnnotation	Annotation
IfcBeam	Beam

3.4.1 Les colonnes

Les colonnes peuvent être intérieures ou extérieures. En CityGML nous utiliserons soit l'entité *BuildingInstallation* soit l'entité *IntBuildingInstallation* (selon qu'il s'agit d'une colonne extérieure ou pas) avec l'attribut *function* de valeur "column".

En IFC, l'entité servant à représenter des colonnes est *IfcColumn* qui comme pour tous ces types de classes, peut être représentée par une sous classe *IfcColumnStandardCase* si la colonne correspond à un certain profil. Notons qu'une *IfcColumn* possède un attribut de type *IfcColumnTypeEnum* pouvant prendre les valeurs "COLUMN", "PILASTER" (pour une colonne encastrée dans un mur), "USERDEFINED" ou "NOTDEFINED".

3.4.2 Les meubles

Pour l'ameublement, les deux modèles présentent un niveau de détail semblable. En effet, en CityGML les meubles sont représentés par un *BuildingFurniture* dont les attributs de fonction (*function*) et d'usage (*usage*) peuvent prendre une valeur parmi une liste de 70 types prédéfinis et très variés (allant de la table jusqu'à l'extincteur en passant par la citerne essence ou encore le distributeur de billets). En IFC nous utiliserons *IfcFurniture* (héritant de *IfcFurnishingElement*) avec un attribut de type *IfcFurnitureTypeEnum* pouvant prendre une des valeurs suivantes : *chair*, *table*, *desk*, *bed*, *filecabinet*, *shelf*, *sofa* ou *userdefined*.

3.4.3 Les "terminaux de flux"

En IFC, la classe *IfcFlowTerminal* permet de modéliser tous les terminaux d'un système de distribution (pour la climatisation, l'eau, les égouts, etc.) Nous pouvons en lister les sous-classes suivantes :

- *IfcAirTerminal* (climatisation, chauffage)
- *IfcAudioVisualAppliance*
- *IfcCommunicationsAppliance*
- *IfcElectricApplicance*
- *IfcFireSuppressionTerminal* (extincteur d'incendie automatique)
- *IfcLamp*
- *IfcLightFixture* (lustres)
- *IfcMedicalDevice* (Pour délivrer certains gaz médicaux comme de l'air "médical", du vide, de l'oxygène, du CO2, du nitrogène, du N2O)
- *IfcOutlet* (prises)
- *IfcSanitaryTerminal* (éviers, toilettes)
- *IfcSpaceHeater* (radiateurs)

— *IfcStackTerminal* (grille permettant de filtrer en sortie d'une bouche d'aération ou d'une goutière, par exemple)
— *IfcWasteTerminal*

Bien qu'il n'existe pas de classe strictement équivalente en CityGML, nous pourrons souvent modéliser ce type d'entité en ayant recours à une *IntBuildingInstallation*. En effet, cette classe prévoit les différents usages suivants :

— Radiator
— Oven
— Fireside
— Ventilator
— Air Conditioning
— Pipe
— Lamp
— Light switch
— Power point
— Cable
— Rafter (Arbalétrier [1])
— Column
— Railing
— Stair

3.4.4 Les escaliers

En CityGML nous utiliserons les classes *BuildingInstallation* ou *IntBuildingInstallation* avec l'attribut *function* de valeur "stairs". Du côté d'IFC le type *IfcStair* possède un attribut de type *IfcStairTypeEnum* décrivant la forme des escaliers (présence d'un palier, avec un angle, en spirale, etc.)

3.4.5 Les rampes

En CityGML nous utiliserons la classe *IntBuildingInstallation* avec l'attribut *function* de valeur "railing". Du côté d'IFC le type *IfcRailing* possède un attribut de type *IfcRailingTypeEnum* pouvant prendre les valeurs suivantes : "HANDRAIL", "GUARDRAIL", "BALUSTRADE", "USERDEFINED" ou "NOTDEFINED".

1. Petites poutres composant la charpente du toit

3.4.6 Les poutres

Les poutres sont représentées par la classe *IfcBeam* en IFC. Son attribut de type *IfcBeamTypeEnum* permet d'ajouter de l'information quant à son utilisation. Celui-ci peut prendre les valeurs suivantes :

- BEAM
- JOIST (Une poutre supportant un plafond ou un sol)
- HOLLOWCORE
- LINTEL (Une poutre au-dessus d'une porte ou d'une fenêtre)
- SPANDREL (Une poutre placée en façade d'un bâtiment)
- T_BEAM
- USERDEFINED
- NOTDEFINED

Il n'existe pas de classe permettant de représenter des poutres en CityGML. Le seul moyen de les représenter est d'utiliser une *IntBuildingInstallation* et de définir une valeur pour l'attribut *function* pour spécifier qu'il s'agit d'une poutre.

3.4.7 Les systèmes HVAC [2]

Pour ce type de système, nous utiliserons en CityGML *IntBuildingInstallation* avec un attribut *class* de valeur "Heating, Ventilation, Climate".

IFC quant à lui définit un domaine particulier pour représenter ce type d'équipements dans le schéma spécifique au domaine *IfcHvacDomain*.

2. Heating, Ventilating, Air Conditioning

3.5 Différence de structure

Notons une barrière importante à l'intégration des deux formats soulevée dans [4] ainsi que dans [9] qui est la différence de structure entre les éléments d'un modèle. Là où dans CityGML, les relations entre les différents éléments sont strictes et directement exprimées dans le schéma, IFC autorise une grande libertée puisqu'il revient à l'utilisateur de définir lui-même ces relations entre les différents éléments du modèle.

Par exemple, il est impossible en IFC de connaître le "chemin" de connexions entre une pièce et ses fenêtres en se basant sur la spécification du modèle IFC, il nous faut analyser les données elles-mêmes. En effet le chemin pourrait être une de ces deux versions :

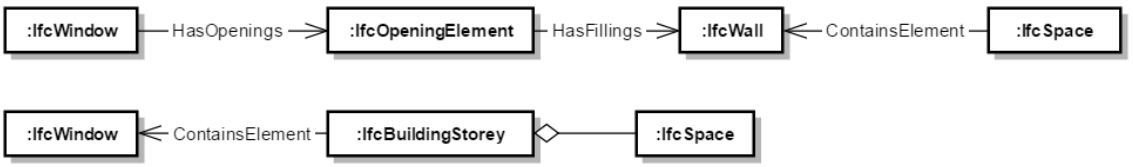

FIGURE 3.4 – Diagrammes d'objets pour une pièce contenant une fenêtre en IFC

En CityGML, le chemin *Window - Wall - Room* est toujours le même car il est directement défini au niveau de la spécification du modèle.

FIGURE 3.5 – Diagrammes d'objets pour une pièce contenant une fenêtre en CityGML

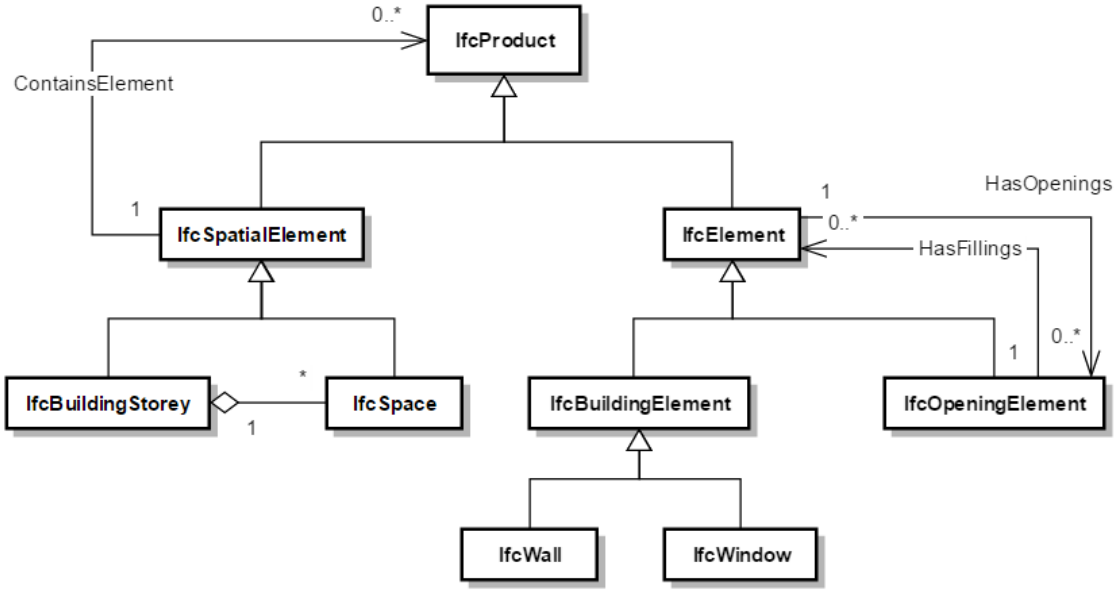

FIGURE 3.6 – Modèle pour une fenêtre et une pièce (Extrait de la spécification IFC, BuildingSMART)

Chapitre 4

Intégration des modèles IFC et CityGML

Dans ce chapitre nous explorerons la question de l'intégration des modèles IFC et CityGML en commençant par lister les différentes méthodes envisagées. Nous terminerons ce chapitre par une comparaison en essayant de mettre en évidence la méthode la plus probante.

4.1 Méthodes d'intégration

L'intégration des formats IFC et CityGML a déjà fait l'objet de nombreuses recherches tant elle promet d'applications. En effet, bien que CityGML soit déjà très utilisé dans de nombreux domaines, aussi bien pour des cas d'utilisation "visuels" que "non-visuels" [12], il montre encore quelques faiblesses qui l'empêchent d'être utilisé à son plein potentiel, notamment son absence de niveaux de détails pour les intérieurs. En dehors du niveau 4 (LOD 4), l'intérieur des bâtiments n'est pas du tout représenté (aussi bien géométriquement que sémantiquement) or les modèles CityGML existants n'offrent que très rarement un tel niveau de détails, ceux-ci se reposant généralement sur de la génération automatique restreignant les détails aux niveaux inférieurs. Si certains chercheurs proposent d'améliorer la situation en permettant la modélisation intérieure pour les niveaux de détails inférieurs [10, 11], la nécessité de le faire "manuellement" restera un frein à leur prolifération.

C'est pourquoi une autre voie de recherche consiste à permettre d'automatiser l'importation de modèles de bâtiments au format IFC (issu du monde BIM) dans des modèles CityGML afin d'augmenter la proportion de modèles de bâtiments détaillés au niveau 4. Cette section détaille l'ensemble des différentes méthodes envisagées pour l'intégration de ces deux formats.

4.1.1 Conversion à sens unique

Une première approche envisagée pour l'intégration des deux types de modèle consiste à convertir un sous ensemble d'un modèle vers l'autre. La restriction à un sous ensemble est en effet nécessaire car certains des concepts présents dans un des modèles n'existent pas dans l'autre, et vice versa.

C'est cette approche qui a été choisie dans le projet "IFC for GIS" (ou IFG) initié par l'autorité de planification de l'état Norvégien (Statens Bygningstekniske Etat) dans le but d'échanger des données entre systèmes GIS et BIM en se reposant sur le modèle IFC. [21] Les exigences du projet étaient :

- le stockage cohérent de données de bâtiments et de données géographiques ;
- le support pour le développement de systèmes d'aide à la planification et à la construction de bâtiments en croisant les données issues de systèmes AEC/FM et GIS.

Pour atteindre ces objectifs, l'approche poursuivie fut de créer un chevauchement entre les deux modèles IFC (dans sa version 2.2) et GML en établissant une correspondance entre les entités dont les occurences et la localisation spatiale pouvaient être identifiées dans les deux. Les entités d'IFC qui ont été retenues sont : *IFCBuilding*, *IFCSpace*, *IFCSite* ainsi que l'historique du bâtiment. Les systèmes de distributions (*IFCSystem* et *IFCDistributionSystem*) ont également été repris et étendus pour tenir compte des systèmes de distribution que l'on rencontre dans les GIS. Les éléments "géographiques" propre aux GIS ont quant à eux été représentés à l'aide de l'entité *IfcGeographicElement*, en utilisant une série de codes pour identifier les différents éléments que l'on peut retrouver dans un document GML. Au final, le projet a pu aboutir à la spécification d'une correspondance entre les géométries d'un modèle IFC étendu et du modèle GML.

Cette méthode de conversion, bien que fonctionnelle, ne constitue pas une méthode d'intégration complète. En effet, en se restreignant ainsi à un sous-ensemble des entités présentes dans l'un des deux modèles, la conversion entraîne une perte d'information inévitable la rendant ainsi à sens unique et limitant donc les applications possibles. De plus, le chevauchement des deux modèles impose de se limiter à des notions purement géométriques (les sémantiques des deux modèles étant très différentes), limitant d'autant plus les applications possibles.

Notons cependant qu'il existe des logiciels permettant des conversions de ce type. Citons notamment :

- IFCExplorer, qui permet une exportation au format CityGML ;
- Feature Manipulation Engine (FME), permettant entre autres des conversions entre CityGML et IFC (Safe Software Inc.)

4.1.2 Utilisation de l'ADE [1]

CityGML proposant un mécanisme d'extension directement dans sa spécification, l'ADE, l'idée de l'exploiter pour définir une extension intégrant les données sémantiques d'IFC au contexte GIS semble être une solution idéale. C'est cette approche qui a été suivie dans l'article [4].

La méthode développée consiste tout d'abord à étendre les types existants de CityGML en leur ajoutant des propriétés venant d'IFC. Ensuite, les classes *Room* et *_AbstractBuilding* sont étendues à l'aide d'un attribut issu d'une nouvelle classe : *VisibleElement*. De cette classe héritent toute une série de sous-classes provenant d'IFC (*Railing, Stair, Column, FlowTerminal*, etc.) permettant ainsi d'étendre le modèle CityGML pour tenir compte des éléments provenant du modèle IFC. Ce modèle étendu est représenté à la figure 4.1.

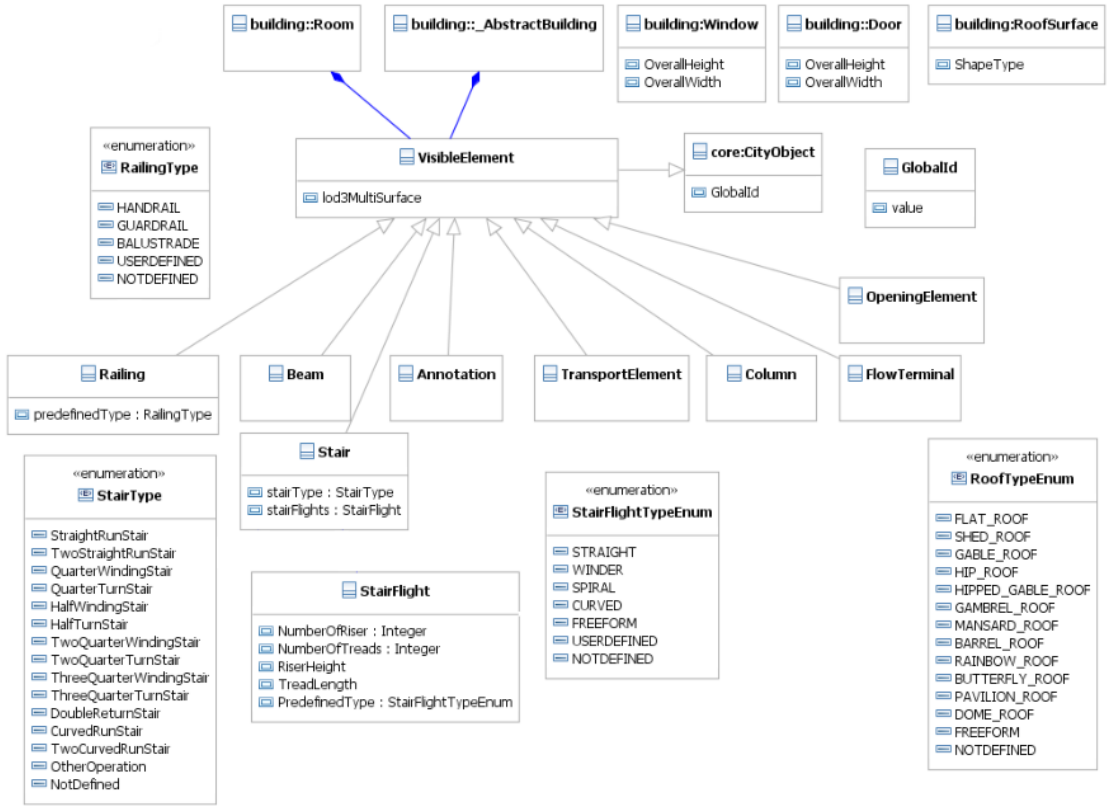

FIGURE 4.1 – Diagramme UML de GeoBIM (extrait de [4])

Cette approche permet donc d'obtenir un modèle CityGML intégrant les concepts d'IFC. Il nous faut toutefois relever un point négatif souligné dans cet article mais également dans [9] : les données géométriques prennent beaucoup plus de place avec le format CityGML dû à son utilisation d'XML (plus de 10 fois plus volumineux qu'un fichier STEP d'IFC). Les fichiers IFC apportant une grande quantité de dé-

1. Application Domain Extension

tails, cela pourrait alourdir le document CityGML au point de le rendre inutilisable pour des données à l'échelle d'une ville.

4.1.3 Utilisation d'ontologies et de Linked Data

Une autre piste d'intégration prometteuse consiste à utiliser des ontologies et le concept de *Linked Data* issu du web sémantique. Le web sémantique est une extension du web ayant pour but d'attribuer un sens aux données présentes sur la toile de telle sorte à ce qu'un programme soit en mesure de les traiter de manière automatique et ainsi de les rendre plus cohérentes et plus intelligentes en en déduisant de nouvelles. Une technique proposée pour créer un web sémantique est d'utiliser des *Linked Data* qui reposent sur 4 principes [22] :

1. l'utilisation d'URI comme identifiant de ressources ;
2. l'utilisation d'URI HTTP pour déréférencer les ressources ;
3. fournir des informations utiles à propos d'une ressource dans des formats ouverts et standards tels que RDF [2], SPARQL [3], etc. ;
4. se référer à d'autres ressources en utilisant leur nom URI HTTP lorsqu'on publie des données sur le web.

C'est ce ce concept de *Linked Data* qui a été appliqué pour étendre CityGML et en particulier son module de Transport dans [5]. La méthode développée consiste à transformer le modèle de CityGML en une ontologie (exprimée en RDF par exemple) et à lui ajouter de nouvelles ontologies pour étendre ses capacités sémantiques. Les liens entre ces différentes ontologies seraient exprimés à l'aide de références sous la forme d'URI, chacun se référant à un élément d'une des ontologies.

Dans cet exemple, une connexion est créée entre une ontologie de mobilité douce et l'ontologie de CityGML grâce à un lien d'inclusion de l'élément *Section* (de cette nouvelle ontologie) dans l'élément *TrafficArea* de CityGML. Cet article montre ensuite comment étendre cette méthode à d'autres domaines comme la qualité de l'air ou encore les sites archéologiques. En se basant sur la géométrie de CityGML, il montre également que la représentation 3D de ses nouvelles informations est possible.

En étendant ce raisonnement à l'intégration de CityGML et d'IFC, nous pourrions créer une ontologie d'IFC et la lier à celle de CityGML en utilisant la méthode précédemment expliquée. Notons que le *BuildingSMART Linked Data Working Group* travaille sur une représentation en *OWL* [4] de l'ontologie d'IFC [28].

2. Resource Description Framework, langage basé sur la définition de *Triple* "*Sujet - Prédicat - Objet*", où chacune de ses parties est identifiée par un URI

3. SPARQL Protocol and RDF Query Language est un langage permettant d'effectuer des requêtes sémantiques sur des données stockées au format *RDF*

4. Web Ontology Language, permettant de définir des ontologies plus complexes qu'en *RDF*

4.1.4 Utilisation d'un modèle intermédiaire

Le modèle UBM

Selon la thèse [6], les deux approches d'intégration tentées jusque-là, à savoir des conversions à sens unique entre les deux modèles en se concentrant principalement sur la géométrie et une simplification d'IFC pour l'intégrer dans les niveaux de détails bas de CityGML, ne sont pas suffisantes.

En effet, ils expliquent que les sémantiques de ces deux types de modèles (et de leur domaine respectif) sont très éloignées. Ils préconisent donc l'utilisation d'une *ontologie de référence*. En définissant une ontologie (en utilisant des langages comme RDF, RDFS ou encore OWL) reprenant les concepts des deux domaines, il est ainsi possible d'obtenir un modèle pouvant servir d'intermédiaire. Le but étant de pouvoir exprimer des liens entre les entités des modèles vers ce modèle intermédiaire et de pouvoir ainsi l'utiliser comme pont.

C'est ainsi qu'ils proposent l'utilisation d'un modèle unifié : l'*Unified Building Model* (UBM). Ce modèle utilise les niveaux de détails de CityGML (voir figure 4.2) et les correspondances suivantes [7] :

IFC	**UBM**	**CityGML**
IfcBuilding	UBMBuilding	_AbstractBuilding
IfcBuildingStorey	UBMStorey	BoundarySurface - RoofSurface - WallSurface - GroundSurface - and other building elements
IfcSpace	UBMSpace - UBMOpenedSpace - UBMClosedSpace	 Room
IfcSlab - (Ground Slab) - (Floor Slab) - (Ceiling Slab)	UBMLevel - UBMGround - UBMFloor UBMCovering - UBMCeiling	 GroundSurface FloorSurface CeilingSurface
IfcRoof	UBMCovering - UBMRoof	RoofSurface
IfcWall - (Exterior Wall) - (Interior Wall) IfcCurtainWall	UBMWall - UBMExteriorWall - UBMInteriorWall UBMWall - UBMCurtainWall	 WallSurface InteriorWallSurface WallSurface
IfcOpeningElement - IfcDoor - IfcWindow	UBMOpening - IfcDoor - IfcWindow	Opening - Door - Window
IfcBeam	UBMBuildingInstallation	BuildingInstallation
IfcColumn	UBMBuildingInstallation	BuildingInstallation
IfcCovering	UBMBuildingInstallation	BuildingInstallation
IfcStair	UBMBuildingInstallation	BuildingInstallation
IfcRailing	UBMBuildingInstallation	BuildingInstallation
IfcRamp	UBMBuildingInstallation	BuildingInstallation

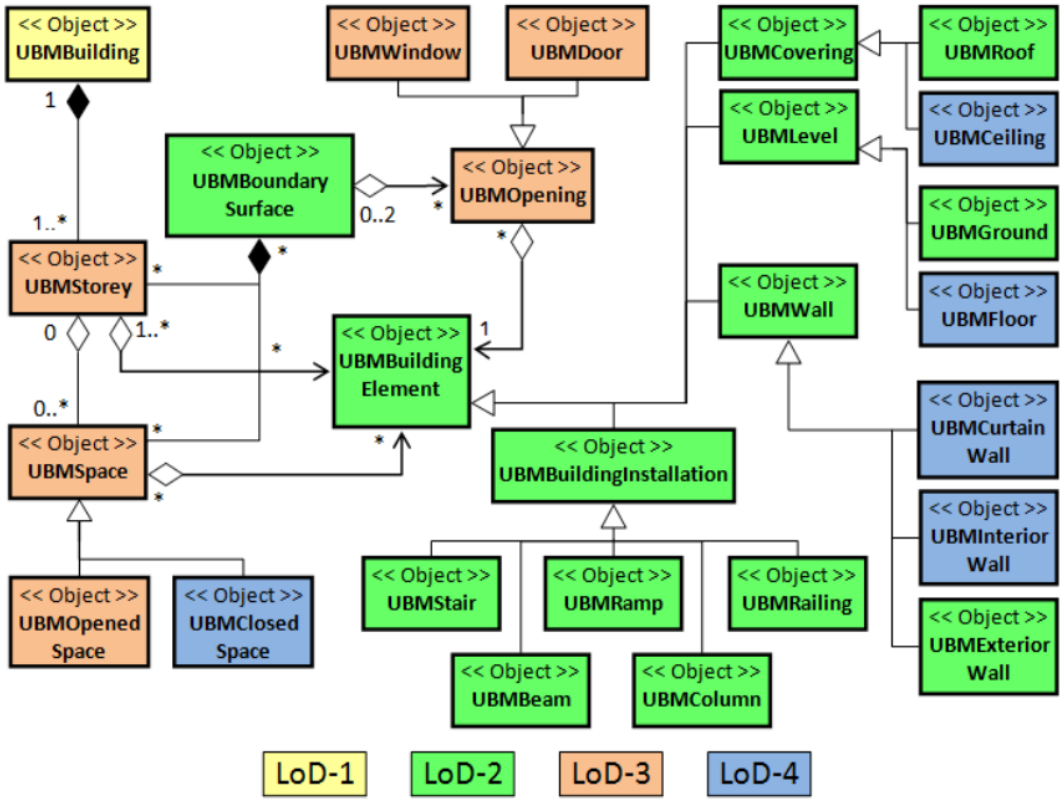

FIGURE 4.2 – Unified Building Model (extrait de [8])

Le modèle CIM

Dans la continuité du modèle "UBM", l'article [3] explique le développement d'un autre modèle intermédiaire : "CIM" (City Information Model). En effet, la méthode de transformation en deux étapes d'UBM (un modèle CityGML est d'abord converti en modèle UBM avant d'être transformé en modèle IFC et inversement) n'explique pas comment réellement intégrer les deux modèles afin de pouvoir gérer des informations venant des deux types de modèles de manière unifiée. De plus, la correspondance sémantique n'est pas démontrée pour UBM. Le modèle CIM est donc développé dans le but de pallier à ces deux manquements.

Leur méthodologie pour concevoir ce modèle consiste à le découper en 5 modules : *Building*, *Transportation*, *CityFurniture*, *MEP*[5] et *Water Body*. Ce modèle intermédiaire s'intéresse donc à un domaine plus large que l'UBM. Ils réalisent ensuite une correspondance entre ces modules et ceux de CityGML (où un module de CityGML peut être découpé pour correspondre à plusieurs modules de CIM) ainsi qu'avec les classes d'IFC.

Malheureusement, l'article n'explique pas plus en détails les correspondances entre

5. Mechanical, Electrical and Plumbing services

les modèles, si ce n'est qu'en ajoutant une référence au tableau 3.4.

4.2 Comparaison et discussion

Au cours de ce chapitre, nous avons passé en revue plusieurs méthodes d'intégration déjà explorées. Nous pouvons classer ces tentatives en 3 catégories :

- une intersection des deux modèles donnant lieu à des conversions à sens unique ;
- une extension du modèle CityGML grâce à son mécanisme d'extension ;
- une union des deux modèles en créant un nouveau modèle intermédiaire ou en utilisant les technologies du web sémantique pour créer des liens entre les données CityGML et les données BIM.

Chacune de ces méthodes présente des avantages et inconvénients.

En ce qui concerne l'utilisation d'une intersection entre les deux modèles, des travaux de recherche (voir [4]) mettent déjà en évidence certains de leurs concepts communs et certains logiciels utilisent ces méthodes pour exporter des données d'un format vers l'autre (avec une perte d'information). Néanmoins, cette approche ne permet pas de tirer parti des avantages d'un modèle dans l'autre dû à cette limitation aux concepts présents dans les deux. C'est pourquoi les logiciels l'utilisant se limitent à exporter uniquement les données géométriques, en se restreignant à un niveau de détails bas. L'objectif premier de cette intégration étant d'obtenir des cas d'utilisation inédits, nous devons donc écarter cette approche.

L'utilisation du mécanisme d'extension de CityGML (l'ADE) pour y intégrer le modèle IFC a pour avantage d'être conçu pour étendre les capacités de CityGML. Il a d'ailleurs déjà été utilisé pour d'autres domaines comme la dispersion du bruit (annexe H de [1]). L'inconvénient majeur de cette approche, outre la nécessité de devoir convertir les données au format IFC vers un format CityGML (avec une extension BIM), est la taille que pourrait prendre les fichiers. En effet, CityGML se plaçant à l'échelle d'une ville, l'utilisation de détails très précis pour l'intérieur des bâtiments (issus des données BIM) pourrait augmenter la taille des fichiers CityGML au point de les rendre inutilisables.

Enfin, les deux dernières méthodes d'intégration vues proposent de créer une union des deux modèles. La première approche consiste à concevoir un modèle intermédiaire comme une union des modèles CityGML et IFC. Les données seraient alors sauvegardées dans ce nouveau format après une conversion. Il est ainsi possible d'exploiter les données géographiques et les données BIM de manière intégrée et donc d'obtenir de nouvelles possibilités d'application. Du côté des points négatifs, nous soulignons la nécessité d'une intégration en deux étapes, il faut en effet toujours passer par ce modèle intermédiaire.

La seconde approche d'unification propose d'exploiter les technologies du web sémantique en définissant des ontologies pour CityGML et pour IFC, dans des langages standards tels que RDF ou OWL. Cette approche, en utilisant une unification des

deux modèles, permet également de tirer parti des forces de chacun. De plus, l'utilisation de *Linked Data* dans des formats standards du web permet d'utiliser les données de manières distribuées sur le web très naturellement (les différents documents seraient localisés par des URL). Notons toutefois qu'il est nécessaire de formater les données (aussi bien CityGML que IFC) pour y intégrer ces liens, ce qui représente un travail conséquent et difficilement automatisable.

Bien que ces trois dernières méthodes d'intégration soient différentes dans leur fonctionnement, le résultat final est identique sur le plan de l'intégration sémantique des modèles. Elles permettent en effet toutes les trois d'établir les liens nécessaires entre les différents concepts des deux modèles, leurs différences résidant dans le format employé pour représenter le modèle complet résultant de cette intégration. Au final, ce sera le domaine d'application qui orientera le choix vers une méthode plutôt qu'une autre, sans autre incidence sur le résultat que la facilité d'implémentation. Dans un domaine se reposant déjà exclusivement sur CityGML et désirant importer un bâtiment IFC de manière sporadique, l'utilisation de l'ADE paraîtra plus naturelle, l'ensemble des données ne devant ainsi pas être converties. Au contraire, dans un domaine où l'utilisation des deux formats se fait déjà de façon régulière et où leurs interactions sont constantes, l'utilisation d'un modèle intermédiaire comme l'UBM pour stocker l'ensemble des données sera probablement plus indiqué. Enfin, s'il s'agit de représenter l'ensemble de ces données sur le web, la méthode reposant sur les technologies du web sémantique se révèlera très efficace en permettant d'en exploiter ses avantages (recherches sous forme de requêtes SPARQL, décentralisation des données entre de nombreux serveurs web, etc.).

Chapitre 5

Exemples d'applications

Dans cette section, nous passons en revue certains exemples de mise en pratique de l'intégration des modèles IFC et CityGML que l'on peut retrouver dans la littérature scientifique, mettant ainsi en évidence l'intérêt d'une intégration pour des cas concrets, en montrant comment obtenir des solutions inédites grâce à cette utilisation simultanée des deux modèles.

5.1 Plan d'évacuation d'un bâtiment

Dans le cadre de la recherche d'un itinéraire d'évacuation, il est primordial de disposer d'une modélisation précise de l'intérieur du bâtiment (portes, fenêtres utilisables, etc.) ainsi que de ce qui l'entoure. Comme le montre l'article [8], l'intégration des modèles GIS et BIM, permet d'obtenir de nouvelles solutions invisibles autrement.

L'article s'attarde principalement sur deux préoccupations pour créer un plan d'évacuation d'un hôpital le plus efficace possible :

- la définition des fenêtres qui peuvent être employées comme des portes en cas d'évacuation ;
- la définition des portes qui donnent sur l'extérieur du bâtiment et de l'usage de l'espace atteint.

Dans l'exemple, l'utilisation de l'UBM permet de détecter des fenêtres qui donnent sur le toit d'un bâtiment voisin, élevant ainsi cette fenêtre au rang de sortie de secours utilisable.

De plus, UBM permet également de répondre à la deuxième question en indiquant pour les portes donnant sur l'extérieur, quelle est l'utilisation du terrain atteint. Il est en effet intéressant de distinguer les routes (qui ne sont pas un espace d'évacuation valable) d'autres types de terrains, plus propices à un scénario d'évacuation.

5.2 Estimation des dégâts d'une inondation

Dans le domaine de l'estimation des dégâts causés par une inondation, l'intégration des modèles BIM et GIS se révèle encore une fois incontournable. L'article [13] montre comment utiliser conjointement le modèle CityGML, pour étudier la hauteur et la vitesse de l'eau, et un modèle BIM pour évaluer l'importance des dégâts sur le bâtiment et sa structure (murs porteurs, poutres, etc.).

L'estimation des dégâts y est décrite comme un processus en trois étapes :
- l'estimation de la santé du bâtiment en analysant l'état de ses structures porteuses ;
- l'estimation du coût total de la réparation ou du remplacement des composants endommagés ;
- la visualisation de la position et de l'ampleur des dommages sur les différents composants.

Une liste des données requises à l'étude est dressée en se basant sur un bâtiment résidentiel Australien typique. Nous y retrouvons notamment la structure globale du bâtiment (périmètre, hauteur, adresse), l'élévation du terrain, la représentation de l'inondation (comme une distribution de points spatio-temporels et comme une surface), la structure interne et externe du bâtiment, les installations électriques, les matériaux dont sont composé le bâtiment et les informations de coût des différents composants du bâtiment.

Un modèle UML intégrant ses exigences aux modèles CityGML et IFC (en tirant parti des points forts de chacun) a ensuite été créé, découpé en 7 modules (*Core, Terrain, Flood, Building, Utility, Valuation, MaterialDomain*). Ce modèle a ensuite été sérialisé en un schéma XML.

La mise en pratique de l'intégration des données consiste à :
- utiliser *ArcGIS* [23] pour convertir les données géométriques du bâtiment d'un format IFC vers des géométries multi surfaces de CityGML (en utilisant les *GenericCityObject*) ;
- extraire les relations sémantiques depuis les fichiers IFC source ;
- faire correspondre les données sémantiques et géométriques en se basant sur un "TAG" identifiant dans les données IFC ;
- faire correspondre les données finales vers le modèle proposé dans l'étude et l'enregistrer dans une base de données.

Les données d'inondation sont obtenues sur base d'une simulation (en partant de données d'élévation de terrain et de position du lit de la rivière de la zone environnante du bâtiment étudié) et sont ensuite converties en données XML pour être elles aussi intégrées au modèle proposé dans l'étude.

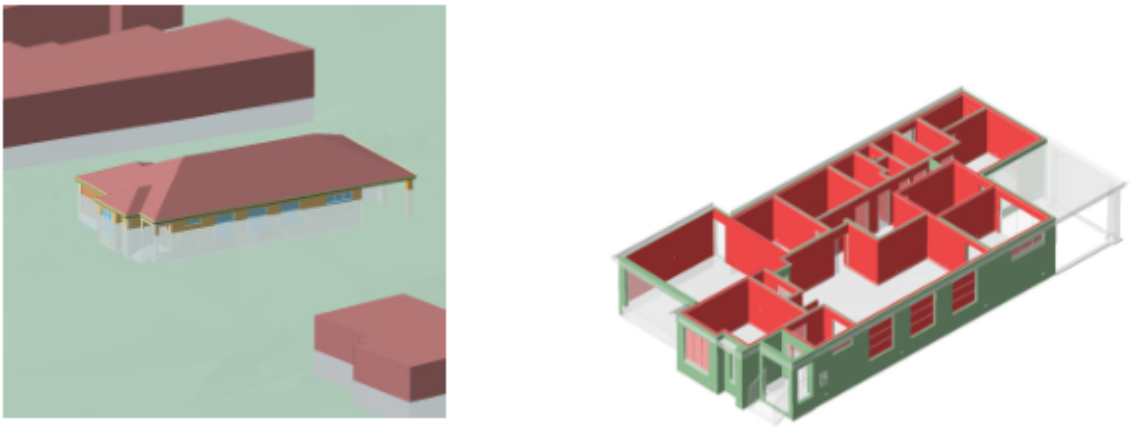

FIGURE 5.1 – Impact d'une inondation sur un bâtiment (image extraite de [13])

5.3 Plan de guidage pour robots

Le Japon, voyant sa population vieillir très rapidement, investit beaucoup dans la robotique pour pallier aux problèmes que cela implique. Le ministère des affaires internes et des communications a lancé un projet de recherche pour des réseaux ubiquitaires de robots (UNR [1]) en 2009. Le but de ce projet est de créer des réseaux de robots afin d'obtenir des services plus performants en les interconnectant. Dans ce cadre, il est important d'avoir une base de données centrale servant les données spatiales de l'environnement entourant ces robots. C'est pourquoi une extension ADE de CityGML permettant d'y intégrer des données venant de CAD fut créée pour en former une base de données cohérentes et accessible en HTTP (OGC CityGML Encoding Standard, Annexe I) [1].

Les robots ont besoin d'une carte indiquant les obstacles (murs, piliers, meubles), le type de sol (pente, bosses) et l'étage de la carte. Le système consiste donc à avoir une base de données centrale intégrant les données CityGML et CAD pour en faire un modèle 3D puis à transformer ces données en cartes 2D découpées sous forme de grille (chaque case indiquant un type de terrain) pour les servir via HTTP aux robots. L'expérience de démonstration du concept fut menée dans un centre commercial, présentant des zones intérieures et extérieures.

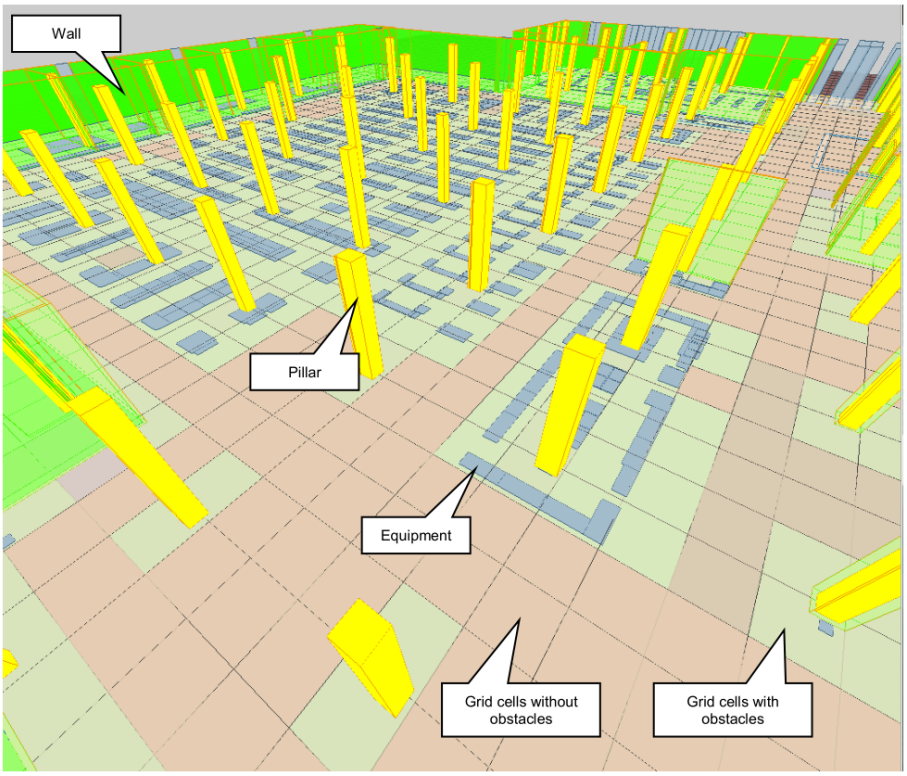

FIGURE 5.2 – Grille indiquant les obstacles (image extraite de [1])

1. Ubiquitous network robot technology

En plus des données déjà disponibles avec CityGML, l'ADE ajoute les concepts de matériaux de surface (*materialType)*), d'étage (*Storey*) et de type d'ouverture pour les portes et les fenêtres (*doorOpereationType* et *windowOperationType*).

5.3.1 Type d'ouverture

Les portes (*bldg : :Door*) et les fenêtres (*bldg : :Window*) se voient respectivement ajouter les attributs *doorOperationType* et *windowOperationType*. Le type est caractérisé par les énumérations *DoorOperationTypeType* et *WindowOperationType*.

Nous pouvons retrouver ces informations directement dans un fichier IFC. En effet nous pouvons mettre en évidence les correspondances suivantes :

UNR ADE	**IFC**
doorOperationType	*IfcDoorType : IfcDoorTypeOperation*
swinging	SINGLE_SWING_LEFT, SINGLE_SWING_RIGHT
double_acting	DOUBLE_DOOR_SINGLE_SWING, DOUBLE_SWING_LEFT, DOUBLE_SWING_RIGHT, DOUBLE_DOOR_DOUBLE_SWING, DOUBLE_DOOR_ SINGLE_SWING_OPPOSITE_LEFT, DOUBLE_DOOR_SINGLE _SWING_OPPOSITE_RIGHT
sliding	SLIDING_TO_LEFT, SLIDING_TO_RIGHT, DOUBLE_DOOR_SLIDING
folding	FOLDING_TO_LEFT, FOLDING_TO_RIGHT, DOUBLE_DOOR_FOLDING
revolving	REVOLVING
rollingup	ROLLINGUP
userdefined	USERDEFINED
notdefined	NOTDEFINED

UNR ADE	**IFC**
windowOperationType	*IfcWindowPanelProperties : IfcWindowPanelOperation*
sidehungrighthand	SideHungRightHand
sidehunglefthand	SideHungLeftHand
tiltandturnrighthand	TiltAndTurnRightHand
tiltandturnlefthand	TiltAndTurnLeftHand
tophung	TopHung
bottomhung	BottomHung
pivothorizontal	PivotHorizontal
pivotvertical	PivotVertical
slidinghorizontal	SlidingHorizontal
slidingvertical	SlidingVertical
removablecasement	RemovableCasement
fixedcasement	FixedCasement
otheroperation	OtherOperation
notdefined	NotDefined

5.3.2 Type de surface

La classe *bldg : :_BoundarySurface* se voit ajouter les attributs : *surfaceMaterialType*, *surfaceJointType*, *surfaceRoofType* et *surfaceOutdoorType*. La classe *bldg : :_Opening* quant à elle se voit ajouter les attributs : *openingMaterialType*, *openingRoofType* et *openingJointType*. À cela viennent s'ajouter les énumérations : *PhysicalMaterialTypeType*, *JointTypeType*, *RoofTypeType*, *InOutdoorTypeType*.

Les différents types énoncés dans l'article manquent de précisions quant à leur fonction, mais nous pouvons néanmoins mettre en évidence certaines classes IFC exploitables. Pour le type de toit, IFC dispose de la classe *IfcRoofType* accompagné de son énumération *IFCRoofTypeEnum*. Celle-ci propose les types de toits suivants : FLAT_ROOF, SHED_ROOF (un toit à une seule pente), GABLE_ROOF (un toit à deux pentes), HIP_ROOF (un toit à 4 pentes), HIPPED_GABLE_ROOF, GAMBREL_ROOF, MANSARD_ROOF, BARREL_ROOF (un toit en demi-cercle), RAINBOW_ROOF (un toit sous forme d'arche gothique), BUTTERFLY_ROOF (un toit à deux pentes mais formant un creux), PAVILION_ROOF (un toit pyramidal), DOME_ROOF, FREEFORM. Les types de rampes (*IfcRampType*) peuvent également se montrer utiles pour des robots à la mobilité réduite. En effet, ce type est accompagné d'une énumération (*IfcRampTypeEnum*) indiquant si la rampe est d'une traite (StraightRunRamp), avec un plat (TwoStraightRunRamp), avec un angle à 90 degrés (QuarterTurnRamp), une spirale (SpiralRamp), etc.

5.3.3 Étage

Pour ajouter le concept d'étage, l'extension ajoute une sous-classe "Storey" à la classe *grp : :CityObjectGroup* comme recommandé dans le standard. Cette nouvelle classe a pour attributs : *heightAboveGround* et *heightToCeiling*.

Encore une fois, nous pouvons exploiter les données issues d'un fichier IFC puisque cette classe "Storey" correspond directement à la classe *IfcStorey* du modèle IFC où son attribut "Elevation" peut être utilisé pour l'attribut "heightAboveGround" de l'extension. L'attribut "heightToCeiling" quant à lui doit être calculé sur base de l'élévation et de la hauteur de l'étage (ce qui correspond à l'attribut "NetHeight" en IFC).

5.4 Gestion de désastre

La coordination d'une mesure d'intervention suite à un désastre demande énormément de réactivité. Dans ce contexte, l'échange et la coordination des données 3D de bâtiments et de leur entourage est capital. Une intégration de ses données permet encore une fois de réaliser des cas d'utilisation inédits. Un bon exemple est l'étude de la mise en place d'un hôpital en urgence près d'un aéroport [14].

Le scénario fictif étudié est celui d'une bombe dans la zone de New-York, entraînant la mise en place d'un centre médical temporaire près d'un aéroport. Le réalisation d'un projet de ce type repose sur 3 étapes clés :

— trouver un bâtiment adéquat près d'un aéroport dans une certaine zone géographique ;
— planifier les espaces requis pour l'hôpital de fortune ;
— concevoir les plans détaillés de la construction.

5.4.1 Trouver un bâtiment adéquat

Le planificateur utilise un client 3D (*LandXplorer CityGML Viewer* dans cet exemple) pour interroger différents services (découverts après avoir interrogé un catalogue) et obtenir une vue globale de l'espace des bâtiments dans une large zone d'intérêt. Ces données sont récupérées au format CityGML qu'elles viennent d'un service BIM ou d'un service CityGML. Il sélectionne ensuite le bâtiment affichant l'espace requis et le sauvegarde dans un catalogue public pour la prochaine tâche.

5.4.2 Planifier les espaces requis pour l'hôpital de fortune

Le planificateur récupère les informations BIM du bâtiment précédemment identifié au format IFC. Il crée ensuite les différents espaces requis dans le bâtiment (salle d'opération, chambres des patients, etc.) puis sauvegarde à nouveau le bâtiment dans le catalogue public.

5.4.3 Concevoir les plans détaillés de la construction

Un troisième planificateur chargé de concevoir les plans détaillés interroge le catalogue public pour obtenir les pièces du bâtiments (au format IFC) et les bâtiments environnants (au format CityGML) afin d'avoir le contexte du bâtiment choisi. Il peut ainsi mettre à jour les plans du bâtiment (en utilisant *MicroStation* de Bentley Systems comme client) avec les détails nécessaires à sa future construction.

5.5 Évaluation de la valeur d'une nouvelle construction

L'évaluation de la valeur d'un bâtiment n'est pas une tâche facile car elle dépend de nombreux facteurs. En effet, si la valeur d'un bâtiment dépend évidemment de sa structure et de sa taille, elle dépend également de facteurs liés à son contexte. Si les entreprises de construction et les cabinets d'architectes disposent de toutes les données du bâtiment lui-même sous la forme de BIM, les données relatives à son contexte font souvent défaut.

Une solution à ce soucis de contextualisation d'un nouveau bâtiment serait une intégration des données BIM de ce dernier au sein de données CityGML de plus grandes échelles. Cette intégration permettrait par exemple d'évaluer la présence de bâtiments voisins et la valeur de l'isolation (thermique et acoustique) avec ceux-ci. On pourrait en effet mesurer l'épaisseur des murs mitoyens et en déterminer la composition. IFC propose à cet effet de décrire la composition matériel d'un mur grâce à l'attribut *IfcMaterialLayerSetUsage* associé à des *IfcMaterial* permettant de décrire des propriétés comme les capacités thermiques (*Pset_MaterialThermal*) ou sa densité et sa porosité (*Pset_MaterialCommon*).

En utilisant les données spatiales couplées à une simulation de la position du soleil, la mise en contexte du bâtiment permettrait également d'en évaluer son exposition solaire. En effet, la position du soleil à tout moment de la journée peut être décrite grâce à ses azimuth [2] et zenith [3] et replacée dans le système de coordonnées de CityGML. L'architecte pourrait alors évaluer la quantité de lumière que le toit reçoit afin d'estimer l'efficacité de panneaux photovoltaïques, des recherches [15] ont d'ailleurs déjà été conduites dans ce domaine, montrant comment mesurer la quantité de lumière atteignant la surface du panneau photovoltaïque.

Il serait également possible d'évaluer la quantité de lumière perçue dans chaque pièce du bâtiment pour en optimiser le confort en changeant l'orientation de la maison si le site de construction le permet ou en choisissant un autre site sinon. Il s'agirait ici encore de simuler la position du soleil, mais d'évaluer la quantité de lumière reçue de certains points clés de la maison (cuisine, chambre à coucher, etc.). Dans le cas d'une ville disposant de points de vues notables (monument, parc, etc.), il serait possible d'évaluer la qualité de la vue depuis les différentes pièces du bâtiment, en utilisant non plus le soleil, mais le point d'intérêt comme cible.

Pouvoir situer le futur bâtiment permet également d'en évaluer les risques naturels. En effet, en localisant les plans d'eau à proximité (module *Waterbody* de CityGML), il est possible de simuler leur débordement et d'évaluer si le nouveau bâtiment se

2. l'angle indiquant la direction du soleil
3. l'angle indiquant l'élévation du soleil

situe ou non dans la zone à risque. De la même façon, cette méthode peut être employée dans les régions à risque pour des feux de forêts (en utilisant le module *Vegetation* de CityGML).

5.6 Gestion des incendies

Pour une réponse optimale à un incendie, certaines informations capitales peuvent être déduites des données spatiales du bâtiment sinistré. Encore une fois, une intégration entre les modèles IFC et CityGML apporte des solutions inédites.

Comme nous l'avons déjà vu, cette intégration permet de trouver des itinéraires d'évacuation hors d'un bâtiment. Ces itinéraires peuvent également être très utiles pour des équipes de secours devant se rendre dans un bâtiment endommagé. En effet, il est possible que les entrées "normales" du bâtiment soient inutilisables, et la possibilité de trouver une entrée "cachée" peut sauver des vies. Un autre point capital pour le contrôle d'un incendie est bien évidemment l'accès aux extincteurs et aux bouches incendies. Pour les premiers, le modèle IFC dispose d'une classe *IfcPlumbingFireProtectionDomain* et le modèle CityGML des codes "fire protection appliance", "fire extinguishing system" qui peuvent être utilisés pour les situer. Pour ce qui est des bouches incendies et points d'eaux, nous retrouvons dans CityGML des plans d'eaux pour incendie ("waterbody for fire-fighting") et des bouches incendies dans les *CityFurniture* en tant que "Hydrant".

Il est également primordial de pouvoir situer le bâtiment par rapport aux brigades incendies les plus proches. Pour cela, nous pouvons recourir aux types "Fire Brigade" des *Room* ou le type "Fire Station" d'un *_AbstractBuilding* en CityGML. La possibilité de voir le bâtiment avant l'intervention permet également d'envoyer le matériel le plus approprié (échelle de bonne longueur, véhicule le plus adapté au terrain avoisinant, etc.) et de prévoir l'évolution de l'incendie en étudiant les bâtiments voisins, et les murs mitoyens (épaisseur, composition).

Chapitre 6

Conclusion

Bien que le modèle CityGML soit de plus en plus utilisé, les méthodes employées pour récupérer des données de ce type entrainent une difficulté à obtenir des modèles de ville avec un haut niveau de détails, ce qui reste un frein dans certaines applications. Une intégration avec les BIM permettrait d'obtenir de nouvelles informations, enrichissant ainsi leur sémantique. L'objectif visé dans ce document était donc d'utiliser le modèle IFC en l'intégrant avec CityGML pour combler les lacunes de ce dernier et ainsi obtenir des applications inédites.

Nous avons donc commencé par présenter les deux modèles en insistant sur leurs forces et leurs faiblesses et en mettant en évidence ce que nous pouvions mobiliser pour chacun dans le cadre d'une intégration. Nous avons donc synthétisé leurs spécifications pour montrer quelles étaient leurs différences majeures. Celles-ci utilisant des formalismes différents, nous avons employé le langage de modélisation UML, et plus particulièrement son diagramme de classes, comme langage commun pour la comparaison des deux modèles.

Dans le troisième chapitre, nous avons pu établir des correspondances entre les deux modèles en se concentrant sur la représentation intérieure d'un bâtiment. Nous avons ainsi démontré l'existence de correspondances entre les deux modèles pour les pièces, les étages d'un immeuble, les fenêtres et les portes. Nous avons également approfondi les résultats de recherches existantes en détaillant les correspondances pour les installations intérieures, et pour chacune de celles-ci comment elles pouvaient être établies. À l'aide du langage UML, nous avons pu en outre illustrer un point soulevé par d'autre chercheurs, problématique pour leur intégration et limitant l'efficacité de celle-ci. Dans le chapitre 4, nous avons synthétisé et comparé plusieurs méthodes d'intégration possibles, rencontrées dans la littérature scientifique, en en listant pour chacune les avantages et les inconvénients.

Enfin, nous avons montré dans le dernier chapitre comment nous pouvions mobiliser l'intégration des modèles CityGML et IFC pour obtenir de nouvelles utilisations,

impossibles autrement. Pour ce faire, nous avons utilisé des exemples déjà existants mais ne tirant parti qu'un des deux modèles, en montrant comment employer l'autre pour améliorer la solution. Nous avons également présenté certains types d'utilisations complètement inédits et exclusifs à cette intégration.

Le domaine de la représentation de données spatiales se révèle être très complexe et les spécifications des modèles CityGML et IFC en sont le reflet. En se limitant à la représentation des bâtiments dans ce document, nous avons pu réduire le cadre de la recherche, mais de nombreux concepts des BIM n'ont pas pu être explorés. En effet, ceux-ci étant à destination de multiples domaines de la construction, leur nombre et leur complexité est très élevé. L'établissement d'un modèle intégré complet demanderait donc beaucoup de temps et l'intervention d'experts de ces différents domaines pour rassurer le modélisateur par rapport à leur sémantique exacte. Pour les mêmes raisons, nous n'avons pas pu explorer les concepts de CityGML en dehors de ceux du module *Building*. Il s'agit là encore d'une piste intéressante qui peut faire l'objet de recherches ultérieures.

Bibliographie

[1] Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, Karl-Heinz Häfele, "OGC City Geography Markup Language (CityGML) Encoding Standard", version 2.0.0, 2012

[2] buildingSMART, "Industry Foundation Classes Specification Standard", version 4, addendum 1, 2015

[3] Xun Xu, Lieyun Ding, Hanbin Luo, Ling Ma, "From building information modeling to city information modeling", *Journal of Information Technology in Construction (ITcon), Special Issue BIM Cloud-Based Technology in the AEC Sector : Present Status and Future Trends*, volume 19, pages 292-307, septembre 2014

[4] Léon van Berlo, Ruben de Laat, "Integration of BIM and GIS : The development of the CityGML GeoBIM extension", *Advances in 3D Geo-Information in Civil Engineering*, pages 211-225, 2011

[5] Claudine Métral, Roland Billen, Anne-Francoise Cutting-Decelle, Muriel van Ruybeke, "Ontology-based approaches for improving the interoperability between 3D urban models", *Journal of Information Technology in Construction 15*, volume15, pages 169-184, 2010

[6] Mohamed El-Mekawy, "Integrating BIM and GIS for 3D city modelling : The Case of IFC and CityGML", PhD Thesis, Royal Institute of Technology (KTH), Stockholm, Suède, novembre 2010

[7] Mohamed El-Mekawy, Anders Östman "Semantic mapping : an ontology engineering method for integrating building models in ifc and citygml", *3rd ISDE digital earth summit*, Nessebar, Bulgarie, 12-14 juin, 2010

[8] Mohamed El-Mekawy, Anders Östman, Ihab Hijazi, "A Unified Building Model for 3D Urban GIS", *ISPRS International Journal of Geo-Information*, volume 1, numéro 2, pages 120-145, 2012

[9] Mohamed El-Mekawy, Anders Östman, Ihab Hijazi, "An Evaluation of IFC-CityGML Unidirectional Conversion", *(IJACSA) International Journal of Advanced Computer Science and Applications*, volume 3, numéro 5, pages 159-171, 2012

[10] Roeland Boeters, Ken Arroyo Ohori, Filip Biljecki, Sisi Zlatanova, "Automatically enhancing CityGML LOD2 models with corresponding indoor geometry",

International Journal of Geographical Information Science, volume 29, numéro 12, pages 2248-2268, 2015

[11] Roland Billen, François Laplanche, Siyka Zlatanova, Ludvig Emgard, "Vers la création d'un méta-modèle générique de l'information spatiale 3D urbaine", *Revue XYZ*, numéro 114, pages 37-42, 2008

[12] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, Arzu Çöltekin, "Applications of 3D City Models : State of the Art Review", *ISPRS International Journal of Geo-Information*, volume 3, numéro 4, pages 2842-2889, 2015

[13] Sam Amirebrahimi, Abbas Rajabifard, Priyan Mendis, Tuan Ngo, "A Data Model for Integrating GIS and BIM for Assessment and 3D Visualisation of Flood Damage to Building", *Locate*, volume 15, pages 10-12, 2015

[14] A. Lapierre, P. Cote, "Using Open Web Services for urban data management : A testbed resulting from an OGC initiative for offering standard CAD/GIS/-BIM services", *Urban and Regional Data Management. Annual Symposium of the Urban Data Management Society*, pages 381-393, 2007

[15] N. Alam a, V. Coors, S. Zlatanova, "Detecting shadow for direct radiation using CityGML models for photovoltaic potentiality analysis", *Urban and Regional Data Management*, CRC Pres, pages 191-196, 2013

[16] Kenneth E. Foote, Margaret Lynch, "Geographic Information Systems as an Integrating Technology : Context, Concepts, and Definitions" , *The Geographer's Craft Project*, Department of Geography, The University of Colorado at Boulder, `http://www.colorado.edu/geography/gcraft/notes/intro/intro.bak16`, 1995

[17] `http://www.opengeospatial.org/ogc`

[18] "OpenGIS® Geography Markup Language (GML) Encoding Standard", version 3.2.1, OGC, 2007

[19] `http://www.opengeospatial.org/standards/citygml`

[20] `http://buildingsmart.org/about/about-buildingsmart/history/`

[21] `http://www.iai.no/ifg/Content/ifg_index.htm`, accédé dans sa version du 12 janvier 2012

[22] `http://www.ted.com/talks/tim_berners_lee_on_the_next_web`, 2009

[23] `http://www.esri.com/software/arcgis`

[24] `http://www.qgis.org/`

[25] `https://www.w3.org/2015/spatial/charter`

[26] `http://www.buildingsmart-tech.org/implementation/get-started/hello-world/example-1`

[27] `http://docs.qgis.org/2.6/en/docs/gentle_gis_introduction/coordinate_reference_systems.html`

[28] `http://www.buildingsmart-tech.org/future/linked-data/ifcowl`

Annexe A

Diagrammes de classes complets de CityGML

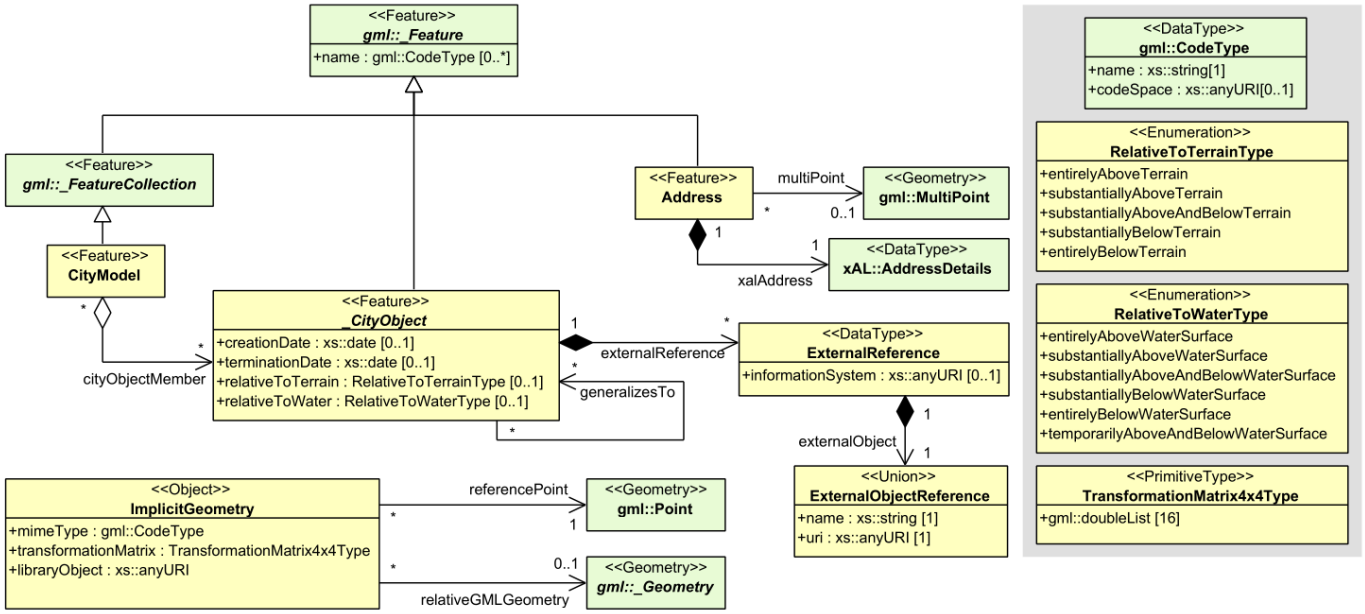

FIGURE A.1 – Diagramme UML du module "core" de CityGML (extrait de OGC [1], section 10.1)

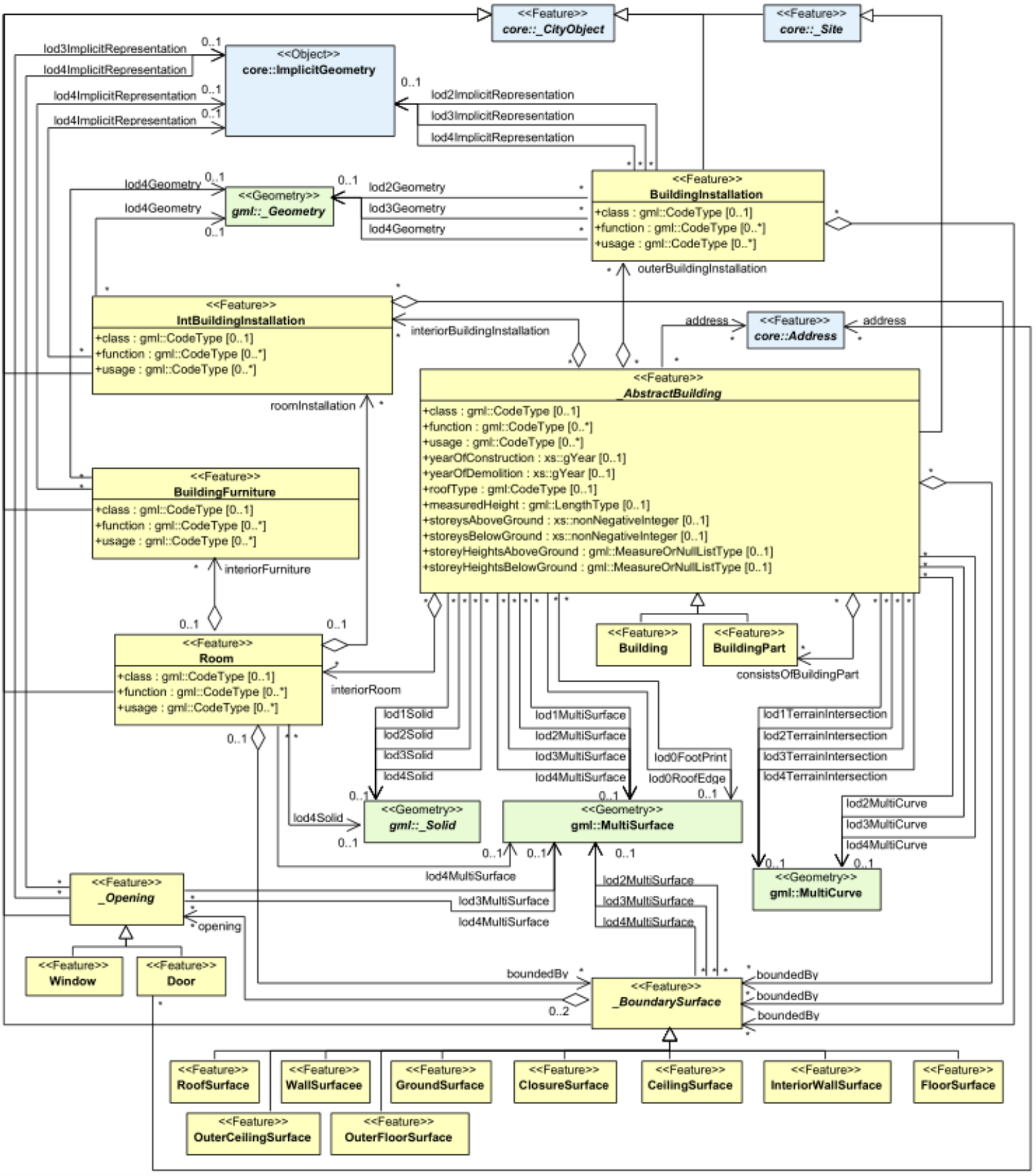

FIGURE A.2 – Diagramme UML du module "building" de CityGML (extrait de OGC [1], section 10.1)

Annexe B

Exemple d'un bâtiment CityGML en LoD 4 [1]

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0"
    xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
    xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0
    http://schemas.opengis.net/citygml/building/2.0/building.xsd
    http://www.opengis.net/citygml/relief/2.0
    http://schemas.opengis.net/citygml/relief/2.0/relief.xsd ">
 <gml:name>Simple 3D city model LOD4 without Appearance</gml:name>
 <gml:boundedBy>
  <gml:Envelope srsDimension="3"
      srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
   <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
   <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
  </gml:Envelope>
 </gml:boundedBy>
 <cityObjectMember>
  <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
   <gml:name>Example Building LOD4 </gml:name>
   <bldg:function
       codeSpace="http://www.sig3d.org/codelists/standard/building
       /2.0/_AbstractBuilding_function.xml">1000
   </bldg:function>
   <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
   <bldg:roofType
       codeSpace="http://www.sig3d.org/codelists/standard/building
```

```
      /2.0/_AbstractBuilding_roofType.xml">1030
   </bldg:roofType>
   <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
   <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
   <bldg:storeyHeightsAboveGround
      uom="#m">3.0</bldg:storeyHeightsAboveGround>
   <bldg:boundedBy>
    <bldg:GroundSurface>
     <gml:name>Ground Slab</gml:name>
     <bldg:lod2MultiSurface>
      <gml:MultiSurface>
       <gml:surfaceMember>
        <gml:Polygon gml:id="GML_d3981803-d4b0-4b5b-969c-53f657594757">
         <gml:exterior>
          <gml:LinearRing>
           <gml:posList>458875.0 5438350.0 112.0 458875.0 5438355.0 112.0
              458885.0 5438355.0 112.0 458885.0 5438350.0
112.0 458875.0 5438350.0 112.0 </gml:posList>
          </gml:LinearRing>
         </gml:exterior>
        </gml:Polygon>
       </gml:surfaceMember>
      </gml:MultiSurface>
     </bldg:lod2MultiSurface>
    </bldg:GroundSurface>
   </bldg:boundedBy>
   <bldg:boundedBy>
    <bldg:WallSurface>
     <gml:name>Wall South</gml:name>
     <bldg:lod4MultiSurface>
      <gml:MultiSurface>
       <gml:surfaceMember>
        <gml:CompositeSurface
           gml:id="GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1">
         <gml:surfaceMember>
          <gml:Polygon gml:id="PolyID10204_1916_571790_369478"/>
         </gml:surfaceMember>
         <gml:surfaceMember>
          <gml:Polygon gml:id="PolyID10205_105_876837_53833">
           <gml:exterior>
            <gml:LinearRing>
             <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0
                112.0 458885.0 5438350.0 115.0
458875.0 5438350.0 115.0 458875.0 5438350.0 112.0 </gml:posList>
            </gml:LinearRing>
           </gml:exterior>
           <gml:interior>
```

```
           <gml:LinearRing>
            <gml:posList>458877.0 5438350.0 114.2 458878.5 5438350.0
                114.2 458878.5 5438350.0 113.2
458877.0 5438350.0 113.2 458877.0 5438350.0 114.2 </gml:posList>
           </gml:LinearRing>
          </gml:interior>
          <gml:interior>
           <gml:LinearRing>
            <gml:posList>458881.5 5438350.0 114.2 458883.0 5438350.0
                114.2 458883.0 5438350.0 113.2
458881.5 5438350.0 113.2 458881.5 5438350.0 114.2 </gml:posList>
           </gml:LinearRing>
          </gml:interior>
         </gml:Polygon>
        </gml:surfaceMember>
        <gml:surfaceMember>...
        </gml:surfaceMember>
       </gml:CompositeSurface>
      </gml:surfaceMember>
     </gml:MultiSurface>
    </bldg:lod4MultiSurface>
    <bldg:opening>
     <bldg:Window gml:id="GML_3b09d6a5-4c24-4847-a8a2-e97475e3de47">
      <gml:name>Window South 1</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
     </bldg:Window>
    </bldg:opening>
    <bldg:opening>
     <bldg:Window gml:id="GML_f75f01cc-c584-4a62-b34a-4a0e2640550d">
      <gml:name>Window South 2</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
     </bldg:Window>
    </bldg:opening>
```

```
    </bldg:WallSurface>
   </bldg:boundedBy>
   <bldg:boundedBy>
    <bldg:WallSurface>
     <gml:name>Wall North</gml:name>... (LoD 2 details)</bldg:WallSurface>
   </bldg:boundedBy>
   <bldg:boundedBy>
    <bldg:WallSurface>
     <gml:name>Wall East</gml:name>
     <bldg:lod4MultiSurface>
      <gml:MultiSurface>
       <gml:surfaceMember>
        <gml:CompositeSurface
            gml:id="GML_6286ffa9-3811-4796-a92f-3fd037c8e668"/>
       </gml:surfaceMember>
      </gml:MultiSurface>
     </bldg:lod4MultiSurface>
     <bldg:opening>
      <bldg:Door gml:id="GML_93096bbb-5155-47fb-ae2c-e2f9327f3007">
       <gml:name>Door East</gml:name>
       <bldg:lod4MultiSurface>
        <gml:MultiSurface>
         <gml:surfaceMember>
          <gml:Polygon gml:id="GML_8f988da9-22d7-41e5-ae94-880afd46a3c9">
</gml:Polygon>
         </gml:surfaceMember>
        </gml:MultiSurface>
       </bldg:lod4MultiSurface>
      </bldg:Door>
     </bldg:opening>
    </bldg:WallSurface>
   </bldg:boundedBy>
   <bldg:boundedBy>
    <bldg:WallSurface>
     <gml:name>Wall West</gml:name>... (LoD 2 details)</bldg:WallSurface>
   </bldg:boundedBy>
   <bldg:boundedBy>
    <bldg:RoofSurface>
     <gml:name>Roof North</gml:name>
     <bldg:lod4MultiSurface>
      <gml:MultiSurface>
       <!-- Roof slab -->
       <gml:surfaceMember>
        <gml:Polygon gml:id="GML_ec6a8966-58d9-4894-8edd-9aceb91b923f">...
            (LoD 2 details)
        </gml:Polygon>
       </gml:surfaceMember>
```

```
      <!-- Roof overhanging -->
      <gml:surfaceMember>
       <gml:Polygon gml:id="GML_70fa738e-80a4-4774-8a3b-322f037fa482">
        <gml:exterior>
         <gml:LinearRing>
          <gml:posList>458874.6 5438352.5 117 458875 5438352.5 117 458875
              5438355 115 458885 5438355 115 458885
5438352.5 117 458885.4 5438352.5 117 458885.4 5438355.312347524
   114.75012198097823 458874.6
5438355.312347524 114.75012198097823 458874.6 5438352.5 117 </gml:posList>
         </gml:LinearRing>
        </gml:exterior>
       </gml:Polygon>
      </gml:surfaceMember>
     </gml:MultiSurface>
    </bldg:lod4MultiSurface>
   </bldg:RoofSurface>
  </bldg:boundedBy>
  <bldg:boundedBy>
   <bldg:RoofSurface>
    <gml:name>Roof South</gml:name>
    <bldg:lod4MultiSurface>
     <!-- Roof slab -->
     <gml:MultiSurface>
      <gml:surfaceMember>
       <gml:Polygon gml:id="GML_b41dc792-5da6-4cd9-8f85-247583f305e3">...
           (LoD 2 details)
       </gml:Polygon>
      </gml:surfaceMember>
      <!-- Roof overhanging -->
      <gml:surfaceMember>
       <gml:Polygon gml:id="GML_db6d8edc-4870-4523-a606-d440f36f8ec8">
</gml:Polygon>
      </gml:surfaceMember>
     </gml:MultiSurface>
    </bldg:lod4MultiSurface>
   </bldg:RoofSurface>
  </bldg:boundedBy>
  <bldg:lod4Solid>
   <gml:Solid>
    <gml:exterior>
     <gml:CompositeSurface>
      <!-- Ground Slab -->
      <gml:surfaceMember
          xlink:href="#GML_d3981803-d4b0-4b5b-969c-53f657594757"/>
      <!-- Wall South -->
```

```
    <gml:surfaceMember
       xlink:href="#GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1"/>
    <!-- Window South 1 -->
    <gml:surfaceMember
       xlink:href="#GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e"/>
    <!-- Window South 2 -->
    <gml:surfaceMember
       xlink:href="#GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea"/>
    <!-- Wall North -->
    <gml:surfaceMember
       xlink:href="#GML_d3909000-2f18-4472-8886-1c127ea67df1"/>
    <!-- Wall East -->
    <gml:surfaceMember
       xlink:href="#GML_6286ffa9-3811-4796-a92f-3fd037c8e668"/>
    <!-- Door East -->
    <gml:surfaceMember
       xlink:href="#GML_8f988da9-22d7-41e5-ae94-880afd46a3c9"/>
    <!-- Wall West -->
    <gml:surfaceMember
       xlink:href="#GML_5cc4fd92-d5de-4dd8-971e-892c91da2d9f"/>
    <!-- Roof Slab North -->
    <gml:surfaceMember
       xlink:href="#GML_ec6a8966-58d9-4894-8edd-9aceb91b923f"/>
    <!-- Roof Slab South -->
    <gml:surfaceMember
       xlink:href="#GML_b41dc792-5da6-4cd9-8f85-247583f305e3"/>
   </gml:CompositeSurface>
  </gml:exterior>
 </gml:Solid>
</bldg:lod4Solid>
<bldg:interiorRoom>
 <bldg:Room>
  <bldg:lod4Solid>
   <gml:Solid>
    <gml:exterior>
     <gml:CompositeSurface>
      <!-- Floor -->
      <gml:surfaceMember>
       <gml:OrientableSurface orientation="-">
        <gml:baseSurface
           xlink:href="#GML_fa89e511-39b2-46de-9a13-9f4621576a46"/>
       </gml:OrientableSurface>
      </gml:surfaceMember>
      <!-- Interior Wall North -->
      <gml:surfaceMember>
       <gml:OrientableSurface orientation="-">
```

```
  <gml:baseSurface
      xlink:href="#GML_592ce9fa-0b98-4225-8d22-20eff4f86fc5"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Interior Wall West -->
<gml:surfaceMember>
 <gml:OrientableSurface orientation="-">
  <gml:baseSurface
      xlink:href="#GML_a9fe597d-c338-43ad-a633-2a0beb273fac"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Interior Wall East -->
<gml:surfaceMember>
 <gml:OrientableSurface orientation="-">
  <gml:baseSurface
      xlink:href="#GML_eaf1db16-56a3-4b86-ae19-2edbb604636f"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Door East -->
<gml:surfaceMember>
 <gml:OrientableSurface orientation="+">
  <gml:baseSurface
      xlink:href="#GML_8f988da9-22d7-41e5-ae94-880afd46a3c9"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Interior Wall South -->
<gml:surfaceMember>
 <gml:OrientableSurface orientation="-">
  <gml:baseSurface
      xlink:href="#GML_a718c157-c948-42cf-a786-0ce61044cff9"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Window South 1 -->
<gml:surfaceMember>
 <gml:OrientableSurface orientation="+">
  <gml:baseSurface
      xlink:href="#GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Window South 2 -->
<gml:surfaceMember>
 <gml:OrientableSurface orientation="+">
  <gml:baseSurface
      xlink:href="#GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea"/>
 </gml:OrientableSurface>
</gml:surfaceMember>
<!-- Ceiling North -->
```

```
        <gml:surfaceMember>
         <gml:OrientableSurface orientation="-">
          <gml:baseSurface
              xlink:href="#GML_989aa5cf-ee07-4fd8-89b6-500a9d5ba8041"/>
         </gml:OrientableSurface>
        </gml:surfaceMember>
        <!-- Ceiling South -->
        <gml:surfaceMember>
         <gml:OrientableSurface orientation="-">
          <gml:baseSurface
              xlink:href="#GML_98841838-ee0b-402f-ba28-64ed61cb10f8"/>
         </gml:OrientableSurface>
        </gml:surfaceMember>
       </gml:CompositeSurface>
      </gml:exterior>
     </gml:Solid>
    </bldg:lod4Solid>
    <bldg:boundedBy>
     <bldg:InteriorWallSurface>
      <gml:name>Interior Wall North</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_592ce9fa-0b98-4225-8d22-20eff4f86fc5">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
     </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
     <bldg:InteriorWallSurface>
      <gml:name>Interior Wall West</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_a9fe597d-c338-43ad-a633-2a0beb273fac">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
     </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
     <bldg:InteriorWallSurface>
      <gml:name>Interior Wall East</gml:name>
      <bldg:lod4MultiSurface>
```

```
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:CompositeSurface
            gml:id="GML_eaf1db16-56a3-4b86-ae19-2edbb604636f">
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
         </gml:CompositeSurface>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
      <bldg:opening>
       <bldg:Door>
        <gml:name>Door East</gml:name>
        <bldg:lod4MultiSurface>
         <gml:MultiSurface>
          <gml:surfaceMember>
           <gml:OrientableSurface orientation="-">
            <gml:baseSurface
               xlink:href="#GML_8f988da9-22d7-41e5-ae94-880afd46a3c9"/>
           </gml:OrientableSurface>
          </gml:surfaceMember>
         </gml:MultiSurface>
        </bldg:lod4MultiSurface>
       </bldg:Door>
      </bldg:opening>
     </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
     <bldg:InteriorWallSurface>
      <gml:name>Interior Wall South</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:CompositeSurface
            gml:id="GML_a718c157-c948-42cf-a786-0ce61044cff9">
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
```

```
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
           <gml:Polygon gml:id="GML_cf0b79ba-f31f-4bae-a10f-5bcc85ce2cf6">
            <gml:exterior>
</gml:exterior>
            <gml:interior>
</gml:interior>
            <gml:interior>
</gml:interior>
           </gml:Polygon>
          </gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
          <gml:surfaceMember>
</gml:surfaceMember>
         </gml:CompositeSurface>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
      <bldg:opening>
       <bldg:Window>
        <gml:name>Window South 1</gml:name>
        <bldg:lod4MultiSurface>
         <gml:MultiSurface>
          <gml:surfaceMember>
           <gml:OrientableSurface orientation="-">
            <gml:baseSurface
                xlink:href="#GML_5e07e2cc-c28c-480e-880f-dfdfe287bb9e"/>
           </gml:OrientableSurface>
          </gml:surfaceMember>
         </gml:MultiSurface>
        </bldg:lod4MultiSurface>
       </bldg:Window>
      </bldg:opening>
      <bldg:opening>
       <bldg:Window>
        <gml:name>Window South 2</gml:name>
        <bldg:lod4MultiSurface>
         <gml:MultiSurface>
          <gml:surfaceMember>
           <gml:OrientableSurface orientation="-">
```

```
            <gml:baseSurface
               xlink:href="#GML_d0ea2b6b-7992-4284-9a20-957a6c5c1cea"/>
           </gml:OrientableSurface>
          </gml:surfaceMember>
         </gml:MultiSurface>
        </bldg:lod4MultiSurface>
       </bldg:Window>
      </bldg:opening>
     </bldg:InteriorWallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
     <bldg:FloorSurface>
      <gml:name>Floor</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_fa89e511-39b2-46de-9a13-9f4621576a46">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
     </bldg:FloorSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
     <bldg:CeilingSurface>
      <gml:name>Ceiling South</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_989aa5cf-ee07-4fd8-89b6-500a9d5ba8041">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
     </bldg:CeilingSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
     <bldg:CeilingSurface>
      <gml:name>Ceiling North</gml:name>
      <bldg:lod4MultiSurface>
       <gml:MultiSurface>
        <gml:surfaceMember>
         <gml:Polygon gml:id="GML_98841838-ee0b-402f-ba28-64ed61cb10f8">
</gml:Polygon>
        </gml:surfaceMember>
       </gml:MultiSurface>
      </bldg:lod4MultiSurface>
```

```
     </bldg:CeilingSurface>
    </bldg:boundedBy>
    <bldg:interiorFurniture>
     <bldg:BuildingFurniture>
      <gml:name>Rocking Chair</gml:name>
      <bldg:function
          codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurnitur
      <bldg:lod4Geometry>
       <gml:MultiSurface>
</gml:MultiSurface>
      </bldg:lod4Geometry>
     </bldg:BuildingFurniture>
    </bldg:interiorFurniture>
   </bldg:Room>
  </bldg:interiorRoom>
  <bldg:address>
   <Address>
    <xalAddress>
     <xAL:AddressDetails>
      <xAL:Country>
       <xAL:CountryName>Germany</xAL:CountryName>
       <xAL:Locality Type="Town">
        <xAL:LocalityName>Eggenstein-Leopoldshafen</xAL:LocalityName>
        <xAL:Thoroughfare Type="Street">
         <xAL:ThoroughfareNumber>1</xAL:ThoroughfareNumber>
         <xAL:ThoroughfareName>Hermann-von-Helmholtz-Platz
         </xAL:ThoroughfareName>
        </xAL:Thoroughfare>
        <xAL:PostalCode>
         <xAL:PostalCodeNumber>76344</xAL:PostalCodeNumber>
        </xAL:PostalCode>
       </xAL:Locality>
      </xAL:Country>
     </xAL:AddressDetails>
    </xalAddress>
    <multiPoint>
     <gml:MultiPoint>
      <gml:pointMember>
       <gml:Point>
        <gml:pos srsDimension="3">458880.0 5438352.7 112.0 </gml:pos>
       </gml:Point>
      </gml:pointMember>
     </gml:MultiPoint>
    </multiPoint>
   </Address>
  </bldg:address>
 </bldg:Building>
```

```
 </cityObjectMember>
 <cityObjectMember>
  <dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766fde6aa67b">
   <gml:name>Example TIN LOD1</gml:name>
   <dem:lod>1</dem:lod>
   <dem:reliefComponent>
    <dem:TINRelief gml:id="GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
     <gml:name>Ground</gml:name>
     <dem:lod>1</dem:lod>
     <dem:tin>
      <gml:TriangulatedSurface>
       <gml:trianglePatches>
        <gml:Triangle>
         <gml:exterior>
          <gml:LinearRing>
           <gml:posList>458868.0 5438362.0 112.0 458875.0 5438355.0 112.0
               458883.0 5438362.0 114.0 458868.0
5438362.0 112.0 </gml:posList>
          </gml:LinearRing>
         </gml:exterior>
        </gml:Triangle>
        <gml:Triangle>...
        </gml:Triangle>(more triangles)</gml:trianglePatches>
      </gml:TriangulatedSurface>
     </dem:tin>
    </dem:TINRelief>
   </dem:reliefComponent>
  </dem:ReliefFeature>
 </cityObjectMember>
</CityModel>
```

Annexe C

Exemple d'un bâtiment IFC

L'exemple venant du site officiel de BuildingSMART [26] consiste en un simple mur muni d'une fenêtre, afin de garder un fichier de taille raisonnable. Le fichier est au format le plus courant pour IFC : *STEP-file*.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION ((’ViewDefinition [CoordinationView,
    QuantityTakeOffAddOnView]’), ’2;1’);
FILE_NAME (’example.ifc’, ’2008-08-01T21:53:56’, (’Architect’),
    (’Building Designer Office’), ’IFC Engine DLL version 1.02 beta’,
    ’IFC Engine DLL version 1.02 beta’, ’The authorising person’);
FILE_SCHEMA ((’IFC2X3’));
ENDSEC;
DATA;
#1 = IFCPROJECT(’3MD_HkJ6X2EwpfIbCFm0g_’, #2, ’Default Project’,
    ’Description of Default Project’, $, $, $, (#20), #7);
#2 = IFCOWNERHISTORY(#3, #6, $, .ADDED., $, $, $, 1217620436);
#3 = IFCPERSONANDORGANIZATION(#4, #5, $);
#4 = IFCPERSON(’ID001’, ’Bonsma’, ’Peter’, $, $, $, $, $);
#5 = IFCORGANIZATION($, ’TNO’, ’TNO Building Innovation’, $, $);
#6 = IFCAPPLICATION(#5, ’0.10’, ’Test Application’, ’TA 1001’);
#7 = IFCUNITASSIGNMENT((#8, #9, #10, #11, #15, #16, #17, #18, #19));
#8 = IFCSIUNIT(*, .LENGTHUNIT., $, .METRE.);
#9 = IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#10 = IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#11 = IFCCONVERSIONBASEDUNIT(#12, .PLANEANGLEUNIT., ’DEGREE’, #13);
#12 = IFCDIMENSIONALEXPONENTS(0, 0, 0, 0, 0, 0, 0);
#13 = IFCMEASUREWITHUNIT(IFCPLANEANGLEMEASURE(1.745E-2), #14);
#14 = IFCSIUNIT(*, .PLANEANGLEUNIT., $, .RADIAN.);
#15 = IFCSIUNIT(*, .SOLIDANGLEUNIT., $, .STERADIAN.);
#16 = IFCSIUNIT(*, .MASSUNIT., $, .GRAM.);
#17 = IFCSIUNIT(*, .TIMEUNIT., $, .SECOND.);
```

```
#18 = IFCSIUNIT(*, .THERMODYNAMICTEMPERATUREUNIT., $, .DEGREE_CELSIUS.);
#19 = IFCSIUNIT(*, .LUMINOUSINTENSITYUNIT., $, .LUMEN.);
#20 = IFCGEOMETRICREPRESENTATIONCONTEXT($, 'Model', 3, 1.000E-5, #21, $);
#21 = IFCAXIS2PLACEMENT3D(#22, $, $);
#22 = IFCCARTESIANPOINT((0., 0., 0.));
#23 = IFCSITE('3rNg_N55v4CRBpQVbZJoHB', #2, 'Default Site', 'Description
   of Default Site', $, #24, $, $, .ELEMENT., (24, 28, 0), (54, 25, 0),
   $, $, $);
#24 = IFCLOCALPLACEMENT($, #25);
#25 = IFCAXIS2PLACEMENT3D(#26, #27, #28);
#26 = IFCCARTESIANPOINT((0., 0., 0.));
#27 = IFCDIRECTION((0., 0., 1.));
#28 = IFCDIRECTION((1., 0., 0.));
#29 = IFCBUILDING('0yf_M5JZv9QQXly4dq_zvI', #2, 'Default Building',
   'Description of Default Building', $, #30, $, $, .ELEMENT., $, $, $);
#30 = IFCLOCALPLACEMENT(#24, #31);
#31 = IFCAXIS2PLACEMENT3D(#32, #33, #34);
#32 = IFCCARTESIANPOINT((0., 0., 0.));
#33 = IFCDIRECTION((0., 0., 1.));
#34 = IFCDIRECTION((1., 0., 0.));
#35 = IFCBUILDINGSTOREY('0C87kaqBXF$xpGmTZ7zxN$', #2, 'Default Building
   Storey', 'Description of Default Building Storey', $, #36, $, $,
   .ELEMENT., 0.);
#36 = IFCLOCALPLACEMENT(#30, #37);
#37 = IFCAXIS2PLACEMENT3D(#38, #39, #40);
#38 = IFCCARTESIANPOINT((0., 0., 0.));
#39 = IFCDIRECTION((0., 0., 1.));
#40 = IFCDIRECTION((1., 0., 0.));
#41 = IFCRELAGGREGATES('2168U9nPH5xB3UpDx_uK11', #2, 'BuildingContainer',
   'BuildingContainer for BuildigStories', #29, (#35));
#42 = IFCRELAGGREGATES('3JuhmQJDj9xPnAnWoNb94X', #2, 'SiteContainer',
   'SiteContainer For Buildings', #23, (#29));
#43 = IFCRELAGGREGATES('1Nl_BIjGLBke9u_6U3IWlW', #2, 'ProjectContainer',
   'ProjectContainer for Sites', #1, (#23));
#44 = IFCRELCONTAINEDINSPATIALSTRUCTURE('2O_dMuDnr1Ahv28oR6ZVpr', #2,
   'Default Building', 'Contents of Building Storey', (#45, #124), #35);
#45 = IFCWALLSTANDARDCASE('3vB2YO$MX4xv5uCqZZG05x', #2, 'Wall xyz',
   'Description of Wall', $, #46, #51, $);
#46 = IFCLOCALPLACEMENT(#36, #47);
#47 = IFCAXIS2PLACEMENT3D(#48, #49, #50);
#48 = IFCCARTESIANPOINT((0., 0., 0.));
#49 = IFCDIRECTION((0., 0., 1.));
#50 = IFCDIRECTION((1., 0., 0.));
#51 = IFCPRODUCTDEFINITIONSHAPE($, $, (#79, #83));
#52 = IFCPROPERTYSET('18RtPv6efDwuUOMduCZ7rH', #2, 'Pset_WallCommon', $,
   (#53, #54, #55, #56, #57, #58, #59, #60, #61, #62));
#53 = IFCPROPERTYSINGLEVALUE('Reference', 'Reference', IFCTEXT(''), $);
```

```
#54 = IFCPROPERTYSINGLEVALUE('AcousticRating', 'AcousticRating',
    IFCTEXT(''), $);
#55 = IFCPROPERTYSINGLEVALUE('FireRating', 'FireRating', IFCTEXT(''), $);
#56 = IFCPROPERTYSINGLEVALUE('Combustible', 'Combustible',
    IFCBOOLEAN(.F.), $);
#57 = IFCPROPERTYSINGLEVALUE('SurfaceSpreadOfFlame',
    'SurfaceSpreadOfFlame', IFCTEXT(''), $);
#58 = IFCPROPERTYSINGLEVALUE('ThermalTransmittance',
    'ThermalTransmittance', IFCREAL(2.400E-1), $);
#59 = IFCPROPERTYSINGLEVALUE('IsExternal', 'IsExternal', IFCBOOLEAN(.T.),
    $);
#60 = IFCPROPERTYSINGLEVALUE('ExtendToStructure', 'ExtendToStructure',
    IFCBOOLEAN(.F.), $);
#61 = IFCPROPERTYSINGLEVALUE('LoadBearing', 'LoadBearing',
    IFCBOOLEAN(.F.), $);
#62 = IFCPROPERTYSINGLEVALUE('Compartmentation', 'Compartmentation',
    IFCBOOLEAN(.F.), $);
#63 = IFCRELDEFINESBYPROPERTIES('3IxFuNHRvBDfMT6_FiWPEz', #2, $, $,
    (#45), #52);
#64 = IFCELEMENTQUANTITY('10m6qcXSj0Iu4RVOK1omPJ', #2, 'BaseQuantities',
    $, $, (#65, #66, #67, #68, #69, #70, #71, #72));
#65 = IFCQUANTITYLENGTH('Width', 'Width', $, 3.000E-1);
#66 = IFCQUANTITYLENGTH('Lenght', 'Lenght', $, 5.);
#67 = IFCQUANTITYAREA('GrossSideArea', 'GrossSideArea', $, 11.500);
#68 = IFCQUANTITYAREA('NetSideArea', 'NetSideArea', $, 10.450);
#69 = IFCQUANTITYVOLUME('GrossVolume', 'GrossVolume', $, 3.450);
#70 = IFCQUANTITYVOLUME('NetVolume', 'NetVolume', $, 3.135);
#71 = IFCQUANTITYLENGTH('Height', 'Height', $, 2.300);
#72 = IFCQUANTITYAREA('GrossFootprintArea', 'GrossFootprintArea', $,
    1.500);
#73 = IFCRELDEFINESBYPROPERTIES('0cpLgxVi9Ew8B08wF2Ql1w', #2, $, $,
    (#45), #64);
#74 = IFCRELASSOCIATESMATERIAL('2zeggBjk9A5wcc3k9CYqdL', #2, $, $, (#45),
    #75);
#75 = IFCMATERIALLAYERSETUSAGE(#76, .AXIS2., .POSITIVE., -1.500E-1);
#76 = IFCMATERIALLAYERSET((#77), $);
#77 = IFCMATERIALLAYER(#78, 3.000E-1, $);
#78 = IFCMATERIAL('Name of the material used for the wall');
#79 = IFCSHAPEREPRESENTATION(#20, 'Axis', 'Curve2D', (#80));
#80 = IFCPOLYLINE((#81, #82));
#81 = IFCCARTESIANPOINT((0., 1.500E-1));
#82 = IFCCARTESIANPOINT((5., 1.500E-1));
#83 = IFCSHAPEREPRESENTATION(#20, 'Body', 'SweptSolid', (#84));
#84 = IFCEXTRUDEDAREASOLID(#85, #92, #96, 2.300);
#85 = IFCARBITRARYCLOSEDPROFILEDEF(.AREA., $, #86);
#86 = IFCPOLYLINE((#87, #88, #89, #90, #91));
#87 = IFCCARTESIANPOINT((0., 0.));
```

```
#88 = IFCCARTESIANPOINT((0., 3.000E-1));
#89 = IFCCARTESIANPOINT((5., 3.000E-1));
#90 = IFCCARTESIANPOINT((5., 0.));
#91 = IFCCARTESIANPOINT((0., 0.));
#92 = IFCAXIS2PLACEMENT3D(#93, #94, #95);
#93 = IFCCARTESIANPOINT((0., 0., 0.));
#94 = IFCDIRECTION((0., 0., 1.));
#95 = IFCDIRECTION((1., 0., 0.));
#96 = IFCDIRECTION((0., 0., 1.));
#97 = IFCOPENINGELEMENT('2LcE70iQb51PEZynawyvuT', #2, 'Opening Element
    xyz', 'Description of Opening', $, #98, #103, $);
#98 = IFCLOCALPLACEMENT(#46, #99);
#99 = IFCAXIS2PLACEMENT3D(#100, #101, #102);
#100 = IFCCARTESIANPOINT((9.000E-1, 0., 2.500E-1));
#101 = IFCDIRECTION((0., 0., 1.));
#102 = IFCDIRECTION((1., 0., 0.));
#103 = IFCPRODUCTDEFINITIONSHAPE($, $, (#110));
#104 = IFCELEMENTQUANTITY('2yDPSWYWf319fWaWWvPxwA', #2, 'BaseQuantities',
    $, $, (#105, #106, #107));
#105 = IFCQUANTITYLENGTH('Depth', 'Depth', $, 3.000E-1);
#106 = IFCQUANTITYLENGTH('Height', 'Height', $, 1.400);
#107 = IFCQUANTITYLENGTH('Width', 'Width', $, 7.500E-1);
#108 = IFCRELDEFINESBYPROPERTIES('2UEO1blXL9sPmb1AMeW7Ax', #2, $, $,
    (#97), #104);
#109 = IFCRELVOIDSELEMENT('3lR5koIT51Kwudkm5eIoTu', #2, $, $, #45, #97);
#110 = IFCSHAPEREPRESENTATION(#20, 'Body', 'SweptSolid', (#111));
#111 = IFCEXTRUDEDAREASOLID(#112, #119, #123, 1.400);
#112 = IFCARBITRARYCLOSEDPROFILEDEF(.AREA., $, #113);
#113 = IFCPOLYLINE((#114, #115, #116, #117, #118));
#114 = IFCCARTESIANPOINT((0., 0.));
#115 = IFCCARTESIANPOINT((0., 3.000E-1));
#116 = IFCCARTESIANPOINT((7.500E-1, 3.000E-1));
#117 = IFCCARTESIANPOINT((7.500E-1, 0.));
#118 = IFCCARTESIANPOINT((0., 0.));
#119 = IFCAXIS2PLACEMENT3D(#120, #121, #122);
#120 = IFCCARTESIANPOINT((0., 0., 0.));
#121 = IFCDIRECTION((0., 0., 1.));
#122 = IFCDIRECTION((1., 0., 0.));
#123 = IFCDIRECTION((0., 0., 1.));
#124 = IFCWINDOW('0LV8Pid0X3IA3jJLVDPidY', #2, 'Window xyz', 'Description
    of Window', $, #125, #130, $, 1.400, 7.500E-1);
#125 = IFCLOCALPLACEMENT(#98, #126);
#126 = IFCAXIS2PLACEMENT3D(#127, #128, #129);
#127 = IFCCARTESIANPOINT((0., 1.000E-1, 0.));
#128 = IFCDIRECTION((0., 0., 1.));
#129 = IFCDIRECTION((1., 0., 0.));
#130 = IFCPRODUCTDEFINITIONSHAPE($, $, (#150));
```

```
#131 = IFCRELFILLSELEMENT('1CDlLMVMv1qw1giUXpQgxI', #2, $, $, #97, #124);
#132 = IFCPROPERTYSET('0fhz_bHU54xB$tXHjHPUZl', #2, 'Pset_WindowCommon',
    $, (#133, #134, #135, #136, #137, #138, #139, #140, #141, #142, #143,
    #144));
#133 = IFCPROPERTYSINGLEVALUE('Reference', 'Reference', IFCTEXT(''), $);
#134 = IFCPROPERTYSINGLEVALUE('FireRating', 'FireRating', IFCTEXT(''), $);
#135 = IFCPROPERTYSINGLEVALUE('AcousticRating', 'AcousticRating',
    IFCTEXT(''), $);
#136 = IFCPROPERTYSINGLEVALUE('SecurityRating', 'SecurityRating',
    IFCTEXT(''), $);
#137 = IFCPROPERTYSINGLEVALUE('IsExternal', 'IsExternal',
    IFCBOOLEAN(.T.), $);
#138 = IFCPROPERTYSINGLEVALUE('Infiltration', 'Infiltration',
    IFCBOOLEAN(.F.), $);
#139 = IFCPROPERTYSINGLEVALUE('ThermalTransmittance',
    'ThermalTransmittance', IFCREAL(2.400E-1), $);
#140 = IFCPROPERTYSINGLEVALUE('GlazingAresFraction',
    'GlazingAresFraction', IFCREAL(7.000E-1), $);
#141 = IFCPROPERTYSINGLEVALUE('HandicapAccessible', 'HandicapAccessible',
    IFCBOOLEAN(.F.), $);
#142 = IFCPROPERTYSINGLEVALUE('FireExit', 'FireExit', IFCBOOLEAN(.F.), $);
#143 = IFCPROPERTYSINGLEVALUE('SelfClosing', 'SelfClosing',
    IFCBOOLEAN(.F.), $);
#144 = IFCPROPERTYSINGLEVALUE('SmokeStop', 'SmokeStop', IFCBOOLEAN(.F.),
    $);
#145 = IFCRELDEFINESBYPROPERTIES('2fHMxamlj5DvGvEKfCk8nj', #2, $, $,
    (#124), #132);
#146 = IFCELEMENTQUANTITY('0bB_7AP5v5OBZ9OTDvo0Fo', #2, 'BaseQuantities',
    $, $, (#147, #148));
#147 = IFCQUANTITYLENGTH('Height', 'Height', $, 1.400);
#148 = IFCQUANTITYLENGTH('Width', 'Width', $, 7.500E-1);
#149 = IFCRELDEFINESBYPROPERTIES('0FmgI0DRX49OXL_$Wa2P1E', #2, $, $,
    (#124), #146);
#150 = IFCSHAPEREPRESENTATION(#20, 'Body', 'SweptSolid', (#151));
#151 = IFCEXTRUDEDAREASOLID(#152, #159, #163, 1.400);
#152 = IFCARBITRARYCLOSEDPROFILEDEF(.AREA., $, #153);
#153 = IFCPOLYLINE((#154, #155, #156, #157, #158));
#154 = IFCCARTESIANPOINT((0., 0.));
#155 = IFCCARTESIANPOINT((0., 1.000E-1));
#156 = IFCCARTESIANPOINT((7.500E-1, 1.000E-1));
#157 = IFCCARTESIANPOINT((7.500E-1, 0.));
#158 = IFCCARTESIANPOINT((0., 0.));
#159 = IFCAXIS2PLACEMENT3D(#160, #161, #162);
#160 = IFCCARTESIANPOINT((0., 0., 0.));
#161 = IFCDIRECTION((0., 0., 1.));
#162 = IFCDIRECTION((1., 0., 0.));
#163 = IFCDIRECTION((0., 0., 1.));
```

```
ENDSEC;
END-ISO-10303-21;
```