Module de base 3D

pour

INTERLIS 2.4

Table des matières

1. Introduction

La présente documentation décrit le module de base 3D de la Confédération. Le module de base 3D complète les modules de base existants de la Confédération [1] en leur ajoutant des types de géométrie 3D.

La documentation est structurée comme suit :

- Le 1er chapitre présente les objectifs et les applications possibles.
- Le 2nd chapitre traite des types de géométrie 3D du module de base.
- L'annexe contient la description du modèle de données dans INTERLIS 2.4 [2].

1.1 Objectifs

Le domaine des types de géométrie 3D est très étendu et il existe déjà différentes normes s'y rapportant (par ex. ISO 19107 Spatial Schema, ISO 16739 IFC, GML, etc.). Le module de base 3D n'a pas pour but de remplacer ces normes. Il s'agit davantage de fournir au modeleur INTERLIS un ensemble d'outils simples mais adaptés à l'usage pratique afin de faciliter l'initiation à la modélisation des données 3D et d'éviter la prolifération de types de données 3D dans les modèles INTERLIS. Des ajouts de types 3D supplémentaires au langage actuel INTERLIS 2.4 ont été volontairement évités.

Le module de base 3D se focalise sur la définition des types de base applicables de manière générale (point, ligne, surface, corps). Les types de base ont été conçus de sorte à pouvoir être facilement étendus à l'avenir à l'aide des moyens fournis par le langage INTERLIS. Pour le moment, seules des surfaces 3D planes sont disponibles par exemple. Mais des surfaces 3D à forme libre (splines) sont par ex. aussi envisageables à l'avenir.

Pour éviter de rendre le module de base 3D plus complexe que nécessaire, l'introduction de « Constructive Solid Geometry » (CSG) a été, d'une manière générale, laissée de côté. Le module de base 3D ne comporte donc pas de corps de base (sphère, cône, cylindre, parallélépipède, etc.) pouvant être combinés par addition ou soustraction pour former de nouvelles formes de corps.

1.2 Applications possibles

Lors du choix des types de base, nous avons veillé à ce que des champs d'application concrets puissent être ainsi couverts. Voici quelques exemples :

- Propriété par étages : le type `Solid3D` peut être utilisé directement pour la modélisation de la propriété par étages.
- Cadastre des conduites : `Pipe3D` constitue un type simple, disponible pour la définition des conduites. Le type `Solid3D` peut être utilisé dans une bibliothèque de symboles 3D, par ex. pour définir des regards.
- Géologie : le type `Tin3D` peut être utilisé pour des surfaces de séparation entre des couches géologiques alors que le type `Solid3D` sert à la modélisation des couches.
- Circulation : des axes routiers peuvent être modélisés avec le type `CompositeCurve3D` et la surface d'une route avec `CompositeSurface3D`.
- etc.

1.3 Conventions

Conventions appliquées :

gras	définitions, nouveaux termes.
`courier`	définition provenant du modèle de données INTERLIS figurant en annexe.
[1]	renvoi à la bibliographie en annexe.

Les termes introduits pour décrire des courbes et surfaces dans [2] (chapitres 2.8.12 et 2.8.13) servent aussi ici à la description des types de données 3D. Pour la représentation 2D, nous utiliserons donc le terme « polyligne » au lieu de « courbe » ou « ligne ».

2. Description des types de données 3D

2.1 Utilisation du module de base 3D

Afin de pouvoir utiliser des systèmes de coordonnées concrets dans un modèle d'application avec les types de données 3D définis ici, il est nécessaire d'importer un contexte INTERLIS 2.4 adapté en plus du modèle de données `Geometry3D_V2`. Exemple pour LV95 :

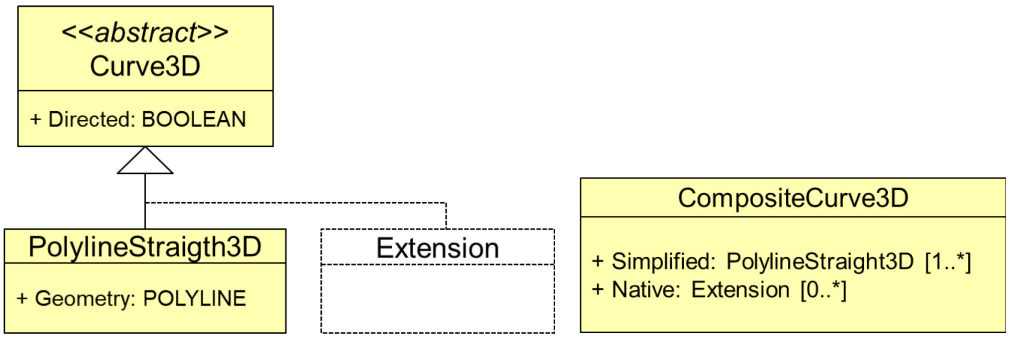
```
MODEL MonApplication =

     IMPORTS Geometry3D_V2, ContextCHLV95_V2;
     ....

END MonApplication.
```

2.2 Types de points

L'espace est une quantité (infinie) de points. Un point comme élément de l'espace est défini de manière unique par la triade de ses coordonnées dans le système de référence correspondant. Pour la définition des points, le type de données générique `Coord3` est repris directement du module de base de géométrie `Geometry_V2`.

2.3 Types de polylignes

Diagramme UML des types de polylignes 3D :



Curve3D

Le type de données `Curve3D` permet de décrire des polylignes (générales) dans l'espace. Le type de données `Curve3D` est, lui-même, abstrait. Tous les types de polylignes simples en sont des spécialisations. Il n'existe actuellement qu'une seule spécialisation concrète (`PolylineSraight3D`).

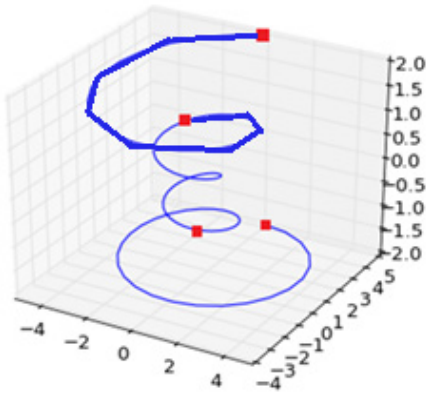
D'autres types concrets de polylignes (splines 3D par ex.) pourront être définis à l'avenir conformément au modèle de conception `Extension`. Il est possible d'orienter ou non des polylignes selon `Curve3D` (attribut `Directed`). En cas de `Curve3D` orienté, l'ordre des points d'appui ne doit plus être modifié dans le transfert.

PolylineStraight3D

Décrit dans l'espace une polyligne dont les parties sont exclusivement des morceaux de droites.

CompositeCurve3D

Décrit une liste (c.-à-d. une suite finie) de polylignes. Les polylignes sont reliées, c.-à-d. que le point initial d'une polyligne de la liste concorde respectivement au point final de la polyligne précédente, hormis pour la première et la dernière polyligne. Les polylignes de `CompositeCurve3D` ne doivent pas se couper dans la projection ou dans l'espace.



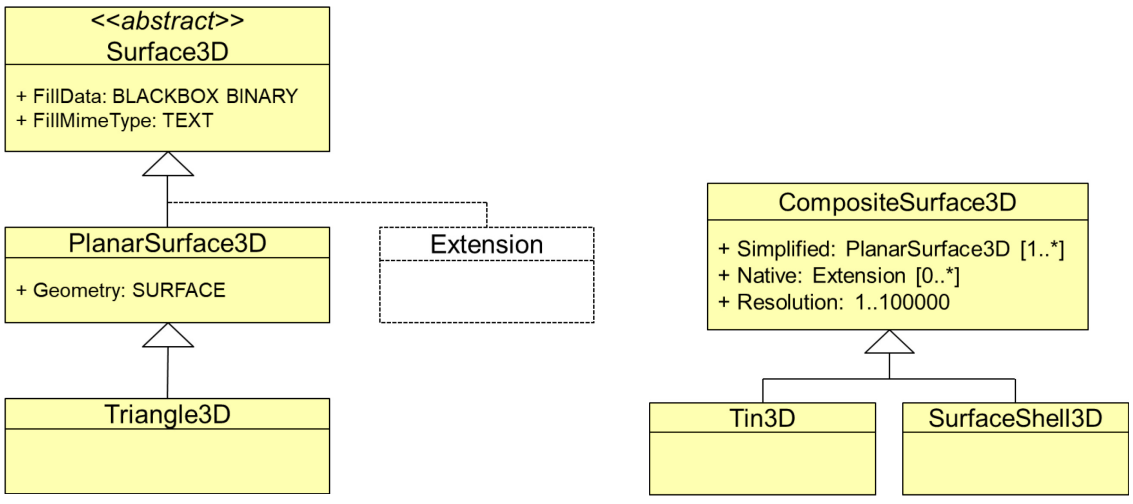
Exemple d'une polyligne du type `CompositeCurve3D`

On voit, dans cet exemple, une polyligne du type `CompositeCurve3D` constituée de trois polylignes simples. Les points initiaux et finaux des polylignes simples sont marqués en rouge. La première polyligne (en partant du haut) est une `PolylineStraight3D`, les autres polylignes sont des spécialisations quelconques de `CompositeCurve3D`.

Comme mentionné précédemment, seule la forme de courbe concrète `PolylineStraight3D` est définie actuellement pour `CompsiteCurve3D`. Pour pouvoir étendre à l'avenir les formes de courbe (par ex. par des splines) tout en assurant l'interopérabilité entre les systèmes, toutes les polylignes doivent toujours être transférées sous forme de `PolylineStraight3D` dans l'attribut `Simplified`. Pour les autres formes de courbes, *il faut de plus* que l'attribut `Native` soit transféré avec la géométrie originale. L'écart entre `Native` et `Simplified` ne doit pas dépasser la résolution multipliée par la résolution géométrique (cf. chapitre 3 Interopérabilité).

2.4 Types de surfaces

Diagramme UML des types de surfaces 3D :



Surface3D

Tous les types de surfaces 3D simples sont des spécialisations du type abstrait `Surface3D`. Il n'existe actuellement que les spécialisations concrètes `PlanarSurface3D` et `Triangle3D`. Il est possible à l'avenir de définir d'autres spécialisations concrètes (surfaces spline 3D par ex.) selon le modèle de conception `Extension`. Le type `Surface3D` correspond aux « surfaces » au chapitre 2.8.13.1 dans [2]. Au sujet de leur frontière, on peut seulement dire qu'elle se compose d'un nombre fini de morceaux de courbes qui n'ont en commun que les points finaux. Les frontières intérieures et extérieures ne peuvent tout d'abord être définies que pour les surfaces planes. Des informations sur l'intérieur de la surface (données de rendering, par ex.) peuvent être transférées avec `FillMimeType` / `FillData`. La structure de ces données n'est pas davantage formalisée ici.

PlanarSurface3D

Décrit une **surface plane**, c.-à-d. une surface dont les points se situent tous dans un plan E. Une surface plane du type `PlanarSurface3D` est limitée par une frontière extérieure et par une, aucune ou plusieurs frontières intérieures. La frontière extérieure est orientée dans le sens anti-horaire alors que les frontières intérieures sont orientées dans le sens horaire. Si l'on suit les frontières dans le sens de rotation correspondant, l'intérieur de la surface se trouve toujours à gauche de l'observateur. Les courbes des frontières intérieures et extérieures ne doivent se composer que de morceaux de droites, elles doivent donc être des polylignes fermées du type `PolylineStraight3D`. La distance de projection des points de frontière sur le plan E doit être au maximum aussi grande que la résolution géométrique dans le modèle. Le plan E doit, de manière générale, être déterminé par un calcul de compensation des angles de la surface plane (excepté pour les triangles).

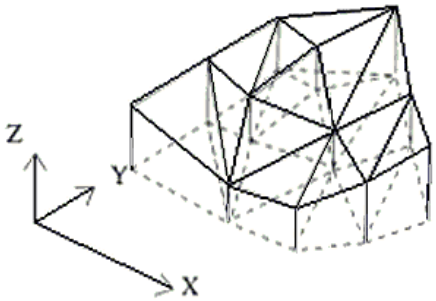
Triangle3D

Décrit un **triangle**, c.-à-d. une limitation du type `PlanarSurface3D`. La frontière de la surface se compose d'exactement trois points d'appui.

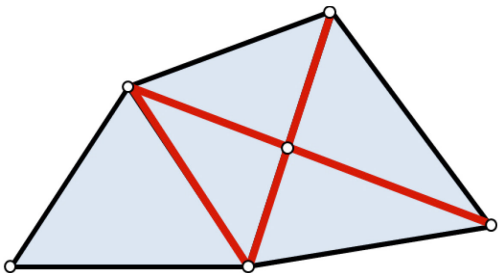
La fonction `getEdgePointCount()` permet de forcer le nombre de points d'appui dans une contrainte. `getEdgePointCount()` compte tous les points d'appui de la frontière extérieure / des frontières intérieures d'une surface. Le point initial et le point final des frontières ne sont comptés qu'une fois.

CompositeSurface3D

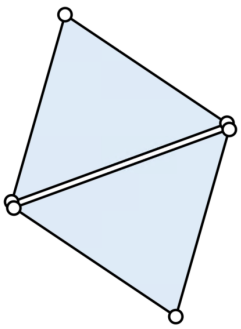
Décrit un **réseau surfacique**, c.-à-d. l'union d'une quantité finie de surfaces qui sont des spécialisations du type `Surface3D` et ont en commun les points de frontière les plus élevés. En clair : les surfaces (partielles) d'un réseau surfacique sont « collées » entre elles au niveau des frontières communes. Cela signifie que chaque polyligne de frontière à l'intérieur d'une surface du type `CompositeSurface3D` est utilisée respectivement par deux surfaces (partielles) du réseau surfacique. Les autres polylignes de frontière forment les limites extérieures ou intérieures du réseau surfacique. L'intérieur du réseau surfacique doit être connexe (évent. plusieurs fois), c.-à-d. que chaque point à l'intérieur du réseau surfacique du type `CompositeSurface3D` est relié à un autre point intérieur par (au moins) un chemin à l'intérieur qui ne coupe pas une frontière du réseau surfacique ni ne contient des points d'appui de frontières extérieures (ou intérieures) du réseau surfacique. Si l'intérieur du réseau surfacique n'est pas connexe, des frontières du réseau surfacique peuvent se toucher aux points d'appui. Les surfaces peuvent s'interpénétrer.



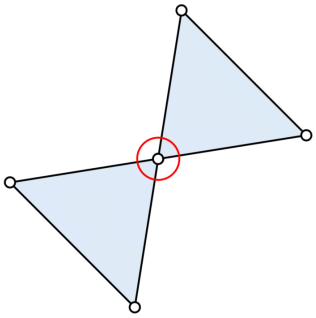
Exemple d'un réseau surfacique du type `CompositeSurface3D` composé de triangles



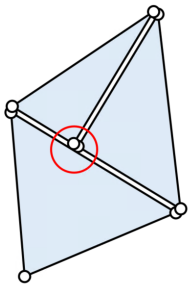
polylignes extérieures (noires) et polylignes intérieures (rouges)



Réseau surfacique valable du type `CompositeSurface3D`



Non valable car intérieur non connexe



Non valable car toutes les polylignes intérieures ne sont pas doubles

Remarques sur l'interopérabilité de `CompositeSurface3D`, cf. chapitre 3.

Tin3D

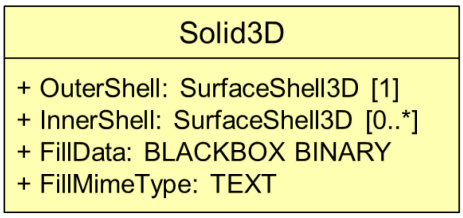
Est une spécialisation de `CompositeSurface3D`. Un réseau surfacique du type `Tin3D` ne se compose que de triangles du type `Triangle3D`. Le type `Tin3D` convient donc particulièrement aux modèles numériques de terrain (MNT).

SurfaceShell3D

Est une spécialisation de `CompositeSurface3D`. Le réseau surfacique du type `SurfaceShell3D` décrit une enveloppe fermée dans l'espace. *Chaque* polyligne de frontière est utilisée par deux surfaces partielles et les surfaces partielles ne doivent pas s'interpénétrer.

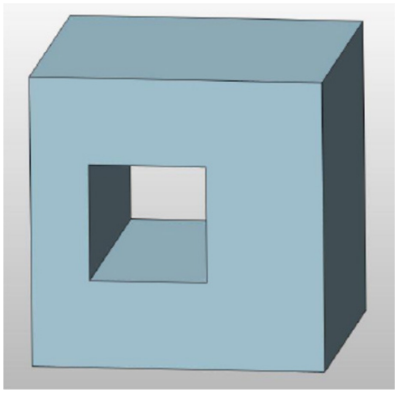
2.5 Types de corps

Diagramme UML des types de corps 3D :

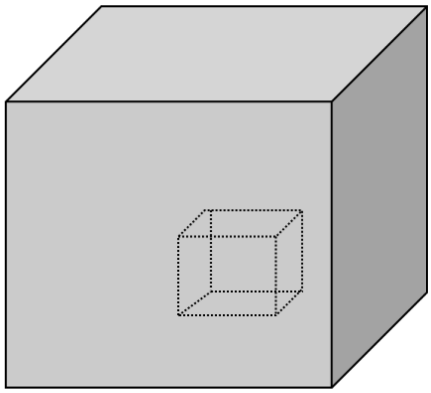


Solid3D

Un corps du type `Solid3D` est limité par une enveloppe extérieure (`OuterShell`) et, en option, par un nombre quelconque d'enveloppes intérieures (`InnerShells`) du type `SurfaceShell3D`. Les enveloppes intérieures et extérieures doivent se toucher au maximum à une arête commune ou à un angle commun. Les enveloppes intérieures doivent se situer entièrement dans l'enveloppe extérieure. L'intérieur du corps doit être connexe (évent. plusieurs fois). Cela signifie que chaque point à l'intérieur du corps est relié à chaque autre point intérieur par (au moins) un chemin (ou plus précisément : une classe de chemins équivalents) à l'intérieur du corps. Ce chemin ne doit ni couper des enveloppes ni contenir des points d'appui d'enveloppes. Si cette condition est remplie, des enveloppes peuvent se toucher à des angles communs ou dans des polylignes communes, mais pas dans des surfaces.

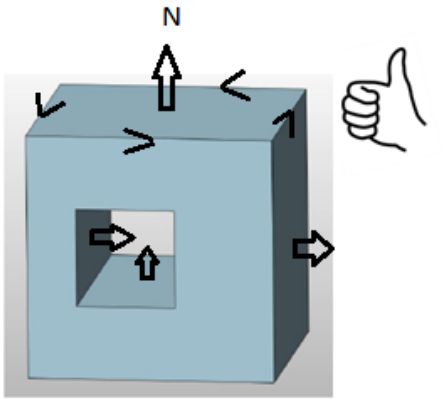


Exemple avec une OuterShell



Exemple avec une OuterShell et une InnerShell

Le vecteur normal des surfaces partielles des enveloppes du type `SurfaceShell3D` pointe toujours depuis l'intérieur dans le sens opposé à l'environnement extérieur du corps. La direction du vecteur normal est définie comme suit : quand les doigts de la main droite suivent le sens de rotation de la frontière extérieure de la surface partielle, le pouce pointe en direction du vecteur normal. Remarque : tandis que le type INTERLIS `SURFACE` ne connaît pas de sens de rotation, le sens de rotation des surfaces partielles du type `SurfaceShell3D` est essentiel.



Vecteur normal N et sens de rotation

Seule l'enveloppe du corps est décrite explicitement dans `Solid3D`. L'intérieur du corps (données de voxels par ex.) peut être transféré avec les attributs `FillMimeType` / `FillData`. La structure de ces données n'est pas davantage formalisée ici.

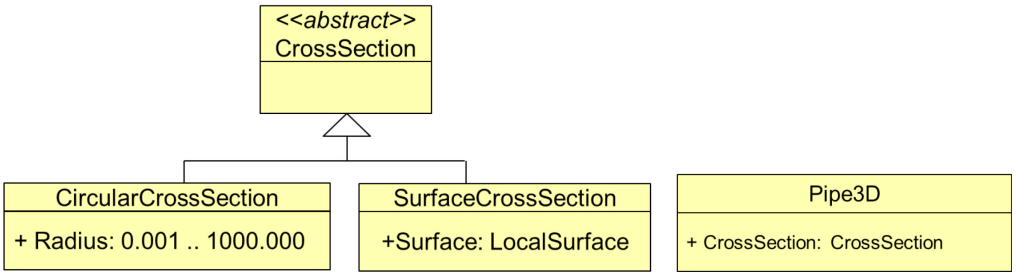
Fonction isCompositeSolid3D()

La fonction `isCompositeSolid()` permet d'exiger dans un (`SET`) `CONSTRAINT` qu'une quantité de `Solid3D` forme un `Solid3D` composé. Les solides partiels de la quantité ne doivent pas s'interpénétrer dans ce cas. La quantité doit, de plus, être connexe. Cela signifie que chaque point à l'intérieur du corps est relié à chaque autre point à l'intérieur par un chemin qui ne croise que des surfaces intérieures, pas des surfaces extérieures.

2.6 Types paramétriques

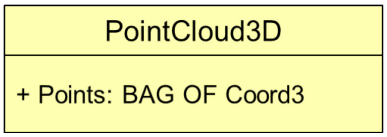
Pipe3D

Le type paramétrique `Pipe3D` est proposé pour ne pas devoir approximer des tuyaux (par ex. dans le cadastre des conduites) avec `Solid3D`. Un `Pipe3D` est une spécialisation du type `CompositeCurve3D` avec section transversale associée (`CrossSection`). Il est possible de saisir un cercle (`CircularCrosssection`) ou une surface quelconque (`SurfaceCrossection`) comme section transversale.



2.7 Types multiples

Un seul type multiple est proposé pour les données Lidar, il s'agit de `PointCloud3D`. Il sera ainsi possible à l'avenir de prévoir des formes de codage plus efficaces pour `PointCloud3D` que ce n'est possible pour le moment avec INTERLIS 2.4.



3. Interopérabilité

Pour faciliter l'interopérabilité entre les systèmes, il est exigé pour `CompositeCurve3D` et `CompositeSurface3D` de transférer, pour les types de géométrie spécialisés selon le modèle de conception `Extension`, la géométrie dans les attributs `Simplified` et `Native`. `Native` contient la définition de géométrie d'origine. `Simplified` se compose d'une représentation simplifiée de la géométrie d'origine par des morceaux de droites ou des surfaces planes. Les programmes qui ne prennent pas (encore) en charge `Extension` peuvent ainsi utiliser la géométrie de `Simplified`.

L'écart maximal entre `Simplified` et `Native` ne doit pas dépasser `Resolution` multipliée par la résolution géométrique dans le modèle, soit par ex. « 1 fois un millimètre ». Il est donc possible de déterminer au moyen de `Resolution` avec quelle précision `Native` est approximé par `Simplified`.

Annexe

A. Basismodul 3D in INTERLIS 2.4

INTERLIS Modelldatei-Name: CHBase_Part8_GEOMETRY3D_V2.ili

```
/* ##########################################################################
   CHBASE - BASE MODULES OF THE SWISS FEDERATION FOR MINIMAL GEODATA MODELS
   ======
   BASISMODULE DES BUNDES           MODULES DE BASE DE LA CONFEDERATION
   FÜR MINIMALE GEODATENMODELLE     POUR LES MODELES DE GEODONNEES MINIMAUX

   PROVIDER: GKG/KOGIS - GCS/COSIG            CONTACT: models@geo.admin.ch
   PUBLISHED: 2023-10-18
   ##########################################################################
*/

/* ##########################################################################
   ##########################################################################
   PART VIII -- GEOMETRY3D
   - Package Geometry3D
*/

!! ##########################################################################
!!@technicalContact=mailto:models@geo.admin.ch
!!@furtherInformation=https://www.geo.admin.ch/de/geoinformation-
schweiz/geobasisdaten/geodata-models.html
TYPE MODEL Geometry3D_V2 (en)
  AT "https://models.geo.admin.ch/CH/" VERSION "2023-10-18" =

  IMPORTS UNQUALIFIED INTERLIS;
  IMPORTS Units;
  IMPORTS CoordSys;
  IMPORTS Geometry_V2;

  DOMAIN

    LocalCoord2 = COORD
       -1000.000 .. 1000.000 [INTERLIS.m],
       -1000.000 .. 1000.000 [INTERLIS.m],
      ROTATION 2 -> 1;
    LocalSurface = SURFACE WITH (STRAIGHTS,ARCS)
       VERTEX LocalCoord2 WITHOUT OVERLAPS > 0.001;

    !! abstract base structure for all 3d curve types
    STRUCTURE Curve3D (ABSTRACT) =
       Directed: MANDATORY BOOLEAN;
    END Curve3D;

    !! polyline in 3d
    STRUCTURE PolylineStraight3D EXTENDS Curve3D =
```

```
      Geometry: MANDATORY POLYLINE WITH (STRAIGHTS) VERTEX  Geometry_V2.Coord3
WITHOUT OVERLAPS > 0.001;
    END PolylineStraight3D;

    !! composite (connected) curve
    STRUCTURE CompositeCurve3D =
      Simplified: LIST {1..*} OF PolylineStraight3D;
      Native: LIST {0..*} OF Curve3D;
      Resolution: 1 .. 100000; !! Default 1
    END CompositeCurve3D;

    STRUCTURE CrossSection (ABSTRACT) =
    END CrossSection;

    STRUCTURE CircularCrossSection EXTENDS CrossSection =
      Radius: MANDATORY 0.001 .. 1000.000 [INTERLIS.m];
    END CircularCrossSection;

    STRUCTURE SurfaceCrossSection EXTENDS CrossSection =
      Surface: MANDATORY LocalSurface;
    END SurfaceCrossSection;

    STRUCTURE Pipe3D EXTENDS CompositeCurve3D =
      CrossSection: MANDATORY CrossSection;
    END Pipe3D;

    !! abtract base structure for all 3d surface types
    STRUCTURE Surface3D (ABSTRACT) =
      FillData: BLACKBOX BINARY;
      FillMimeType: TEXT*100;
    END Surface3D;

    !! function for calculating edge point count
    FUNCTION getEdgePointCount(geometry:SURFACE):NUMERIC;

    !! planar surface in 3D with no arcs
    STRUCTURE PlanarSurface3D EXTENDS Surface3D =
      Geometry: MANDATORY SURFACE WITH (STRAIGHTS) VERTEX Geometry_V2.Coord3
WITHOUT OVERLAPS > 0.001;
    END PlanarSurface3D;

    !! planar surface with only three points no arcs
    STRUCTURE Triangle3D EXTENDS PlanarSurface3D =
    MANDATORY CONSTRAINT
      getEdgePointCount(Geometry) == 3;
    END Triangle3D;

    !! non planar surface in 3D
    !! all inner edge curves are shared by two surfaces
    !! except edge curves of composite boundary(ies)
    !! surfaces may intersect
    STRUCTURE CompositeSurface3D =
      Simplified: BAG {1..*} OF Triangle3D;
      Native: BAG {0..*} OF Surface3D;
      Resolution: 1 .. 100000; !! Default 1
    END CompositeSurface3D;

    !! triangle network (TIN)
    STRUCTURE Tin3D EXTENDS CompositeSurface3D =
    END Tin3D;
```

```
    !! closed surface shell
    !! all surface edge curves are shared by two surfaces
    !! surfaces may not intersect
    STRUCTURE SurfaceShell3D EXTENDS CompositeSurface3D =
    END SurfaceShell3D;

    !! 3D solid with one outer shell and 0..* inner shells
    !! shells may not intersect
    !! inner shells are enclosed by outer shell
    STRUCTURE Solid3D =
      OuterShell: MANDATORY SurfaceShell3D;
      InnerShells: BAG {0..*} OF SurfaceShell3D;
      FillData: BLACKBOX BINARY;
      FillMimeType: TEXT*100;
    END Solid3D;

    !! collection of 3D points
        STRUCTURE PointCloud3D =
      Points: BAG {1..*} OF Geometry_V2.Coord3;
    END PointCloud3D;

    !! function for testing composite solids in set constraints
    FUNCTION isCompositeSolid3D(solids:BAG OF Solid3D):BOOLEAN;

END Geometry3D_V2.

!! ##########################################################################
```

B. Bibliographie

[1] Modules de base de la Confédération pour les « modèles de géodonnées minimaux »
https://backend.geo.admin.ch/fileservice/sdweb-docs-prod-geoadminch-files/files/2023/03/02/4f9f4099-11f4-4129-9717-5370bd29dceb.pdf

[2] eCH-0031 : INTERLIS 2.4 Manuel de référence
https://www.ech.ch/dokument/266625ed-0a4a-4c20-b7ff-504cd908c0d6