

UML – GML- Classi di Oggetti

Claudio Rocchini
Istituto Geografico Militare



Introduzione

- Per lavorare nel GIS serve sapere anche queste cose?
- Sì, perché:
 - I dati geografici verranno scambiati nel formato GML o forniti da servizi di rete basati su GML
 - Le specifiche del progetto INSPIRE sono UML-XML-GML
 - Le specifiche dei metadati e i dati di scambio obbligatori per il CNIPA sono in XML-XSD
 - Per progettare un PersonalGeoDB di ArcGIS con Visio bisogna utilizzare UML
 - Eccetera



UML

- Unified Modeling Language
- Linguaggio di modellazione generico *standardizzato*
- E' un linguaggio per lo più *grafico*: le descrizioni in questo linguaggio sono rappresentate da schemi e diagrammi
- Nasce per la progettazione di software ma l'utilizzo si è esteso ad altri campi (GIS)



Non tutto l'UML

- L'UML si divide in vari tipi di diagramma, es.:
 - Struttura
 - **Classi di Oggetti**
 - Componenti
 - Pacchetti (Package)
 - Installazione (Deployment)
 - Comportamento
 - Casi d'uso
 - Interazione
 - Attività
- Noi ci occuperemo solo delle Classi di Oggetti



UML e Orientamento agli Oggetti

- Spesso i simbolismi dell'UML servono a rappresentare concetti che derivano dagli Oggetti
- Per questo via via che introduciamo gli schemi UML, daremo una breve descrizione della struttura orientata agli oggetti associata
- *“Non ci si faccia prendere dal panico! Tutto sarà chiaro al momento opportuno”*. B. Stroustrup, C++ terza edizione



Classe

- La classe (di oggetti) descrive la nostra entità ed è rappresentata con un rettangolo con in testa il nome della classe
- Di seguito sono elencati gli attributi (in questo è simile agli schemi ER delle basi di dati)
- Un terzo riquadro riporta i metodi (le operazioni possibili) della classe: in questa sede non ne parleremo

Persona
-nome 1: string -cognome 1 -indirizzo [0..1] -corsi [0..n] : lista



Attributi di una Classe

- Gli attributi hanno un nome (come ER)
- La specifica può essere più o meno dettagliata
- Gli attributi possono avere un tipo di dato (come ER)
- Il tipo di dato è semplice (numeri, parole), oppure può essere un'altra classe (nelle basi di dati questo non è possibile). Es. in GML la classe polilinea ha un attributo coordinate che è una lista di punti (altra classe)
- Gli attributi hanno una cardinalità minima e massima (non presente nei DB)



Cardinalità degli attributi

- Possibili Cardinalità:
 - 1 : esattamente un'istanza, attributo obbligatorio semplice
 - 0..1: zero od una istanza: attributo opzionale semplice
 - 0..n: zero o più istanze: lista di elementi opzionale
 - 1..n: una o più istanza: lista di elementi, almeno uno è obbligatorio



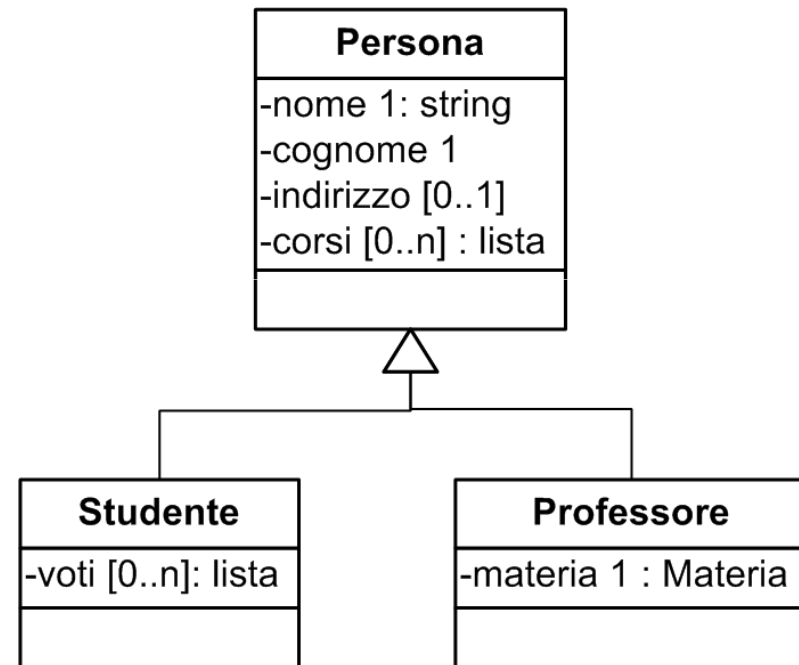
Generalizzazione

- Concetto base dell'Object Oriented
- Indica una relazione fra due classe, in cui una è considerata una specializzazione dell'altra
- Se una classe B (specifica) è una specializzazione di una più generale A, vuol dire che B è anche un A (oppure B si può sostituire ad A)
- La classe specifica *eredita* tutte le proprietà della classe generale (attributi e funzionalità)



Es. Generalizzazione

- Le classi studente e professore sono classi specializzate di persone
- Questo si traduce nel fatto che uno studente (od un professore) sono anche persone
- Studente e professore ereditano gli attributi di persona (nome, cognome,...)
- In UML la generalizzazione si rappresenta con una freccia dalla testa bianca.



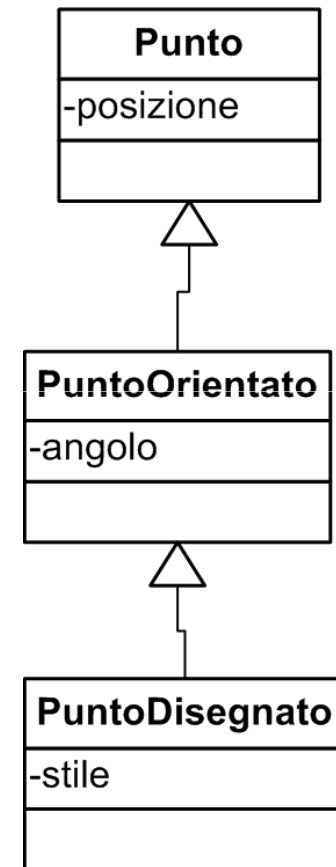
Generalizzazione

- La generalizzazione permette di specificare le proprietà comuni di una serie di oggetti, attraverso la definizione di una classe generale
- Bisogna ricordare che una classe eredità tutte le proprietà delle classi padri
- La strutturazione a gerarchie di classi è molto comune in programmazione e nella definizione delle primitive geometriche (GML)



Es. Generalizzazione

- La classe punto ha l'attributo posizione
- La classe PuntoOrientato specializza punto, aggiungendo l'attributo angolo
- Un PuntoOrientato è anche un Punto ed eredita da questo l'attributo posizione
- Il PuntoDisegnato ha l'attributo stile ed eredita la posizione da Punto e l'angolo da PuntoOrientato



Es. Reale Generalizzazione

- Nelle specifiche GML appare spesso la generalizzazione
- Addirittura, tutti gli oggetti GML sono figli di un unico oggetto padre *GM_Object*
- Nell'esempio a destra la classe *punto* è una specializzazione della classe *primitiva* geometrica
- Il *punto* ha l'attributo posizione, ma anche tutte le proprietà di *primitiva* e di tutti i suoi antenati

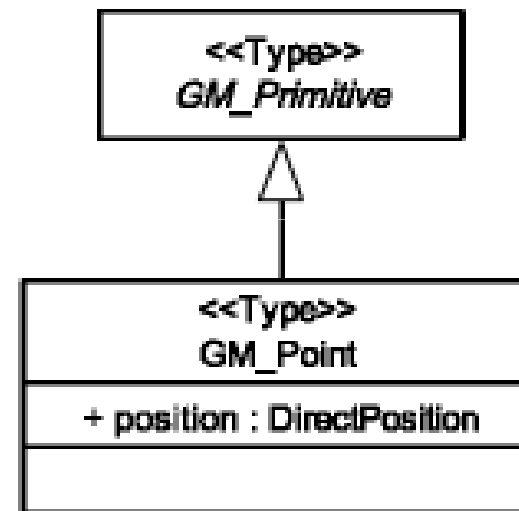


Figure D.5 — Point



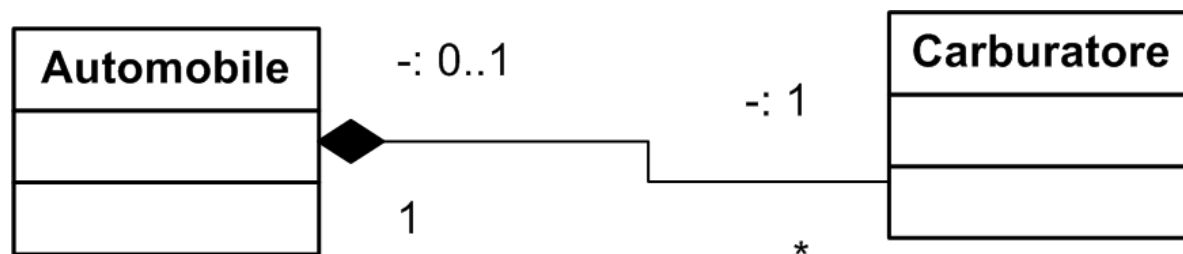
Composizione

- La composizione è un'altra relazione fra classi
- Indica che una classe padre è composta da una o più classi figlio
- La composizione lega classi fortemente interconnesse
- La classe figlio di solito, non può essere utilizzata senza la classe padre (life cycle dependency)
- In UML la relazione di composizione è rappresentata da una freccia che ha come testa una losanga nera
- La relazione ha di solito indicata la cardinalità (minima e massima) degli elementi in gioco



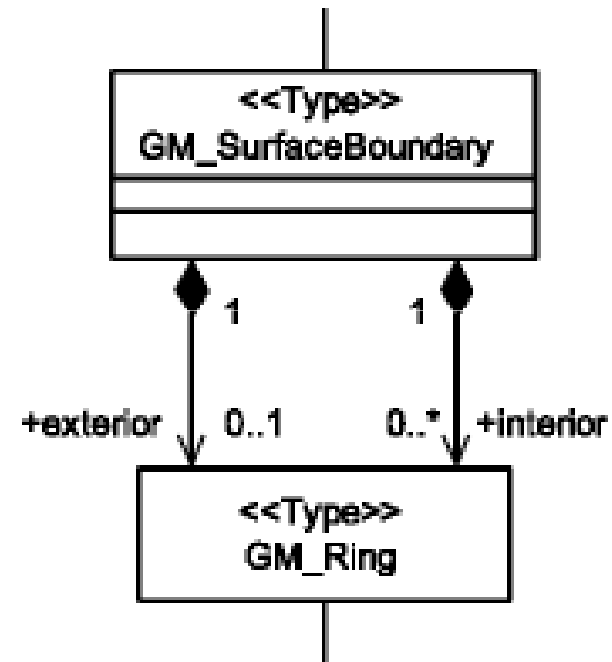
Es. Composizione

- Una automobile è composta (anche) da uno ed un solo carburatore, mentre un carburatore può o meno essere inserito in una macchina
- Il carburatore di per sé non a funzionalità se non per comporre macchine
- Nota: la generalizzazione non c'entra: un carburatore non è una automobile



Es. Composizione GML

- In GML, la classe *bordo* (di area) è composta da un ed un solo *anello* esterno, e da eventualmente zero o più anelli interni (che rappresentano i buchi)
- Per *anelli* si intende curve chiuse rappresentate da polilinee
- Di per sé gli *anelli* non hanno funzione, se non quella di definire i *bordi*



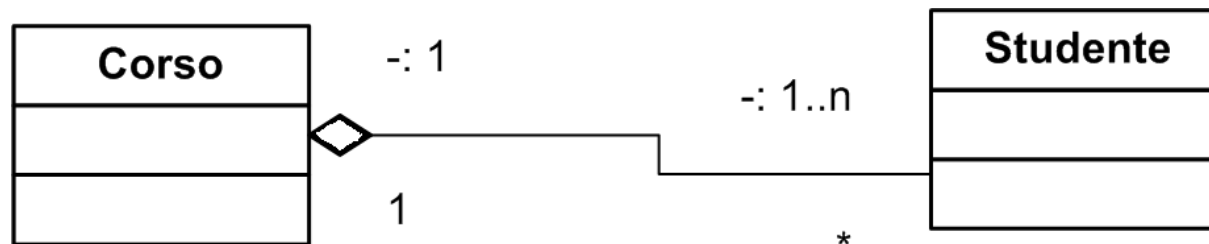
Aggregazione

- L'aggregazione è una relazione fra classi che indica il formare una classe come aggregazione di altre classi
- La relazione fra classi coinvolte è minore (della composizione): la classe facente parte dell'aggregazione ha funzionalità di per sé, anche senza la classe aggregante
- Tipico esempio di queste classi sono le classi “contenitore”
- Anche per questa relazione viene specificata la cordialità minima e massima
- In UML l'aggregazione viene rappresentata da una freccia con in testa una losanga bianca



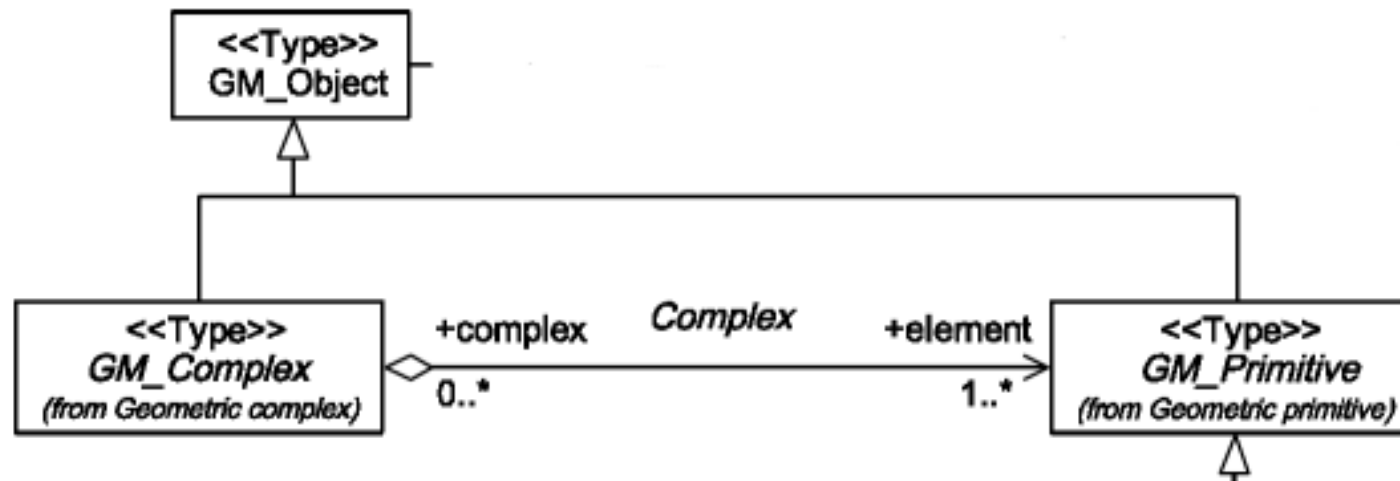
Es. Aggregazione

- L'aggregazione di uno o più studenti forma un corso
- La classe studente però ha funzionalità di per sé (es. amministrativa)
- Ovviamente, uno studente non è un corso (ma al massimo fa parte di un corso)



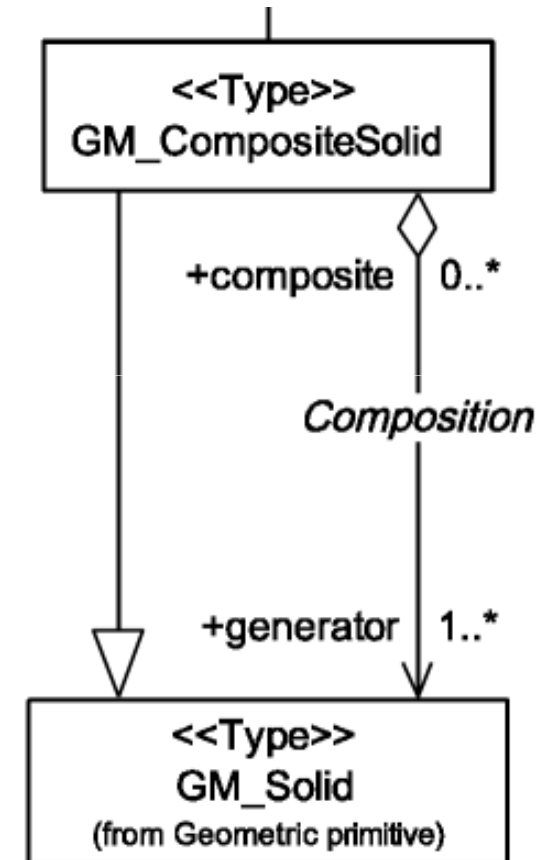
Es. Aggregazione GML

- Un oggetto *complesso* (una multi-area) è l'aggregazione di più *primitive* geometriche (aree)
- Allo stesso tempo, il complesso e la primitiva sono oggetti GM (sono specializzazioni diverse di *oggetto*)



Es. Aggregazione GML (2)

- Ancora più difficile:
- Un solido Composto è l'aggregazione di più solidi semplici (freccia a losanga “in su”)
- Allo stesso tempo, il solido composto è una specializzazione del solido semplice (è anche un solido, freccia triangolare “in giù”)



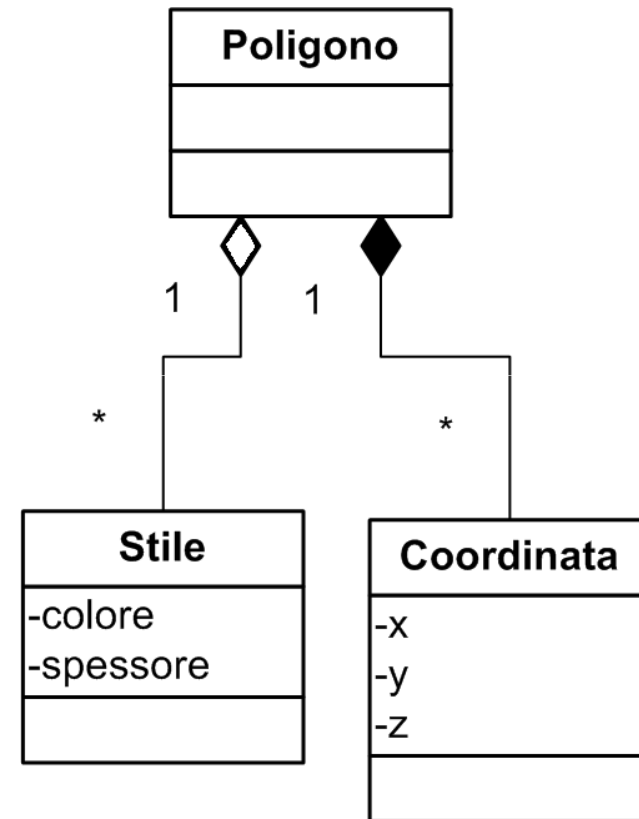
Composizione VS Aggregazione

- I due concetti possono facilmente essere confusi
- Nella *composizione* una classe appartiene ad un solo tutto (un carburatore sta in una macchina), *nell'aggregazione* un oggetto può far parte di molti aggregati
- Nella composizione il ciclo di vita degli oggetti è collegato: se l'oggetto contenitore non serve più, non servono più neanche le parti



Composizione + Aggregazione

- Il poligono è *composto* da coordinate puntuali, che sono incluse in un solo poligono. Le coordinate servono solo a definire il poligono
- Ad un poligono è *aggregato* un certo stile di visualizzazione, che può essere utilizzato anche per altre figure geometriche: lo stile ha funzionalità di per sé



Associazione

- Due classi possono essere legate in modo più generico dal concetto di associazione generica che non è una generalizzazione, ne una composizione ne una aggregazione
- Come le altre relazioni è specificata anche dalla sua cardinalità
- L'associazione si rappresenta in UML da una freccia semplice



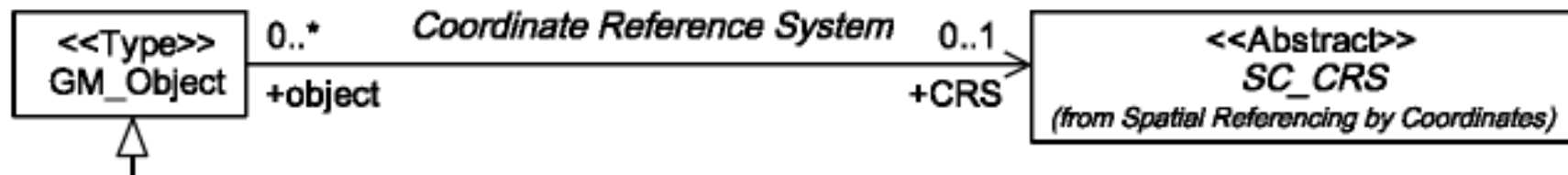
Es. Associazione

- Ad un ufficio IGM è associato un ed un solo CapoUfficio (questa relazione ricorda le basi di dati)



Es. Associazione GML

- Ad ogni oggetto GM (oggetto geometrico) è associato un oggetto *Sistema di Riferimento*
- Questo vuol dire che tutte le forme geometriche GML sono associate ad un sistema di coordinate, come è in tutti i dati GIS moderni
- Questo tipo di relazione assomiglia a quelle tipiche delle basi di dati

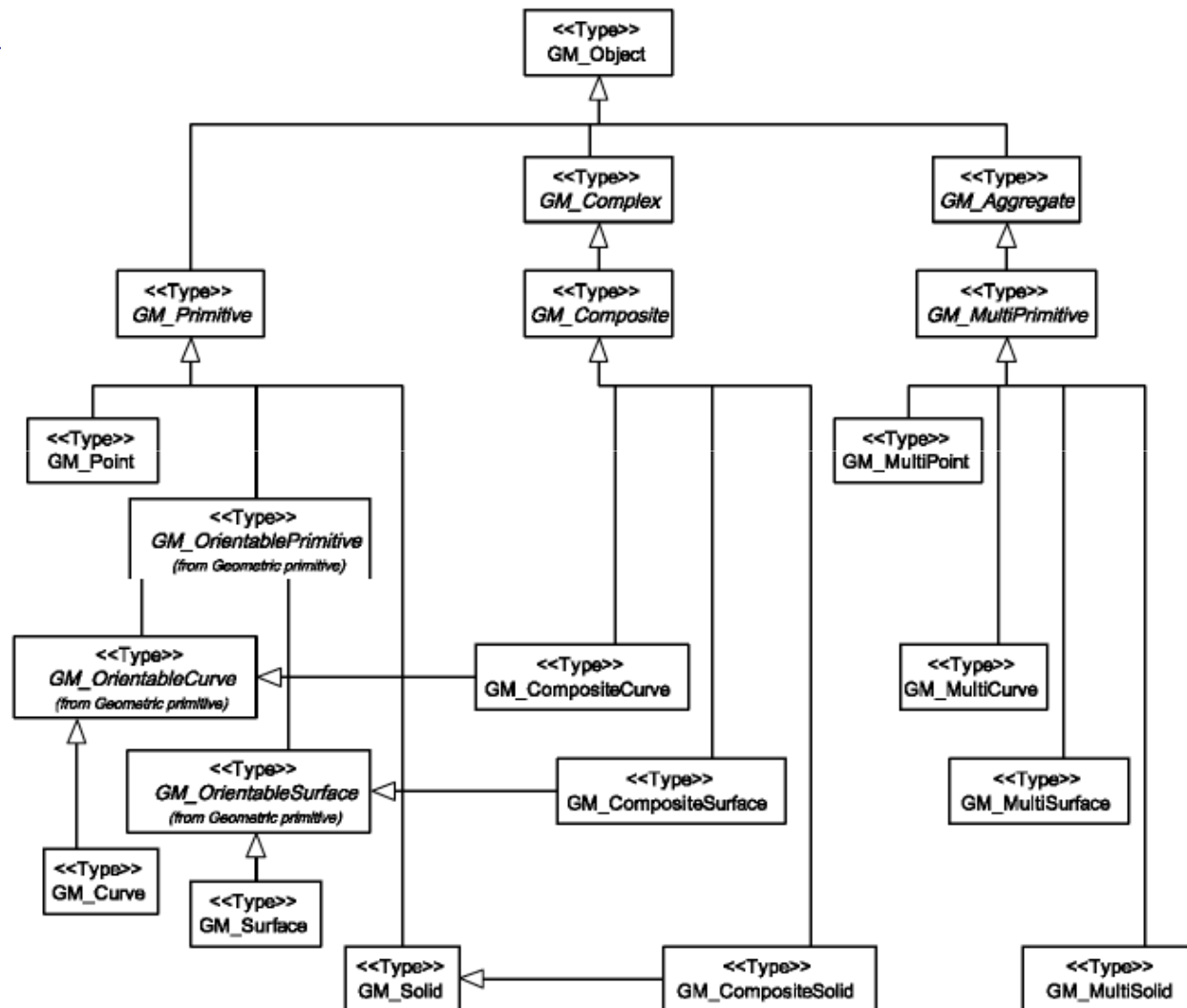


Esempi GML

- Di seguito vediamo alcuni esempi della struttura GML
- Cerchiamo di ritrovare, in alcune parti dei diagrammi UML, i concetti appena descritti
- Questa non vuole essere un'introduzione esaustiva al GML, ma solo un esempio di come si utilizza UML



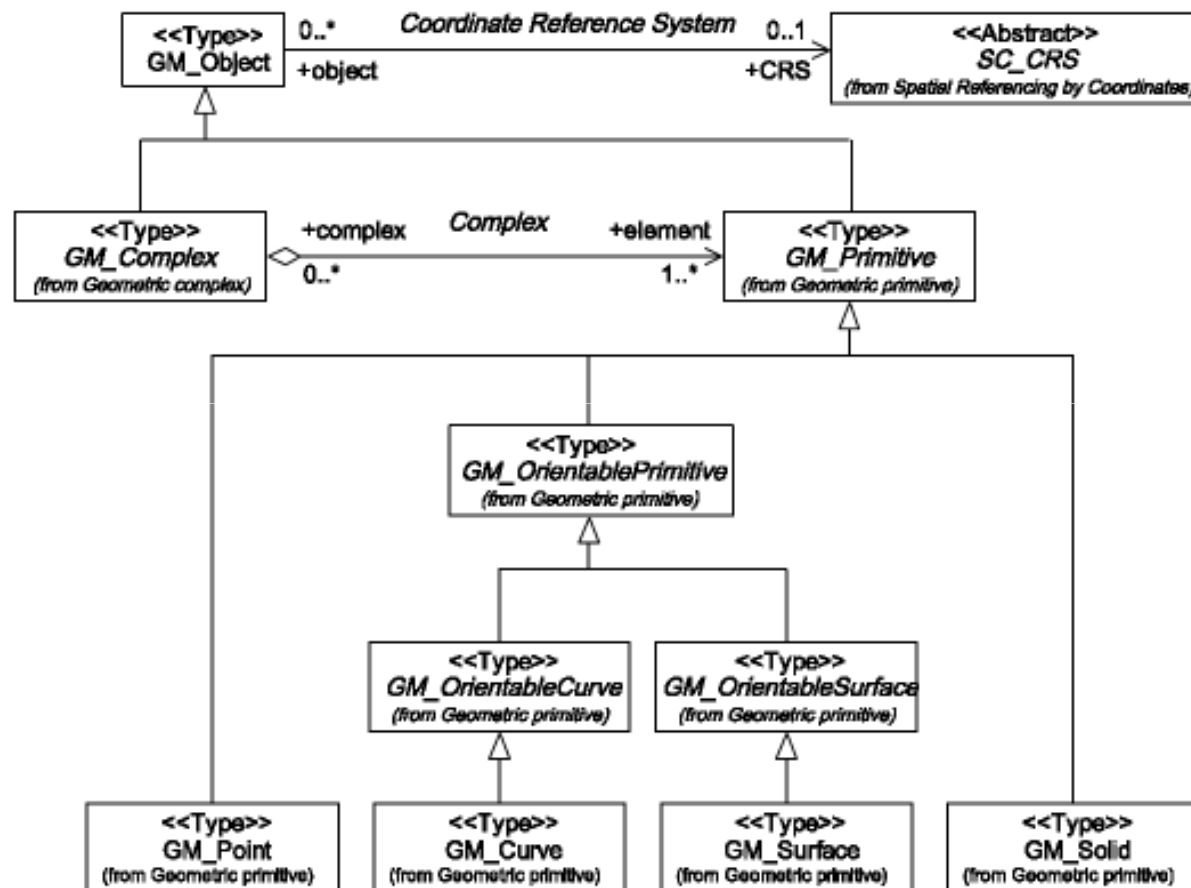
GML



Claudio Rocchini – Istituto Geografico Militare



GML



Aree e Bordi

- Interessante è la definizione di aree e bordi
- Un'area è definita come la composizione di una serie di “anelli”
- Il primo anello indica il bordo esterno
- Gli altri anelli descrivono i buchi interni



GML

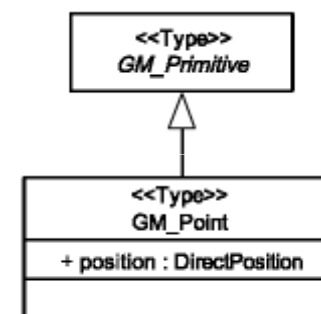
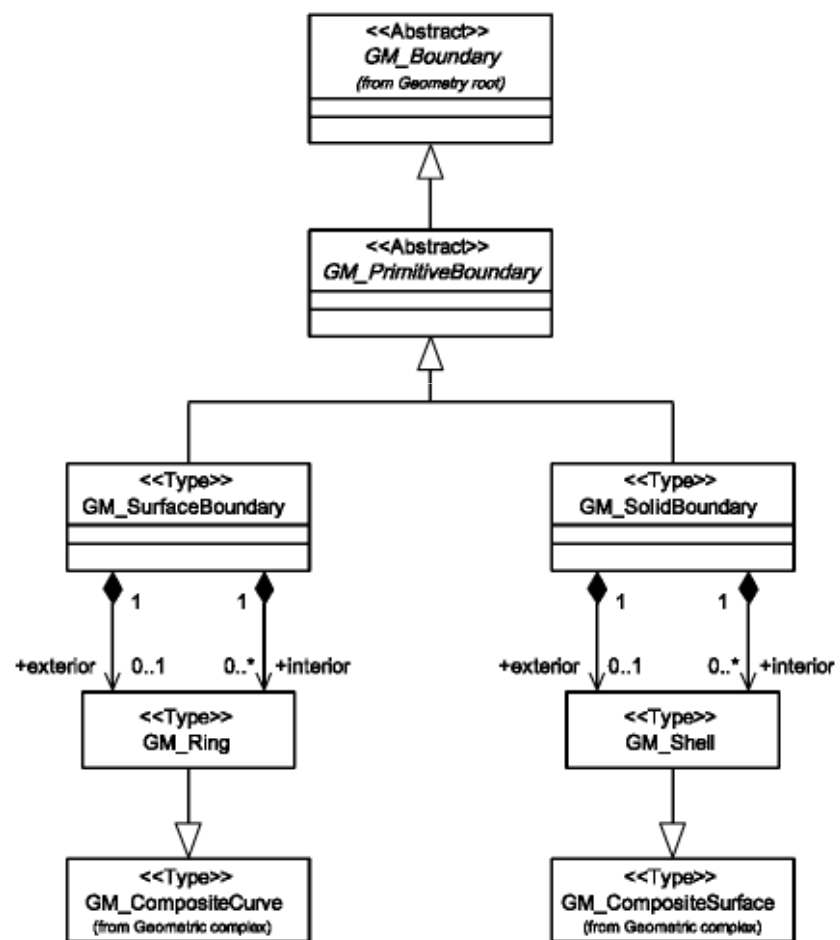
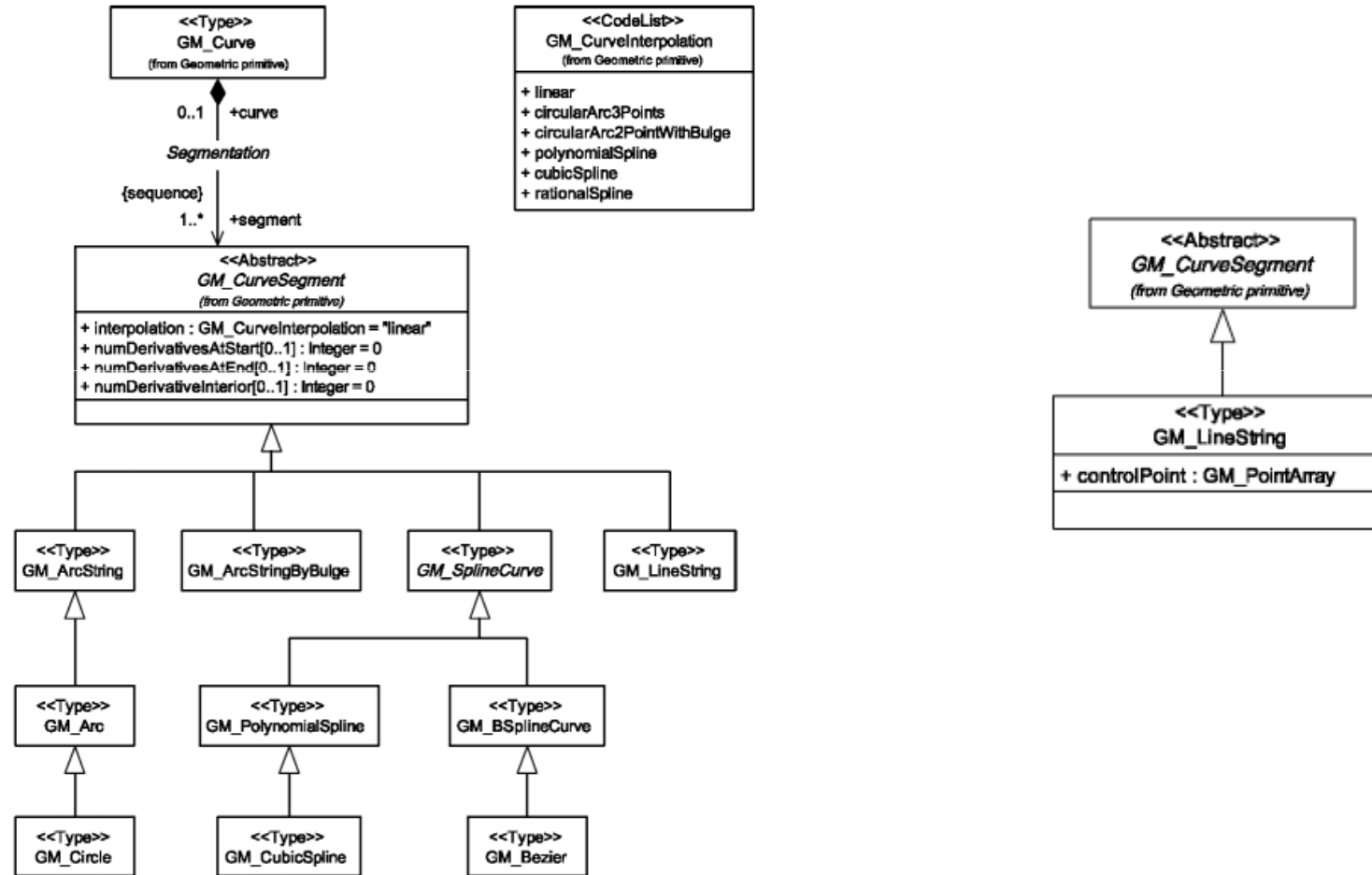


Figure D.5 — Point



GML

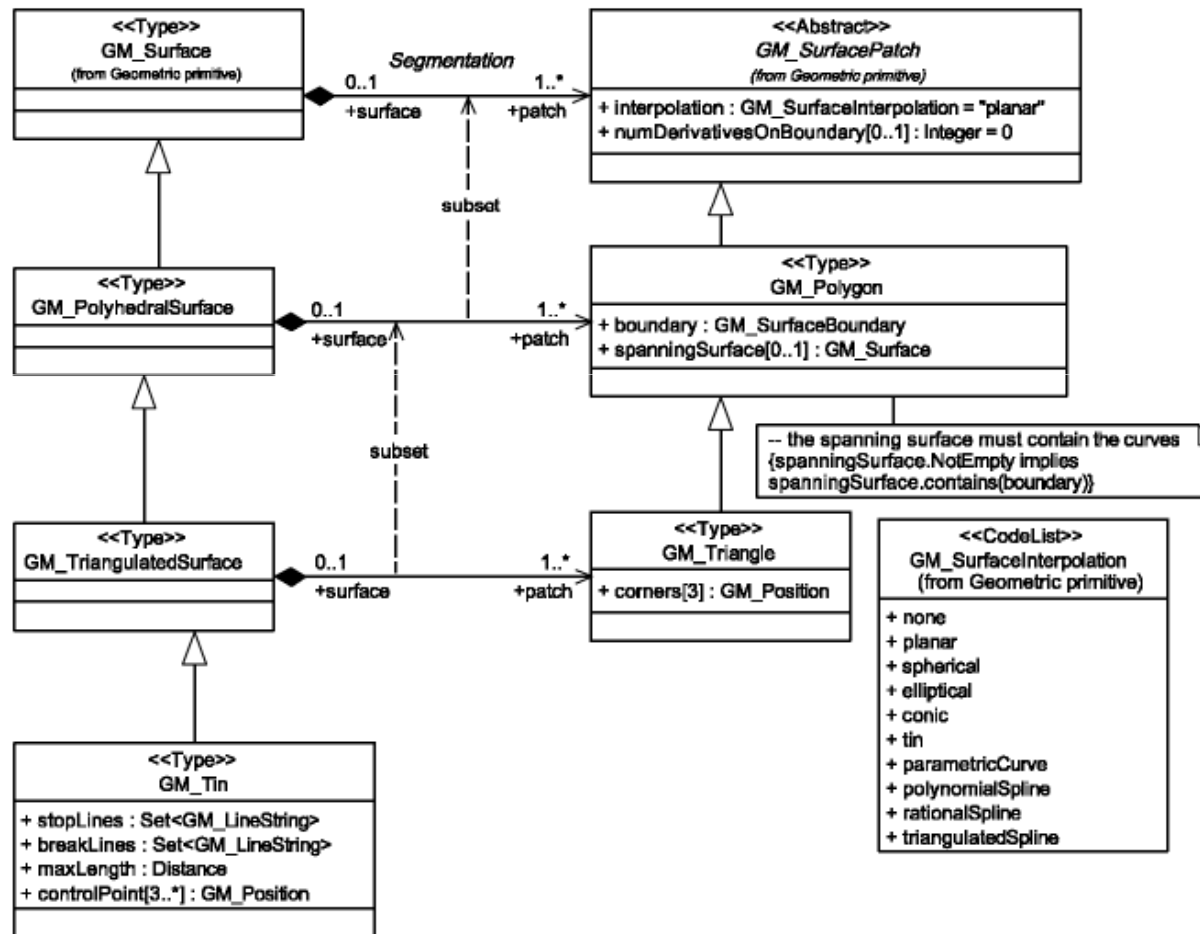


GML Avanzato

- L'ultima versione del GML prevede molti concetti avanzati:
 - Oggetti volumetrici
 - Topologia esplicita (winged-edge)
 - Superfici poliedrali e griglie, come ad esempio i TIN (triangolazioni irregolari del terreno, che descrivono il modello altimetrico)



GML



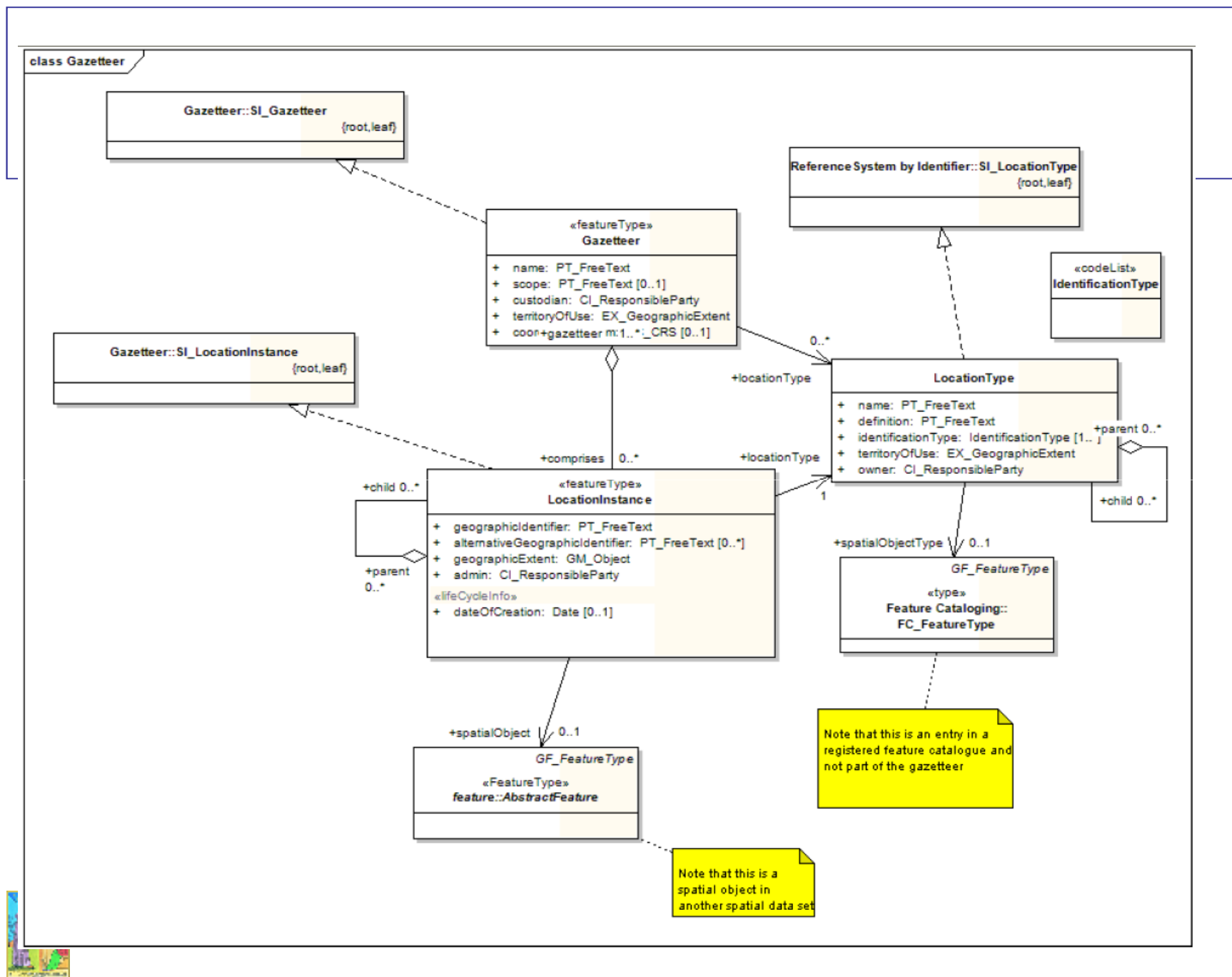
Toponimi INSPIRE

- INSPIRE: progetto europeo di standardizzazione e fruizione dei dati geografici
- Le specifiche dei toponimi prevedono la struttura base delle classi di dati definita in UML:



Inspire Toponimi





Riferimenti

- OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2
(<http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF>)
- OpenGIS® Geography Markup Language (GML) Encoding Standard, Version: 3.2.1
- Schemi originali disegnati con Microsoft **Visio**© e **Liquid XML Studio** (Free)



Bibliografia del Corso

- *Concepts and Techniques of Geographic Information Systems* 2° edition, C.P.Lo A. K. W. Yeung, Pearson Prentice Hall
- *UML Distilled*, 1° ed italiana, M. Fowler, Addison – Wesley
- *C++*, 3° edizione italiana, B. Stroustrup, Addison Wesley

