Documentation sur la création d’une ADE pour CityGML

# Qu’est-ce qu’une ADE ?

CityGML est un langage de modélisation des objets de la ville, basé sur XML. Il permet d’exprimer de nombreuses propriétés de ces objets, liées aussi bien à leur géométrie qu’à leur sémantique, etc.

Cependant, le modèle ne peut bien évidemment pas couvrir tous les besoins possibles dans tous les cas de figure. C’est pour cela que CityGML comporte un mécanisme d’extension du modèle, afin de permettre aux utilisateurs d’exprimer des concepts qui ne sont pas présents dans le CityGML de base. Ce mécanisme est appelé Application Domain Extension (ADE).

Dans le cadre d’une ADE (par opposition à l’utilisation d’attributs génériques, aussi possible dans CityGML), les nouveaux éléments du modèle doivent être explicitement spécifiés dans un document de définition de schéma XML (fichier XSD), et chaque ADE doit avoir son namespace dédié.

Il faut donc, lors de la création d’une ADE, commencer par bien concevoir son modèle de données puis rédiger le fichier XSD qui le spécifie.

Une ADE permet d’effectuer deux types d’extensions du modèle : rajouter de nouveaux attributs aux classes déjà existantes de CityGML, et la création de toutes nouvelles classes. S’agissant d’une *extension* du modèle, une ADE ne peut pas remplacer des propriétés ou des relations entre classes déjà existantes.

# Principe de fonctionnement

Les nouvelles classes sont, de façon générale, définies de la même manière que les applications de GML. Les nouvelles classes hériteront de classes déjà existantes dans CityGML ou GML (telles que \_AbstractFeature, ou \_CityObject par exemple)

Les classes déjà existantes peuvent être étendues afin de recevoir de nouvelles propriétés (qui font partie du namespace de l’ADE). Ce type d’extension est réalisé grâce à un système de crochet propre à CityGML.  
Dans la définition de schéma XML de CityGML, chaque classe possède, en plus de ses éléments « utiles », un élément \_GenericApplicationPropertyOf*<Featuretypename>*, où <Featuretypename> est le nom de la classe qu’on souhaite étendre. Cet élément peut être de n’importe quel type et peut apparaitre autant de fois qu’on le veut dans l’instance XML. Ceci veut dire qu’on peut rajouter autant de propriétés de n’importe quel type que l’on veut à une classe CityGML.  
Pour se raccorder à ce « crochet », les nouvelles propriétés devront porter dans leur définition l’attribut substitutionGroup="<Featurenamespace>:\_GenericApplicationPropertyOf*<Featuretypename>*"

Par exemple, l’ADE Noise introduit une toute nouvelle classe, NoiseCityFurnitureSegment, représentant les barrières anti-bruit. Cette nouvelle classe hérite de core:AbstractCityObject. L’ADE définit donc d’abord le nouveau type de données NoiseCityFurnitureSegmentType, puis un élément de ce type.

|  |
| --- |
| <xsd:complexType name="NoiseCityFurnitureSegmentType">  <xsd:complexContent>  <xsd:extension base="core:AbstractCityObjectType">  <xsd:sequence>  <xsd:element name="type" type="NoiseCityFurnitureSegmentTypeType" minOccurs="0"/>  <xsd:element name="reflection" type="xsd:string" minOccurs="0"/>  <xsd:element name="reflectionCorrection" type="gml:MeasureType" minOccurs="0"/>  <xsd:element name="height" type="gml:LengthType" minOccurs="0"/>  <xsd:element name="distance" type="gml:LengthType" minOccurs="0"/>  <xsd:element name="lod0BaseLine" type="gml:CurvePropertyType"/>  </xsd:sequence>  </xsd:extension>  </xsd:complexContent> </xsd:complexType> |
| <xsd:element name="NoiseCityFurnitureSegment" type="NoiseCityFurnitureSegmentType" substitutionGroup="core:\_CityObject"/> |

L’ADE introduit ensuite ces barrières dans le modèle en tant que propriétés de la classe CityFurniture. On crée alors un nouveau type de données définissant une « propriété », c’est-à-dire une association avec un élément « NoiseCityFurnitureSegment »

|  |
| --- |
| <xsd:complexType name="NoiseCityFurnitureSegmentPropertyType">  <xsd:sequence minOccurs="0">  <xsd:element ref="NoiseCityFurnitureSegment" minOccurs="0"/>  </xsd:sequence>  <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/> </xsd:complexType> |

Puis on rajoute une nouvelle propriété de ce type à la classe CityFurniture en utilisant le système de « crochet » discuté plus haut.

|  |
| --- |
| <xsd:element name="noiseCityFurnitureSegmentProperty" type="NoiseCityFurnitureSegmentPropertyType" substitutionGroup="frn:\_GenericApplicationPropertyOfCityFurniture"/> |

# Méthodologie de création

Rédiger le fichier XSD à la main est une tâche fastidieuse, et qui devient vite très compliquée si l’ADE comporte beaucoup de nouveaux éléments.

Pour simplifier la création d’ADE, il existe une méthode pour générer automatiquement le fichier XSD à partir d’un diagramme de classe UML. Cette méthode utilise les logiciels Enterprise Architect [1] pour dessiner le diagramme UML, et ShapeChange [2] pour générer le XSD à partir du fichier projet d’Enterprise Architect.

Le formalisme utilisé pour représenter le mécanisme de l’ADE en UML est introduit dans l’article *Modelling an application domain extension of CityGML in UML* [3]. Les nouvelles classes de l’ADE sont modélisées de façon traditionnelle par des classes UML. Les nouveaux membres à ajouter à des classes existantes sont modélisés grâce à des sous-classes portant le stéréotype <<ADEElement>>.

Un exercice détaillé de création d’ADE de Snowflake intitulé « *Developing CityGML ADEs  
Training Course*»présente la modélisation d’une ADE sous Enterprise Architect et l’utilisation de ShapeChange pour générer le fichier XSD. Cet exercice ne semble plus accessible sur le site web, mais une copie a pu être récupérée. Les fichiers joints à cet exercice incluent une modélisation de CityGML (pour l’extension de classes) et un template de fichier de configuration de ShapeChange. A noter que lors de l’étape de génération du XML, l’exercice de Snowflake propose d’exporter le diagramme UML au format XMI pour ensuite exécuter ShapeChange. Cette manipulation peut cependant causer des erreurs et il est recommandé de générer le XSD directement depuis le fichier de projet Enterprise Architect (extension .eap).

**[1]** Enterprise Architect **:** <http://www.sparxsystems.com/>

**[2]** ShapeChange : <http://shapechange.net/get-started/>

**[3]** Van den Brink, L., Stoter, J. E., & Zlatanova, S. (2012, May). Modelling an application domain extension of CityGML in UML. In *ISPRS Conference 7th International Conference on 3D Geoinformation, The International Archivees on the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-4, part C26, 16–17 May 2012, Québec, Canada*. ISPRS.