Institute of Information Science
Academia Sinica
中央研究院 資訊科學研究所

GML基礎與實務

-於台北市政府資訊中心-

The fundamental and practice of GML

-At Computer Center, Taipei City Government-

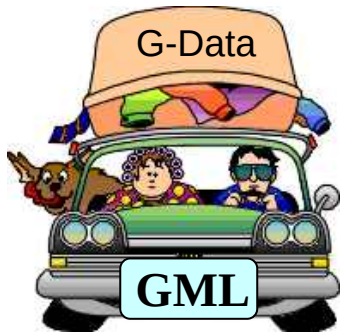
鄧東波
Dongpo Deng

2007-3-21

dongpo@iis.sinica.edu.tw

從XML說起

- GML (Geography Mark-up Language)是基於XML (eXtensible Mark-up Language)編碼技術所發展，以對於地理空間資訊進行模式化、傳輸和交換。
  - GML繼承了XML的所有語法與規則，如XML Namespaces, XML Schema, Xlinks.
  - 具有處理地理空間資料的語彙，如feature, geometry, topology, 3D, ...



控制標籤的格式(tag)

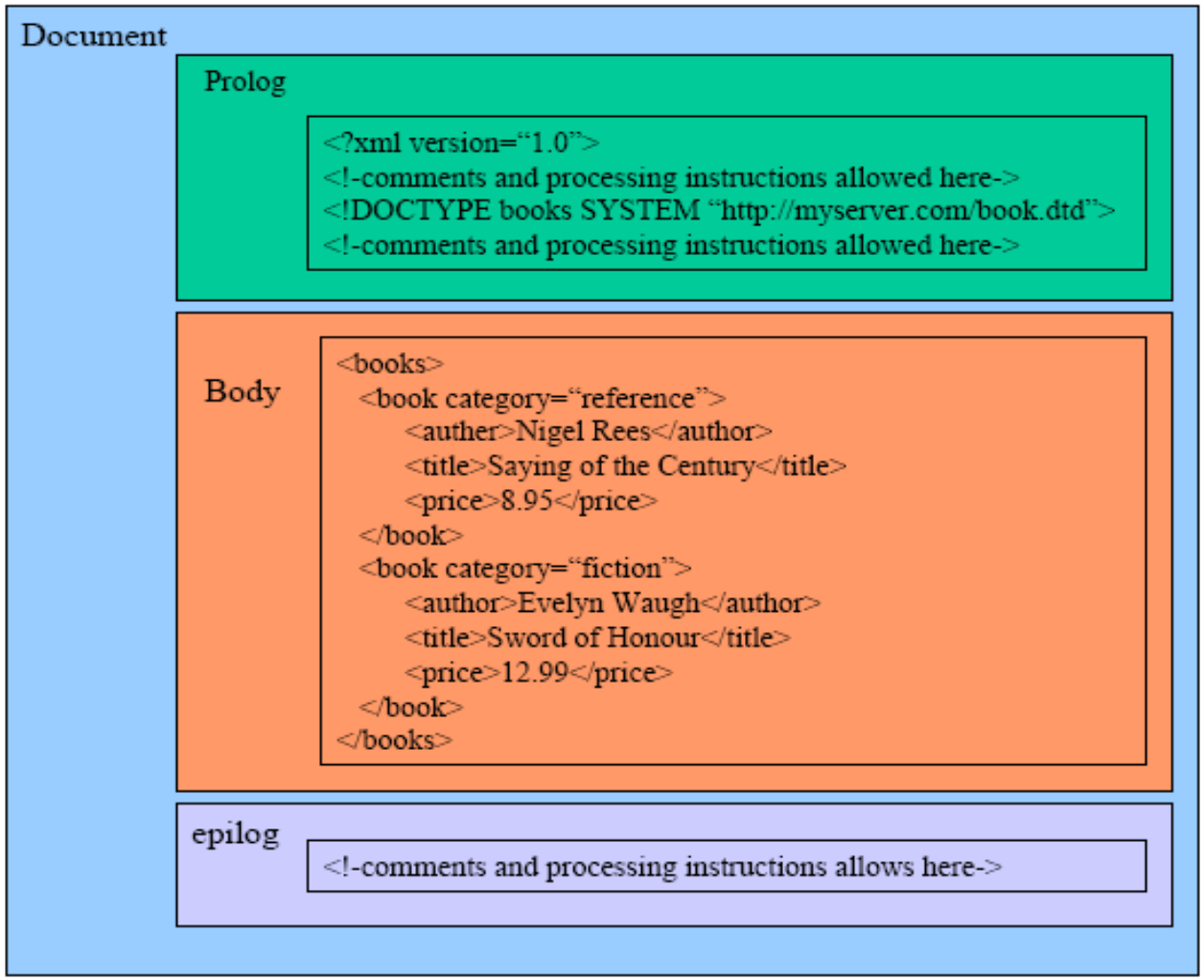
- **Syntax**
  - <tagname>
  - <tagname attribute=”value”>
  - </tagname>
  - <tagname/>
  - <tagname attr1=”value1”
  - attr2=”value2”/>
- **Component**
  - 開始標籤 (Element start-tag)
  - 有屬性開始標籤 (Start-tag with an attribute)
  - 結束標籤 (End-tag)
  - 空標籤 (Empty-element tag)
  - 有屬性空標籤 (Empty-element tag with two attributes)

元素(element)

- 元素(element)爲XML文件的基本組件
- 元素(element)以標籤(tag)來分隔
- Elements must have a closing tag
- Elements must be properly nested

```
<book>
        <title>My First XML</title>
        <prod id="33-657" media="paper"></prod>
        <chapter>Introduction to XML
                <para>What is HTML</para>
                <para>What is XML</para>
        </chapter>
        <chapter>XML Syntax
                <para>Elements must have a closing tag</para>
                <para>Elements must be properly nested</para>
        </chapter>
</book>
```

文件架構(Document Parts)



A example of XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.0.1 U (http://www.xmlspy.com) by Alexander
Pilz (private) -->
<Company xmlns="http://my-company.com/namespace"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://my-company.com/namespace.xsd">
  <Address xsi:type="US-Address">
    <Name>US dependency</Name>
    <Street>Noble Ave.</Street>
    <City>Dallas</City>
    <Zip>04812</Zip>
    <State>Texas</State>
  </Address>
  <Person Manager="true" Degree="BA" Programmer="false">
    <First>Fred</First>
    <Last>Smith</Last>
    <PhoneExt>22</PhoneExt>
    <Email>Smith@work.com</Email>
  </Person>
</Company>
<!– This xml document just for educational demonstration, don't  be
shareable-->
```

Prolog

Body

Epilog

Parent

Child

Parent

XML Schema

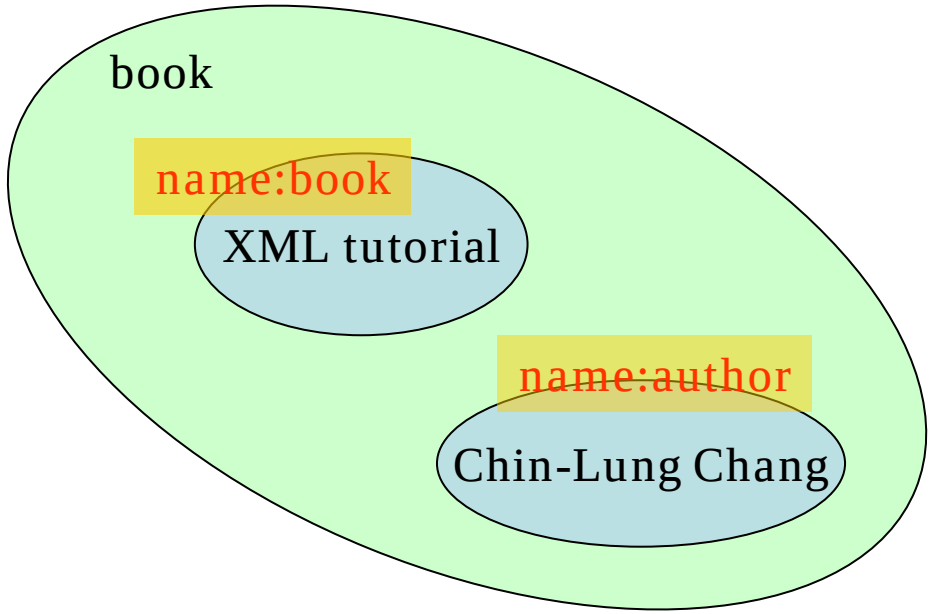
1. Namespace
2. Declaration
3. Simple Type
4. Complex Type
5. Substitution group
6. Identity constraint
7. Schema Composition

More details to www.iis.sinica.edu.tw/~evirt/report.htm

Institute of Information Science
Academia Sinica
中央研究院 資訊科學研究所

XML Namespace

Why namespace?



- A book record

```
<book>
   <name>XML tutorial</name>
</book>
```

- An author record

```
<author>
   <name>Chin-Lung Chang</name>
<author>
```

- Combining book and author records? A conflict of element name!

```
<book>
   <book_name>XML tutorial</book_name>
   <author_name>Chin-Lung Chang</author_name>
</book>
```

XML Namespace

- Namespace: A collection of element and attribute names
  - The namespace is identified by a URI.
  - Two-part naming convention:
    - The prefix name
    - The URI of the XML Namespace

- xmlns:foo="http://www.foo.org/"

Declaring Namespace

- Namespace are declared using a special attribute that starts with the xmlns attribute name.
- It is not possible to associate a prefix to an empty URI string. Ex. xmlns:prod=“”
- An example:

```
<prod:product xmlns:prod=“http://example.org/prod”>
  <prod:number>557</prod:number>
  <prod:size system=“US-DRESS”>10</prod:size>
</prod:product>
```

Target Namespace

- XSDL allows a schema document to define ONE namespace, known as its target namespace
  - A schema document cannot have more than one target namespace.
  - Elements defined in the schema document will be referred to by the target namespace.

- Every component declared or defined by a global declaration is associated with that target namespace.

- Local declarations may or may not use the target namespace.

Target Namespace: An Example

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://example.org/prod"
    targetNamespace="http://example.org/prod">

  <xsd:element name="number" type="xsd:integer"/>
  <xsd:element name="size" type="SizeType"/>

  <xsd:simpleType name="SizeType">
      <--……-->
  </xsd:simpleType>

</xsd:schema>
```

Institute of Information Science
Academia Sinica
中央研究院 資訊科學研究所

Complex type

Complex Type Definition: Examples

```
<xsd:complexType name="ProductType">
    <xsd:sequence>
       <xsd:elementname="number“ type="ProdNumType"/>
       <xsd:element name="name" type="xsd:string"/>
       <xsd:element name="size" type="SizeType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="product" type="ProductType"/>
< ---------------------------------------------------------- >
<xsd:element name="product">
    <xsd:complexType>
          <xsd:sequence>
                <xsd:element name="number" type="ProdNumType"/>
                <xsd:element name="name" type="xsd:string"/>
                <xsd:element name="size" type="SizeType"/>
          </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Simple
Element-only
Mixed
Empty

Model Group

- The order and structure of the children of a complex type are known as its "content model".
- Model group allow you to group child element declarations or references together to construct more meaningful content models.
- There are 3 kinds of model groups:
  - Sequence
  - Choice
  - All
- Every complex type has exactly one model group child.

Sequence Groups

```
<xsd:complexType name="ProductType">
    <xsd:sequence>
        <xsd:element name="number" type="ProdNumType"/>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="size" type="SizeType" minOccurs="0"/>
        <xsd:element name="color" type="ColorType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<product>
    <number>557</number>
    <name>Short-Sleeved Linen Blouse</name>
    <size system="US-DRESS">10</size>
    <color value="blue"/>
</product>
```

Attribute Declarations

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:attribute  name="effDate“ type="xsd:date"/>
    <xsd:complexType name="ProductType">
          <xsd:sequence>
                  <!--...-->
          </xsd:sequence>
          <xsd:attribute ref="effDate"/>
        <xsd:attribute name=“local” type=“xsd:string”/>
    </xsd:complexType>
</xsd:schema>
```

How to Derive Complex Types?

- Restriction
  - Restricting the valid contents of a type.
  - Values of new type is a subset of those of the base type.
  - All values of the restricted type are valid with respect to the base type.

- Extension
  - Adding additional children and/or attributes to a type.

Complex Type Extension (Example)

Extended:

```
<xsd:complexType name="ShirtType">
    <xsd:complexContent>
        <xsd:extension base="ProductType">
          <xsd:choice maxOccurs="unbounded">
              <xsd:element name="size" type="SizeType"/>
              <xsd:element name="color" type="ColorType"/>
          </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

```
<xsd:complexType name="ProductType">
    <xsd:sequence>
        <xsd:element name="number" type="ProdNumType"/>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="size" type="SizeType" minOccurs="0"/>
        <xsd:element name="color" type="ColorType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="RestrictedProductType">
    <xsd:complexContent>
        <xsd:restriction base="ProductType">
            <xsd:sequence>
                <xsd:element name="number" type="ProdNumType"/>
                <xsd:element name="name" type="xsd:string"/>
            </xsd:sequence>
        </xsd:restriction>
    </xsd:complexContent>
</xsd:complexType>
```

Attribute Restriction: Examples

```
<xsd:complexType name="BaseType">
    <xsd:attribute name="a" type="xsd:integer"/>
    <xsd:attribute name="b" type="xsd:string"/>
    <xsd:attribute name="c" type="xsd:string" default="c"/>
    <xsd:attribute name="d" type="xsd:string"/>
    <xsd:attribute name="e" type="xsd:string"/>
    <xsd:attribute name="x" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="DerivedType">
    <xsd:complexContent>
        <xsd:restriction base="BaseType">
            <xsd:attribute name="a" type="xsd:positiveInteger"/>
            <xsd:attribute name="b" type="xsd:string" default="b"/>
            <xsd:attribute name="c" type="xsd:string" default="c2"/>
            <xsd:attribute name="d" type="xsd:string" use="required"/>
            <xsd:attribute name="e" type="xsd:string" use="prohibited"/>
        </xsd:restriction>
    </xsd:complexContent>
</xsd:complexType>
```

Institute of Information Science
Academia Sinica
中央研究院 資訊科學研究所

Substitution group

substitution group(1)

- Each substitution group consists of
  - a head
  - One or more members

- Wherever the head element declaration is referenced in a content model, one of the member element declarations may be substituted in place of the head
  - Element substitution

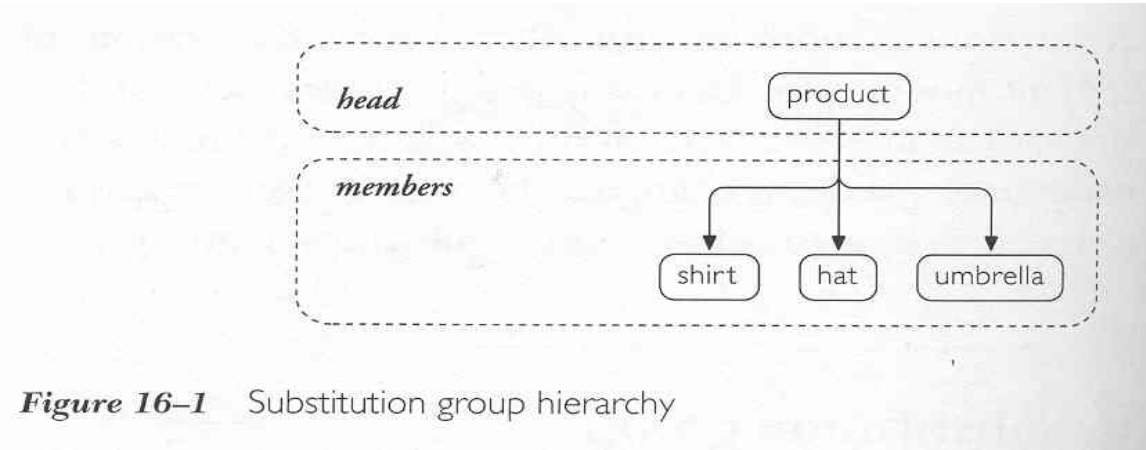
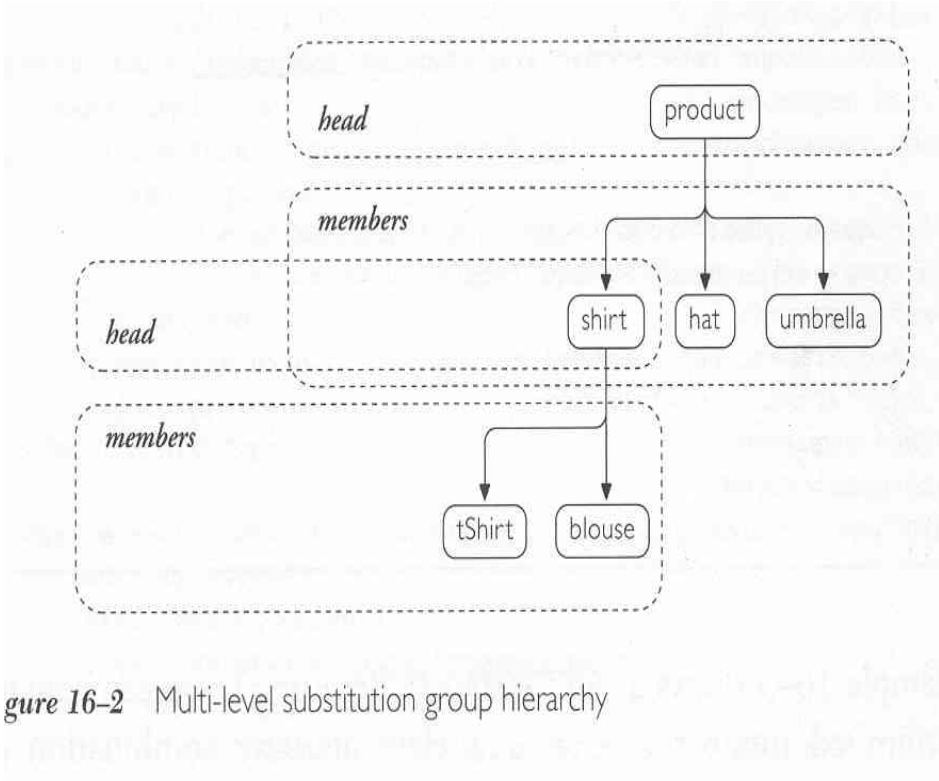


Figure 16–1 Substitution group hierarchy

substitution groups(2)

- Each element declaration can only be a member of one substitution group
- A member of one group may be the head of another group
- tShirt and blouse may substitute for product (or shirt)



gure 16–2 Multi-level substitution group hierarchy

Example: head

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="items" type="ItemsType"/>
    <xsd:complexType name="ItemsType">
        <xsd:sequence>
            <xsd:element ref="product" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="product" type="ProductType"/>
    <xsd:complexType name="ProductType">
        <xsd:sequence>
            <xsd:element ref="number"/>
            <xsd:element ref="name"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Example: member

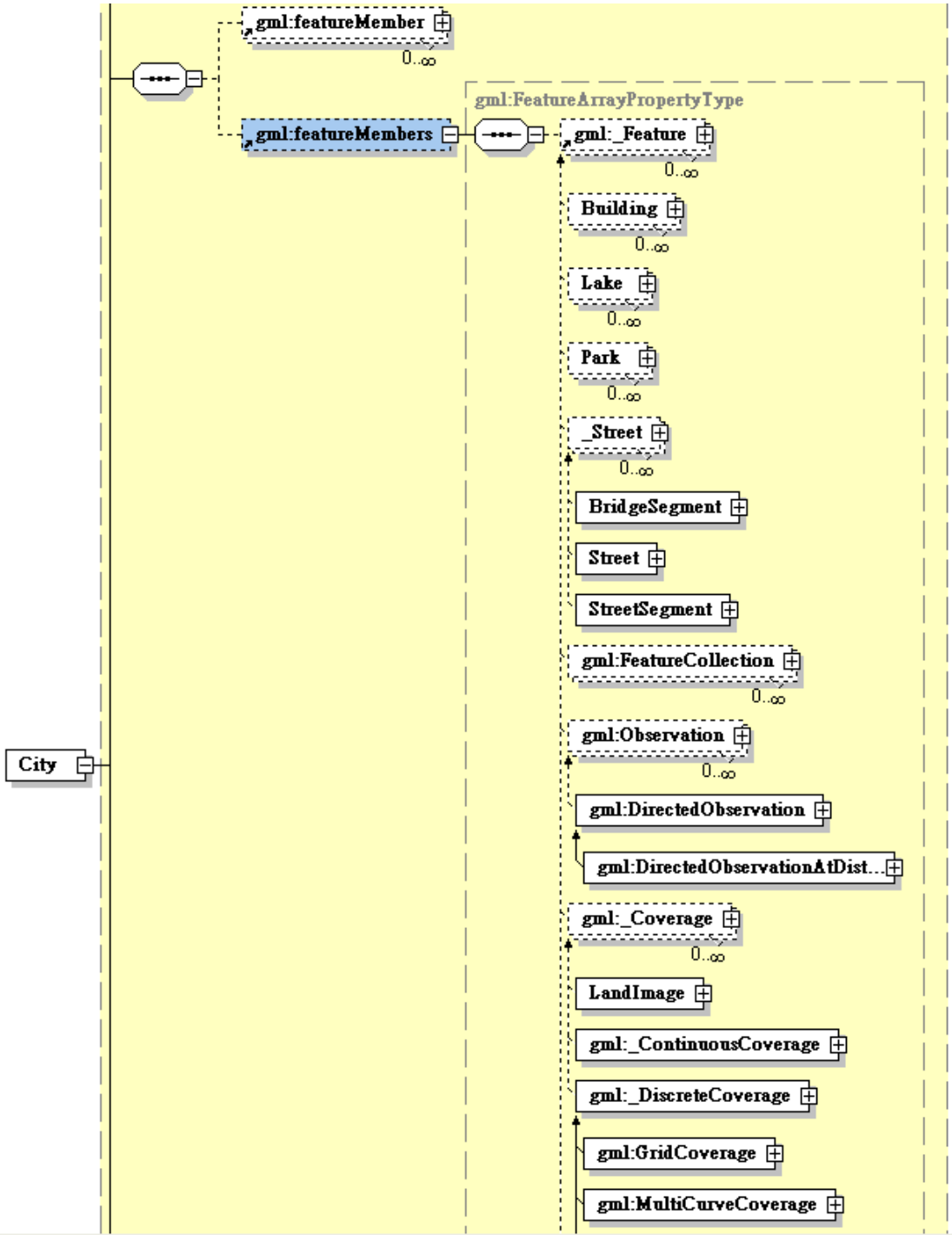
```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="shirt" type="ShirtType“ substitutionGroup="product"/>

    <xsd:complexType name="ShirtType">
      <xsd:complexContent>
        <xsd:extension base="ProductType">
          <xsd:sequence>
            <xsd:element name="size" type="ShirtSizeType"/>
            <xsd:element name="color" type="ColorType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!--...-->
    <xsd:element name="umbrella" substitutionGroup="product"/>
    <!--...-->
</xsd:schema>
```

Example: GML

```
<element name="City" type="app:CityType"
substitutionGroup="gml:_FeatureCollection"/>
<complexType name="CityType">
 <complexContent>
  <extension base="gml:AbstractFeatureCollectionType">
   <sequence>
    <element name="topologyModel"
type="gml:TopoComplexMemberType" minOccurs="0"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
<element name="Lake" type="app:LakeType"
substitutionGroup="gml:_Feature"/>
<complexType name="LakeType">
 <complexContent>
  <extension base="gml:AbstractFeatureType">
   <sequence>
    <element name="extent" type="gml:SurfacePropertyType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

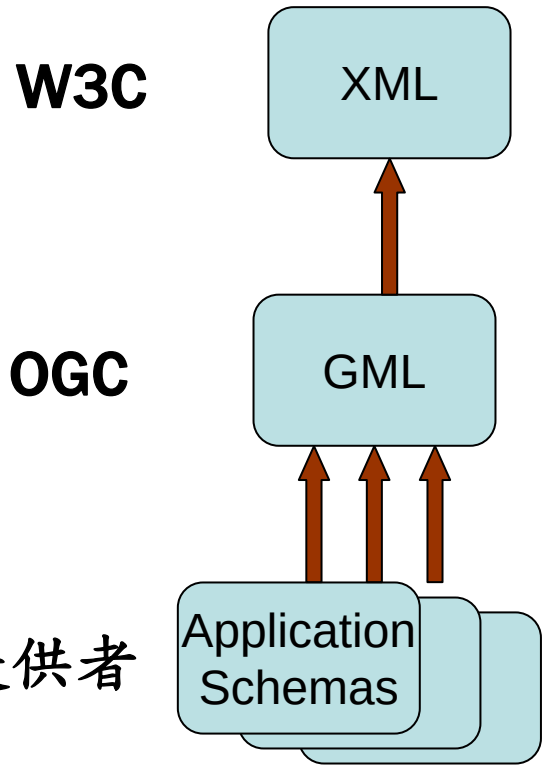

City
gml:featureMember
0..∞
gml:featureMembers
gml:FeatureArrayPropertyType
gml:_Feature
0..∞
Building
0..∞
Lake
0..∞
Park
0..∞
_Street
0..∞
BridgeSegment
Street
StreetSegment
gml:FeatureCollection
0..∞
gml:Observation
0..∞
gml:DirectedObservation
gml:DirectedObservationAtDist...
gml:_Coverage
0..∞
LandImage
gml:_ContinuousCoverage
gml:_DiscreteCoverage
gml:GridCoverage
gml:MultiCurveCoverage

Institute of Information Science
Academia Sinica
中央研究院 資訊科學研究所

What's GML

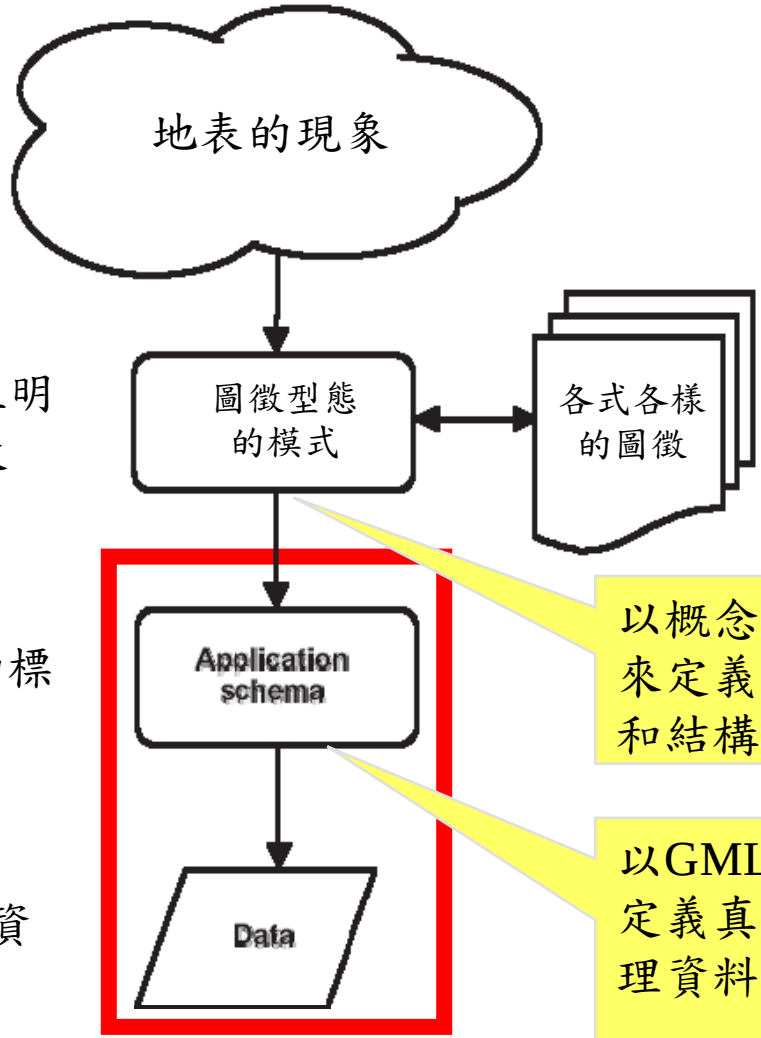
GML 是一種資料格式

- GML 是XML的一種擴充。



修改自 www.flakesoftware.co.uk

一份GML文件來自於...



對於真實世界，以廣泛且明確的方式來抽取地理圖徵

以GML schemas定義application schemas中的標準元素和型態之使用

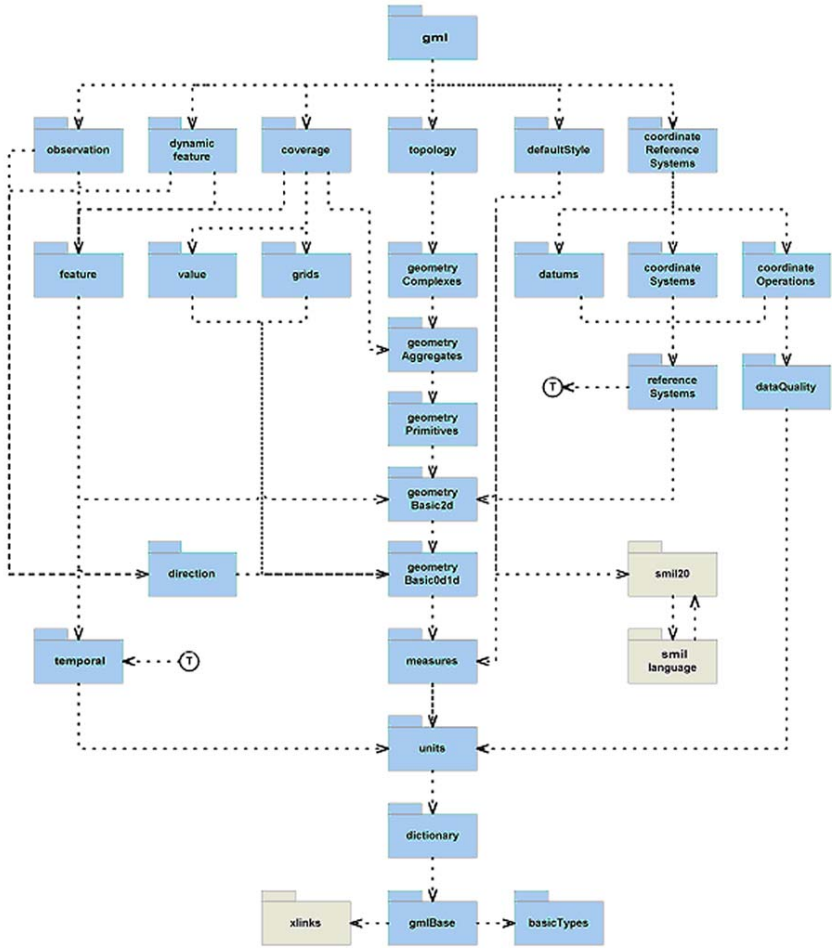
根據application schema資料有邏輯結構

以概念性的schema language來定義資料的內容和結構，如UML和ER model

以GML Application Schema定義真實世界中的物件，使地理資料成為GML文件

GML Schemas

- GML 3.1.1中包含29 個 GML Schemas。
- GML Schemas 只有基礎地理空間定義， 如feature, geometry, topology...， 且這些schemas是平行的且不針對任何一個應用領域。



GML Schemas

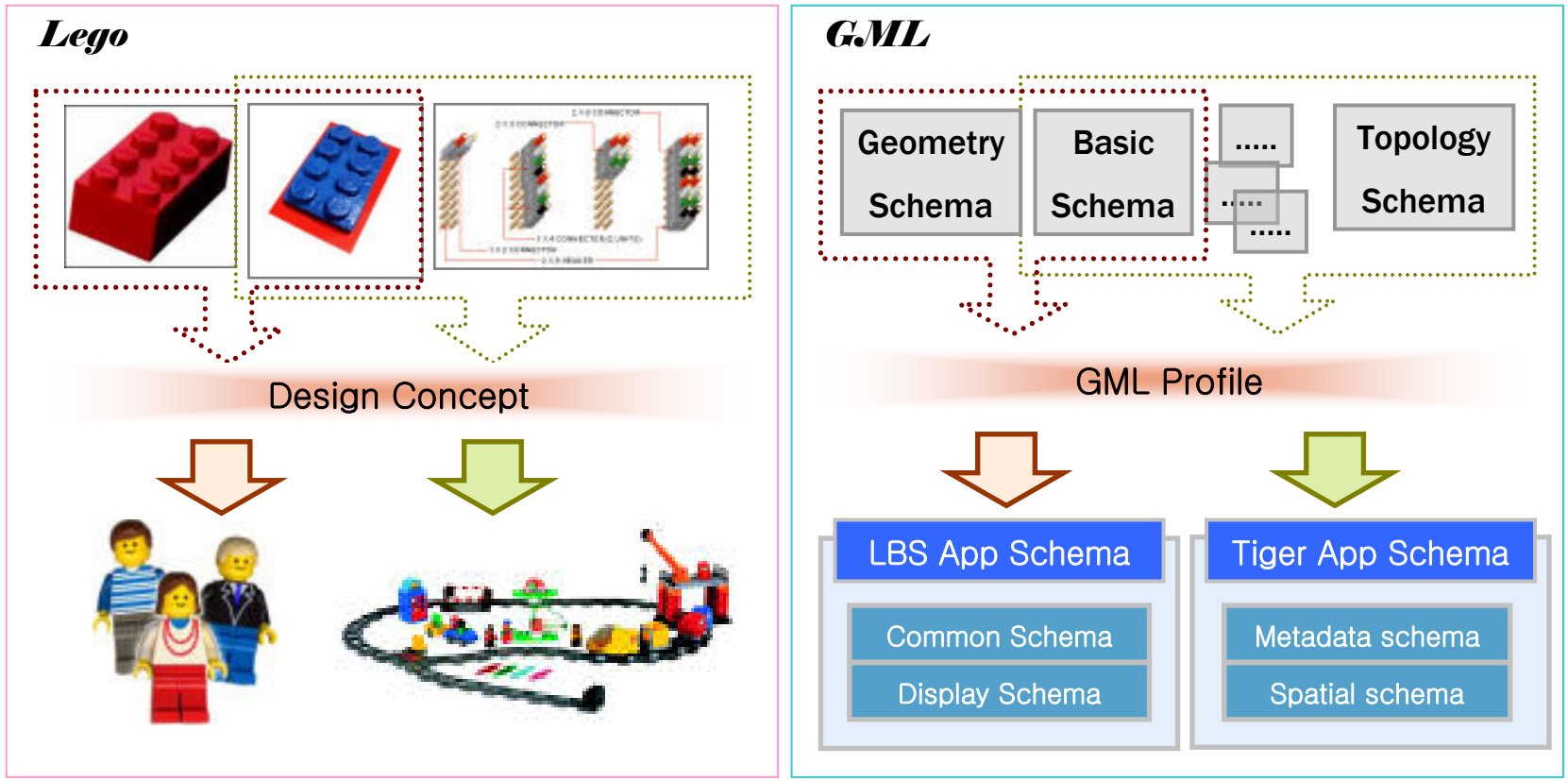
- 基礎的 schemas、 一般的語法(syntax)
- 圖徵模式(Feature Model)
- 基本幾何圖元定義 (0d, 1d, 2d)
- 由附加的幾何圖元(Additional geometric primitives) (0d, 1d, 2d, 3d)
- 幾何組成(Geometric composites)
- 幾何聚集(Geometric aggregates)
- 座標參考系統(Coordinate reference systems)
- 詮釋資料(metadata)機制

GML Schemas

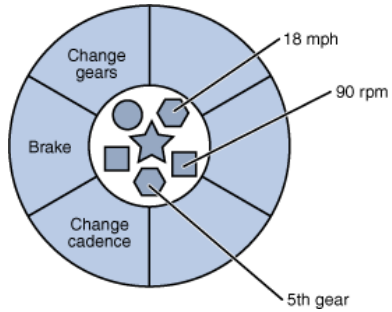
- 撲拓關係(Topology)。
- 時間資訊和動態圖徵。
- 資料典(Definitions and dictionaries)
- 單位、量測值(Unit of Measure)。
- 方向(Directions)
- 觀察資料(Observations)
- 網格式資料(Coverages)
- 預設樣式(Default styling)

GML Application Schema

- GML Application Schema是一個組件式的架構, 如樂高

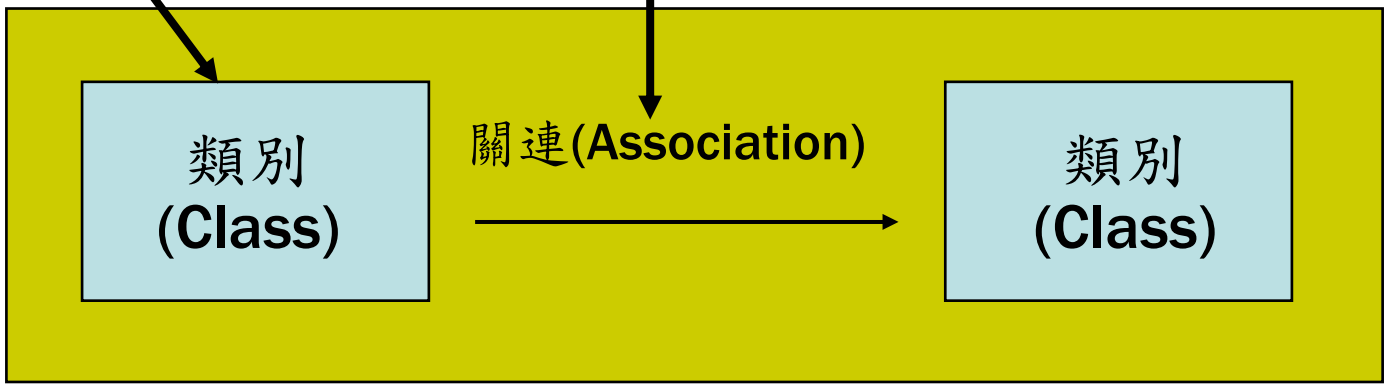


GML 模型



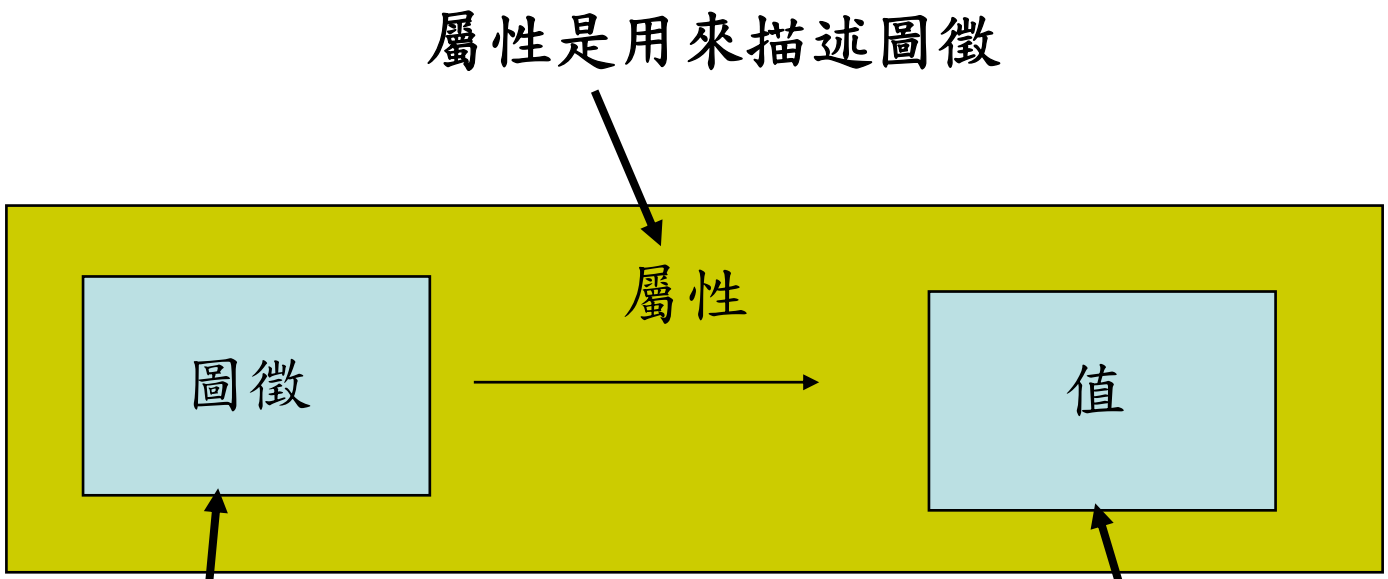
真實世界中的
“實體”或“現象”

類別之間的關連
(其名稱的角色是描述之間的關係)



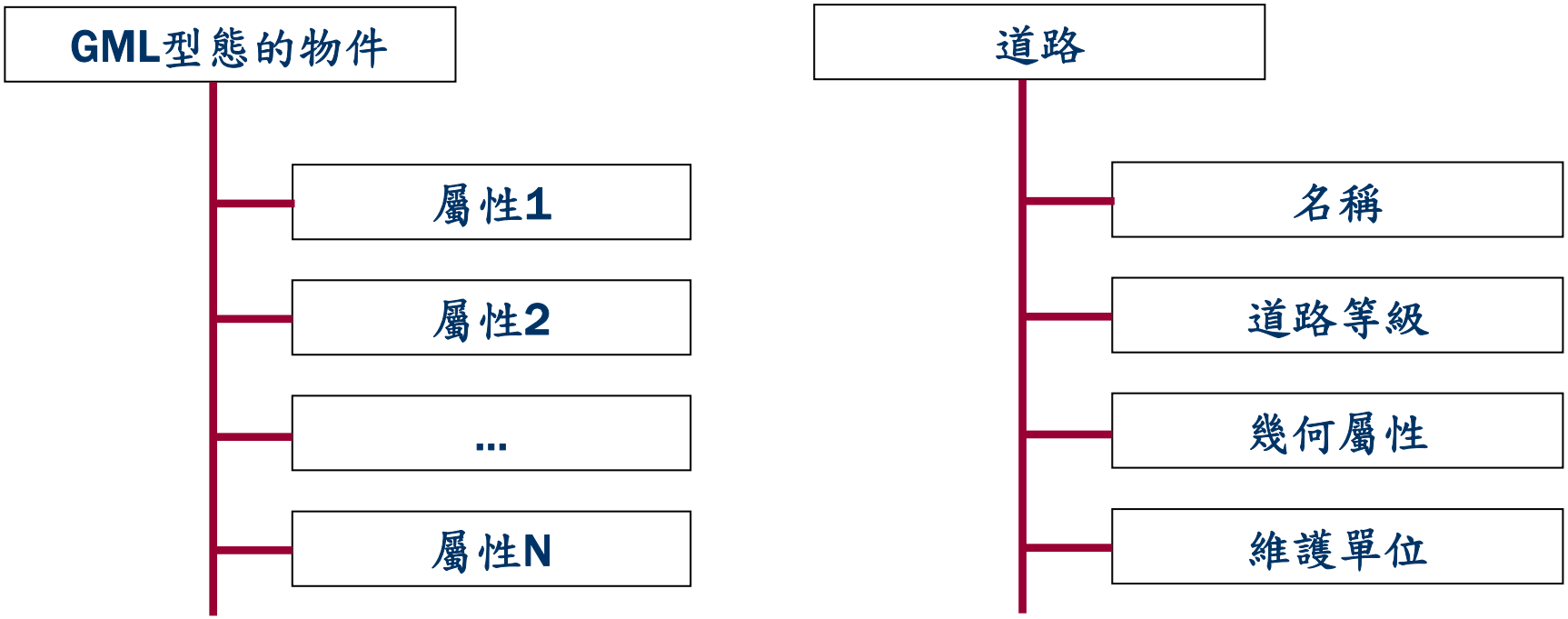
Ex.	忠孝東路	交叉	新生北路
	台北市	包含	信義區
	都市發展局	屬於	台北市政府

GML 模型



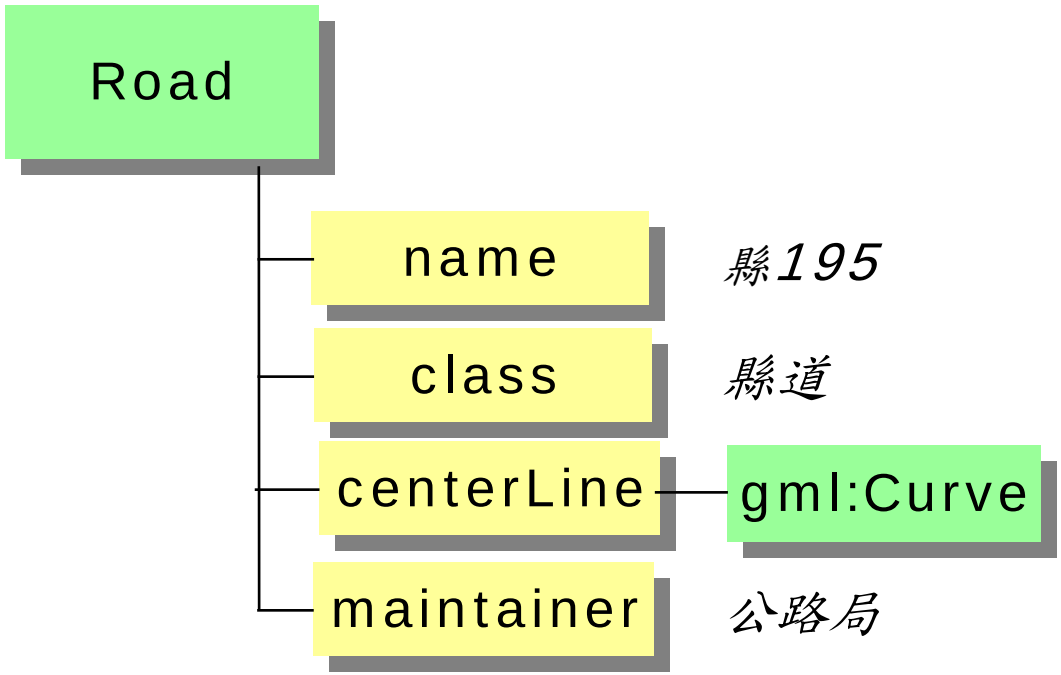
Ex.　台北101　　高度　　584公尺

GML 模型



GML物件是由一系列的屬性來描述。

模式化的圖徵型態

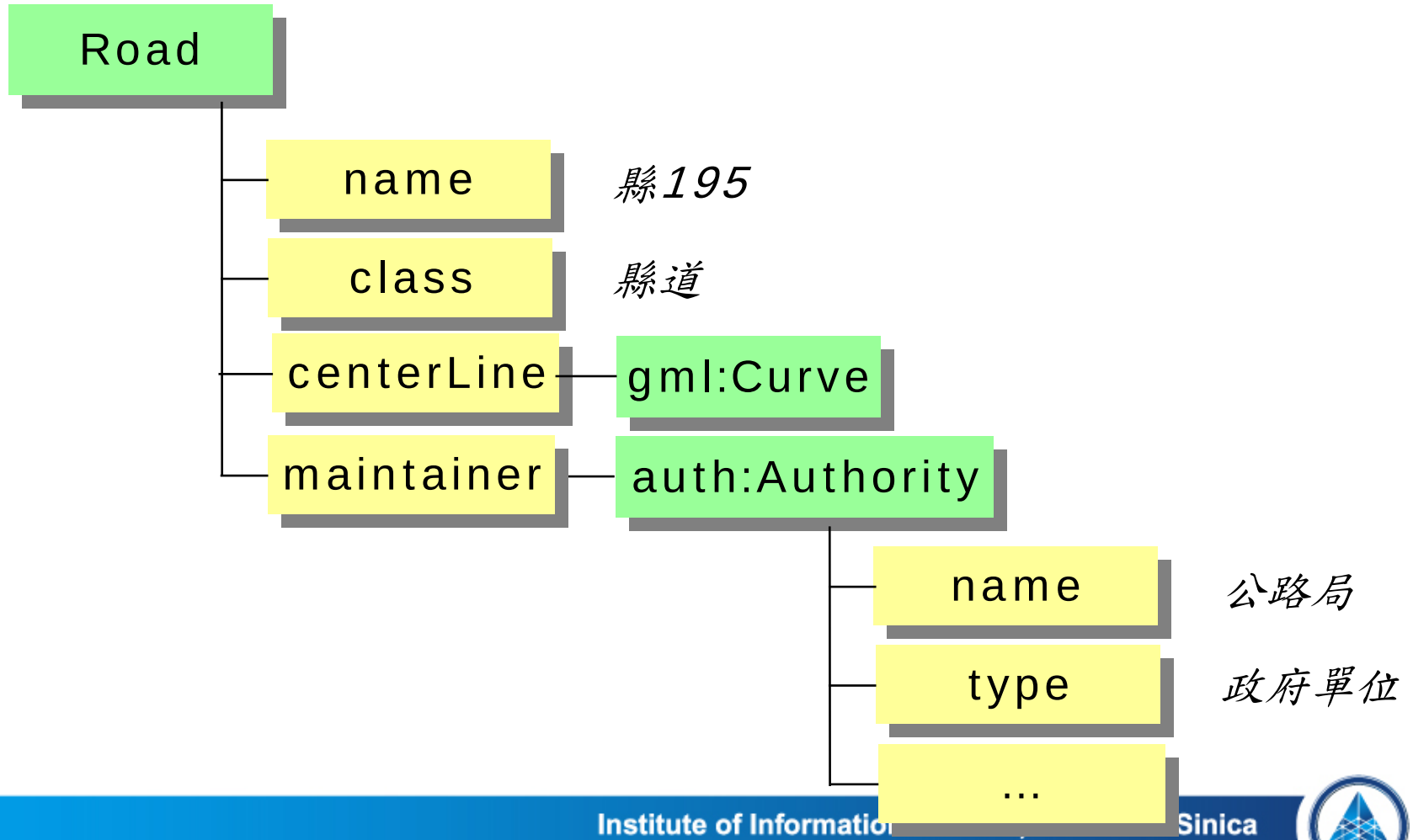


建構一個資訊社群→ 達到語彙(圖徵型態和它們的屬性)一致。

模式化的圖徵型態

```
<Road gml:id="CR195">
        <name>縣195</name>
        <class>縣道</class>
        <centerLine>
                <gml:Curve>...</gml:Curve>
        </centerLine>
        <maintainer>公路局</maintainer>
</Road>
```

模式化的圖徵型態


Road
name
縣195
class
縣道
centerLine
gml:Curve
maintainer
auth:Authority
name
公路局
type
政府單位
...

模式化的圖徵型態

```
<Road gml:id=" CR195">
        <name>縣195</name>
        <class>縣道</class>
        <centerLine>
                <gml:Curve>...</gml:Curve>
        </centerLine>
        <maintainer>
                <auth:Authority gml:id= "AU028">
                        <name>公路局</name>
                        <type>政府單位</type>
                </auth:Authority>
        </maintainer>
</Road>
```

模式化的圖徵型態

```
<Road gml:id= "CR195">
        <name>縣195</name>
        <class>縣道</class>
        <centerLine>
                <gml:Curve>...</gml:Curve>
        </centerLine>
        <maintainer xlink:href="urn:x-auth:AU028"/>
</Road>
```

- 屬性的子元素亦可是"物件"，且可透過**xlink:href**來取得。
- **xlink:href**以連結方式來關連其它物件的屬性值。
- 所要關連的物件可以是同一份**GML**文件的一部份或任何網際網路和內部網路之中。

GML 模型

基礎模型是圖徵型態/屬性

<圖徵型態名稱>
    <屬性名稱1>
        <型態名稱> ... </型態名稱>
    </屬性名稱1>
    ...
    <屬性名稱N>
    ...
    </屬性名稱N>
</圖徵型態名稱>

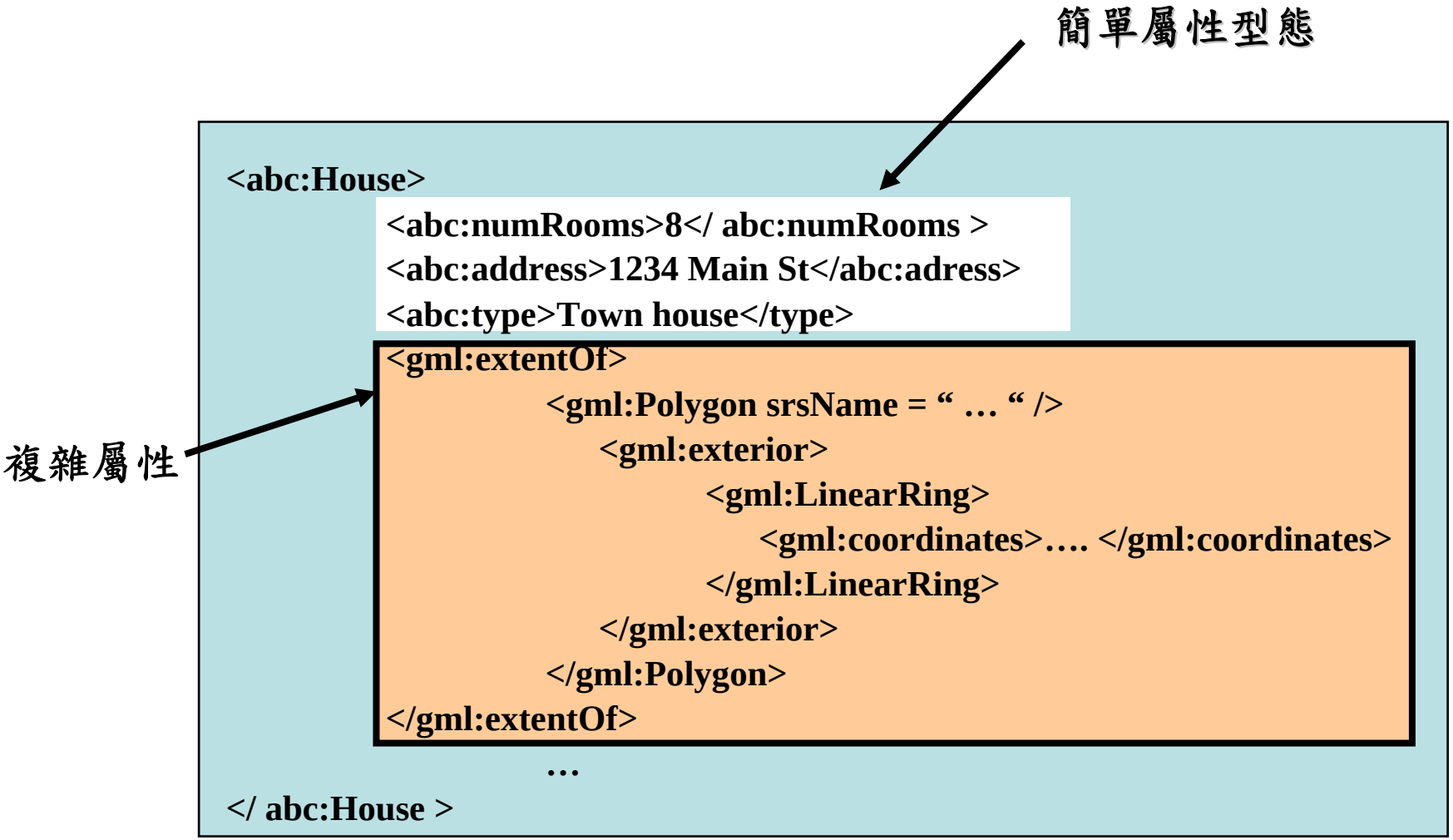
GML 模型

```
<圖徵型態名稱>
        <屬性名稱1>
                <型態名稱>...</型態名稱>
        </屬性名稱1>
        <屬性名稱2>
                <型態名稱>
                        </屬性名稱>
                        ...
                        </屬性名稱>

                </型態名稱>
        </屬性名稱2>
                ...
        <屬性名稱N>
                ...
        </屬性名稱N>
</圖徵型態名稱>
```

屬性/型態可被巢狀化，可無限延伸至你想要

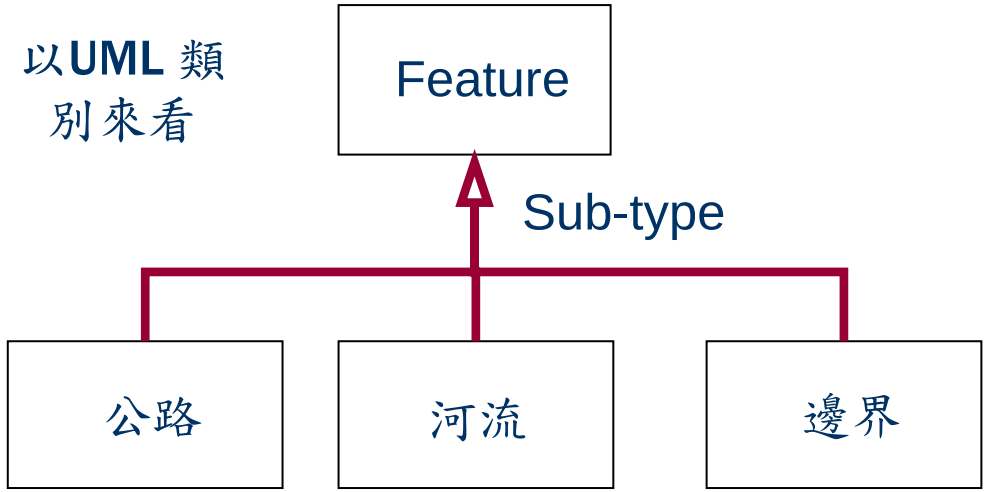
GML 模型


簡單屬性型態
複雜屬性
<abc:House>
<abc:numRooms>8</ abc:numRooms >
<abc:address>1234 Main St</abc:adress>
<abc:type>Town house</type>
<gml:extentOf>
<gml:Polygon srsName = " … " />
<gml:exterior>
<gml:LinearRing>
<gml:coordinates>…. </gml:coordinates>
</gml:LinearRing>
</gml:exterior>
</gml:Polygon>
</gml:extentOf>
…
</ abc:House >

GML – 圖徵

在此的圖徵是一個有意義的物件或觀念

以UML 類別來看



公路、河流和邊界是圖徵的子類別(subclass)

An example of feature collection

```
<Rivers>
 <name>淡水河</name>
 <gml:featureMember>
  <Segment id="001">
   <gml:centerLineOf>
    <gml:curveProperty>
     ...
    </gml:curveProperty>
   </gml:centerLineOf>
  </Segment>
 </gml:featureMember>
 <crossArea>
  <Towns>三重市</Towns>
  <Towns xlink:type = “simple” xlink:href = “#淡水鎮”/>
 </crossArea>
</Rivers>
```

The GML application schema for above example

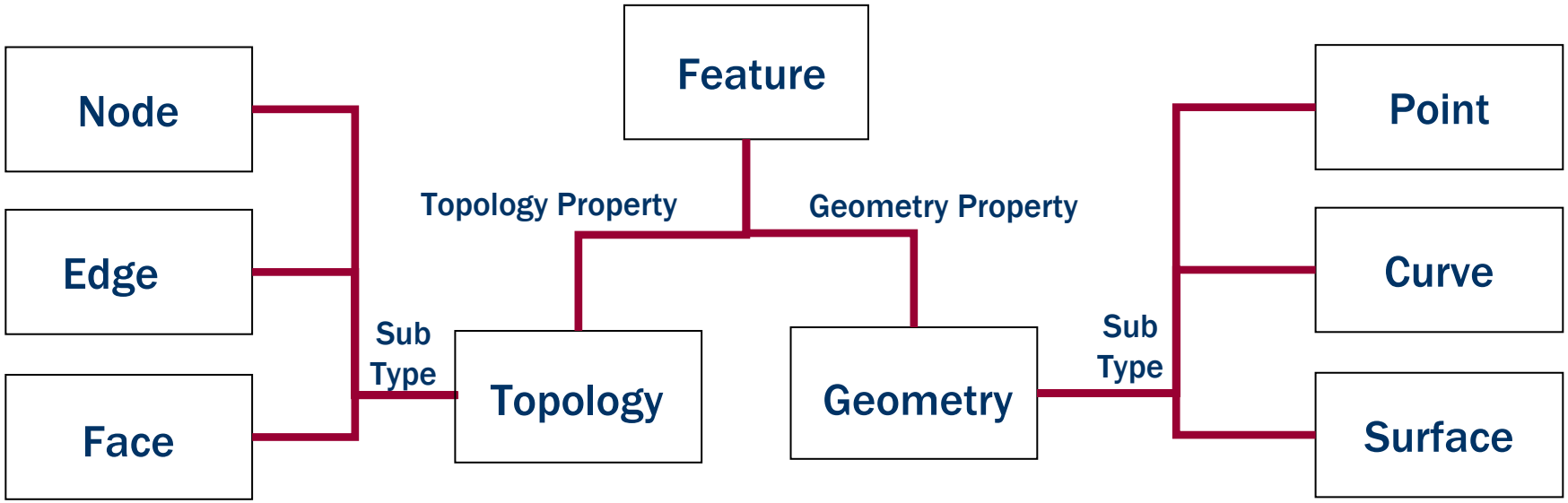
```
<element name="Rivers" type="Rivers"
substitutionGroup="gml:_FeatureCollection"/>
<complexType name="Rivers">
 <complexContent>
  <extension base="gml:AbstractFeatureCollectionType">
   <sequence>
    <element name="name" type="string"/>
    <element name="gml:featureMember"/>
    <element name="crossArea" type="crossAreaType"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

基於FeatureCollection的GML文件

```
<abc:FeatureCollection>
    <abc:featureMember>
        < .. 一些圖徵 .. >
    </abc:featureMember>
    <abc:featureMember xlink:href="http:// ..."/>
    <abc:featureMember>
        < .. 一些圖徵 .. >
    </abc:featureMember>
    <abc:featureMember>
        < .. 圖徵集合中的其它屬性 .. >
</abc:FeatureCollection>
```

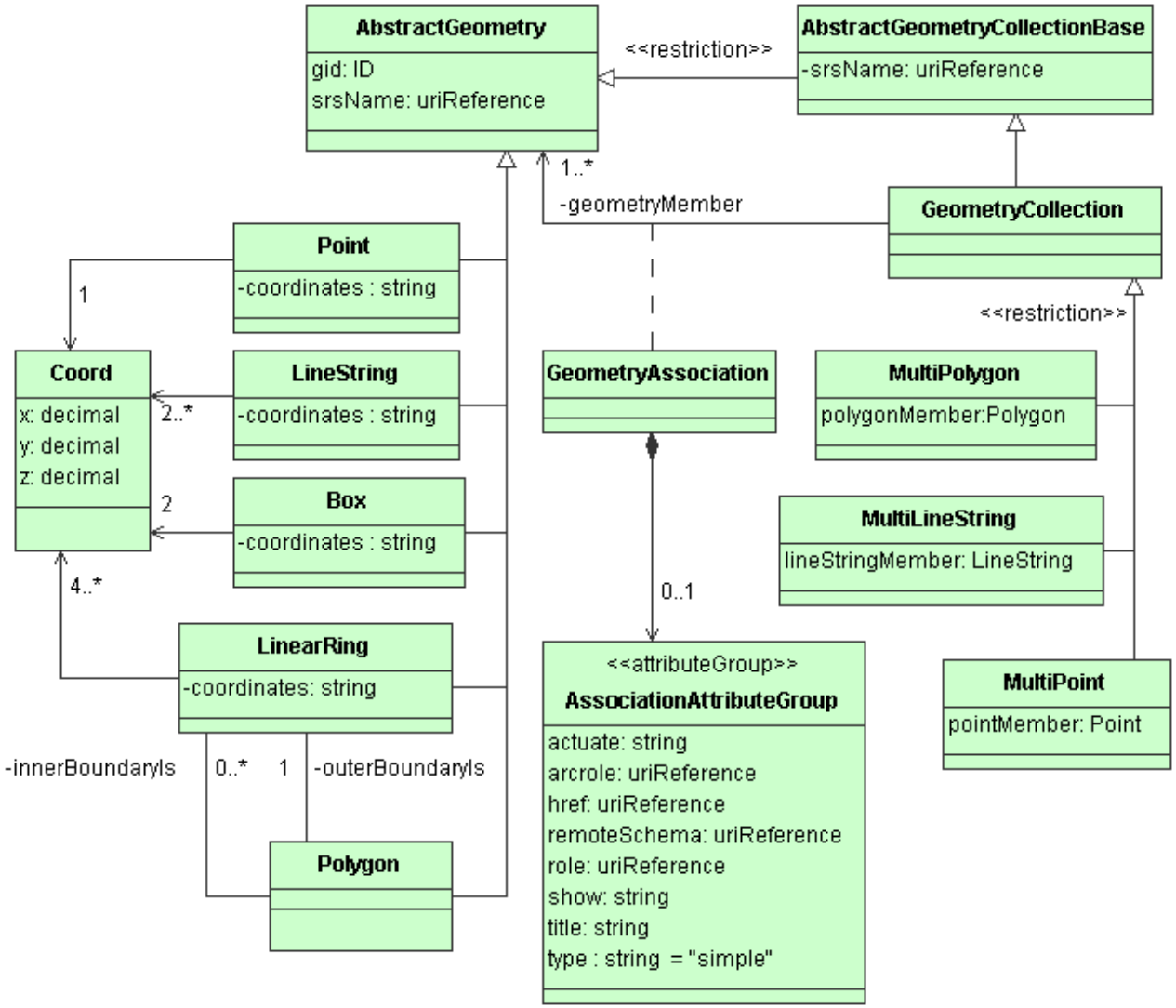
其中 **<abc:FeatureCollection>** 是一個元素(element)，它的內容模式(content model)是由 **gml:AbstractFeatureCollectionType**衍生。

模式化地理資料 – 圖徵(Features)



圖徵可有3D幾何或拓撲屬性。

GML 幾何模型


AbstractGeometry
gid: ID
srsName: uriReference
<<restriction>>
AbstractGeometryCollectionBase
-srsName: uriReference
1..*
-geometryMember
GeometryCollection
Point
-coordinates : string
1
Coord
x: decimal
y: decimal
z: decimal
LineString
-coordinates : string
2..*
Box
-coordinates : string
2
4..*
LinearRing
-coordinates: string
-innerBoundaryIs
0..*
1
-outerBoundaryIs
Polygon
GeometryAssociation
0..1
<<attributeGroup>>
AssociationAttributeGroup
actuate: string
arcrole: uriReference
href: uriReference
remoteSchema: uriReference
role: uriReference
show: string
title: string
type : string = "simple"
<<restriction>>
MultiPolygon
polygonMember:Polygon
MultiLineString
lineStringMember: LineString
MultiPoint
pointMember: Point

幾何屬性

GML提供幾何屬性的想法。

幾何屬性是“幾何的值”(“geometry-valued”.)

屬性名稱描述在圖徵關係中屬性角色

e.g.

```
<百貨公司 gml:id = “台北101”>
        <gml:centerOf>
                <gml:Point srsName = “...” > ...
                </gml:Point>
        </gml:centerOf>
</百貨公司>
```

GML圖徵幾何

```
<tgml:建物 gml:id = “台北101”>
        <tgml:樓層數>101</tgml:樓層數>
        <tgml:土地使用>商業區</tgml:土地使用>
        <tgml:面積 uom=“#m2”>100000</tgml:面積>
        <tgml:相鄰街道>市府路</tgml:相鄰街道>
        <tgml:街號>45</tgml:街號>
        <gml:location>
                <gml:Point srsName = “ … “>
                        <gml:coordinates>55661.1454,
                                454656.67</gml:coordinates>
                </gml:Point>
        </gml:location>
</tgml:建物>
```

GML Namespace

幾何屬性

GML提供一些已定義的幾何屬性值：

點(point-valued)屬性

- centerOf
- location
- position

線(curve-valued)屬性

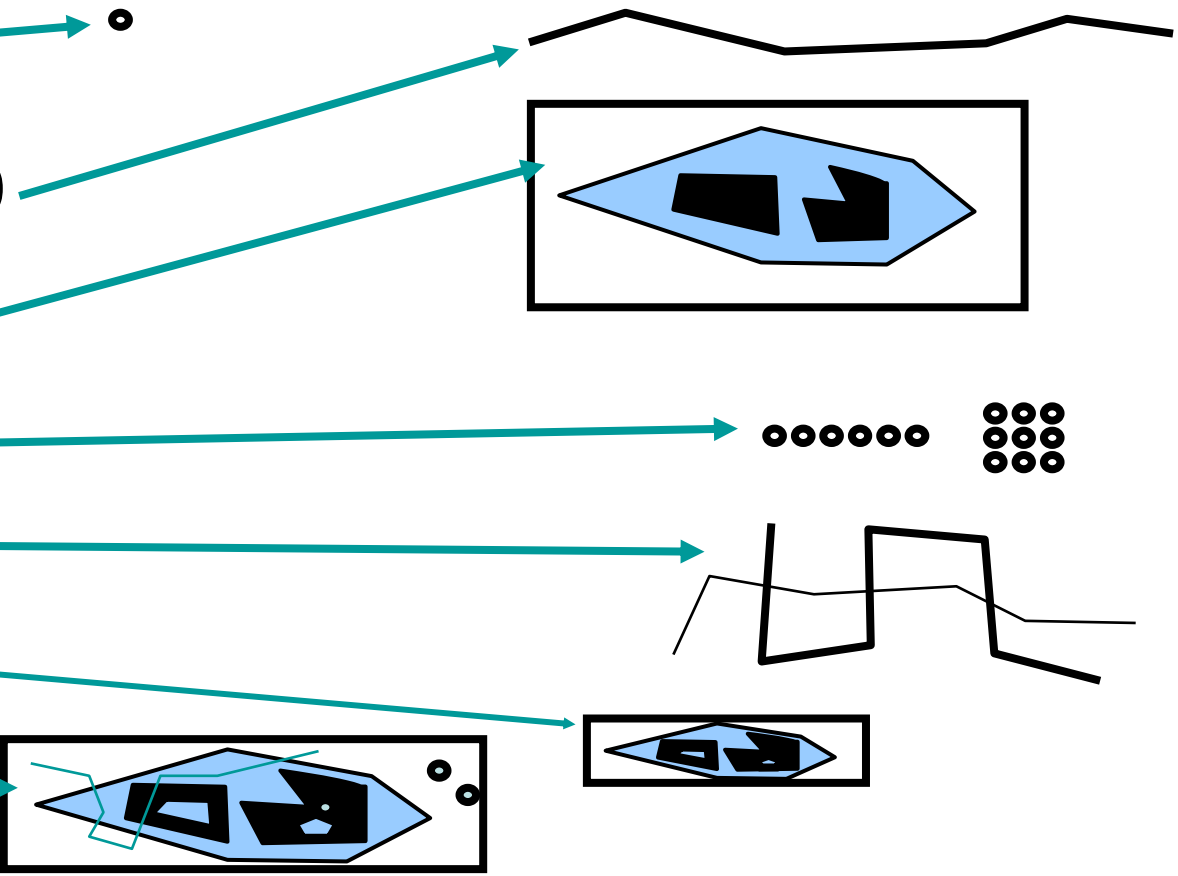
- centerLineOf
- edgeOf

面（Surface Valued）屬性

- extentOf
- coverage

一般被使用的 GML幾何類別

- Point
- Line String (linear)
- Polygon (linear)
- MultiPoint
- MultiLineString
- MultiPolygon
- MultiGeometry



Point

```
<gml:pointArrayProperty>
    <gml:Point>
        <gml:coordinates>311153,2770508</gml:coordinates>
    </gml:Point>
</gml:pointArrayProperty
```

Line

```
<gml:curveProperty>
 <gml:Curve>
  <gml:segments>
   <gml:LineStringSegment>
    <gml:coordinates>311176,2770535
311177,2770530..</gml:coordinates>
   </gml:LineStringSegment>
  </gml:segments>
 </gml:Curve>
</gml:curveProperty>
```

Polygon

```
<gml:polygonProperty>
 <gml:Polygon>
  <gml:exterior>(若是中空的多邊形之內框，則用gml:interior)
   <gml:Ring>
    <gml:curveMember>
     <gml:Curve>
      <gml:segments>
       <gml:LineStringSegment>
        <gml:coordinates>311030,2770504 .. 311030,2770504
</gml:coordinates>(第一個座標值和最後一個座標值是相同)
       </gml:LineStringSegment>
      </gml:segments>
     </gml:Curve>
    </gml:curveMember>
   </gml:Ring>
  </gml:exterior>
 </gml:Polygon>
</gml:polygonProperty>
```

Some examples

- The ESRI's GML document
- The example of GeoWeb Workshop
- 台北市政府地籍資料

簡報結束
敬請提問

本投影片採用CC授權(姓名標示-非商業性-相同方式分享)
http://creativecommons.org/licenses/by-nc-sa/2.5/tw/