INTESA STATO REGIONI ENTI - LOCALI
SISTEMI INFORMATIVI TERRITORIALI

COMITATO TECNICO DI COORDINAMENTO

SPECIFICHE PER LA REALIZZAZIONE DEI DATA BASE TOPOGRAFICI DI INTERESSE GENERALE

TITOLO:

IL MODELLO CONCETTUALE GEOUML

INQUADRAMENTO GENERALE ED INTRODUZIONE ALL'USO

Data di emissione:	7 aprile 2004
Versione.sottoversione:	2.1
Tipo di documento:	Versione definitiva per la sperimentazione
Emesso da:	Intesa GIS / WG 01
Riferimenti:	1n1007_1 , 1n1007_2 , 1n1010_1
Nome del file:	1n1010_2.pdf
URL:	http://www.intesagis.it

Proprietà intellettuale e limitazioni d'uso: La proprietà intellettuale è condivisa dagli Enti partecipanti all'IntesaGIS.
Il contenuto può essere liberamente utilizzato e riprodotto, nell'ambito degli scopi previsti dall'IntesaGIS e delle finalità del documento, con obbligo di citazione della fonte.

NOTA: INTESAGIS STA PER INTESA STATO REGIONI ENTI LOCALI SUI SISTEMI INFORMATIVI TERRITORIALI

Intesa GIS / WG 01 **1n 1010_2**

Abstract:

Documento per la specifica formale in UML
Inquadramento generale ed introduzione all'uso del modello concettuale GeoUML

Redazione:	Giuseppe Pelagatti
Note	Lavoro svolto dal Dipartimento di Elettronica e Informazione (DEI) del Politecnico di Milano su contratto di ricerca finanziato dal progetto IntesaGIS
Intesa GIS /WG 01: Gruppo di lavoro Specifiche dei DB Topografici	Gennaro Afeltra, Alberto Belussi, Flavio Bernabino, Lorenzo Bottai, Manuela Corongiu, Stefania Crotta, Lino Di Rienzo, Dario Dominico, Marco D'Orazi, Roberto Gaspani, Gabriele Garnero, Franco Guzzetti, Federica Liguori, Mauro Negri, Mauro Nordio, Stefano Olivucci, Sergio Panella, Giuseppe Pelagatti, David Remotti, Mauro Rossi (coordinamento), Umberto Sassoli, Antonio Trebeschi, Mauro Vasone, Antonio Zampieri
Supporto Scientifico DB Spaziali	Giuseppe Pelagatti (PoliMI)
Esperti incaricati della revisione dei documenti	Sergio Dequal (PoliTo), Mario Fondelli (Iuav), Riccardo Galetto (UniPv), Luciano Surace (IIM)
La struttura dell'IntesaGIS	*Il coordinamento ed indirizzo complessivo sulle attività dell'IntesaGIS è svolto dal Comitato Tecnico di Coordinamento composto dai rappresentanti dello Stato (organi cartografici), delle Regioni e degli Enti Locali*
Comitato Tecnico di Coordinamento	Carlo Cannafoglia - presidente (Agenzia Territorio), Maurizio De Gennaro e Aldo Marolla - segreteria CTC (Reg.Veneto), Gianfranco Amadio (IGM), Vincenza Buccino (Reg. Basilicata), Claudio Cattena (Reg.Lazio), Maria Donatella Borsellino (Reg. Sicilia), Elettra Cappadozzi (CNIPA), Raffaele Caputo (ANCI), Carlo Dardengo (IIM), Mario Di Massa (CONFSERVIZI), Roberto Gavaruzzi (Reg. Emilia Romagna), Roberto Laffi (Reg. Lombardia), Angelo Lisi (APAT). Domenico Longhi (Reg. Abruzzo), Enrico Nardelli (UNICEM), Sebastiano Rao (Reg. Piemonte), Giovanni Tomei (UPI), Giampaolo Turco (CIGA), Marcello Vitiello (Reg. Molise).
Struttura di coordinamento e verifica DB Topografici per il CTC	Mario Desideri (Reg. Toscana) e Gianfranco Amadio (IGM) - responsabili, Giampaolo Artioli (Reg. Emilia-Romagna), Maria Donatella Borsellino (Reg. Sicilia), Elettra Cappadozzi (CNIPA),, Stefania Crotta (Reg. Lombardia), Sergio Farruggia (Comune Genova) , Roberto Gaspani (Comune Bergamo), Antonio Venditti (Min. Ambiente) Marcello Vitiello (Reg. Molise)

Parole chiave: Specifiche di contenuto, inquadramento generale, introduzione all'uso, Modello concettuale, GeoUML, UML

PREMESSA

Con la pubblicazione di questa versione dei documenti di "Specifica per la realizzazione dei Data Base Topografici di interesse generale" termina la fase di definizione preliminare dei contenuti e si avvia la sperimentazione attraverso alcune applicazioni pilota anche su scala estesa, della durata indicativa di un biennio. Una modalità del tutto simile a quanto avviene per gli standard Europei di settore, che prevedono una fase di validazione biennale (ENV).
Nel corso della sperimentazione si provvederà a completare i documenti e le parti ancora mancanti e a sviluppare gli approfondimenti già previsti.
Completata questa ulteriore fase le Specifiche verranno proposte alla Conferenza Stato Regioni Enti locali per la loro approvazione così come stabilito dall'Intesa sui sistemi informativi geografici.

Una tale sperimentazione risulta quanto mai necessaria a fronte della complessità derivante dalla convergenza di molteplici aspetti e dall'innovazione tecnologica sottintesa, ed ha come scopo primario la verifica dei seguenti aspetti:

- **Le modalità di effettiva realizzazione della Base Dati Topografica**. Con quali parametri di qualità a fronte di quali tempi e costi. Una verifica complessiva e di dettaglio sia per una fornitura di primo impianto, sia per la derivazione, con o senza aggiornamento fotogrammterico, da CTR numerica esistente presso gli Enti. La sperimentazione deve permettere di sottoporre a controllo ogni suo aspetto in un contesto di una casistica estesa e non solo più prototipale e deve coinvolgere in questa fase l'esperienza di tutti gli operatori del settore, dagli utenti alle ditte fornitrici di cartografia e GIS;

- **la fruibilità della Base dati Topografica.** Il grado di adeguatezza a fronte dei tanti e tanto dissimili utilizzi con cui deve integrarsi, intendendo con questo sia la fruibilità diretta dei suoi contenuti, ma e soprattutto, la sua adeguatezza ad essere integrata nelle diverse basi dati delle applicazioni di settore.
  Quale sia la sua potenzialità effettiva a costituire una prima base condivisa, che possa esser anche il presupposto per una più vasta opera di integrazione e condivisione tra basi dati. Una fruibilità diretta quindi che si innesti nel flusso informativo di un Ente, garantendosi in tal modo l'aggiornamento dei suoi dati in tempo reale, ed una fruibilità tematica e applicativa, come nucleo condiviso e condivisibile di tutte le informazioni territoriali;

- **L'effettivo grado di interoperabilità**. La sperimentazione di quale grado di interoperabilità si può instaurare tra i diversi Enti od Uffici che aderiscono all'IntesaGIS, a verifica di uno dei presupposti fondanti di tutto il progetto. Con quali modalità, quali regole e con quale efficienza. Quale la reale suddivisione e distribuzione tra gli Enti e nel territorio, nell'ambito del contesto operativo nazionale;

- **la derivabilità del DB25** in tutti i casi reali e soprattutto cercando di minimizzare i requisiti necessari per tale derivazione;

- **la sua integrazione nel Sistema Informativo** di un Ente o di un Ufficio. Quali problematiche e quali soluzioni ottimali nella progettazione e la realizzazione del proprio Database, del proprio ambiente di elaborazione spaziale e di gestione dell'informazione territoriale (GIS). Quali problemi e quali soluzioni per una condivisione in rete efficiente e con quali tecnologie.

Risulta evidente come i punti precedentemente elencati si intreccino e si intersechino in una sperimentazione complessiva rivolta tutti gli aspetti.

Per garantire la massima ricaduta, nella fase di revisione dei documenti, dei risultati conseguiti dalle sperimentazioni, risulterà fondamentale un loro coordinamento con la direzione del progetto IntesaGIS, cui potranno rivolgersi anche per ogni approfondimento delle Specifiche stesse.

Un ulteriore aspetto che dovrà esser affrontato in questa fase riguarda l'aggiornamento professionale connesso alla produzione e utilizzo dei DB topografici . Una tale competenza, sia degli utenti sia dei fornitori di dati, è tutt'altro che secondaria e risulterà decisiva per un reale successo di tutto il progetto.

Come meglio specificato nel documento "**Le Specifiche per la realizzazione dei Database Topografici di interesse generale - lo stato dell'arte ed alcune proposte per una prosecuzione**", le Specifiche sin qui prodotte rappresentano un primo nucleo che richiede di essere ulteriormente integrato da approfondimenti relativi all'informazione catastale, alla codifica delle Entità e degli attributi, ad una presentazione cartografica dinamica, adeguata alle nuove tecnologie di rete, alla derivazione della presentazione a scale di sintesi oltre che del DB25, solo per citare i più importanti.

Non meno importante sarà stabilire quale precisione dei dati sarà necessaria a fronte dell'imminente impiego del GPS associato ad una rete UMTS e quale struttura dati. Quale precisione a fronte delle elaborazioni necessarie alla gestione del dissesto idrogeologico, o quale densità informativa e quale aggiornamento sono richiesti da una efficiente gestione del Servizio Nazionale di Protezione Civile.

Occorre inoltre approfondire quali frontiere stabilire per la terza dimensione a fronte delle nuove tecnologie, quali ad esempio quella del Lidar, e delle funzionalità di elaborazione delle stesse e delle emergenti esigenze.

Una caratteristica del progetto, non meno importante e quanto mai attuale, riguarda la sua naturale convergenza in quello più complessivo che sta nascendo per una Infrastruttura Nazionale di gestione dei Dati Spaziali, NSDI integrata a livello europeo, ESDI: il progetto INSPIRE di cui il progetto IntesaGIS può costituire la modalità di realizzazione del nucleo nazionale di base posizionato tra i più evoluti.

Uno sforzo coordinato in questa direzione permetterà a tutto il contesto nazionale di collocarsi adeguatamente in quello europeo e di far fronte in modo efficiente alle nuove emergenti e pressanti richieste nel campo dell'elaborazione dei dati territoriali, dotandosi di quella che ormai risulta esser una infrastruttura fondamentale per la gestione e lo sviluppo del territorio.

Si è giunti alla fine di questa fase del lavoro e alla soddisfazione di un obiettivo raggiunto si unisce la consapevolezza delle difficoltà che abbiamo ancora davanti, degli ostacoli da superare per migliorare gli elaborati e completare le parti mancanti e soprattutto per farle diventare patrimonio comune e base di un programma nazionale di produzione dell'informazione geografica.

Ci preme infine ringraziare tutti coloro che hanno collaborato per raggiungere questi risultati: in primo luogo il Gruppo di lavoro e i diversi redattori dei documenti; i colleghi del Gruppo di coordinamento DB topografici, gli esperti di riferimento rappresentativi della Comunità scientifica nazionale, tutti i tecnici, professionisti ed utenti degli enti pubblici, dei centri di ricerca, delle imprese ed associazioni che hanno animato gli incontri ed i confronti finora realizzati e che non mancheranno di partecipare al prossimo Convegno di Venezia di presentazione dei risultati.

Carlo Cannafoglia
Mario Desideri
Gianfranco Amadio

Indice

Introduzione

Questo documento descrive il modello GeoUML in maniera non formale, nella misura in cui ciò è possibile nella descrizione di un modello formale di specifica.
La definizione formale di GeoUML è fornita dall'altro documento di questa serie [1n1010_1], la cui lettura è piuttosto faticosa e richiede una conoscenza approfondita del linguaggio UML [UML], del suo Object Constraints Language (OCL) basato sul calcolo dei predicati, e dello standard ISO TC211 Spatial Schema [SPATIAL], indicato nel seguito semplicemente come Spatial Schema.
Una presentazione formale e sintetica del linguaggio sarà anche resa disponibile in lingua inglese in [BNP04].
Questo documento tenta di rendere il GeoUML comprensibile senza richiedere come prerequisiti la conoscenza di tali documenti. Questa scelta ha comportato l'utilizzazione di alcune semplificazioni che potrebbero apparire come scorrettezze o ambiguità all'esperto di UML e di Spatial Schema. Per la soluzione di tali ambiguità si rimanda ai documenti originali referenziati. Si ritiene anche che, per il lettore non esperto di UML che volesse affrontare lo studio dello standard ISO Spatial Schema, questo documento possa costituire una utile introduzione.

Riferimenti

[1n1010_1] "Il modello concettuale GeoUML – Specifica formale in UML", Intesa GIS/WG 01, N 1010_1
[BNP04] A.Belussi, M.Negri, G.Pelagatti, "GeoUML: an ISO TC211 compatible Conceptual Model for Geographical Databases", rapporto interno DEI-Politecnico di Milano, in corso di preparazione
[UML] "Geographic information – Part3: Conceptual Schema Language", ISO/TC211
[OMG_UML] "OMG Unified Modelling Language Specification", Object Management Group
[SPATIAL] "Geographic information – Spatial Schema", ISO/TC211
[RULES] "Geographic information – Rules for Application Schema", ISO/TC211
[SFM99] "Information technology - Database languages – SQL Multimedia and Application Packages - Part 3: Spatial", ISO/IEC 13249-3, dicembre 1999
[GML 3] "OpenGIS Geography Markup Language (GML) Implementation Specification – Version 3.0", OGC 2003

Indice.

1. Caratteristiche generali di GeoUML

Una definizione sintetica di GeoUML è la seguente:
*GeoUML è un **linguaggio di specifica di un database geografico** a **livello concettuale** compatibile con gli aspetti rilevanti dello **standard ISO TC211***.
Gli aspetti rilevanti di tale definizione sono:

- linguaggio di specifica di un database geografico: questa affermazione indica che GeoUML non serve per descrivere il contenuto informativo di un database, ma per descrivere le proprietà generali di tale contenuto informativo; uno schema scritto in GeoUML rappresenta quindi "informazione sull'informazione", tecnicamente detta ***metainformazione***;
- livello concettuale: questa affermazione indica un preciso livello di astrazione al quale questo contenuto viene descritto; l'importanza di tale livello di astrazione viene precisato più avanti, differenziandola sia dai livelli più bassi (livello fisico o interno), sia dai livelli più alti (livelli semantici o ontologici, ecc...)
- compatibile con gli standard ISO TC211: questa affermazione implica che per la parte generale GeoUML è una specializzazione del linguaggio UML, per la parte più specificamente geografica è una specializzazione degli standard rilevanti prodotti dal ISO TC211, in particolare "Spatial Schema" [SPATIAL] e "Rules for Application Schema" [RULES].

Prima di iniziare, nel capitolo 2, la descrizione approfondita del linguaggio GeoUML, sembra opportuno in questo capitolo introduttivo descrivere alcuni aspetti di inquadramento generale e di motivazione delle scelte adottate nel linguaggio, procedendo nel modo seguente:

- nel paragrafo 1.1 si precisa il ruolo del modello concettuale nei database, mettendo a fuoco ciò che tale modello deve definire e ciò che NON deve definire, i motivi per cui è importante rispettare questo livello di astrazione e il legame tra i livelli di astrazione e i diversi usi della metainformazione
- nel paragrafo 1.2 si fornisce una introduzione agli aspetti strettamente geometrici di GeoUML (il Modello Geometrico di GeoUML), allo scopo di mostrare il legame tra le scelte operate nella specializzazione dello Spatial Schema e l'impostazione generale delle specifiche di progetto del database topografico
- nel paragrafo 1.3 si forniscono alcune indicazioni sul processo che ha portato alla definizione della attuale versione di GeoUML
- nel paragrafo 1.4 infine si forniscono alcune indicazioni sulle regole seguite nella scrittura del documento.

1.1 Livelli di astrazione della Metainformazione e applicazioni

La possibilità di memorizzare, elaborare ed interpretare informazioni si basa sulla conoscenza delle regole adottate per rappresentare un contenuto informativo astratto, dotato di un significato per l'essere umano, tramite opportune strutture di segnali fisici di varia natura. L'insieme di queste regole, che non sono l'informazione, ma sono informazione sull'informazione, viene comunemente indicato col termine di metainformazione.
Non vi è dubbio che la strutturazione dell'informazione e della relativa metainformazione debba avvenire per "livelli di astrazione"; nessuno pretenderebbe infatti di descrivere il significato di documenti complessi direttamente in base alla disposizione dei bit su un disco magnetico. Per questo motivo, esistono numerosi standard e convenzioni, anche conflittuali tra loro, che descrivono come rappresentare i livelli più bassi della rappresentazione dell'informazione, come la rappresentazione di numeri o di stringhe di caratteri. Per questo motivo, quando si desidera trasferire informazioni tra due sistemi, bisogna preoccuparsi di definire anche tutte le conversioni necessarie a questi livelli.
Tuttavia, normalmente nessuno reputa rilevante, nella scelta di un contenuto informativo di un database, occuparsi di quale sarà la rappresentazione dei numeri o delle stringhe; viene cioè accettato il principio che certi aspetti di basso livello sono da trattare separatamente e indipendentemente da aspetti di più alto livello, come appunto la scelta del contenuto di un database.
In particolare, è importante osservare che un livello troppo basso di astrazione nella definizione di un contenuto informativo non solo ha il difetto di obbligare chi deve occuparsi di definire il contenuto di tener conto di dettagli irrilevanti (e questo è già sufficientemente grave), ma lega impropriamente aspetti generali dell'informazione, di ampia validità e di lunga durata, ad aspetti implementativi legati a tecnologie di validità meno ampia e di assai minor durata temporale (e questo è ancora più grave).
Come esempio di una conseguenza di queste considerazioni largamente condivise possiamo dire che la definizione di cosa siano una strada o un edificio in certi contesti applicativi non dovrebbe essere legata alla loro rappresentazione in specifici formati di rappresentazione su file.
Talvolta l'errore di legare la specifica di contenuti a una rappresentazione fisica viene commesso per validi motivi, in particolare i due seguenti:

- in assenza di un modello astratto precisamente definito, la definizione di strutture fisiche sembra essere l'unico modo per evitare una descrizione in linguaggio naturale, che quasi sempre risulta prolissa nel tentativo di non essere ambigua, e infine rimane ambigua ugualmente;
- in assenza di un modello astratto precisamente definito manca in genere la garanzia sulla effettiva corrispondenza tra una possibile implementazione e le specifiche astratte di contenuto.

Il modello concettuale standardizzato dall'ISO TC211 e la sua specializzazione GeoUML possono rimediare ad ambedue queste carenze, fornendo un modello preciso e non ambiguo per la definizione dei contenuti di un database del quale sia possibile definire (o, meglio, siano già definite) le corrispondenze (mapping) con possibili implementazioni.

Allo scopo di inquadrare il ruolo svolto dal modello concettuale in alcuni processi fondamentali che coinvolgono il database, in particolare, oltre alla definizione del contenuto, lo scambio di dati e l'interoperabilità, nel seguito si fornisce un quadro del

legame esistente tra i livelli di astrazione e alcune funzioni rilevanti in questo contesto che devono essere svolte sul database.

1.1.1 Livelli fondamentali di definizione dell'informazione

Nell'analizzare i livelli di astrazione conviene considerare tre macrolivelli fondamentali, ognuno dei quali può a sua volta contenere diversi sottolivelli specializzati; la terminologia in questo campo non è univoca, pertanto riportiamo in neretto il termine utilizzato in questo documento:

- livello 1 (***semantico***, ontologico, ecc...): proprietà dell'informazione che interessano solamente l'utente umano (ad esempio, cosa siano una "carreggiata" o un "edificio" nel mondo reale)
- livello 2 (***concettuale***, logico, ecc...): proprietà dell'informazione che interessano sia il sistema che l'utente umano, e l'interazione uomo/macchina (ad esempio, il fatto che una carreggiata sia rappresentata da un poligono, abbia attributi che ne descrivono il numero di corsie e la larghezza, ecc..);
- livello 3 (***fisico***, interno, ecc...): proprietà dell'informazione che interessano solamente la macchina (ad esempio, il modo in cui il poligono che costituisce la carreggiata è rappresentato in macchina, il modo in cui sono rappresentati i numeri, ecc...)

Ovviamente, il progresso informatico tenta di alzare il livello 2 per coprire aspetti del livello 1, quindi l'identificazione concreta di questi livelli varia (lentamente) nel tempo.
E' importante osservare che, mentre nel campo dell'intelligenza artificiale e di alcune sue recenti derivazioni (WEB ontologico, ecc...) si discutono intensamente aspetti di livello 1, semantico, nel campo della costruzione dei sistemi informativi e dei database ci si muove realisticamente in quello che a una certa data può essere considerato il livello 2, concettuale.

1.1.2 Livelli di schema e rilevanza per diversi usi o funzioni

Possiamo rapportare i diversi livelli di metainformazione ai diversi usi che vengono fatti di tale metainformazione da parte sia del sistema che degli utenti umani.
In tabella 1.1 si indicano, per alcuni usi significativi del database, i livelli di metainformazione che risultano rilevanti.
La tabella mostra che il livello concettuale è l'unico livello rilevante per quanto riguarda l'interazione uomo/macchina e la definizione dei contenuti di un database. Per ottenere invece una interpretazione umana del dato, in particolare a fronte di trasferimenti di informazione tra utenti con culture diverse, in genere il modello concettuale viene integrato con altra metainformazione di tipo più "semantico".
Si noti che questa informazione semantica deve "agganciarsi" al modello concettuale per poter correlare l'aspetto interpretativo alla effettiva informazione presente nel database (ad esempio, un complesso modello interpretativo della nozione di edificio deve rapportarsi alla definizione di "Edificio" data nel modello concettuale, perchè a questa sono poi attribuite le informazioni presenti nel sistema).
Nel contesto geografico l'interpretazione umana è rilevante per la rilevazione del dato, cioè per il corretto riconoscimento degli oggetti nella realtà osservata. Quindi, in base a quanto detto sopra, lo schema concettuale corredato di opportune metainformazioni semantiche aggiuntive, costituisce la base per la definizione di capitolati di acquisizione e/o regole di aggiornamento del database.

Infine, la tabella mostra che per molte operazioni che coinvolgono il sistema, in particolare l'interoperabilità o il trasferimento dati tra sistemi, e ovviamente per l'implementazione del database, è necessario avere sia lo schema concettuale che quello fisico. In particolare, lo schema fisico descrive le strutture fisiche di memorizzazione collegandole alle definizioni dello schema concettuale, perchè il sistema non manipola le strutture fisiche in maniera autoreferenziale ma interpretando le richieste che provengono dall'utente umano attraverso l'interfaccia uomo/macchina.

USI della METAINFORMAZIONE	**livello 1 semantico**	**livello 2 concettuale**	**livello 3 fisico**
interpretazione umana del dato	Rilevante	Rilevante	
rilevamento del dato	Rilevante	Rilevante	
trasferimento di informazioni tra utenti	Rilevante	Rilevante	
interazione uomo/macchina		Rilevante	
definizione dei contenuti del DB		Rilevante	
interoperabilità tra sistemi		Rilevante	Rilevante
trasferimento dati tra sistemi		Rilevante	Rilevante
implementazione del database		Rilevante	Rilevante

Tabella 1.1

Lo standard Spatial Schema e collegati permette di definire uno schema concettuale e, a partire da questo, anche aspetti di schema fisico, perchè è orientato all'interoperabilità completa tra sistemi.
I prodotti conformi allo Spatial Schema implementeranno tale modello concettuale e potranno interoperare; per quanto riguarda lo scambio di dati è stato definito un formato XML dello Spatial Schema, detto GML [GML 3], che costituisce una possibile rappresentazione fisica di quel modello concettuale.
Dato che lo Spatial Schema è molto generale, esso prevede la possibilità di "ritagliarlo" e "specializzarlo" per affrontare specifici contesti applicativi.
GeoUML costituisce una specializzazione creata per facilitare in primo luogo la definizione dei contenuti del database topografico, mentre, per quanto riguarda l'implementazione, si rimanda allo Spatial Schema o a altre specifiche.

1.2 Caratteristiche generali del modello geometrico di GeoUML

Gli aspetti propriamente geometrici di GeoUML costituiscono la parte qualitativamente e quantitativamente principale del modello, anche se devono essere inquadrati entro la struttura più generale del modello stesso, costituita dalle nozioni relative al modello a oggetti (classi, attributi, associazioni e ereditarietà), trattata al capitolo 2.
Gli aspetti geometrici sono trattati dettagliatamente nei capitoli 3, 4 e 5, pertanto l'introduzione fornita qui vuole essere solamente una guida all'interpretazione del contenuto di tali capitoli e forse può essere compresa a fondo solamente dopo la loro lettura.

1.2.1 Oggetti geometrici e Relazioni Spaziali tra oggetti geometrici

La rappresentazione della geometria in molti sistemi geografici tende a riflettere uno dei due seguenti criteri:

- orientamento ai singoli oggetti: seguendo questo criterio la geometria dei singoli oggetti viene rappresentata oggetto per oggetto; in questo modo si possono individuare bene i singoli oggetti ma si tende a sacrificare l'interazione tra i diversi oggetti (esempi di questo approccio: gli Shapefile, ecc...)
- orientamento alle relazioni spaziali (topologiche): seguendo questo criterio l'enfasi è sulla strutturazione complessiva della rappresentazione geometrica, che deve rappresentare le relazioni spaziali tramite la condivisione di componenti geometriche, ad esempio l'adiacenza tra poligoni tramite la condivisione di porzioni (archi) dei poligoni stessi (esempi di questo approccio: VPF, coverage di ArcInfo, ecc...).

Ognuno di questi criteri possiede delle caratteristiche positive, ma purtroppo è difficile farli coesistere, perchè tendono a produrre strutturazioni differenti, in quanto l'orientamento alle relazioni spaziali richiede di spezzare gli oggetti in modo da evidenziare le componenti condivise con altri oggetti, e questo approccio, in assenza di strutturazioni più complesse della geometria, tende a impedire il riconoscimento degli oggetti stessi.
In GeoUML, in totale conformità con lo Spatial Schema, si cerca di ottenere la sintesi di questi aspetti basandosi su una struttura più ricca di tipi geometrici. L'idea fondamentale consiste nel distinguere il livello delle ***primitive geometriche***, che in prima approssimazione coincide con quello delle geometrie di sistemi tradizionali (esempio: point, polyline, polygon, ecc...), dal livello degli ***oggetti geometrici***, che sono costituiti da ***insiemi di primitive***.
Basandosi su questa distinzione è possibile definire le relazioni spaziali tra oggetti geometrici in termini di condivisione di primitive tra di loro, in un modo che generalizza quanto fatto dai sistemi tradizionali orientati alla rappresentazione della topologia.
I diversi modi in cui un oggetto geometrico struttura le primitive che lo costituiscono produce un ricco assortimento di tipi geometrici, descritti in dettaglio al capitolo 3. Astraendo dai dettagli, possiamo dire che la costruzione degli oggetti geometrici come insiemi di primitive si basa sui seguenti principi:

- creazione di ***complessi***: sono oggetti le cui primitive soddisfano requisiti molto generali relativi alla struttura topologica;
- creazione di ***composti***: sono oggetti le cui primitive, oltre a soddisfare i requisiti generali dei complessi, sono organizzate in modo tale che l'intero oggetto ha il comportamento di una singola primitiva (tali oggetti sono

isomorfi a una primitiva); ad esempio, una linea composta (vedi cap. 3) è costituita da molte linee primitive, ma si comporta anche come una singola primitiva;
- creazione di ***aggregati***: sono insiemi generici di primitive, senza vincoli di strutturazione; a questi tipi di oggetti non viene dedicato molto spazio in questo documento, perchè non danno luogo a molte proprietà definibili nello schema.

Il capitolo 3 di questo documento, dedicato alla definizione dei tipi geometrici, definisce quindi le caratteristiche dei singoli tipi di oggetti geometrici in quanto insiemi di primitive.
Il capitolo 4, partendo dalla definizione dei singoli tipi geometrici, descrive come si possono definire strutture che rappresentano l'interazione di diversi oggetti. Mentre il capitolo 3 essenzialmente traduce applicando alcune restrizioni i tipi già previsti dallo Spatial Schema, il capitolo 4 definisce strutturazioni già molto più orientate ad aspetti applicativi, sia generali, sia specifici del contesto del GeoUML (vedi ad esempio il paragrafo seguente relativo alla terza dimensione).
In particolare, tali strutturazioni consistono nei seguenti aspetti:

- la condivisione di primitive tra diversi oggetti geometrici, ottenuta creando gerarchie di oggetti più semplici che compongono oggetti più complessi (ad esempio, ponendo oggetti diversi a comporre uno stesso "strato", ma anche ponendo oggetti come le "UnitàVolumetriche" a comporre oggetti come gli "Edifici");
- la trattazione di attributi che variano in funzione della geometria di un oggetto; ad esempio l'attributo "pavimentazione" associato ad un'Area Stradale, che varia in base ai diversi punti dell'area stessa. Questo aspetto riguarda la strutturazione, perchè le "sottoaree" che definiscono i vari tipi di pavimentazione sono geometrie più o meno complesse che devono essere correttamente strutturate.
- una specie di tipo derivato che riguarda la terza dimensione (vedi prossimo paragrafo)

Infine, il capitolo 5 descrive la possibilità di definire ***vincoli topologici*** generali, indipendentemente da una particolare strutturazione. Ad esempio, possiamo definire un vincolo del tipo "per ogni AreaDiPertinenzaStradale deve esistere un'AreaStradale alla quale essa è adiacente". Questi vincoli hanno una struttura logica (in questo caso si richiede l'esistenza di un certo tipo di oggetto) e fanno riferimento ad una relazione topologica (in questo caso l'adiacenza tra poligoni).
L'affermazione che questo tipo di vincoli non definiscono una particolare strutturazione delle primitive non significa che una strutturazione opportuna non possa essere utile nella verifica del vincolo stesso; con riferimento all'esempio citato, potrebbe essere conveniente porre le AreaDiPertinenzaStradale in uno stesso oggetto complesso (ad esempio, in uno "strato") insieme agli oggetti di tipo AreaStradale, in modo che l'adiacenza sia verificabile in base alla condivisione di primitive. Tuttavia, questa strutturazione deve essere definita secondo le regole del capitolo 4, non è implicita nella definizione dei vincoli topologici.

1.2.2 Gestione della terza dimensione

Un aspetto che ha profondamente influenzato il modello geometrico di GeoUML è la scelta di utilizzare al meglio la disponibilità di informazioni tridimensionali, di tipo

puntiforme o lineare, senza però rappresentare un modello tridimensionale completo delle superfici e dei volumi.
Questa scelta si basa quindi sui seguenti principi:

- gli oggetti geometrici puntiformi e lineari sono rappresentabili in 3 dimensioni (spazio 3D);
- per gli oggetti geometrici che sarebbero superfici nello spazio può essere rappresentata la frontiera poligonale (anelli) in 3 dimensioni, ma le proprietà di tipo areale possono solamente (ove ciò è possibile e sensato) essere riferite alla superficie ottenuta proiettando il poligono sul piano bidimensionale (spazio 2D).

Ovviamente, il primo principio implica che le proprietà topologiche relative a punti e linee siano applicate pienamente nello spazio tridimensionale; ad esempio, un punto è contenuto in una linea se vi appartiene nello spazio 3D, due linee sono connesse se lo sono nello spazio 3D, ecc...
Più complesse sono le conseguenze del secondo principio, perchè alcune proprietà topologiche possono fare riferimento agli anelli tridimensionali, ma altre richiedono di fare riferimento a superfici. Come esempio di una proprietà rappresentabile con riferimento agli anelli 3D possiamo considerare un punto posizionato su un anello 3D (ad esempio, un cartello stradale sul bordo di un'area stradale).
Come esempio di una proprietà che non è possibile rappresentare con riferimento diretto agli anelli 3D si consideri il contenimento di un anello che rappresenta un cortile dentro un anello che rappresenta un edificio – in termini topologici tale contenimento richiederebbe di avere le superfici tridimensionali. Tuttavia, grazie al fatto che nei sistemi geografici l'asse verticale ha un significato particolare, possiamo in molti casi surrogare la carenza di superfici 3D proiettando gli anelli sul piano 2D e considerando le relative superfici 2D; nel nostro esempio ci contenteremo di rappresentare la proprietà che la superficie 2D che rappresenta il cortile sia contenuta dentro la superficie 2D che rappresenta l'edificio.
Nel linguaggio GeoUML l'applicazione dei principi enunciati comporta le seguenti conseguenze:

- tutti i tipi geometrici hanno un'indicazione di dimensionalità; tale dimensionalità può essere 2D oppure 3D per le linee e i punti, ma è solamente 2D per le superfici;
- esiste una funzione ***planar*** che, dato un oggetto 3D produce la sua rappresentazione 2D, ottenuta come proiezione di tutti i suoi punti;
- viene definito un tipo strutturato ***SurfaceB3D*** (vedi cap. 4.6), costituito da un anello in 3D e da una superficie in 2D legati dalla condizione che la proiezione dell'anello sul piano 2D produce la frontiera della superficie.

1.2.3 Rappresentazione Interna della Geometria

Con rappresentazione interna della geometria si intende il modo in cui le primitive geometriche sono rappresentate in termini di strutture dati.
GeoUML non prescrive tale rappresentazione interna, perchè si tratta di un aspetto considerato pertinente al livello fisico e non a quello logico/concettuale.
Il fatto che GeoUML definisca la strutturazione degli oggetti non deve essere confuso con la rappresentazione interna; la strutturazione degli oggetti prescrive come devono essere correttamente costituiti gli insiemi di primitive che rappresentano gli oggetti e le loro relazioni spaziali, ma non prescrive ne la rappresentazione interna delle singole primitive, ne le strutture dati utilizzate per rappresentare gli insiemi di primitive che costituiscono

gli oggetti. In particolare, quest'ultimo aspetto è complesso proprio perchè la struttura degli oggetti è complessa e perchè le primitive sono condivise.
Ovviamente, per implementare un database o per produrre un file di trasferimento è necessario definire tale rappresentazione; ciò può essere fatto a parte rispetto allo schema concettuale in GeoUML. La scelta di una rappresentazione può essere legata a degli standard oppure a formati proprietari, ed è opportuno che sia separata dalla definizione dei contenuti a livello concettuale.
Anche se la rappresentazione interna di uno schema GeoUML può essere qualsiasi, scegliendo certi formati può essere molto difficile riuscire a rappresentare tutto il contenuto di uno schema; alcune osservazioni a questo proposito si trovano nel capitolo 4.
In ogni caso, dato che GeoUML è consistente con lo Spatial Schema, qualsiasi formato di trasferimento o di database conforme allo Spatial Schema è adatto a rappresentare un insieme di dati definito in GeoUML; in particolare questo vale per il formato GML [GML 3].

1.3 Metodo e criteri di definizione del GeoUML

Il linguaggio GeoUML è stato definito con un processo rivolto ad ottenere due obiettivi principali, talvolta in contrasto tra loro:

- la compatibilità con lo standard ISO TC211 e la definizione formale di tutti i suoi elementi;
- la funzionalità adeguata a definire tutte le proprietà dei contenuti del database topografico (definiti dal gruppo di lavoro sui contenuti) in maniera il più semplice e diretta possibile.

Per ottenere questi obiettivi il processo di definizione del linguaggio si è svolto in parallelo col processo di definizione dei contenuti: ogni volta che la definizione dei contenuti incontrava delle difficoltà nell'esprimere in maniera semplice alcune proprietà dell'informazione si è proceduto a rivedere il linguaggio, definendo in maniera formale e compatibile con ISO TC211 i nuovi costrutti. Per evitare che questo processo conducesse ad una aggregazione incoerente di caratteristiche disomogenee, è stato compiuto uno sforzo per rivedere ogni volta le scelte operate in un'ottica di buona progettazione del linguaggio, cercando di ottenere ortogonalità, completezza e una struttura chiara di livelli di astrazione (alcuni costrutti costituiscono una base minima, altri sono derivabili da quelli di base).
Uno dei criteri fondamentali relativi alla semplicità del linguaggio è stato quello di fare in modo che non fosse mai necessario nella definizione dei contenuti ricorrere a espressioni in Object Constraint Language (OCL); OCL è una parte del linguaggio UML utilizzata per la descrizione di proprietà degli oggetti e basata sul calcolo dei predicati. Il motivo di questa scelta è costituito dal fatto che OCL è un linguaggio molto difficile da usare per gli utenti non specialisti.
In pratica, dato che la definizione formale dei costrutti di GeoUML [1n1010_1] è basata pesantemente su definizioni in OCL, possiamo dire che GeoUML "incapsula" la complessità di OCL all'interno di costrutti linguistici "prefabbricati" che possono essere usati con relativa facilità dagli utenti.
Infine, fin dall'inizio del processo di definizione di GeoUML è emerso un ulteriore e non marginale vantaggio della definizione di costrutti (ovvero strutture) "prefabbricati", cioè la possibilità di interpretare queste strutture ai fini dell'implementazione, mentre le formule OCL, essendo estremamente flessibili non forniscono alcune indicazione relativa alla strutturazione.
In pratica, questo significa che normalmente un progettista definisce le formule OCL in base a certe strutture che ha in mente, ma le formule non rendono tale struttura comune facilmente riconoscibile; i "prefabbricati" GeoUML invece rappresentano le strutture che sono state individuate nel processo di definizione dei contenuti esplicitamente, fornendo indicazioni utili per l'implementazione.

1.4 Regole seguite da questo documento

Si richiamano qui alcuni aspetti relativi al modo in cui questo documento definisce il linguaggio.

1.4.1 Sintassi

GeoUML possiede sia una sintassi grafica che una sintassi alfanumerica (detta anche "forma testuale" del linguaggio).
La sintassi grafica è quella del linguaggio UML standard estesa con i costrutti nuovi. La rappresentazione adottata per tali costrutti è stata progettata per essere facilmente implementata in Rational Rose, che è il prodotto classico di design UML ed è stato utilizzato dagli estensori dello standard ISO TC211 Spatial Schema; tuttavia GeoUML può essere implementato su qualsiasi prodotto di design orientato a UML.
I diagrammi riprodotti in questo documento sono stati realizzati con Rational Rose arricchito con un "plug-in" realizzato appositamente.
La sintassi alfanumerica è stata definita in maniera originale, per permettere di scrivere schemi in forma non grafica anche se il linguaggio UML non possiede una sintassi alfanumerica. Per non appesantire la trattazione, la forma alfanumerica è spiegata in questo documento solo tramite esempi, mentre la definizione formale della sintassi può essere trovata nel documento 1n1010_1. Non esistendo ancora un compilatore per la forma testuale i dettagli della sintassi non sono particolarmente rilevanti e potrebbero essere modificati se risultasse necessario o conveniente; in questo documento si è adottata la convenzione di scrivere le parole chiave del linguaggio in corsivo sottolineato – ad esempio *classe*, *attributi*, ecc... – e una indentazione delle clausole che aumenta la leggibilità.
Esula dagli scopi di questo documento discutere i vantaggi e gli svantaggi della forma testuale rispetto a quella grafica; in generale, la forma grafica risulta vantaggiosa per mettere in evidenza i legami esistenti tra le varie classi, mentre quella testuale può risultare vantaggiosa nella specifica di una singola classe. Pertanto gli esempi contenuti in questo documento, che sono piccoli e riferiti al numero minimo indispensabile di classi di oggetti, potrebbero suggerire l'inutilità della forma grafica; se però si considerano schemi reali, in cui sono presenti molte associazioni e vincoli tra classi diverse, allora si scopre che la forma grafica è indispensabile per capire i complessi legami esistenti tra classi, che costituiscono l'aspetto fondamentale di uno schema.
Dato che lo stesso schema, cioè lo stesso contenuto, può essere sempre espresso in ambedue le forme, in questo documento vengono presentate ambedue le forme di ogni possibile costrutto del linguaggio.

1.4.2 Esempi

Gli esempi in questo documento hanno sempre esclusivamente lo scopo di illustrare i costrutti del linguaggio, e non hanno mai lo scopo di suggerire particolari contenuti. Per rendere un esempio utile si deve fare riferimento ad una concezione intuitiva del tipo di oggetto utilizzato nell'esempio stesso, e tale visione è in genere assai semplificata rispetto alla definizione presente in uno schema reale.
Ad esempio, utilizzando le classi UnitàEdilizia, ElementoStradale, ecc... con alcuni attributi, in questo documento non si vuole assolutamente entrare nel merito della opportunità di definire un tipo particolare di classi e di attributi, ma semplicemente illustrare il concetto di classe e di attributo e la sua definizione in GeoUML. L'eventuale coincidenza o conflitto di un esempio di questo documento con definizioni presenti nei

documenti dedicati alle definizioni di contenuto (come la serie 1n1007) è da considerare totalmente casuale. Come indicato sopra, per comprendere veramente il linguaggio è necessario considerare anche esempi più estesi di quelli elementari contenuti in questo documento, che servono a capire il singolo costrutto isolatamente; un esempio di grande dimensione è sicuramente costituito dal documento 1n1007_4, che contiene la definizione dei contenuti del database topografico in GeoUML, sia in versione testuale che in versione grafica.

2 Aspetti non geometrici di GeoUML

Il linguaggio GeoUML è un'estensione del linguaggio UML orientata ai dati geografici. In questo capitolo si fornisce una presentazione degli aspetti non geometrici di GeoUML, che sono essenzialmente quelli presenti nei "Class Diagrams" del linguaggio UML, cioè:

- classe
- attributo (con i diversi domini e la cardinalità)
- associazione
- ereditarietà
- vincolo

A questi in GeoUML sono aggiunte le due nozioni di

- chiave primaria
- dominio gerarchico

Alla base della definizione di uno schema in UML c'è la nozione di ***oggetto (o entità)***, che fa riferimento a un elemento della realtà al quale è utile e sensato attribuire informazioni in forma di attributi e/o di associazioni con altri oggetti. Lo schema definisce le ***proprietà*** (properties) degli oggetti che potranno popolare la base di dati. Ovviamente, nel momento in cui si definisce lo schema gli oggetti non esistono; ciò che viene definito nello schema sono le ***classi*** di oggetti che potranno entrare a far parte della base dati.
Una classe definisce le proprietà comuni a un insieme di oggetti omogenei. Tali proprietà possono essere ***attributi***, ***operazioni*** o ***ruoli in associazioni*** con altre classi. In GeoUML le operazioni risultano poco importanti e saranno trascurate. Un attributo associa agli oggetti di una classe un valore. Una associazione permette di relazionare tra loro oggetti di diverse classi.
L'intero schema è costituito da una serie di definizioni di classi, e per ogni classe dalla definizione degli attributi e delle associazioni. A queste nozioni si aggiunge l'***ereditarietà*** fra classi. L'ereditarietà è un meccanismo di classificazione gerarchica per definire sottoclassi che ereditano le proprietà della superclasse, esattamente come i cani "ereditano" tutte le proprietà dei mammiferi e questi "ereditano" le proprietà degli animali, ecc...
Nei paragrafi seguenti vengono illustrate le regole di definizione di questi aspetti in GeoUML.

2.1 Costrutti UML standard

Questa sezione può essere saltata da chi conosce già il linguaggio UML; l'unico elemento non UML è costituito dalla definizione della forma testuale.

2.1.1 Classe

Una classe definisce le proprietà comuni a un insieme di oggetti omogenei. In ogni istante di vita della base di dati geografica una classe ha un'estensione costituita dall'insieme di oggetti contenuti nella base di dati che appartengono alla classe.
Una classe possiede un nome, eventualmente un nome abbreviato ed è rappresentata in uno schema grafico secondo la forma generale mostrata nel diagramma 2.1

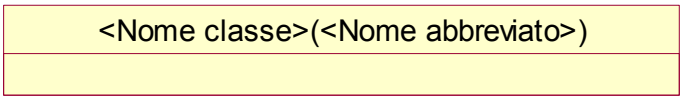

Diagramma 2.1

La definizione testuale si basa sulla parola chiave *classe* seguita dal nome della classe e eventualmente dal nome abbreviato, come nel seguente esempio:

classe UnitàEdilizia (UNEDIL)

In forma grafica lo stesso esempio è mostrato nel diagramma 2.2

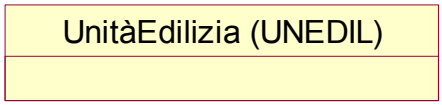

Diagramma 2.2

2.1.2 Attributo

La nozione di ***attributo*** fa riferimento ad una proprietà di una classe esprimibile attraverso un valore scelto in un dominio.
Un attributo A di una classe C, con dominio D, è una funzione che associa ad ogni oggetto della classe C un valore appartenente al dominio D.
I domini di base per gli attributi non geometrici sono: *String*, *Integer* e *Real*.
Ogni attributo ha un nome che deve essere univoco nell'ambito della classe. La rappresentazione grafica generica di una classe dotata di un attributo è mostrata nel diagramma 2.3.

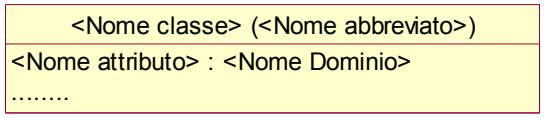

Diagramma 2.3

La definizione testuale si basa sulla parola chiave *attributi* seguita dal nome e dal tipo degli attributi, come nel seguente esempio:

classe UnitàEdilizia (UNEDIL)
attributi: Codice: *Integer*;
Nome: *String*

In forma grafica lo stesso esempio è mostrato nel diagramma 2.4

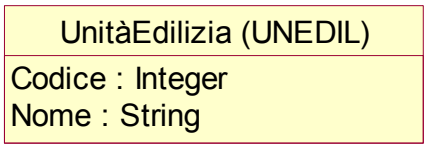

Diagramma 2.4

Talvolta è necessario scrivere delle espressioni che fanno riferimento a un attributo di una classe. A questo scopo si utilizza una notazione (***dot notation***) basata sulla scrittura del nome della classe seguito da un punto e dal nome dell'attributo; ad esempio la notazione "UnitàEdilizia.Codice" fa riferimento all'attributo Codice della classe UnitàEdilizia. Come vedremo trattando le associazioni, la dot notation è molto utile e potente.

2.1.3 Attributo enumerato

Un ***attributo enumerato*** è un attributo con dominio finito i cui valori sono predefiniti ed elencati nello schema.

La forma testuale della definizione di un attributo enumerato è duplice: i valori possono essere elencati direttamente nella definizione dell'attributo oppure essere elencati in una definizione separata del dominio (questa è la forma che corrisponde più strettamente alla forma grafica).
Un esempio di definizione separata è il seguente:

classe UnitàEdilizia
attributi: Codice: *Integer*;
Destinazione: CATEGORIA_DEST

dominio CATEGORIA_DEST: (ospedale, luogo-di-culto,poste, ...)

mentre la definizione diretta nell'attributo sarebbe la seguente:

classe UnitàEdilizia
attributi: Codice: *Integer*;
Destinazione: (ospedale, luogo-di-culto,poste, ...)

Questa forma più sintetica è indicata quando i valori sono pochi e il dominio non deve essere condiviso con altri attributi, altrimenti è preferibile la forma con definizione separata del dominio.
La rappresentazione grafica di questo esempio è mostrata nel diagramma 2.5

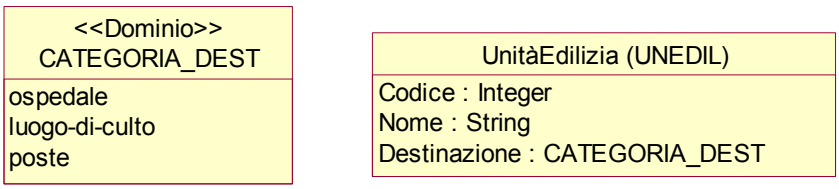

Diagramma 2.5

2.1.4 Cardinalità degli attributi (attributo multivalore)

Un attributo associa normalmente un unico valore ad ogni oggetto di una classe. Tale caratteristica dell'attributo può essere modificata specificando esplicitamente la cardinalità dell'attributo.

La cardinalità di un attributo definisce il numero minimo e il numero massimo di valori che l'attributo può assumere.

Si indica sintatticamente nel seguente modo: [min..max], dove min e max indicano la cardinalità minima e la cardinalità massima dell'attributo. Il valore di default (vale a dire, la cardinalità attribuita in caso di assenza della esplicita indicazione [min..max]) è [1..1]; talvolta tale valore è indicato anche semplicemente come [1].

I casi che si possono verificare sono i seguenti:

cardinalità	significato
[1..1]	Attributo obbligatorio ad un sol valore
[0..1]	Attributo opzionale ad un sol valore
[1..*]	Attributo obbligatorio e multivalore
[0..*]	Attributo opzionale e multivalore
[1..n]	Attributo obbligatorio con al massimo n valori (n>1)
[0..n]	Attributo opzionale con al massimo n valori (n>1)
[n..m]	Attributo obbligatorio con al minimo n valori e al massimo m valori (n>1, e m≥n)

La rappresentazione grafica della cardinalità di un attributo è mostrata nel diagramma 2.6:

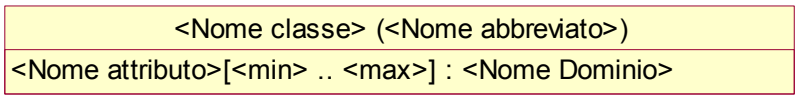

Diagramma 2.6

Nella notazione testuale si usa la stessa forma, come nel seguente esempio:

classe UnitàEdilizia
attributi: Codice: *Integer*;
Destinazione [1..*]: CATEGORIA_DEST

Il diagramma 2.7 mostra l'esempio in forma grafica

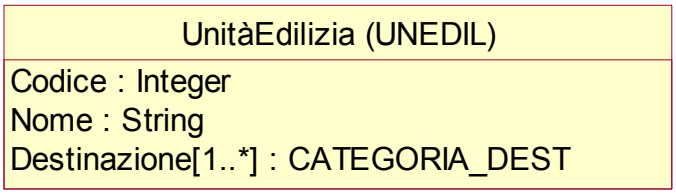

Diagramma 2.7

2.1.5 Associazione (binaria)

Un'associazione rappresenta un legame logico tra due classi. Le istanze di una associazione rappresentano coppie di oggetti appartenenti alle classi coinvolte nell'associazione.

Per ogni classe partecipante ad una associazione si da un nome al ***ruolo*** che essa svolge nell'associazione e si specifica un vincolo di cardinalità simile a quello degli attributi. I ruoli infatti possono essere visti come degli attributi particolari che restituiscono, invece di un valore di un dominio normale, un oggetto (o più oggetti) della classe associata. Pertanto ogni ruolo ha un suo ruolo inverso definito nell'altra classe.

La forma grafica generale per la rappresentazione delle associazioni è mostrata nel diagramma 2.8.

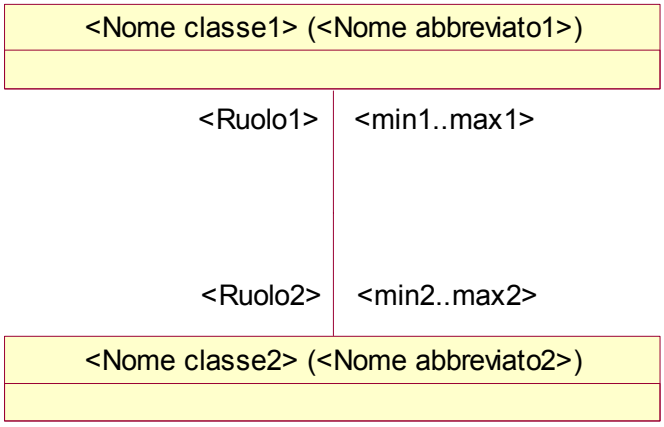

Diagramma 2.8

La notazione testuale per la definizione di associazioni consiste nel dichiarare, dopo la parola chiave *ruoli*, il ruolo della classe collegata e eventualmente, dopo la parola chiave *inverso*, il ruolo opposto, come mostrato dal seguente esempio:

classe Civico
attributi: numerazione: *String*;
ruoli: StradaAppartenenza [1..1] : ToponimoStradale
inverso CiviciDellaStrada [0..*]
classe ToponimoStradale
attributi: codice: *String*;
nome: *String*
ruoli: CiviciDellaStrada [0..*] : Civico
inverso StradaAppartenenza [1..1]

La riga

ruoli: StradaAppartenenza [1..1] : ToponimoStradale

definisce l'esistenza di un'associazione di questa classe (la classe Civico) con la classe ToponimoStradale; dice anche che StradaAppartenenza è un ruolo (cioè una funzione) che, dato un Civico, restituisce esattamente un oggetto della classe ToponimoStradale
La riga successiva, cioè

inverso CiviciDellaStrada [0..*].

dice che, dato un oggetto della classe ToponimoStradale, la funzione CiviciDellaStrada restituisce un insieme, eventualmente vuoto, di Civici che sono associati all'oggetto considerato.
Si noti che nella forma testuale è presente ridondanza, perchè l'associazione e i ruoli in questo esempio sono stati definiti in ambedue le classi interessate, in forma simmetrica; questa ridondanza può essere evitata rinunciando alla dichiarazione dell'associazione in una delle due classi, ma in questo modo può essere difficile per il lettore capire in quali associazioni è coinvolta una classe.

Questo esempio in forma grafica è mostrato nel diagramma 2.9. Nella forma grafica non c'è ridondanza e i legami tra le classi dovuti alle associazioni sono più evidenti che nella forma testuale.

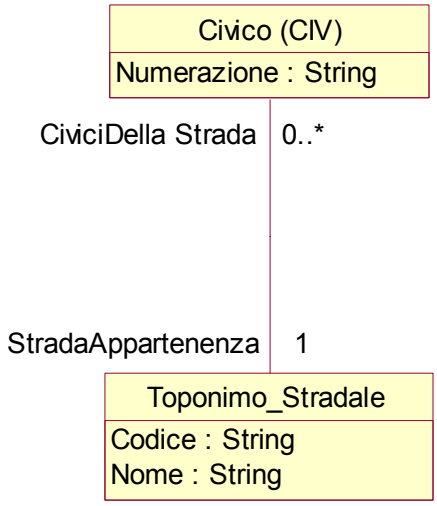

Diagramma 2.9

L'analogia dei ruoli con gli attributi permette di estendere le "dot notation" ai ruoli; pertanto, la notazione

"Civico.StradaAppartenenza"

fa riferimento alla strada che viene raggiunta seguendo il ruolo StradaAppartenenza a partire da un oggetto della classe Civico.
Questa notazione applicata ai ruoli risulta molto potente, perchè può essere applicata ricorsivamente e permette di scrivere espressioni che "navigano" attraverso allo schema; ad esempio, la scrittura

"Civico.StradaAppartenenza.nome"

si riferisce al nome della strada di appartenenza di un civico, ma se la classe ToponimoStradale partecipasse in altre associazioni con altre classi, allora potremmo raggiungere queste altre classi a partire dalla classe Civico. Le espressioni scritte in questo modo risulteranno utili nei capitoli relativi alla struttura e ai vincoli topologici.

Talvolta un'associazione possiede attributi propri. In questo caso la rappresentazione grafica è quella mostrata nel diagramma 2.10

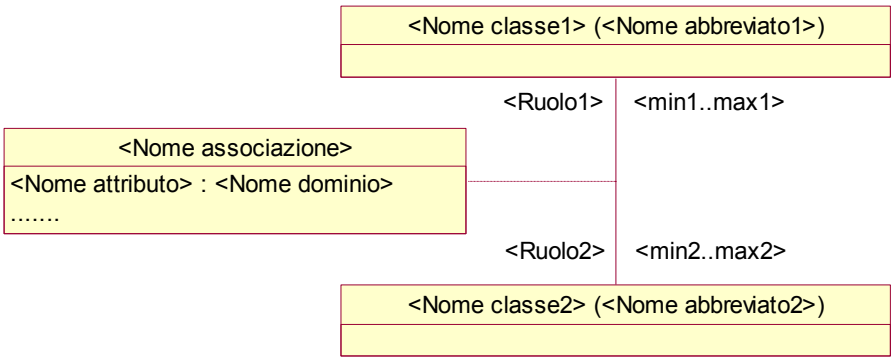

Diagramma 2.10

In questo caso la forma testuale richiede di dichiarare, in maniera indipendente rispetto alle dichiarazioni dei ruoli all'interno delle classi interessate, anche un'associazione, utilizzando la parola chiave *associazione*.
In forma testuale il diagramma 2.10 diventa quindi la seguente dichiarazione:

classe Nome_classe1
    attributi: ...;
        ruoli: Ruolo2 [min2..max2] : Nome_classe2
        inverso Ruolo1 [min1..max1]
classe Nome_classe2
    attributi: ...;
        ruoli: Ruolo1 [min1..max1] : Nome_classe1
        inverso Ruolo2 [min2..max2]
associazione Nome_associazione
        attributi: ...;

Questo tipo di dichiarazione delle associazioni può essere utilizzato nella forma testuale anche se l'associazione non possiede attributi; in questo modo si ottiene la possibilità di leggere le associazioni in maniera indipendente dalle classi, ma si inserisce ulteriore ridondanza nelle dichiarazioni e quindi è consigliabile utilizzarlo in maniera limitata.

2.1.6 Ereditarietà tra classi

La più semplice forma di gerarchia di ereditarietà rappresenta una relazione supertipo/sottotipo tra due classi, dove una classe è sottoclasse dell'altra classe (superclasse).

Le caratteristiche di questa relazione tra classi sono sia intensionali che estensionali; il livello estensionale è tipico dei database:

- a livello intensionale indica che un elemento della sottoclasse possiede tutte le proprietà che caratterizzano un elemento della superclasse (come ad esempio un cane possiede tutte le caratteristiche di un mammifero)
- a livello estensionale dichiara che un esemplare della sottoclasse appartiene anche all'estensione della superclasse (cioè un cane appartiene all'estensione dei cani e anche a quella dei mammiferi, ovvero nel database l'insieme dei cani deve essere un sottoinsieme di quello dei mammiferi)

Un esempio di definizione in GeoUML di una gerarchia di ereditarietà semplice in notazione testuale è il seguente:

classe Area di Circolazione
attributi: ...
classe Area di Circolazione Stradale
sottoclasse di Area di Circolazione
...

Si osservi che una gerarchia semplice è **incompleta**, cioè tutti gli elementi della sottoclasse appartengono alla superclasse, ma non viceversa (nell'esempio si dichiara quindi che possono esistere Aree di Circolazione che non sono Aree di Circolazione Stradale).

Una gerarchia di ereditarietà in generale può coinvolgere più di una sottoclasse; in tal caso la forma testuale è la seguente:

classe Area di Circolazione *superclasse* (Area di Circolazione Stradale,
Area di Circolazione Pedonale, Area di Circolazione Ciclabile)
attributi: ...

La forma grafica di questo esempio è rappresentata nel diagramma 2.11a.

Si precisa che con questa forma di dichiarazione la gerarchia è considerata **completa**, cioè per ogni elemento della superclasse esiste un corrispondente elemento di una sottoclasse (ovviamente continua a valere anche la proprietà che ogni elemento di una sottoclasse appartiene alla superclasse). Inoltre la gerarchia è **disgiunta**, cioè non esistono due elementi diversi nelle sottoclassi in corrispondenza dello stesso elemento della superclasse.

Per modificare la proprietà di completezza e disgiunzione è possibile aggiungere dopo la parola chiave *superclasse* le parole chiave (standard UML) *incomplete* e/o *overlapping* come nel seguente esempio in forma testuale, la cui rappresentazione grafica è data nel diagramma 2.11b:

classe Area di Circolazione *superclasse incomplete* (Area di Circolazione Stradale,
Area di Circolazione Ciclabile)
attributi: ...

Con questa dichiarazione possono esistere Aree di Circolazione che non sono nè Aree di Circolazione Stradale nè Aree di Circolazione Ciclabile, ma non possono esistere Aree di Circolazione sono sia Aree di Circolazione Stradale sia Aree di Circolazione Ciclabile, perchè non abbiamo eliminato la disgiunzione (per ottenere quest'ultimo effetto avremmo dovuto scrivere *superclasse incomplete overlapping*)

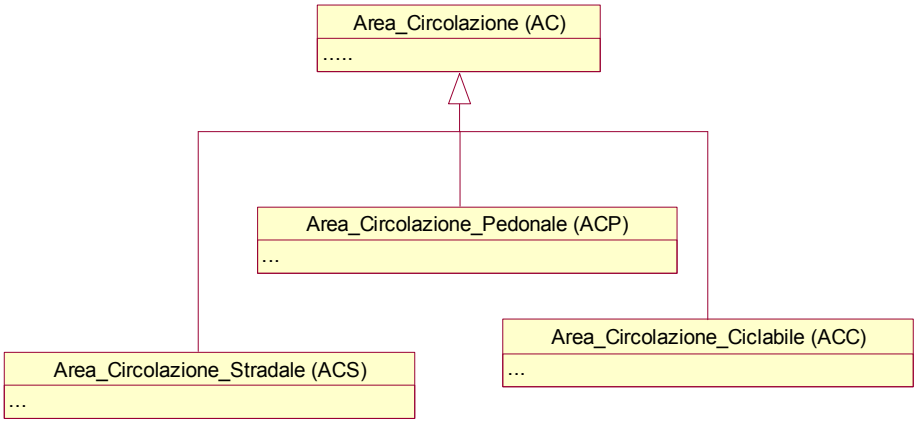

Diagramma 2.11 a

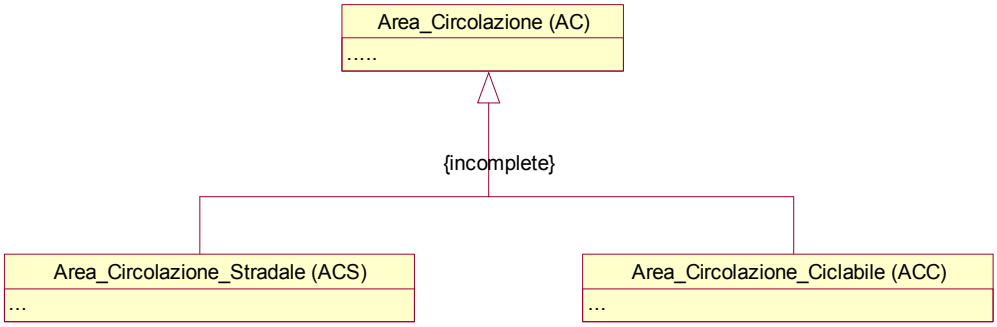

Diagramma 2.11 b

2.1.7 Vincoli generici in OCL (Object Constraint Language)

Sia sulle classi che sulle associazioni del modello è possibile definire vincoli di integrità. Tali vincoli stabiliscono l'insieme degli stati consistenti per le istanze delle classi o delle associazioni a cui sono legati. Il linguaggio adottato per la specifica dei vincoli è il linguaggio OCL (Object Constraint Language) di UML.

Un vincolo di integrità definito su una classe C definisce una proprietà che deve essere sempre verificata su tutte le istanze della classe.

Il linguaggio OCL è molto difficile da utilizzare da parte di chi scrive i vincoli e di scarsa leggibilità per chi li deve leggere e interpretare; inoltre, i vincoli generici in OCL definiscono delle proprietà dell'informazione senza fornire alcuna indicazione sulla loro implementazione. Per questo motivo un obiettivo di GeoUML è quello di permettere di definire tutti i vincoli necessari tramite costrutti specializzati senza dover mai ricorrere al linguaggio generale OCL.

In questo documento non si descrive OCL ma si rimanda alla specifica dell'OMG indicata nei riferimenti generali e al documento parallelo di definizione formale del GeoUML (documento 1n1010_1).

2.2 Estensioni rispetto al linguaggio UML standard

2.2.1 Chiave primaria

Tutti gli oggetti sono dotati di un identificatore automatico interno (OID - object identifier). Un OID è una sequenza di bit che è associata a un solo oggetto e lo identifica in tutta la base di dati.

L'identificazione degli oggetti è però in generale un problema che richiede di fare scelte diverse in base ai requisiti della classe di oggetti che viene considerata. I meccanismi di identificazione infatti variano nelle seguenti caratteristiche:

- contesto di unicità (scope): definisce in quale ambito l'identificatore è univoco
- visibilità esterna: definisce se l'identificatore può essere utilizzato all'esterno del database

Contesto di unicità

Il contesto minimo di unicità è l'unicità nell'ambito esclusivo del database (inteso in senso ristretto come dataset): un identificatore di questo tipo è garantito unico solo nel database. Il massimo di universalità è l'unicità universale tramite un ***UUID*** (Universal Unique Identifier); l'oggetto è identificato in maniera univoca rispetto a tutti gli altri oggetti esistenti al mondo.

Esistono diversi schemi di definizione di UUID; uno dei più diffusi è quello definito nell'ambito del DCE (Distributed Computing Environment), che utilizza 128 bit. Si sta affermando anche lo schema degli URI di Internet, che specializza la nozione di UUID al contesto di Internet.

Tutti questi schemi di identificazione sono molto potenti, ma usano identificatori molto lunghi rispetto agli usuali OID dei sistemi di database. In GeoUML non è stato previsto per ora alcuno schema particolare di UUID.

Visibilità

L'identificatore automatico interno (OID) non è "visibile" all'esterno del sistema, cioè non può essere utilizzato fuori, generalmente perchè il sistema si riserva di modificarlo se necessario (si pensi a certi identificatori di primitive geometriche in alcuni tipi di GIS); perciò in molti casi è opportuno definire un identificatore esterno "visibile", detto, secondo la terminologia del modello relazionale, ***chiave primaria***.

La chiave primaria è costituita da un insieme di attributi e/o ruoli in associazioni che hanno la proprietà di identificare in modo univoco (almeno in un certo contesto) un oggetto di una classe. Ad esempio, possiamo identificare una strada attraverso la sua associazione col comune di appartenenza (ruolo) e il suo codice unico all'interno del comune stesso.

Per quanto riguarda i ruoli viene richiesto che i vincoli di cardinalità posti sul ruolo siano [1..1]: vale a dire, da un oggetto della classe identificata attraverso il ruolo si deve raggiungere uno e un solo oggetto della classe identificante (nell'esempio precedente questo significa che ogni strada deve appartenere a uno e un solo comune).

La scelta di attribuire o meno una chiave primaria a una classe deve essere valutata con attenzione, perché può avere rilevanti conseguenze implementative. Un tipo particolare di chiave primaria è costituito dagli UUID citati sopra.

Con riferimento all'esempio citato sopra, la definizione di chiave primaria in forma testuale è la seguente:

classe Strada

attributi: PK Codice: *String*;

...

ruoli: *PK* ComuneAppartenenza [1..1] : Comune

inverso StradaDelComune [0..*]

classe Comune

attributi: CodiceIstat: *String*;

ruoli StradaDelComune [0..*]: Strada

inverso ComuneAppartenenza [1..1]

La forma grafica dello stesso schema è mostrata nel diagramma 2.12

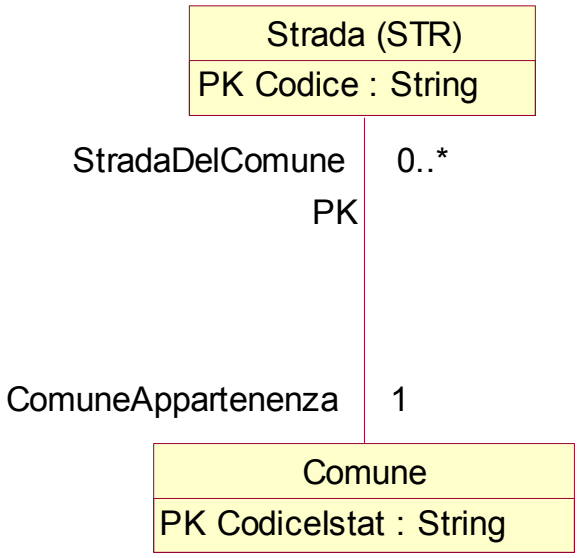

Diagramma 2.12

2.2.2 Attributo enumerato gerarchico

In alcuni casi è necessario rappresentare attributi i cui valori sono definiti attraverso una classificazione gerarchica.

Si ottiene questo tramite una versione arricchita dell'attributo enumerato, detta attributo enumerato gerarchico, nella quale dopo ogni valore della lista è possibile inserire uno o più altri attributi.

Nel seguente esempio l'attributo "tipo" sarebbe un semplice enumerato se fosse definito come

tipo (Piazza, AreaStrutturata,Tronco)

Però, dopo il primo valore "Piazza" troviamo un nuovo attributo (riconoscibile dai 2 punti finali) "circolazione: ", il cui tipo è un enumerato (rotatoria, non-rotatoria), e dopo il secondo valore "AreaStrutturata" troviamo due attributi, "sottotipo" e "dimensione", dei quali uno è enumerato e l'altro real, ecc...

classe elementoStradale
attributi: tipo (Piazza circolazione: (rotatoria, non-rotatoria),
AreaStrutturata sottotipo: (casello, incrocio),
dimensione: real,
Tronco lunghezza: real)

Si noti che si può nidificare questo tipo di attributo anche più profondamente e che si può utilizzare la definizione separata del dominio enumerato invece dell'elencazione diretta dei valori utilizzata nell'esempio, secondo le regole viste nel paragrafo relativo agli attributi enumerati.

3. Attributi e Domini Geometrici

In questo capitolo trattiamo le caratteristiche dei singoli oggetti geometrici utilizzati per rappresentare le proprietà spaziali degli oggetti delle classi applicative. Gli oggetti geometrici sono considerati isolatamente, non vengono quindi trattati gli aspetti derivanti dalle relazioni spaziali tra oggetti diversi; questi aspetti saranno trattati nel successivo capitolo.

3.1 Attributi geometrici

Una classe può possedere, oltre agli attributi già visti, anche attributi geometrici. Un attributo geometrico è un attributo i cui valori appartengono ad un tipo geometrico (che costituisce il dominio di tale attributo).
In figura 3.1 è mostrata la rappresentazione di una classe dotata di un attributo geometrico. L'unica differenza rispetto ad un attributo normale consiste nel tipo di dominio dell'attributo, che deve essere uno dei tipi geometrici.

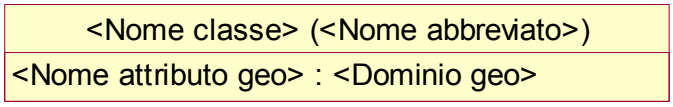

Figura 3.1

Un esempio di classe con attributo geometrico è mostrato in figura 3.2. Tale figura mostra un aspetto importante delle classi GeoUML, cioè che una classe può possedere più di un attributo geometrico. In particolare l'esempio attribuisce a ogni oggetto della classe Strada un attributo "percorso", di tipo GU_CXCurve3D (sostanzialmente si tratta di una linea tridimensionale) e un attributo "estensione", di tipo GU_CXSurface2D (sostanzialmente si tratta di una superficie nel piano).

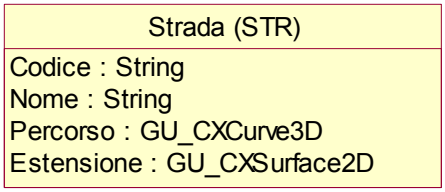

Figura 3.2

La rappresentazione testuale dello stesso esempio di figura 3.2 è la seguente:

classe Strada (STR):
attributi: Codice: *String*;
Nome: *String*;
Percorso: GU_CXCurve3D;
Estensione: GU_CXSurface2D.

3.2 Caratteristiche generali degli oggetti e dei tipi geometrici

3.2.1 Proprietà generali degli oggetti geometrici

I valori di un attributo geometrico devono appartenere a un insieme di domini particolari, detti domini o tipi geometrici. I tipi geometrici del modello GeoUML sono stati definiti specializzando i tipi dello standard ISO TC211 Spatial Schema [SPATIAL] tenendo conto dei seguenti requisiti:

- la necessità di rappresentare i punti e le linee sia in 2D che in 3D, mentre le superfici sono rappresentate solamente in 2D
- la necessità di rappresentare relazioni e vincoli di natura geometrica e topologica tra le geometrie

Le istanze di un tipo geometrico sono dette valori o oggetti geometrici. Tutti gli oggetti geometrici, indipendentemente dal loro tipo specifico, possiedono alcune proprietà comuni (questo fatto è espresso nello Spatial Schema definendo i tipi come sottoclassi della classe GM_Object, cioè della classe generale degli oggetti geometrici).

La dimensione degli oggetti geometrici è definita nel modo seguente: dimensione 0 per i punti, dimensione 1 per le linee e dimensione 2 per le superfici (gli oggetti di dimensione 3, cioè i volumi, non sono trattati nella versione attuale del GeoUML).

Un aspetto importante del modello geometrico dell'ISO TC211 e del GeoUML è la distinzione, all'interno di un oggetto geometrico, di due parti: la sua parte interna (*interior*) e la sua frontiera (*boundary*). La definizione formale di frontiera di un oggetto geometrico generico è piuttosto complicata, ma nella maggior parte degli oggetti geometrici appartenenti a un tipo geometrico di GeoUML la sua individuazione è intuitivamente semplice, come vedremo discutendo i singoli tipi.

Se consideriamo la frontiera di un oggetto geometrico a sua volta come un oggetto geometrico, vale la seguente proprietà generale: la dimensione della frontiera è inferiore di una unità rispetto a quella dell'oggetto (ad esempio, la frontiera delle linee è costituita da punti, la frontiera delle superfici è costituita da linee).

Un oggetto geometrico è detto *semplice* se e solo se non possiede punti di autointersezione – figura 3.3.a (la figura mostra un oggetto non-semplice, contenente un punto di autointersezione).

Un oggetto geometrico è detto *ciclico* se e solo se la sua frontiera è vuota (ad esempio, una circonferenza, una linea chiusa, una superficie sferica sono ciclici) – figura 3.3.b

Osservazione sulla terminologia: gli oggetti ciclici sono più comunemente detti chiusi; questa terminologia sarà però evitata in questo documento, coerentemente con lo standard Spatial Schema, perchè il termine chiuso è utilizzato anche per indicare se gli oggetti sono insiemi chiusi o aperti di punti (un oggetto geometrico è chiuso se include i suoi punti di frontiera, aperto in caso contrario).

Le seguenti funzioni, definite su tutti gli oggetti geometrici, permettono di analizzare le caratteristiche appena esposte di un oggetto geometrico:

- isSimple( ): è una funzione booleana che restituisce TRUE (vero) se l'oggetto è semplice, FALSE in caso contrario;
- isCycle( ): è una funzione booleana che restituisce TRUE (vero) se l'oggetto è ciclico, FALSE in caso contrario;
- boundary( ): è una funzione definita per tutti gli oggetti geometrici, anche se viene specializzata poi nei diversi tipi; dato un oggetto geometrico G, G.boundary( ) restituisce il valore della frontiera di G (ovvero gli oggetti geometrici che costituiscono la frontiera di G)

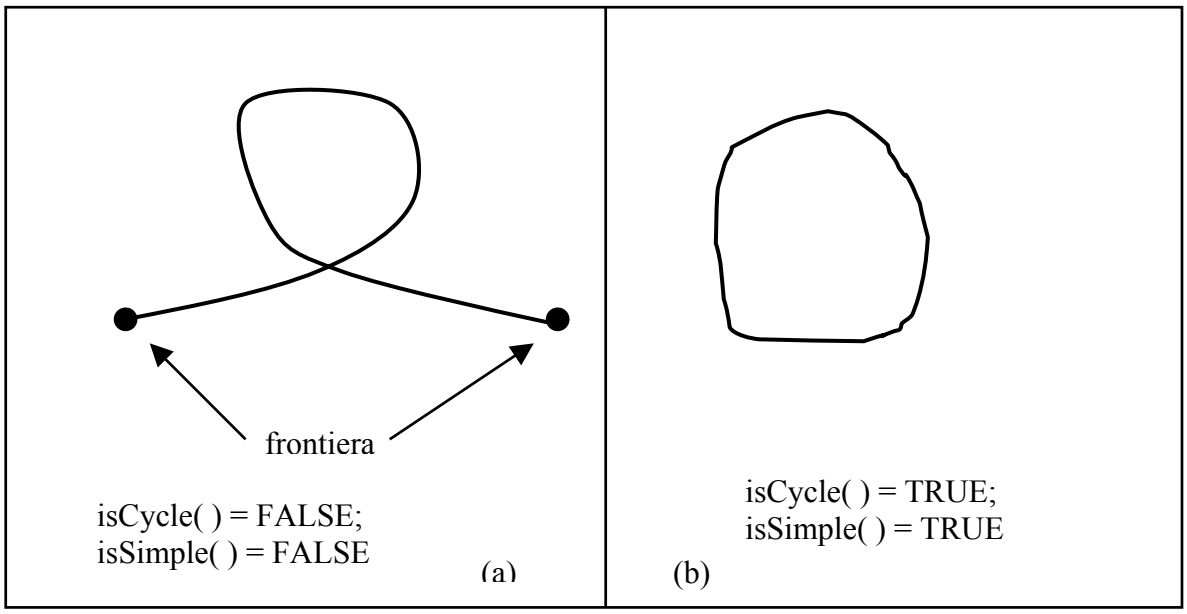

Figura 3.3

Un oggetto che costituisce la frontiera di un altro oggetto deve essere ciclico; quindi, dato un oggetto geometrico G, l'applicazione in cascata delle funzioni boundary e isCycle deve produrre sempre TRUE, ovvero il seguente predicato è sempre vero

G.boundary( ).isCycle( ) = TRUE

3.2.2 Spazi di riferimento

Una ulteriore proprietà posseduta da ogni oggetto geometrico è costituita dal suo spazio di riferimento. La funzione CRS( ), definita per tutti gli oggetti geometrici, restituisce il suo spazio di riferimento.

Lo spazio di riferimento possiede alcune proprietà che non interessano particolarmente il GeoUML, come:

- validArea – porzione di territorio dove il sistema si applica
- dati di riferimento geodetico, come il datum (a sua volta articolato in datum.type, datum.datum, ecc...) ellipsoid, coordinate system, ecc...

In GeoUML si suppongono predefiniti due spazi di riferimento, detti GU_CRS-2D e GU_CRS-3D, aventi tutti i parametri uguali tranne il parametro che esprime il numero di coordinate:

- coordinateSystem.dimension

con l'ovvia condizione che tale parametro vale 2 in GU_CRS-2D e 3 in GU_CRS-3D.

Qualsiasi oggetto geometrico deve appartenere ad uno di questi due spazi, ma la sua appartenenza non è determinata oggetto per oggetto; essa dipende dallo specifico tipo geometrico al quale l'oggetto appartiene, perchè tutti gli oggetti appartenenti ad uno stesso tipo appartengono allo spazio di riferimento associato al tipo stesso.

Tutti i tipi geometrici definiti in GeoUML hanno un nome che inizia con GU_ e termina con 2D oppure 3D, ad indicare lo spazio di riferimento al quale gli oggetti di quel tipo appartengono.

Dato che in GeoUML i nomi dei tipi geometrici hanno la struttura

GU_ttttt2D oppure GU_ttttt3D

dove ttttt indica gli aspetti più specifici del tipo (punti, linee, superfici di vario genere), quando in questo documento vorremo fare affermazioni relative ad un tipo ttttt indipendentemente dallo spazio di riferimento, parleremo di oggetti di tipo GU_ttttt.
Quindi, in base alle definizioni viste, se G è un oggetto appartenente a un tipo GU_ttttt2D la seguente affermazione è sempre vera:

G.CRS( ).coordinateSystem.dimension = 2

(ovvero: il valore dell'attributo coordinateSystem.dimension dell'oggetto di tipo CRS restituito dalla funzione CRS( ) applicata all'oggetto G è sempre 2). Analogamente per un oggetto in 3D.
Dato che gli oggetti geometrici dei tipi GU_ttttt2D e GU_ttttt3D appartengono a due spazi diversi, non esistono relazioni spaziali dirette tra loro. Tuttavia, in alcuni casi è necessario esprimere proprietà spaziali tra due oggetti geometrici appartenenti ai due diversi spazi; a questo scopo è definita, per tutti gli oggetti geometrici, la seguente funzione:

- planar( ) – restituisce un oggetto geometrico in GU_CRS-2D che è ottenuto proiettando tutti i punti dell'oggetto di partenza sul piano 2D (se l'oggetto di partenza è già in 2D, planar restituisce lo stesso oggetto)

Riassumendo, dato un oggetto geometrico generico G, appartenente a un qualsiasi tipo geometrico del GeoUML, sono definite le seguenti funzioni:

- boundary( )
- isSimple( )
- isCycle( )
- CRS( )
- planar( )

ma, in base al tipo specifico al quale l'oggetto appartiene potranno esistere dei vincoli sui valori restituiti da tali funzioni.

3.2.3 Le primitive geometriche

Gli oggetti geometrici appartenenti ai tipi geometrici del GeoUML si compongono di 5 categorie di primitive geometriche fondamentali.
Le primitive geometriche sono il punto (point), la linea (curve) e la superficie (surface). La scelta di rappresentare le linee e i punti in 2 oppure 3 dimensioni, mentre le superfici sono solamente in due dimensioni, produce due categorie di punti e due categorie di curve e una sola categoria di superfici. Quindi le primitive geometriche appartengono a 5 categorie diverse.
La nozione di primitiva, cioè di oggetto elementare, indica che a livello di GeoUML non siamo interessati alle caratteristiche di struttura interna di questi oggetti. Gli aspetti che ci interessa conoscere di una primitiva geometrica sono solamente quelli che abbiamo considerato precedentemente, cioè conoscere la sua frontiera e il suo spazio di riferimento, sapere se è semplice, sapere se è ciclica.
Le curve primitive hanno una frontiera costituita da due punti. Ogni curva primitiva inoltre è orientata e quindi i due punti di frontiera si distinguono in punto di inizio (startpoint) e punto di fine (endpoint).
Una curva primitiva ciclica e semplice come quella di figura 3.3.b è detto anello.
Una primitiva di superficie è costituita da un'unica superficie connessa la cui frontiera (esterna) è costituita da un anello e può possedere alcuni "buchi" o "isole", ognuno dotato di un anello come frontiera; quindi, la frontiera esterna di una superficie primitiva è costituita da un anello e la frontiera interna è costituita da un insieme (eventualmente vuoto) di anelli.

Nota: un anello che costituisce la frontiera di una superficie può anche essere formato da un certo numero di curve primitive concatenate in modo da formare un'unica curva composta semplice e ciclica; questo caso sarà definito precisamente più avanti.
La rappresentazione interna della geometria delle primitive, costituita generalmente dalle coordinate di un certo numero di punti e dall'indicazione del metodo di interpolazione (lineare, archi, spline, o altro) adottato, non ci interessa; nelle figure di questo documento la geometria è rappresentata quindi arbitrariamente, senza ipotizzare in particolare uno specifico metodo di interpolazione, ma evidenziando solamente gli aspetti significativi illustrati sopra. Ad esempio, in figura 3.3.a possiamo immaginare una qualsiasi rappresentazione interna della curva rappresentata, ma è significativa l'esistenza del punto di autointersezione e dei due punti di frontiera. I punti di frontiera delle curve primitive sono a loro volta delle primitive di tipo puntiforme e sono evidenziati nelle figure; tutti gli altri punti interni della curva, anche se servono per rappresentarne internamente la geometria, non sono evidenziati nelle figure.

Cenni alla rappresentazione interna e allo standard ISO TC211 Spatial Schema
Lo standard prevede una rappresentazione complessa delle curve e delle superfici, ammettendo molti diversi metodi di interpolazione, mentre i punti sono rappresentati da una coppia (in 2D) oppure da una terna (in 3D) di coordinate.
Nello standard ogni curva primitiva è suddivisa in segmenti concatenati, che differiscono in base al loro metodo di interpolazione. Ogni segmento è caratterizzato da un certo numero di punti e da un metodo di interpolazione.
Analogamente, ogni superficie è suddivisa in alcune "Surface patch", ognuna delle quali può possedere un suo metodo di interpolazione; lo standard prevede molti metodi complessi. Ogni surface patch deve essere connessa e può avere dei buchi.
Lo standard prevede che un Application Schema selezioni un sottoinsieme di tali rappresentazioni. Come già detto, in GeoUML la scelta di tale sottoinsieme non è rilevante e può essere modificata in base all'evoluzione delle esigenze applicative e del supporto fornito dai sistemi.
Allo stato attuale (2003) sembra realistico prevedere un modello piuttosto semplice di curve, già supportato dai sistemi che aderiscono al "simple feature model"(cioè lo standard di rappresentazione della geometria adottato anche nel linguaggio SQL Multimedia, vedi [SFM99]), cioè il modello "linestring"; in questo modello una primitiva lineare (curva primitiva) è costituita da una sequenza di segmenti di retta concatenati. I punti terminali dei segmenti sono detti vertici. Si ricorda che tale modello può essere sia bidimensionale che tridimensionale.
Per le superfici esiste una restrizione molto forte che deriva dal fatto che le superfici sono solamente nello spazio di riferimento bidimensionale, quindi esse sono per forza planari.

3.3 I tipi geometrici del GeoUML

I tipi geometrici del GeoUML, cioè i tipi che costituiscono i domini degli attributi geometrici, sono stati determinati in modo da caratterizzare il più possibile in maniera precisa i valori che tali attributi possono possedere.

3.3.1 I tipi GU_Point2D e GU_Point3D

Questi due tipi definiscono oggetti geometrici puntiformi in 2 e 3 dimensioni.
In effetti, gli oggetti geometrici puntiformi coincidono in pratica con la primitiva geometrica punto, quindi non richiedono ulteriori commenti.
Si precisa che la frontiera dei punti è vuota, quindi se P è un oggetto di tipo GU_Point, P.boundary( ) restituisce l'insieme vuoto, ovvero

P.boundary( ).isEmpty( ) = TRUE

è sempre vera (isEmpty( ) è una funzione booleana applicabile agli oggetti di tipo insieme che restituisce TRUE se l'insieme è vuoto).

3.3.2 I tipi GU_CPCurve2D e GU_CPCurve3D

Questi due tipi geometrici costituiscono i tipi fondamentali per la rappresentazione di geometrie lineari in 2 e 3 dimensioni.

Essi sono derivati, come richiamato dalle lettere CP nel loro nome, dalle classi GM_compositeCurve dello Spatial Schema, cioè dalla nozione di "curva composta".

Una curva composta è costituita da una sequenza di curve primitive semplici, concatenate e prive di qualsiasi intersezione tra i loro punti interni.

Il termine "concatenate" significa che è possibile ordinare le N curve primitive che costituiscono la curva composta in modo che:

- per 1<=i<N, lo endpoint della primitiva i-esima coincida con lo startpoint della primitiva (i+1)-esima;
- se la curva composta non è ciclica, allora lo startpoint della prima primitiva sia startpoint della curva composta e lo endpoint della N-esima primitiva sia endpoint della curva composta.
- se invece la curva composta è ciclica, lo startpoint della prima primitiva coincida con lo endpoint della N-esima primitiva.

In figura 3.4 è mostrato un esempio di curva composta costituita da 3 curve primitive. Per ogni curva primitiva sono indicate le primitive puntiformi che ne costituiscono la frontiera.

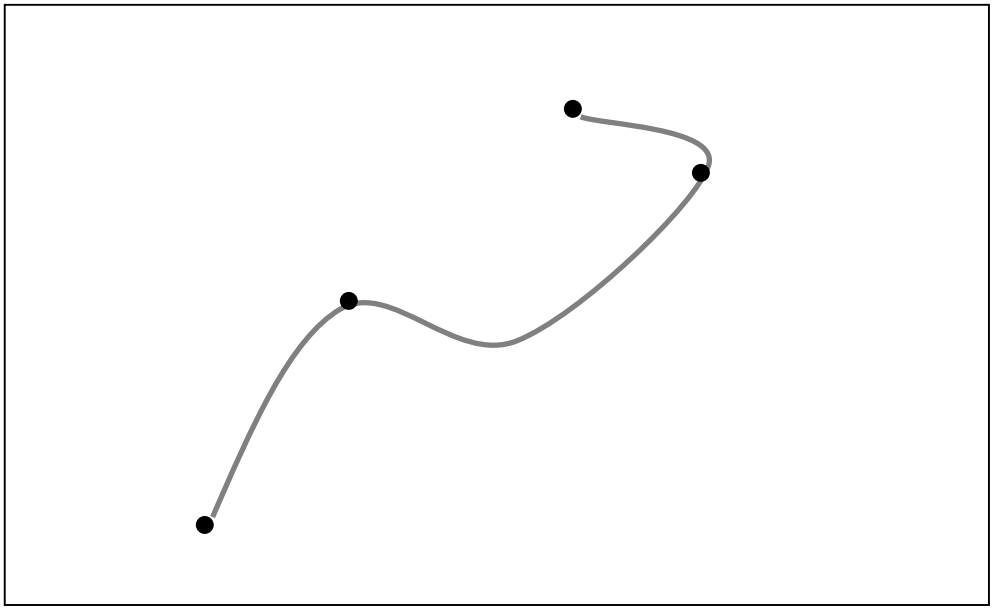

Figura 3.4

La funzione boundary( ) di un oggetto geometrico L di tipo GU_CPCurve restituisce i due punti terminali della curva; quindi l'espressione L.boundary( ) produce una coppia di punti. Si noti che i punti terminali delle singole curve primitive condivisi da più di una primitiva sono punti interni della curva composta.

Le linee di tipo GU_CPCurve sono *orientate*, quindi i due punti che costituiscono la frontiera sono distinguibili in un punto di inizio e un punto di fine, restituiti dalle due funzioni startpoint( ) e endpoint( ) applicate al boundary; le due espressioni

L.boundary.startpoint
L.boundary.endpoint

restituiscono rispettivamente il punto di inizio e di fine di L.
Si osservi che una curva composta possiede tutte le proprietà di una curva primitiva; in particolare, una curva composta può essere costituita da una sola curva primitiva.
I motivi per cui in GeoUML le curve più elementari sono date dal tipo "curva composta" e non da un tipo "curva primitiva" sono i seguenti:

1. la curva composta possiede tutte le proprietà della curva primitiva, quindi quest'ultima non è necessaria,
2. una curva composta può essere sempre decomposta in un numero maggiore di primitive senza che il suo tipo cambi (mentre la decomposizione di una curva primitiva la trasformerebbe in una curva composta) – vedremo trattando degli attributi a tratti che questo motivo è importante.

Consideriamo ora alcuni esempi particolari di curve composte (figura 3.5)

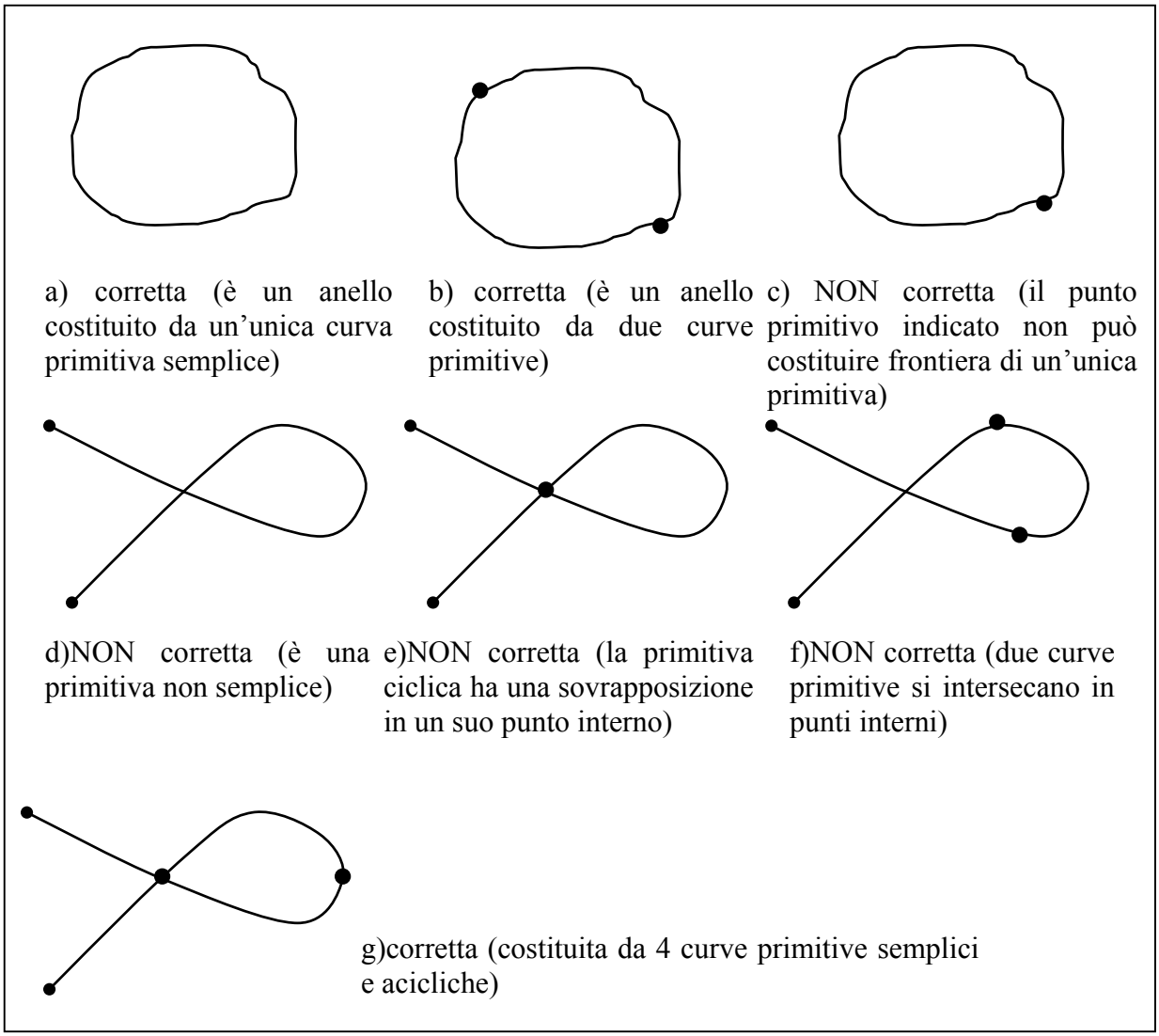

Figura 3.5

I casi di figura mostrano che il vincolo di semplicità delle curve primitive che costituiscono una curva composta è importante; a causa di tale vincolo un'asola deve essere composta da almeno 4 curve primitive.
I due anelli di figura a e b sono anelli corretti, ma la curva di figura c è errata perchè il punto indicato non può essere una frontiera (sarebbe una frontiera di una primitiva con se stessa); non si confonda a questo proposito la nozione di inizio e fine nel processo di digitalizzazione di una curva con il risultato finale, che in un anello costituito da un'unica primitiva non possiede le nozioni di inizio e fine.
E' importante tenere ben distinte le proprietà di una curva composta da quelle delle sue primitive; mentre le primitive che costituiscono una curva composta devono essere semplici, la curva composta può essere non semplice; in effetti la curva composta di figura 3.5.g è non semplice.

3.3.3 I tipi GU_CPRing2D e GU_CPRing3D

Abbiamo già visto che un caso particolare di curve composte è costituito dalle curve composte cicliche semplici o anelli (figura 3.5.a e 3.5.b).

Un anello (Ring) è una curva composta semplice e ciclica

Per definire in GeoUML un attributo di tipo anello esistono i due tipi GU_CPRing2D e GU_CPRing3D. Si osservi che un attributo di tipo GU_CPCurve ammette già gli anelli tra i propri valori; definire un attributo di tipo GU_CPRing serve quindi per specificare che si vogliono solamente anelli come possibili valori.

In figura 3.6 sono mostrati alcuni casi di oggetti geometrici che sono anelli e alcuni che non lo sono.

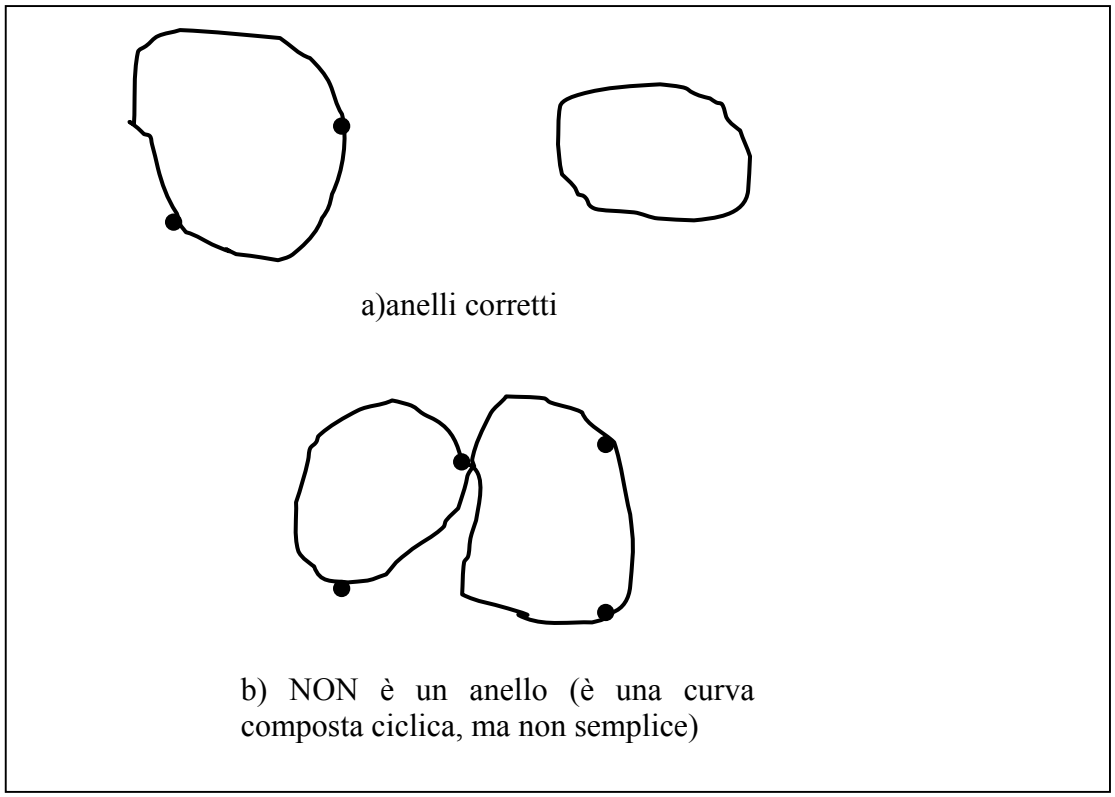

Figura 3.6

Possiamo interpretare figura 3.6.b come 2 anelli con un punto interno in comune (gli anelli infatti hanno solamente punti interni, essendo cicli e quindi essendo la loro frontiera vuota).

3.3.4 I tipi GU_CXCurve2D e GU_CXCurve3D

Questi tipi, detti curve complesse (CXCurve) permettono di rappresentare oggetti geometrici costituiti da un insieme di curve primitive che possono intersecarsi esclusivamente nei loro punti terminali.

Un oggetto di tipo CX_Curve è:

- *un insieme di curve primitive semplici*
- *prive di intersezioni tra i loro punti interni*
- *che possono condividere i loro punti terminali*

In figura 3.7 è mostrato un oggetto geometrico di questo tipo. Si noti che NON è richiesto che l'insieme sia connesso; un oggetto di tipo CXCurve è quindi molto generale.

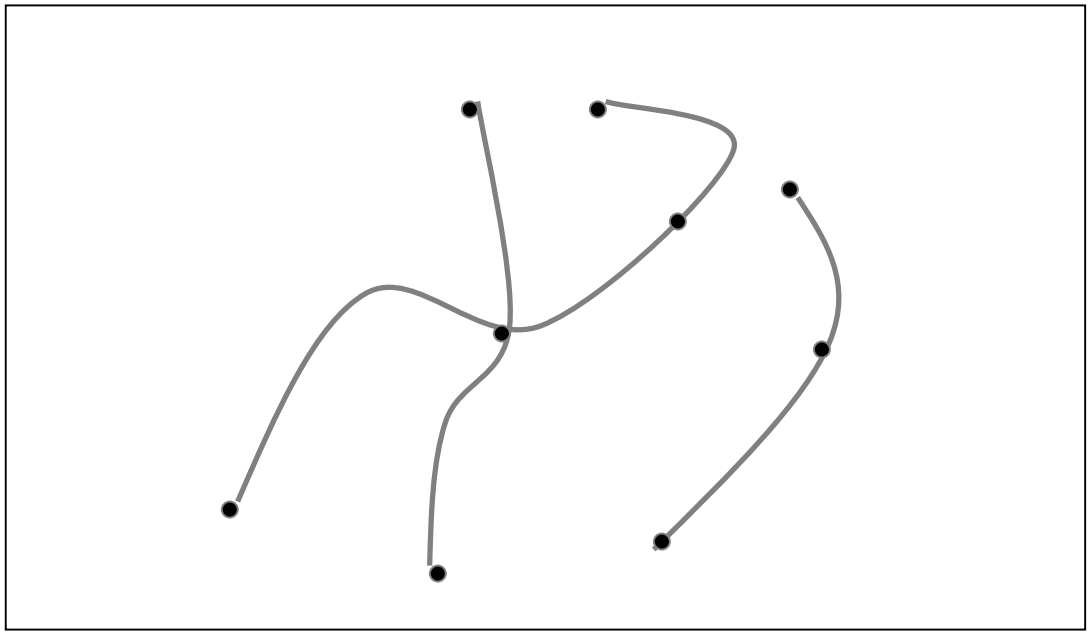

Figura 3.7

Nelle curve complesse la frontiera non è costituita da due punti, come nelle curve composte, ma da tutti i punti che sono terminali di una unica curva primitiva. Quindi, se L è una curva complessa, la funzione L.boundary( ) produce un insieme di punti.
Si noti che le curve composte (CPCurve) sono un caso particolare di curve complesse, nelle quali ogni curva primitiva si concatena con un'altra curva primitiva.

3.3.5 I tipi GU_CXRing2D e GU_CXRing3D

Una sottoclasse di GU_CXCurve è costituita dalla classe GU_CXRing, i cui oggetti sono insiemi di anelli disgiunti; più precisamente:

Un oggetto di tipo GU_CXRing è:

- *un insieme di anelli*
- *privi di intersezioni nei loro punti interni*

La conseguenza di questa definizione è che i singoli anelli che costituiscono un CXRing devono essere disgiunti tra loro.
In figura 3.8.sono mostrati alcuni esempi di CXRing corretti e alcuni casi simili, ma non corretti.

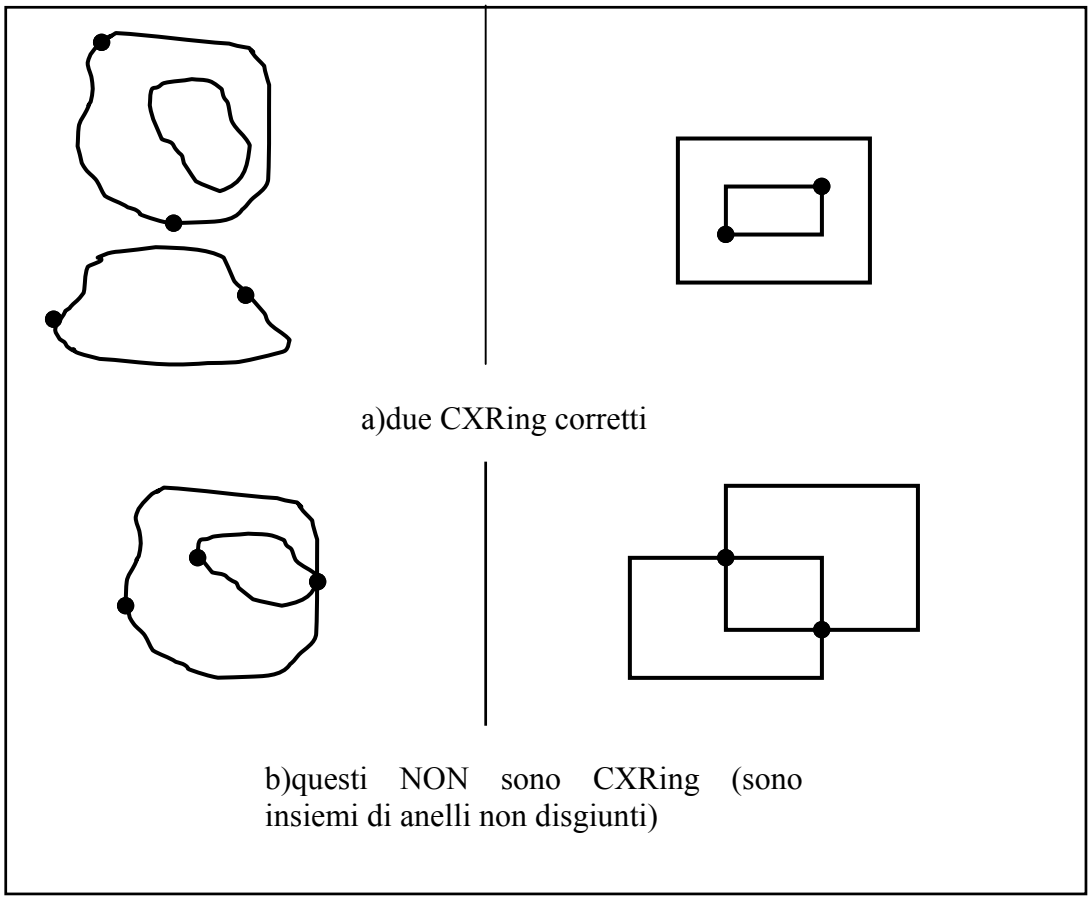

Figura 3.8

L'utilità del tipo CXRing può apparire limitata dal vincolo che gli anelli siano disgiunti, ma questo vincolo è necessario, perchè una CXCurve è un insieme di primitive e non un insieme di sottoinsiemi, come sarebbe necessario per descrivere in maniera univoca anelli non disgiunti.
Le strutture di figura 3.8.b possono essere definite in GeoUML utilizzando gli aspetti strutturali definiti nel capitolo successivo, creando oggetti composti da altri oggetti; un tipo invece definisce la struttura di oggetti singoli.

3.3.6 I tipi GU_CNCurve2D e GU_CNCurve3D (curve complesse connesse)

Gli oggetti di tipo CNCurve sono una sottoclasse degli oggetti di tipo CXCurve caratterizzata dal fatto di essere connessi. Valgono quindi per questi oggetti tutte le proprietà viste per le curve complesse con in più la propietà di connessione.
In figura 3.9 è mostrato un esempio di GU_CNCurve.

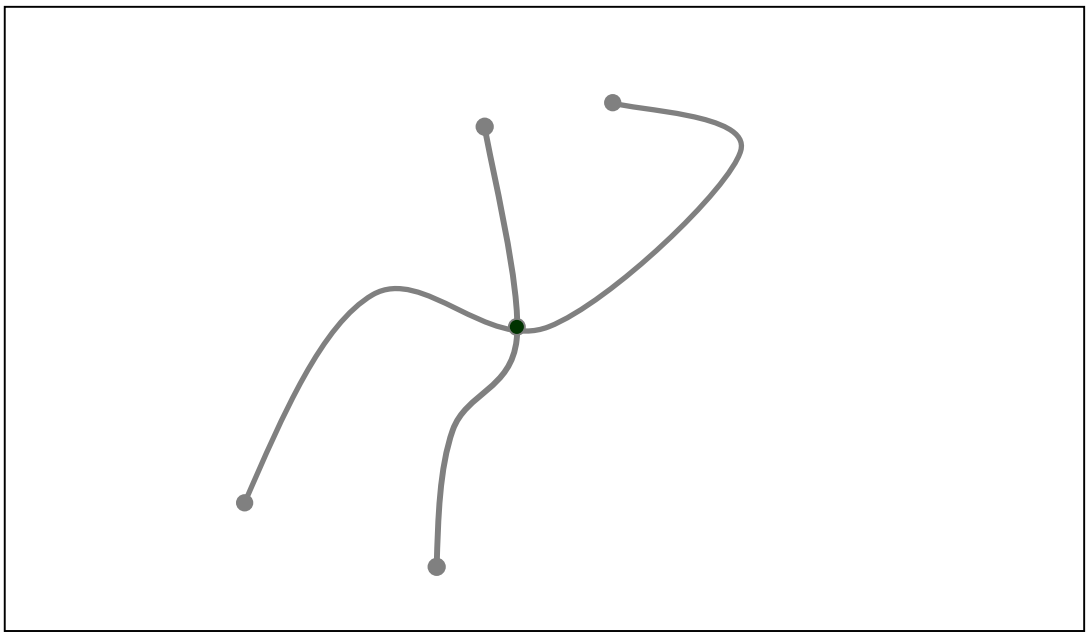

Figura 3.9

Confrontando l'esempio di figura 3.9 con quello di figura 3.7 si può verificare che le curve connesse costituiscono il punto di passaggio tra le curve complesse e quelle composte; l'oggetto di figura 3.7, di tipo CXCurve, può infatti essere considerato come una coppia di oggetti di tipo CNCurve, uno dei quali è rappresentato in figura 3.9, mentre l'altro è un caso particolare di CNCurve, perchè soddisfa anche i requisiti di una CPCurve.
Riassumendo: le curve composte sono un caso particolare di curve connesse che sono a loro volta un caso particolare di curve complesse.

3.3.7 Il tipo GU_CPSurface2D

Questo tipo rappresenta le superfici composte nel piano 2D, analogamente a quanto visto per le curve composte.

Una superficie composta è costituita da un insieme di superfici primitive adiacenti tra loro in modo tale che da ogni primitiva si possa raggiungere ogni altra primitiva senza attraversare la frontiera della superficie composta.

Inoltre, deve valere la seguente proprietà:

Ogni porzione di frontiera condiviso tra due superfici primitive adiacenti deve essere costituito da un insieme ben definito di curve primitive

In Figura 3.10 è mostrata una superficie composta da 3 superfici primitive sp1, sp2 e sp3 adiacenti.

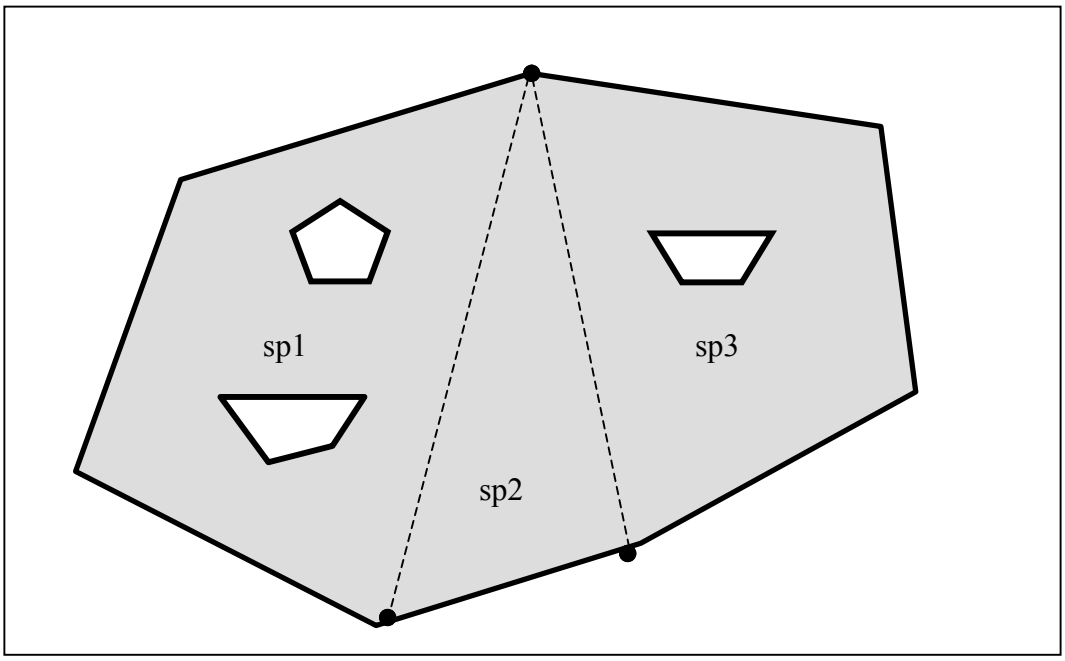

Figura 3.10

La frontiera di una superficie composta è quella complessiva (linee continue in figura), ed è ottenuta eliminando dall'insieme delle frontiere delle singole superfici primitive i lati condivisi da più di una primitiva (linee tratteggiate in figura). Tale frontiera complessiva è costituita da un anello esterno e da un insieme (eventualmente vuoto) di anelli interni. Dato un oggetto S di tipo GU_CPSurface gli elementi della frontiera sono ottenibili tramite le seguenti funzioni:

- S.boundary( ).exterior: restituisce un oggetto di tipo GU_CPRing2D, che costituisce la frontiera esterna
- S.boundary( ).interior: restituisce un insieme (eventualmente vuoto) di oggetti di tipo GU_CPRing2D, che costituisce la frontiera interna

In figura 3.10 il numero di curve primitive è quello minimo per soddisfare tutte le proprietà degli anelli di frontiera; tali anelli potrebbero essere costituiti da primitive più piccole e numerose. In particolare è importante osservare che, in base alla definizione, nessuna curva primitiva può appartenere contemporaneamente a una frontiera condivise tra due superfici primitive (linee tratteggiate in figura) e alla frontiera complessiva.

Si osservi che una superficie composta possiede tutte le proprietà di una superficie primitiva; in particolare, una superficie composta può essere costituita da una sola superficie primitiva.

3.3.8 Il tipo GU_CXSurface2D

Una superficie complessa (CXSurface) è costituita da un insieme di superfici primitive che possono essere adiacenti ma devono essere prive di intersezioni tra i loro punti interni.
Ogni porzione di frontiera condiviso tra due superfici primitive adiacenti deve essere costituito da un insieme ben definito di curve primitive

In sostanza, nelle superfici complesse è eliminato il vincolo di continuità presente nelle superfici composte.
Un esempio è mostrato in figura 3.11.
Il rapporto tra le superfici complesse e quelle composte è analogo a quanto visto relativamente a quello tra le curve complesse e quelle composte, cioè

1 le superfici composte sono un caso particolare delle superfici complesse, e
2 una superficie complessa può essere considerata come un insieme di superfici composte disgiunte tra loro.

Tuttavia, non esiste l'analogo del tipo intermedio costituito dalle curve connesse, perchè nel piano 2D le superfici connesse coincidono con le superfici composte.

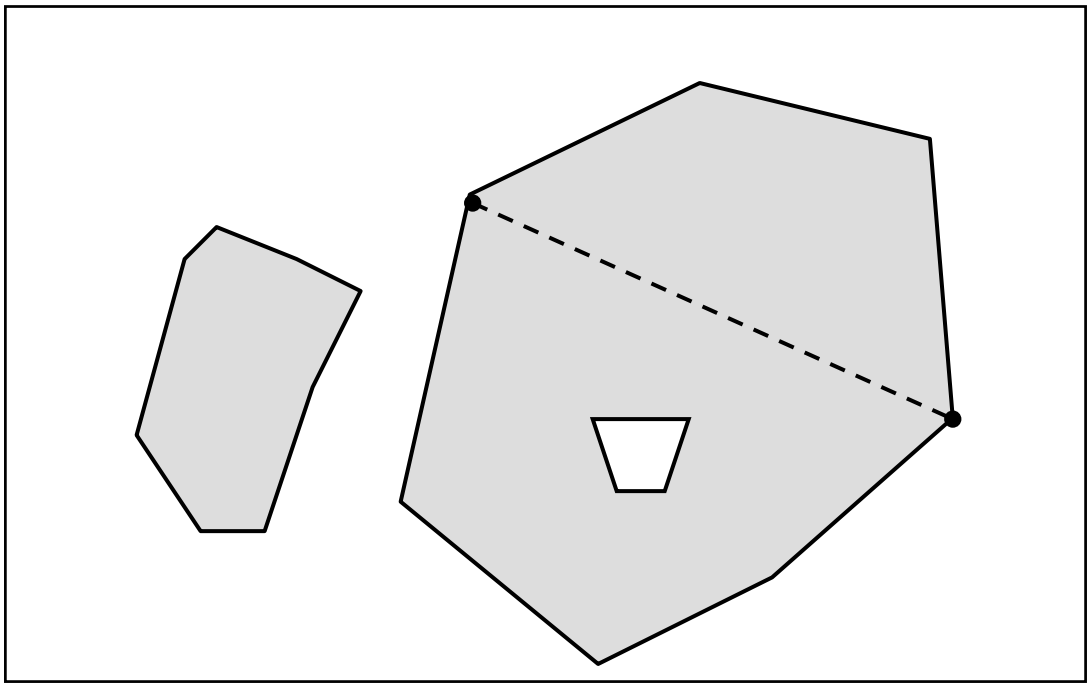

Figura 3.11

La frontiera delle superfici complesse è ottenuta eliminando dall'insieme delle frontiere delle singole superfici primitive i lati condivisi da più di una primitiva, come per le superfici composte; però, a differenza delle superfici composte, il risultato di questa definizione produce un insieme di anelli esterni.
Dato un oggetto S di tipo GU_CXSurface gli elementi della frontiera sono ottenibili tramite le seguenti funzioni:

- S.boundary( ).exterior : restituisce un insieme di oggetti di tipo GU_CPRing2D, perchè la frontiera esterna, a differenza delle GU_CPSurface, non è costituita da un unico anello
- S.boundary( ).interior : restituisce un insieme di oggetti di tipo GU_RingCP2D

3.3.9 Il tipo GU_Complex2D e GU_Complex3D

Gli oggetti di tipo GU_Complex permettono di rappresentare oggetti geometrici costituiti dai cosiddetti "complessi".

Un complesso è costituito da un insieme di primitive geometriche di vario tipo, per le quali valgono le seguenti proprietà:

1. *sono tutte semplici,*
2. *i loro punti interni sono tutti disgiunti,*
3. *se una primitiva geometrica appartiene a un complesso, anche le primitive che ne costituiscono la frontiera appartengono al complesso.*

I complessi che contengono superfici esistono solo in 2D; i complessi 3D sono costituiti solamente da curve e da punti.
In figura 3.12 è mostrato un oggetto di tipo GU_Complex2D.

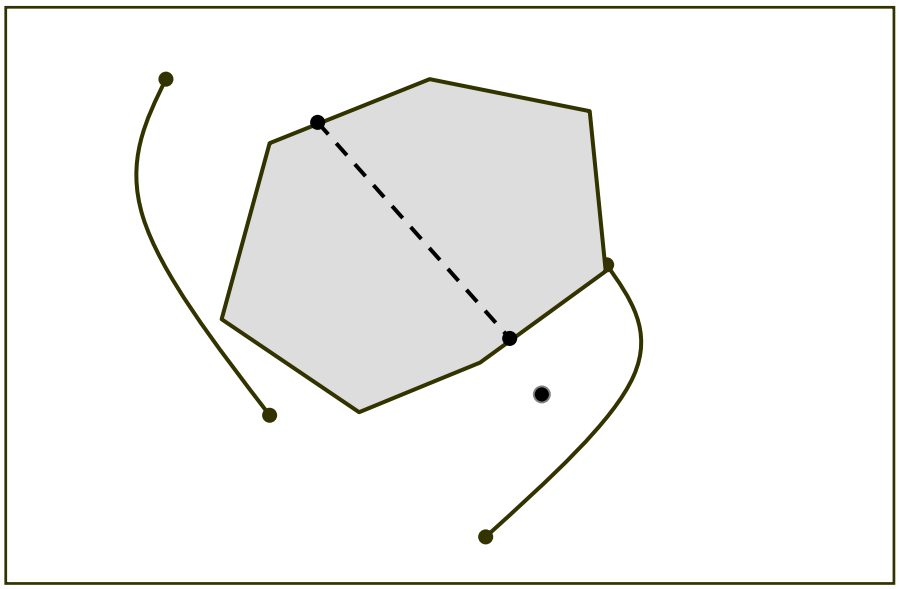

Figura 3.12

Nei complessi generici multidimensionali, cioè costituiti da primitive di diversa dimensione, è problematico dare una definizione univoca della frontiera. Pertanto, in GeoUML non è permesso fare riferimento alla funzione boundary( ) di un complesso generico (vedremo che per questo motivo alcune relazioni spaziali tra complessi generici non possono essere espresse).
Osserviamo che per i tipi GU_CXCurve2D e GU_CXSurface2D, che sono dei casi particolari, cioè dei sottotipi, di GU_Complex2D costituiti da primitive omogenee, la frontiera è invece ben definita, come abbiamo visto in precedenza (in realtà questo è il motivo per cui questi tipi sono stati definiti, riducendo la necessità di riferirsi a complessi generici ai soli casi in cui ciò risulta inevitabile).
Un'osservazione importante riguarda il caso in cui in un complesso multidimensionale una linea sia contenuta in una superficie. Per la regola di disgiunzione dei complessi i punti (interni) di una curva primitiva non possono sovrapporsi ai punti interni di una superficie primitiva, ma solo ai suoi punti di frontiera; se si verifica un caso simile è necessario definire una frontiera interna (un "buco") nella superficie costituita da una curva primitiva chiusa in anello con se stessa.

3.3.10 I tipi aggregati

Un aggregato è costituito da un insieme di primitive geometriche senza nessun vincolo particolare. Oltre agli aggregati generici:

- GU_Aggregate2D
- GU_Aggregate3D,

che possono contenere primitive geometriche di ogni tipo appartenenti al proprio spazio di riferimento, esistono gli aggregati omogenei, detti Multipoint, Multicurve, ecc..., costituiti da primitive omogenee; tali tipi in GeoUML sono:

- GU_MPoint2D: aggregato di soli punti in 2D;
- GU_Mpoint3D: aggregato di soli punti in 3D;
- GU_Mcurve2D: aggregato di sole curve in 2D;
- GU_Mcurve3D: aggregato di sole curve in 3D;
- GU_Msurface2D: aggregato di sole superfici in 2D;
- GU_MRing2D: aggregato di anelli in 2D;
- GU_MRing3D: aggregato di anelli in 3D;

Gli aggregati sono trattati brevemente, perchè la nozione di aggregato non è in realtà fondamentale in GeoUML, in quanto un aggregato può essere costruito combinando la nozione di insieme del UML con un tipo geometrico non aggregato. Si consideri ad esempio una classe i cui oggetti possono avere un attributo "puntiRappresentativi" costituita da un insieme di punti; le due dichiarazioni seguenti

attributi:
puntiRappresentativi: [0..*] GU_Point3D

e

attributi:
puntiRappresentativi: [1..1] GU_Mpoint3D

differiscono solo nel fatto che nel primo caso il valore dell'attributo è costituito da un insieme di oggetti geometrici di tipo GU_Point3D mentre nel secondo caso tale valore è costituito da un unico oggetto geometrico di tipo GU_Mpoint3D, e quindi la nozione di insieme è incorporata nella geometria.
Vale invece la pena di richiamare la differenza tra gli aggregati e i complessi: ambedue sono insiemi, ma i complessi (e i loro derivati, cioè le curve e superfici complesse e composte) hanno delle proprietà supplementari, per cui un complesso è anche un aggregato, ma non viceversa. I complessi sono rilevanti perchè, come vedremo nel capitolo successivo, le loro proprietà supplementari rispetto al fatto di essere un insieme di primitive esprimono molte proprietà strutturali dell'informazione geometrica.

3.3.11 Riepilogo di tutti i tipi e legami di classe/sottoclasse

I legami esistenti tra i tipi GeoUML, già enunciati nella presentazione dei singoli tipi, sono rappresentati sinteticamente in figura 3.13, dove le frecce vanno da un sottotipo a un supertipo, a condizione di essere nello stesso spazio di riferimento. Ad esempio, un oggetto di tipo GU_CPCurve2D è anche di tipo GU_CNCurve2D e quindi anche di GU_CXCurve2D e di GU_Complex2D.
Anche se un complesso costituisce un caso particolare di aggregato, e quindi ad esempio un GU_Complex2D costituisce un caso particolare di un GU_Aggregate2D e una GU_CXCurveND costituisce un caso particolare di GU_McurveND, non sono indicati i legami tra i derivati dei complessi e degli aggregati, essendo questo tipo di legame generalmente irrilevante.

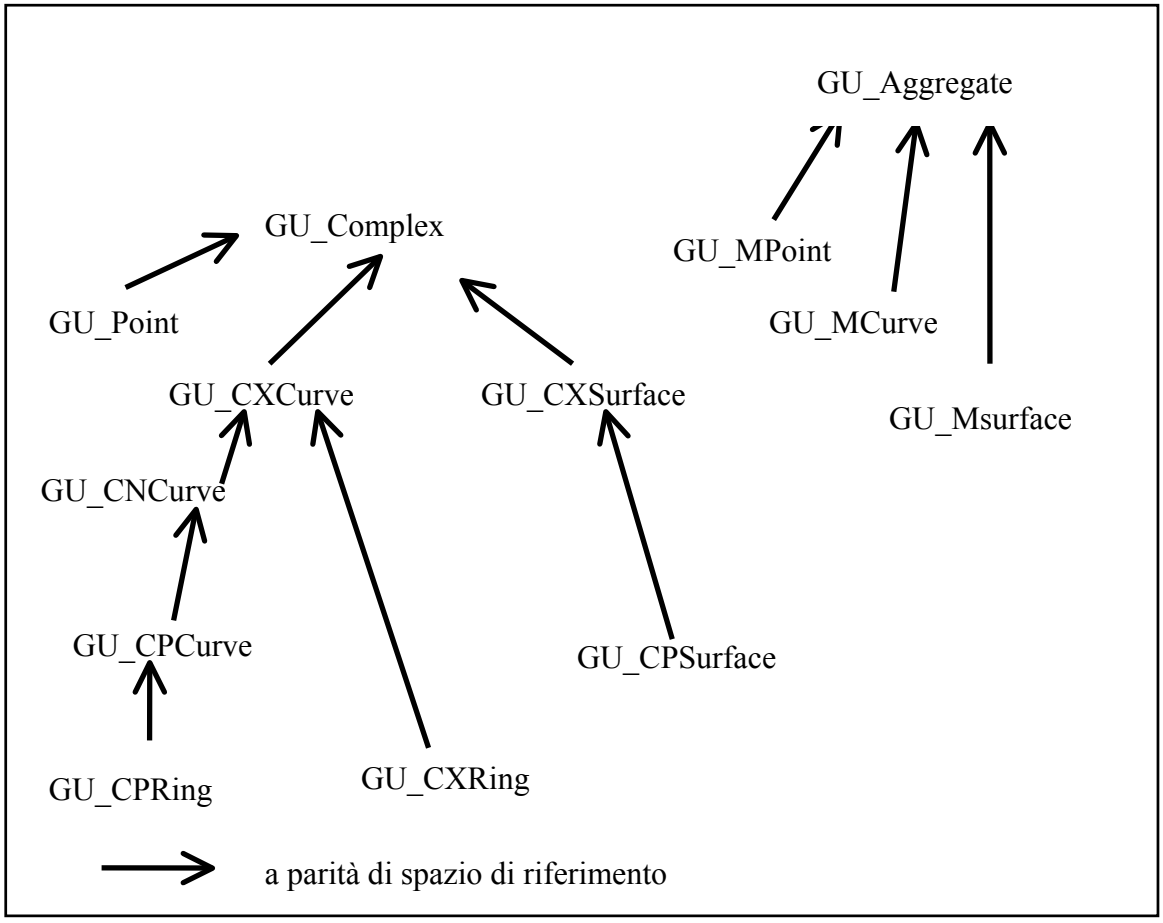

Figura 3.13

La gerarchia relativa ai complessi permette di sostituire le definizioni dei vari tipi date in precedenza con definizioni basate sul raffinamento dei tipi a partire da quello più generale; pertanto, partendo dalla definizione già data di complesso:

Un complesso è costituito da un insieme di primitive geometriche di vario tipo per le quali valgono le seguenti proprietà:

1. *i loro punti interni sono tutti disgiunti,*
2. *se una primitiva geometrica appartiene a un complesso, anche le primitive che ne costituiscono la frontiera appartengono al complesso*

possiamo dare le seguenti definizioni:

- *Una curva complessa è un complesso costituito da sole curve;*
- *Una curva connessa è una curva complessa connessa;*
- *Una curva composta è una curva connessa con due soli punti di frontiera;*
  (oppure, seguendo lo Spatial Schema: una curva composta è un complesso isomorfo con una curva primitiva)
- *Un anello complesso è una curva complessa semplice e ciclica (isCycle=TRUE);*
- *Un anello composto è una curva composta semplice e ciclica;*
- *Una superficie complessa è un complesso costituito solamente da superfici;*
- *Una superficie composta è una superficie complessa la cui frontiera esterna è costituita da un solo anello*
  (oppure, seguendo lo Spatial Schema: una superficie composta è un complesso isomorfo con una superficie primitiva)

Questo tipo di ridefinizione rappresenta più esplicitamente il legame di specializzazione esistente tra i vari tipi, anche se esplicita meno il significato intuitivo dei tipi.

3.3.12 Rapporto tra i tipi GeoUML e lo standard ISO TC211 Spatial Schema

In figura 3.14 sono riportate le derivazioni dei tipi GeoUML dai tipi dello standard ISO "Spatial Schema", indicati dal prefisso GM_ . Per una definizione formale e dettagliata del GeoUML come specializzazione dello standard si rimanda al documento parallelo 1n1010_1.

Classe di GeoUML	Classe di Spatial Schema da cui è derivata
GU_Point2D	GM_Point
GU_Point3D	GM_Point
GU_CPCurve2D	GM_CompositeCurve
GU_CPCurve3D	GM_CompositeCurve
GU_CPRing2D	GM_CompositeCurve
GU_CPRing3D	GM_CompositeCurve
GU_CPSurface2D	GM_CompositeSurface
GU_CNCurve2D	GM_Complex
GU_CNCurve3D	GM_Complex
GU_CXCurve2D	GM_Complex
GU_CXCurve3D	GM_Complex
GU_CXRing2D	GM_Complex
GU_CXRing3D	GM_Complex
GU_CXSurface2D	GM_Complex
GU_Complex2D	GM_Complex
GU_Complex3D	GM_Complex
GU_Aggregate2D	GM_Aggregate
GU_Aggregate3D	GM_Aggregate
GU_MPoint2D	GM_MultiPoint
GU_MPoint3D	GM_MultiPoint
GU_MCurve2D	GM_MultiCurve
GU_MCurve3D	GM_MultiCurve
GU_MSurface2D	GM_MultiSurface
GU_MRing2D	GM_Aggregate
GU_MRing3D	GM_Aggregate

Figura 3.14

4. Struttura derivante dalle Relazioni Spaziali tra Attributi Geometrici

Nel capitolo precedente sono state descritte le proprietà degli oggetti appartenenti ai singoli tipi geometrici senza considerare la interazione tra oggetti diversi. In questo capitolo si descrivono gli aspetti di GeoUML che permettono di controllare tale interazione in modo strutturale, cioè definendo proprietà che determinano la strutturazione delle primitive geometriche, mentre nel successivo capitolo verranno definite relazioni spaziali più generali, che non si traducono in proprietà strutturali.

4.1 Le relazioni tra complessi e la struttura geometrica

Alla base delle relazioni strutturali definibili in GeoUML c'è la nozione di complesso e la appartenenza di più oggetti geometrici allo stesso complesso. In questo capitolo mostreremo la seguente caratteristica dell'appartenenza di più oggetti allo stesso complesso:

> *se due oggetti geometrici appartengono allo stesso complesso, la strutturazione delle loro primitive è tale da permettere di derivare le relazioni spaziali tra i due oggetti dalle proprietà di condivisione delle loro primitive*.

Nel seguito useremo il termine "*tipo complesso*" per indicare uno qualsiasi dei sottotipi di GU_Complex; pertanto, l'affermazione che un oggetto geometrico G è di tipo complesso significa che G appartiene ad uno dei seguenti tipi: GU_Complex, GU_Point, GU_CXCurve, GU_CNCurve, GU_CPCurve, GU_CXRing, GU_CPRing, GU_CXSurface e GU_CPSurface.

4.1.1 La relazione tra Supercomplesso e Sottocomplesso (Supercomplex e Subcomplex)

Dato che tutti gli oggetti di tipo complesso sono degli insiemi di primitive geometriche, la relazione di contenimento tra insiemi permette di definire la associazione di contenimento tra oggetti complessi nel modo seguente:

> *Dati due oggetti geometrici G1 e G2, ambedue di tipo complesso, diciamo che G1 è un Sottocomplesso di G2 se l'insieme di primitive che appartengono a G1 è contenuto nell'insieme di primitive che appartengono a G2.*
> *Inoltre, se G1 è un Sottocomplesso di G2, diciamo che G2 è un Supercomplesso di G1 e che esiste l'associazione Contains tra G1 e G2.*

4.1.2 Conseguenze della relazione di Sottocomplesso

La relazione di Sottocomplesso è fondamentale per strutturare la geometria in modo topologicamente corretto. Infatti, in base alle proprietà dei complessi, vale la seguente regola:

> *Se un certo numero di oggetti g1, g2, g3, ecc... sono tutti sottocomplessi di un unico oggetto complesso G, allora tutte le intersezioni, le sovrapposizioni e le adiacenze tra gli oggetti g1, g2, g3, ecc... sono rappresentate tramite condivisione di primitive geometriche appartenenti a G.*

Un esempio è mostrato in figura 4.1; in figura 4.1.a sono mostrati 3 oggetti geometrici g1,g2 e g3 molto semplici (sono 3 curve composte da una sola curva primitiva retta ciascuna), mentre in figura 4.1.b si ipotizza che tali 3 oggetti costituiscano un unico oggetto complesso G (supercomplesso). Dato che in un complesso le primitive non possono intersecarsi, gli oggetti originali sono stati spezzati in primitive più piccole, in modo che g1, g2 e g3 si intersechino solamente nei punti terminali delle primitive che li costituiscono (i nuovi punti primitivi di intersezione sono p4, p5 e p6).

In figura 4.1 oltre alla rappresentazione geometrica è esemplificata *l'informazione strutturale* associata agli oggetti.

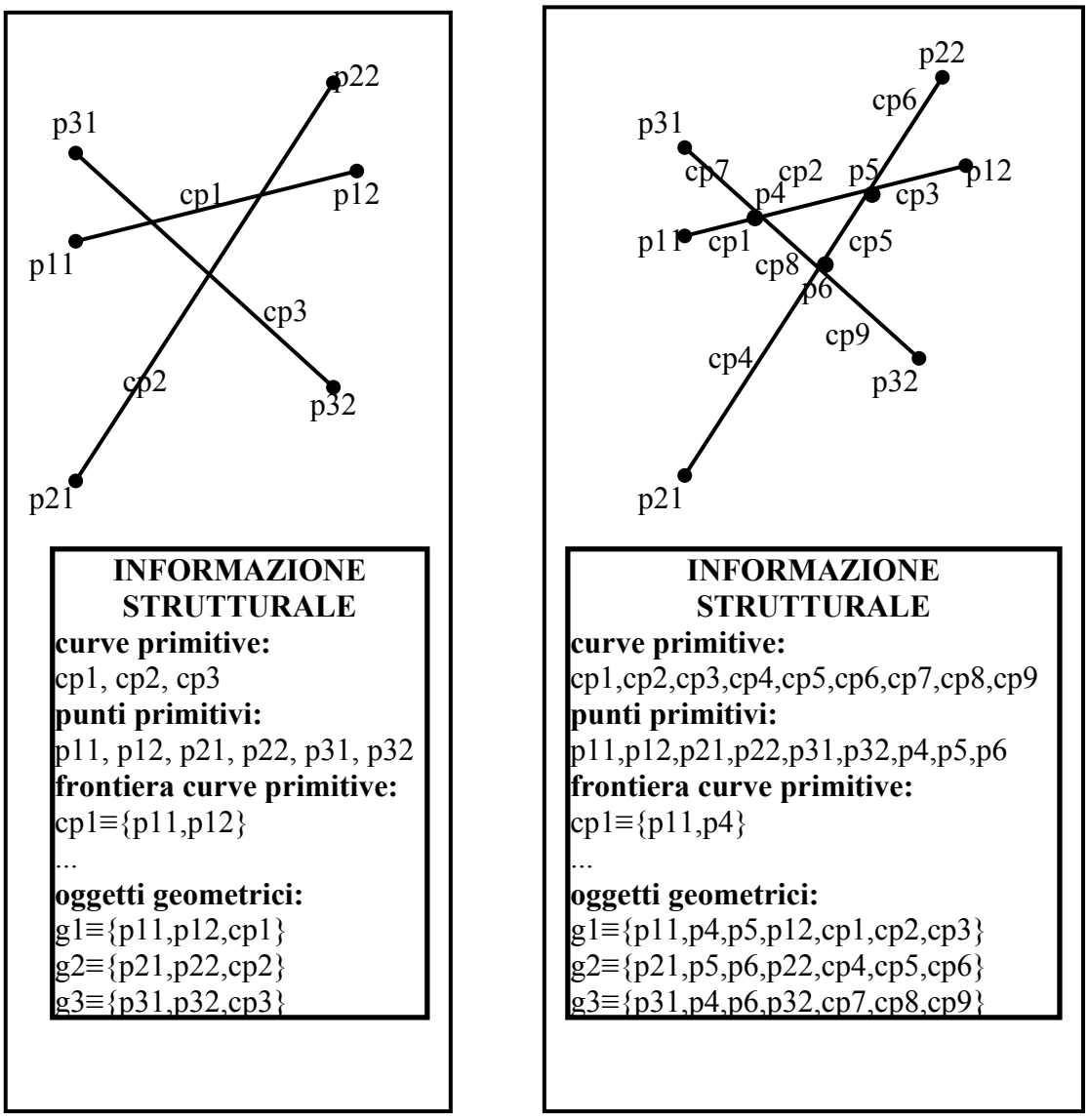

Figura 4.1

In figura 4.1.b l'informazione strutturale permette non solo di ricostruire i 3 oggetti geometrici, ma anche di determinare quali siano le loro relazioni spaziali. Si osservi che i punti p4, p5, p6 sono le uniche primitive condivise tra i 3 oggetti geometrici; quindi, ad esempio, dal fatto che p4 sia condiviso tra g1 e g3 ma non sia un loro punto terminale, e che g1 e g3 non condividono nessuna curva primitiva possiamo dedurre che g1 e g3 hanno solamente una intersezione puntiforme in p4.
Il fatto che questa deduzione sia possibile senza analizzare la geometria degli oggetti significa che non è necessario eseguire un algoritmo di geometria computazionale, ma è sufficiente un algoritmo combinatorio sull'informazione strutturale, cioè un algoritmo

computazionalmente molto più leggero il cui costo non cambierebbe se in figura sostituissimo la semplice geometria rettilinea con curve dalla geometria molto più complessa.
Osserviamo inoltre che aver dovuto spezzare g1, g2 e g3 nel complesso G costituisce anche un esempio dell'importanza che esse siano di tipo composto; se fossero state delle curve primitive, spezzandole avrebbero cambiato tipo.

Considerazioni simili possono essere svolte sulle superfici. Nelle superfici le primitive condivise possono essere sia superfici primitive, sia punti e curve primitive che appartengono alla loro frontiera, ma il tipo di analisi possibile sulla struttura non cambia.

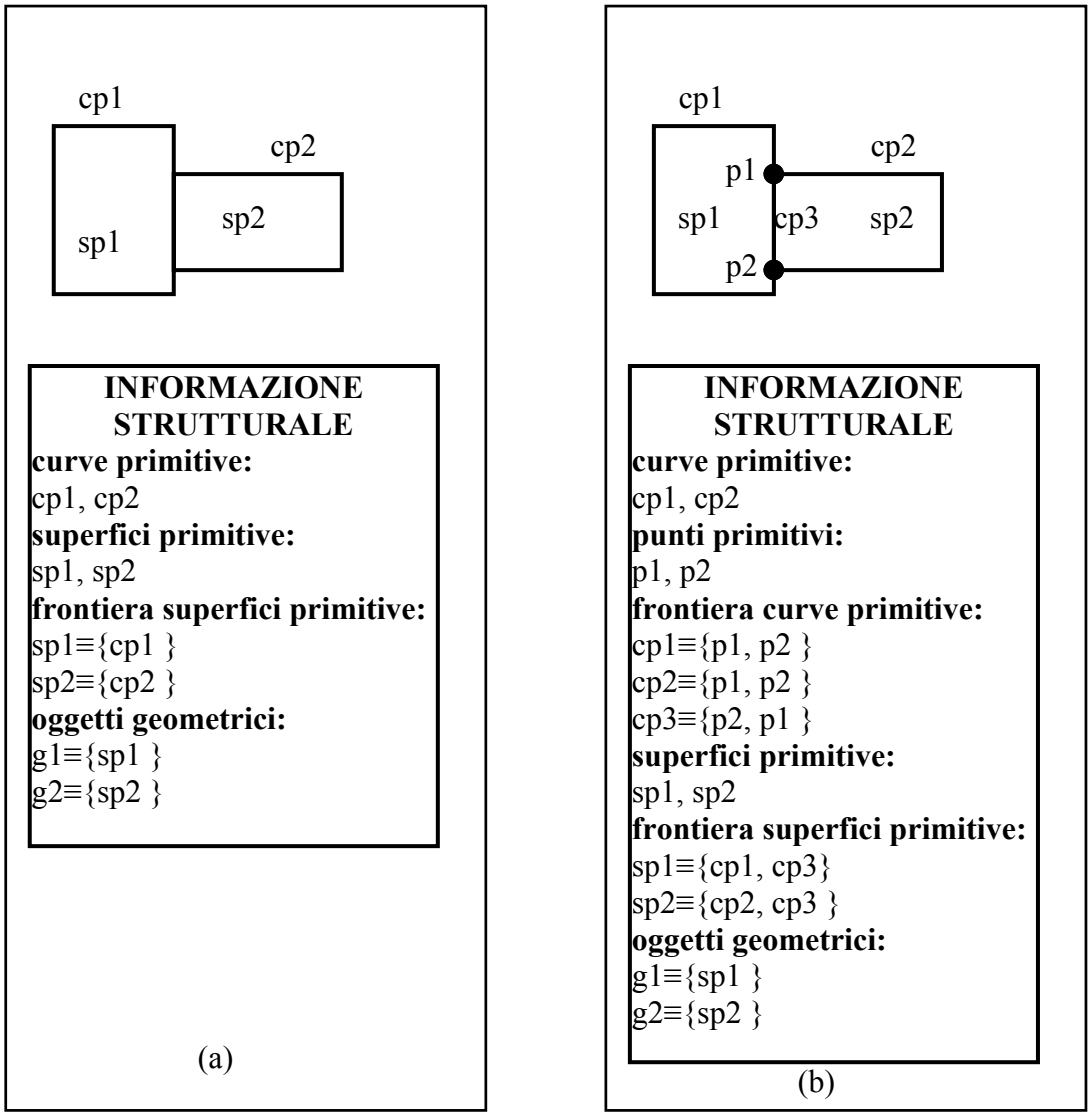

Figura 4.2

L'esempio di figura 4.2 è analogo, con riferimento alle superfici, dell'esempio di figura 4.1 sulle curve. In questo caso due oggetti indipendenti costituiti da un'unica superficie primitiva di figura 4.2.a sono portati in un complesso condiviso in figura 4.2.b e quindi le intersezioni sono individuate e rese primitive indipendenti (la curva primitiva cp3).

In figura 4.2.b si vede che l'informazione strutturale permette di dedurre che i due oggetti geometrici sono adiacenti, perchè condividono solamente una curva primitiva appartenente alle loro frontiere.

4.1.3 Correttezza topologica dei dati geometrici in un complesso

Il fatto che tutte le intersezioni esistenti tra gli oggetti che sono sottocomplessi di uno stesso supercomplesso debbano costituire delle primitive geometriche indipendenti comporta che la geometria deve essere topologicamente corretta. La correttezza topologica della geometria dipende dal processo di aggiornamento; in generale, se tale processo è basato su una rilevazione che non garantisce intrinsecamente tale correttezza, allora la geometria deve essere sottoposta ad un processo di ripulitura (cleaning) che elimini le inconsistenze topologiche; tipici esempi di inconsistenze topologiche sono:

- una linea che dovrebbe raggiungerne un'altra, ma non la raggiunge (undershoot)
- un linea che dovrebbe terminare quando ne raggiunge un'altra, ma la supera (overshoot)
- due superfici che dovrebbero essere adiacenti, condividendo un tratto di frontiera, ma formano invece piccole aree di sovrapposizioni o piccoli buchi tra di loro (sliver poligons)

La correttezza topologica deve essere verificata con riferimento alla realtà rappresentata, perchè non è ad esempio possibile distinguere formalmente il caso in cui si ha un "undershoot" dal caso in cui una linea si ferma realmente prima di raggiungerne un'altra (si pensi ad esempio ad una strada in rappresentazione lineare che non si inserisce in un'altra a causa di una sottile barriera).

In ogni caso, in GeoUML vale il seguente principio:

> *All'interno di uno stesso complesso deve essere garantita la correttezza topologica dei dati tramite la corretta individuazione delle primitive condivise.*

Con riferimento all'esempio di figura 4.2 possiamo dire che la vera trasformazione è più sostanziale di quella mostrata, perchè in generale la situazione di partenza sarà simile a quella di figura 4.3.a e quindi la creazione di un complesso avrà l'effetto anche di ripulire il dato dagli sliver poligons, come mostrato in figura 4.3.b (in figura 4.3 l'informazione strutturale sarebbe identica a quella di figura 4.2 e perciò non è riportata).

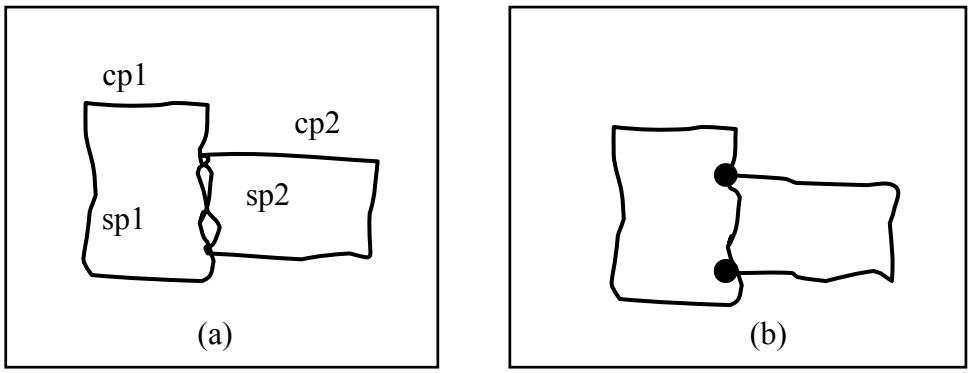

Figura 4.3

4.1.4 Implementazione di riferimento

Il modello GeoUML si pone a livello logico/concettuale, non a livello fisico, quindi possono esistere molte rappresentazioni fisiche che lo implementano. E' compito del sistema che adotta una certa implementazione garantire che l'interfaccia logica si comporti secondo quanto specificato dal modello. Questo processo è detto *"virtualizzazione"* ed è alla base di tutti i sistemi informatici complessi. In genere la virtualizzazione è tanto più complessa e costosa quanto più la rappresentazione fisica differisce dal modello logico.
Dato che il modello logico basato sui complessi è definito facendo riferimento a una rappresentazione logica concreta (primitive, insiemi, ecc...) e non in maniera puramente astratta, è possibile e sensato definire una *"implementazione di riferimento"* per gli oggetti che sono sottocomplessi dello stesso supercomplesso nel modo seguente:

- viene rappresentata la geometria di ogni singola primitiva geometrica
- viene rappresentata l'informazione strutturale

In questo modo una primitiva condivisa da due oggetti geometrici viene rappresentata una sola volta e nei due oggetti esistono due riferimenti a quella stessa primitiva, come esemplificato nelle figura 4.1.b. e 4.2.b (in terminologia Object-Oriented abbiamo una rappresentazione per identità, perchè una primitiva condivisa, ad esempio p4 in figura 4.1.b, viene riconosciuta in quanto di tratta dello stesso oggetto, ovvero è identificata dallo stesso identificatore di oggetto - OID).
Altre implementazioni potrebbero invece rappresentare ogni oggetto geometrico separatamente, e determinare la condivisione attraverso un meccanismo di uguaglianza delle primitive, purchè i dati siano stati ripuliti e quindi la loro correttezza topologica sia garantita.
Ad esempio, un sistema potrebbe adottare una rappresentazione di oggetti indipendenti come quella di figura 4.1.a, a condizione di avere eseguito una operazione di "ripulitura" dell'informazione capace di garantirci che esistano i punti di intersezione tra le 3 curve e siano rappresentabili alla precisione adottata. In pratica ciò richiede generalmente che il punto di intersezione diventi un "vertice" nella rappresentazione geometrica interna delle curve. Nell'esempio di figura 4.1 il punto p4 di intersezione tra g1 e g3 verrebbe determinato non in base al meccanismo di identità (di una primitiva), ma di uguaglianza delle coordinate di un punto interno di g1 e di un punto interno di g3.

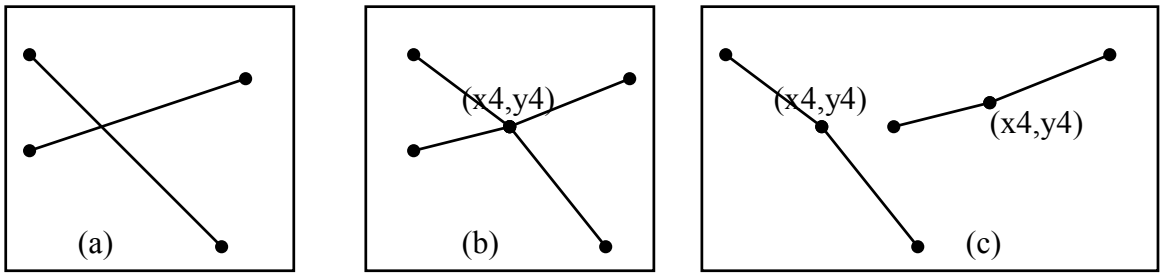

Figura 4.4

In figura 4.4 si mostra questo procedimento: in figura (a) esistono due oggetti indipendenti, in figura (b) viene determinata un'intersezione rappresentabile nel punto di coordinate (x4,y4), ma, a causa della precisione finita, ciò comporta una leggera modificazione dei due oggetti, infine in figura (c) i due oggetti sono resi indipendenti. La condivisione di un punto è ora riconoscibile grazie all'uguaglianza dei valori delle coordinate, senza condivisione di un punto primitivo.

Ovviamente queste considerazioni sull'implementazione sono solamente degli esempi; dal punto di vista del GeoUML si ribadisce che:

- qualsiasi rappresentazione fisica che permetta di produrre il modello logico è ammissibile;
- esiste una implementazione di riferimento.

Per quanto riguarda i formati di trasferimento si ritiene che una implementazione di riferimento del GeoUML possa essere definita tramite lo standard GML3 dell'OGC; questo aspetto esula però dagli scopi di questo documento.

4.2 Sottocomplessi e classi GeoUML

Nel paragrafo precedente abbiamo analizzato la relazione di sottocomplesso tra due oggetti geometrici e le sue implicazioni; tuttavia, quando definiamo le classi in GeoUML non operiamo a livello dei singoli oggetti ma a livello di intere classi. Per questo motivo il linguaggio GeoUML permette di definire dei vincoli che determinano quali classi possono o devono avere oggetti che sono sottocomplessi dello stesso supercomplesso e quindi, in base a quanto detto al paragrafo precedente, forniscono delle indicazioni per una strutturazione topologicamente corretta della geometria.

4.2.1 Il vincolo di appartenenza tra classi

Definiamo la proprietà di appartenenza degli oggetti di una classe a quelli di un'altra classe nel modo seguente:

> *Date due classi CLX e CLY dotate di due attributi geometrici X e Y di tipo complesso diciamo che*
>
> *CLX.X appartiene CLY.Y*
>
> *se per ogni oggetto geometrico gx di X esiste un corrispondente oggetto geometrico gy di Y tale che gx è un sottocomplesso di gy.*

In GeoUML è possibile definire un vincolo che stabilisce che CLX.X appartiene a CLY.Y, nel modo seguente:

> *classe* CLX
>
> *attributi*: X: GU_....;
>
> *...altri attributi ...*
>
> *classe* CLY
>
> *attributi*: Y: GU_....;
>
> *...altri attributi ...*
>
> *struttura*: CLX.X *appartiene* CLY.Y

I vincoli di struttura possono essere scritti in qualsiasi punto dello schema, perchè contengono tutti gli elementi necessari per individuare le classi e gli attributi coinvolti; spesso però i vincoli di struttura sono scritti subito dopo la classe vincolata.

La rappresentazione grafica di questa proprietà è mostrata nel diagramma 4.1

E' importante osservare che la definizione del vincolo di appartenenza nell'esempio precedente NON impone che tutti gli oggetti geometrici della classe CLX appartengano ad uno stesso complesso, ma permette a diversi oggetti geometrici gx1, gx2, ecc... dell'attributo CLX.X di essere sottocomplessi di uno stesso oggetto gy1 dell'attributo CLY.Y. Sono possibili tutte le situazioni intermedie tra i seguenti due estremi:

- ogni oggetto gx_i è un sottocomplesso di un diverso oggetto gy_j
- tutti gli oggetti di CLX.X sono sottocomplessi dello stesso oggetto gy

Inoltre, si noti che solamente la classe CLX è vincolata; nella classe CLY possono esistere oggetti dell'attributo Y che non sono supercomplessi di alcun oggetto dell'attributo CLX.X.

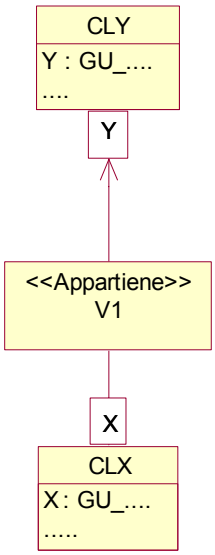

Diagramma 4.1

Da un punto di vista formale l'appartenenza così definita è un vincolo, in quanto si limita a stabilire che gli oggetti della classe devono soddisfare una certa condizione per poter esistere nel database, ma in base a quanto visto sopra relativamente all'informazione strutturale nella relazione tra sottocomplessi e supercomplessi, questo vincolo ha delle evidenti conseguenze sulla strutturazione dell'informazione a livello del modello logico/concettuale (e, se si tiene conto delle considerazioni relative all'implementazione di riferimento, può avere anche conseguenze sulla struttura implementativa). Per questo motivo questo tipo di proprietà e le altre presentate nei prossimi paragrafi vengono introdotte in GeoUML dalla parola chiave "struttura".

4.2.2 Il vincolo di appartenenza disgiunta e quasi-disgiunta

In molti casi, oltre a richiedere che gli oggetti geometrici di una classe siano sottocomplessi degli oggetti geometrici di un'altra classe, si richiede anche che gli oggetti geometrici che sono sottocomplessi dello stesso oggetto abbiano insiemi di primitive interne (cioè primitive che non appartengono alla frontiera) disgiunti:

> *Due oggetti complessi della stessa dimensione sono disgiunti se i sottoinsiemi costituiti dalle loro primitive interne (cioè tutte le primitive che li costituiscono eccetto quelle di frontiera) sono disgiunti*

In altri termini, due oggetti complessi sono disgiunti se non condividono primitive *interne a ambedue* gli oggetti. In particolare, questa definizione considera disgiunte due linee complesse che si toccano in qualche estremo oppure una delle quali è incidente all'altra, ma non considera disgiunte due linee complesse che si intersecano in qualche punto primitivo interno.

In GeoUML il vincolo di appartenenza definito precedentemente può essere modificato richiedendo che *tutti gli oggetti che sono sottocomplessi di uno stesso sottocomplesso abbiano insiemi di primitive interne disgiunti*; a questo scopo si utilizza la parola chiave *dj-appartiene* invece del semplice *appartiene*; ad esempio:

classe CLX
attributi: X: GU_....;
...altri attributi ...
classe CLY
attributi: Y: GU_....;
...altri attributi ...

struttura: CLX.X *dj-appartiene* CLY.Y

Una proprietà più debole della completa disgiunzione delle primitive interne è la seguente, che chiameremo "quasi-disgiunzione":

Due oggetti complessi della stessa dimensione sono quasi-disgiunti se i sottoinsiemi di primitive di dimensione massima sono disgiunti.

Ad esempio, i 3 oggetti geometrici di figura 4.1.b sono quasi-disgiunti, perchè sono di dimensione 1 e condividono solamente punti primitivi (dimensione 0), ma non curve primitive (dimensione 1). Si osservi che tali insiemi non sono disgiunti, perchè condividono punti interni.

In GeoUML il vincolo di appartenenza può essere modificato richiedendo che *tutti gli oggetti che sono sottocomplessi di uno stesso sottocomplesso siano quasi-disgiunti tra loro*; a questo scopo si utilizza la parola chiave *qdj-appartiene* invece del semplice *appartiene*; ad esempio:

classe CLX
attributi: X: GU_....;
...altri attributi ...
classe CLY
attributi: Y: GU_....;
...altri attributi ...

struttura: CLX.X qdj-appartiene CLY.Y

La rappresentazione grafica di queste proprietà è mostrata nel diagramma 4.2

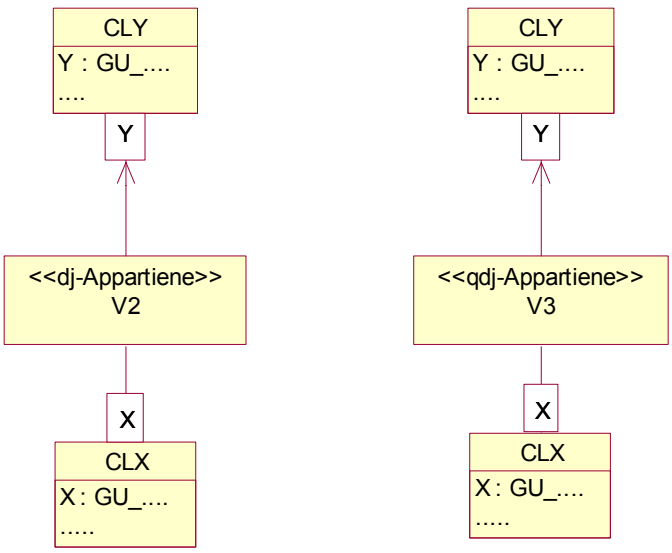

Diagramma 4.2

In pratica, nel caso di oggetti di tipo linea composta o complessa, l'appartenenza disgiunta permette che le linee si tocchino solo nei punti di frontiera, l'appartenenza quasi-disgiunta permette alle linee di intersecarsi in qualsiasi punto primitivo, ma non permette di condividere una curva primitiva.
Nel caso di oggetti di tipo superficie composta o complessa, l'appartenenza disgiunta permette che le superfici si tocchino solo nelle curve o nei punti di frontiera, cioè permette l'adiacenza; l'appartenenza quasi-disgiunta in teoria permetterebbe alle superfici di intersecarsi in qualsiasi curva primitiva, ma in pratica, essendo le superfici considerate nell'attuale versione di GeoUML solo planari, questo vincolo coincide con quello di appartenenza disgiunta (nello spazio 3D invece due superfici potrebbero intersecarsi lungo una linea, e quindi la distinzione avrebbe senso).

4.2.3 Il vincolo di composizione
In taluni casi si vuole imporre che per ogni oggetto geometrico gy di un attributo complesso Y di una classe CLY esistano uno o più oggetti gx1,gx2,...gxN di un attributo complesso X di una classe CLX tali che le primitive di gy siano contenute nell'unione delle primitive di gx1, gx2, ... gxN.
In sostanza si vuole che tutti gli oggetti dell'attributo complesso CLY.Y siano costituiti da primitive appartenenti agli oggetti di un attributo complesso di un'altra classe CLX.X.
In GeoUML questo vincolo viene espresso nella sezione di struttura della classe che deve essere composta dall'altra, CLY nell'esempio considerato, nel modo seguente:

classe CLY
attributi: Y: GU_....;
...altri attributi ...
classe CLY
attributi: Y: GU_....;
...altri attributi ...

struttura: CLY.Y *compostoDa* CLX.X

La rappresentazione grafica di questa proprietà è mostrata nel diagramma 4.3

Si noti che il vincolo di composizione impone a tutti gli oggetti di CLY.Y di essere composti da primitive di CLX.X, ma non impone che tutte le primitive di CLX.X appartengano a CLY.Y

4.2.4 Il vincolo di partizione
In molti casi si vogliono combinare gli effetti del vincolo di appartenenza disgiunta o quasi disgiunta e quello di composizione; abbiamo infatti visto che:

- nel caso "*CLX.X appartiene CLY.Y*" solamente la classe CLX è vincolata; nella classe CLY possono esistere oggetti dell'attributo Y che non sono supercomplessi di alcun oggetto dell'attributo CLX.X.
- nel caso "*CLY.Y compostoDa CLX.X"* il vincolo di composizione impone a tutti gli oggetti di CLY.Y di essere composti da primitive di CLX.X, ma non impone che tutte le primitive di CLX.X appartengano a CLY.Y

Combinando i due vincoli (nella versione disgiunta o quasi-disgiunta) otteniamo il vincolo di partizione "CLY.Y partizionato da CLX.X", che impone le seguenti condizioni:

- per ogni oggetto geometrico di CLX.X deve esistere un oggetto geometrico di CLY.Y di cui è un sottocomplesso,
- l'insieme delle primitive geometriche di ogni oggetto geometrico di CLY.Y deve coincidere con l'unione delle primitive degli oggetti geometrici di CLX.X che sono suoi sottocomplessi,
- gli oggetti di CLX.X che sono sottocomplessi dello stesso oggetto geometrico di CLY sono disgiunti o quasi-disgiunti tra loro.

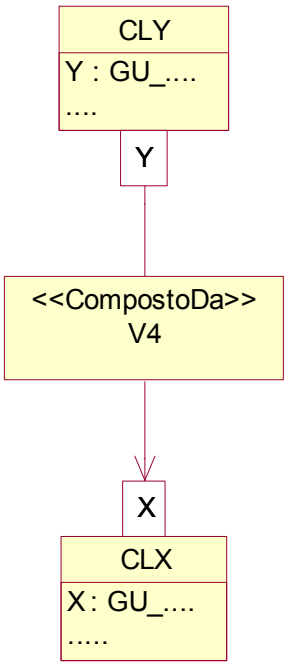

Diagramma 4.3

La definizione in GeoUML del vincolo di partizione è espressa utilizzando le parole chiave *partizionato* se le primitive devono essere disgiunte e *q-partizionato* se devono essere quasi disgiunte nel modo seguente:

classe CLY
    attributi: Y: GU_....;
        ...altri attributi ...
classe CLY
    attributi: Y: GU_....;
        ...altri attributi ...

struttura: CLY.Y *partizionato* CLX.X
oppure
struttura: CLY.Y *q-partizionato* CLX.X

La rappresentazione grafica di questa proprietà è mostrata nel diagramma 4.4

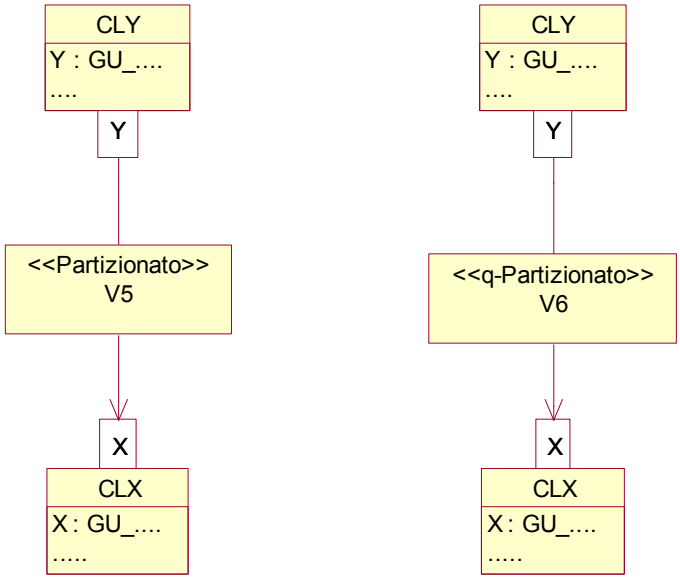

Diagramma 4.4

4.3 Varianti dei vincoli strutturali

In GeoUML è possibile utilizzare delle varianti dei vincoli strutturali definiti in precedenza che permettono di definire alcune importanti proprietà dell'informazione.

4.4.1Vincoli strutturali collegati ad associazioni – Associazioni spaziali

I vincoli strutturali talvolta non devono semplicemente fare riferimento a 2 classi indipendenti, ma devono appoggiarsi ad una associazione.
Ad esempio, si considerino due classi relative alle Regioni ed alle Province: ovviamente queste due classi sono legate da un'associazione Regioni-Province che è indipendente dalla rappresentazione geometrica dei loro territori. Se però realizziamo tale rappresentazione geometrica in forma di poligoni, allora è evidente che esiste un vincolo di appartenenza di ogni provincia ad una regione, che potremmo definire con il vincolo *appartiene* visto in precedenza; tuttavia, è anche evidente che una provincia P ha una rappresentazione poligonale che non solo deve appartenere (essere contenuta) nel poligono di una regione R, ma anche che tale regione R deve essere proprio quella legata a P nell'associazione Regioni-Province.
Quanto descritto in questo esempio può essere rappresentato in GeoUML nel modo seguente:

classe Regioni
attributi: estensione: GU_CPSurface2D;
nome: *String*;
ruoli: ProvinceDiRegione [0..*]: Province
inverso RegioneDiProvincia [1..1];

classe Province
attributi: estensione: GU_CPSurface2D;
nome: *String*;
ruoli: RegioneDiProvincia [1..1]: Regioni
inverso ProvinceDiRegione [0..*];

struttura: Province.estensione *appartiene*
Provincia.RegioneDiProvincia.estensione

(vedi anche diagramma 4.5)

L'elemento nuovo è costituito dal riferimento, all'interno del vincolo di struttura, al ruolo RegioneDiProvincia per indicare che l'appartenenza fa riferimento alla regione collegata dall'associazione.
E' possibile aggiungere dopo il riferimento al ruolo anche un richiamo al tipo di oggetto restituito, se lo si ritiene utile per leggibilità; il vincolo assume la forma seguente:

struttura: Province.estensione *appartiene*
Provincia.RegioneDiProvincia::Regioni.estensione

La notazione ::Regione serve a richiamare il fatto che l'oggetto restituito dall'associazione è di tipo Regione, ma non modifica il significato del vincolo scritto precedentemente.
Lo stesso meccanismo può essere utilizzato sostituendo al vincolo di appartenenza gli altri vincoli strutturali; in effetti, nel caso delle regioni e province il vincolo più significativo sarebbe quello che dichiara che una regione è *partizionata* nelle sue province.

Il legame forte esistente tra una associazione e un vincolo strutturale può essere interpretato anche nel modo seguente: partendo dalle geometrie possiamo dire che l'appartenenza spaziale di una provincia a una regione determina il valore della sua associazione. Per questo motivo nei diagrammi Rose l'associazione e il vincolo sono stati fusi in una "associazione spaziale" invece di essere evidenziati separatamente.

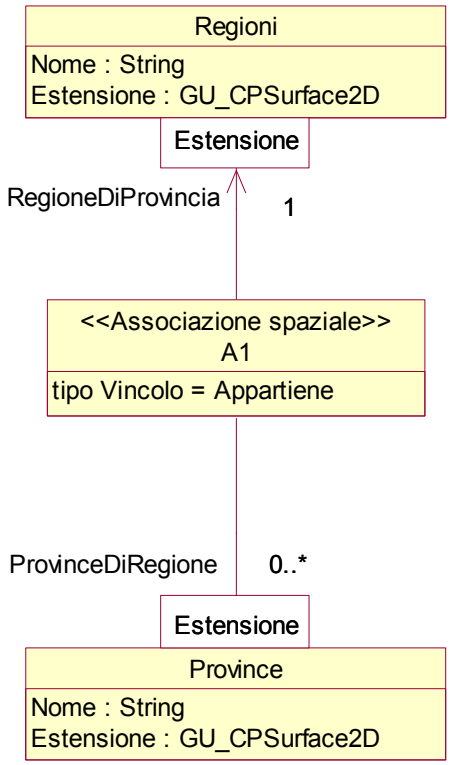

Diagramma 4.5

4.4.2 La funzione boundary nell'espressione di vincoli strutturali

Nell'espressione dei vincoli strutturali è possibile utilizzare la funzione boundary; ad esempio, se vogliamo imporre il seguente vincolo:

> i punti terminali del percorso lineare di un oggetto della classe ElementoStradale devono essere le posizioni puntiformi di due oggetti della classe GiunzioneStradale

possiamo in GeoUML scrivere le seguenti definizioni di classi con un vincolo strutturale:

classe ElementoStradale
    attributi: percorso: GU_CPCurve3D;
        ...
classe GiunzioneStradale
    attributi: posizione: GU_Point3D;
        ...
struttura: ElementoStradale.percorso.*bnd* *compostoDa*
        GiunzioneStradale. posizione

(vedi anche diagramma 4.6)

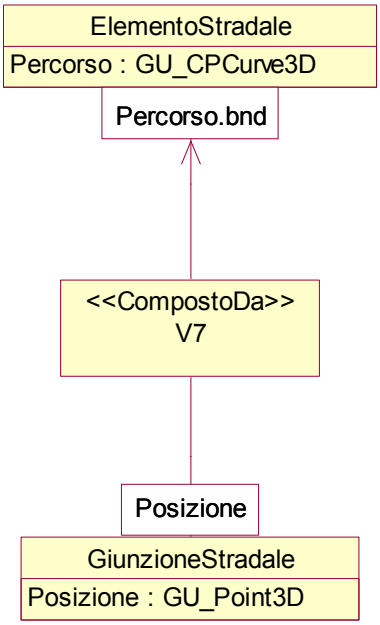

Diagramma 4.6

4.4.3 Vincoli con riferimento all'unione di più classi

In alcuni casi si vuole che un vincolo *compostoDa* oppure *partizionato* facciano riferimento a diversi attributi geometrici di diverse classi. Le diverse classi con i rispettivi attributi devono in questo caso essere elencate tra parentesi.

Ad esempio, si vuole definire un "percorso misto" come costituito da tratti stradali e tratti ferroviari, come nell'esempio seguente.

classe Strada
    attributi: percorso: GU_CPCurve
classe Ferrovia
    attributi: percorso: GU_CPCurve
classe PercorsoMisto
    attributi: percorso: GU_CPCurve

struttura PercorsoMisto.Percorso *CompostoDa*
        (Strada.percorso, Ferrovia.percorso)

(vedi anche diagramma 4.7)

Il significato di questa dichiarazione è che un oggetto di tipo "percorso misto" possiede un attributo geometrico costituito da curve primitive che appartengono agli attributi geometrici delle strade o delle ferrovie.

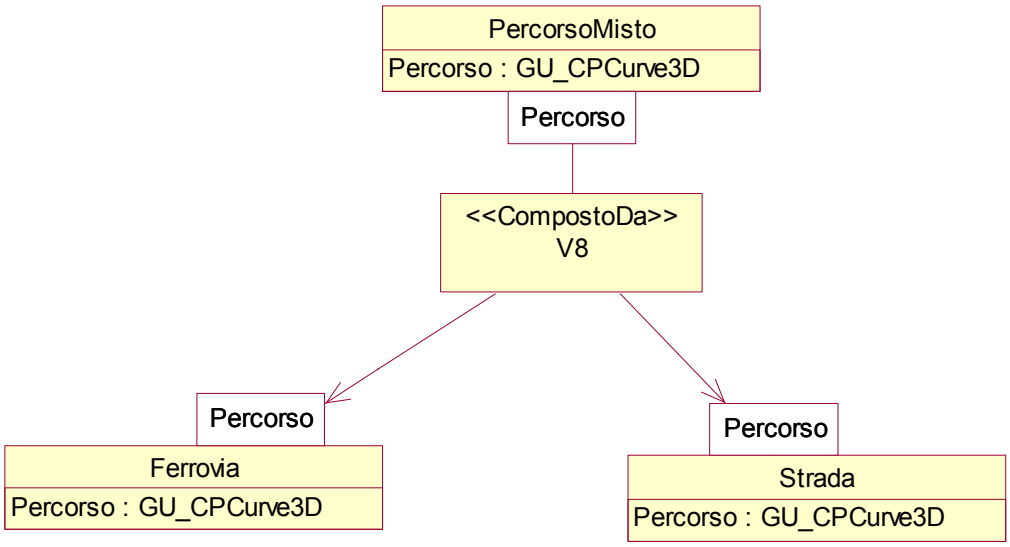

Diagramma 4.7

4.4 Gli strati topologici

Uno strato topologico è una classe dotata di due caratteristiche particolari:

- nella base dati può esistere una sola istanza di oggetto di questa classe (classe monoistanza)
- questa classe ha un unico attributo geometrico monovalore che può essere di tipo GU_Complex (strato generico) oppure GU_CXCurve (strato lineare) oppure GU_CXSurface (strato areale)

Gli strati vengono utilizzati normalmente per imporre condizioni più stringenti di condivisione di un complesso da parte di oggetti geometrici; abbiamo visto infatti che la definizione del vincolo di appartenenza NON impone che tutti gli oggetti geometrici della classe vincolata appartengano ad uno stesso complesso.
Se però quando scriviamo un vincolo del tipo

CLX.X appartiene CLY.Y

la classe Y è uno strato, allora l'esistenza di un'unica istanza di oggetto in questa classe e il fatto che il suo attributo sia monovalore impone che tutti gli oggetti di CLX.X siano sottocomplessi dello stesso oggetto.
In questo modo risultano esplicitate a livello di primitive geometriche le relazioni spaziali esistenti tra tutti gli oggetti che costituiscono l'attributo CLX.X.
Dato che i vincoli di appartenenza quasi-disgiunta e di partizione hanno le stesse caratteristiche del vincolo di appartenenza semplice, questa considerazione si applica anche a questi vincoli.
Negli strati non vengono definiti altri attributi oltre a quello geometrico; lo strato è solamente un oggetto "di strutturazione" che definisce che le primitive geometriche di altri oggetti devono appartenenere allo stesso complesso.
Inoltre, se lo strato ha un vincolo CompostoDa oppure Partizionato, come spesso accade, le primitive che costituiscono lo strato devono esistere comunque come primitive di altri oggetti, quindi la definizione di uno strato non causa la creazione di alcuna primitiva aggiuntiva rispetto a quelle che costituiscono gli oggetti delle classi.

Da un punto di vista sintattico la dichiarazione di uno strato è simile a quella di una classe, ma la parola chiave *classe* è sostituita dalla parola chiave *strato* e il nome dell'unico attributo geometrico è standard: "*geometria*".

Esempio di dichiarazione di una strato lineare in 3D:

strato ReteA *geometria*: GU_CXCurve3D

Esempio di dichiarazione di una strato areale:

strato CoperturaA *geometria*: GU_CXSurface2D

La rappresentazione grafica di queste definizioni è mostrata nel diagramma 4.8

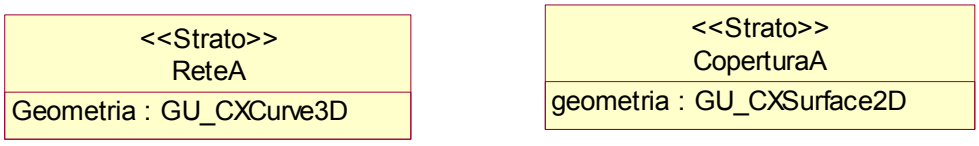

Diagramma 4.8

I seguenti esempi illustrano alcuni possibili modi di utilizzazione degli strati.
Strutturazione di strati topologici lineari
Si supponga di voler definire uno strato topologico costituito da elementi lineari (ad esempio, una rete stradale costituita da elementi stradali o una rete ferroviaria costituita da elementi ferroviari). Le due classi fondamentali e la relazione che le unisce possono essere dichiarate nel modo seguente:

strato: ReteA *geometria*: GU_CXCurve
classe: ElementoA
attributi: tracciato: GU_CPCurve
.....
struttura: ReteA. *geometria partizionato* ElementoA.tracciato

(vedi anche diagramma 4.9)

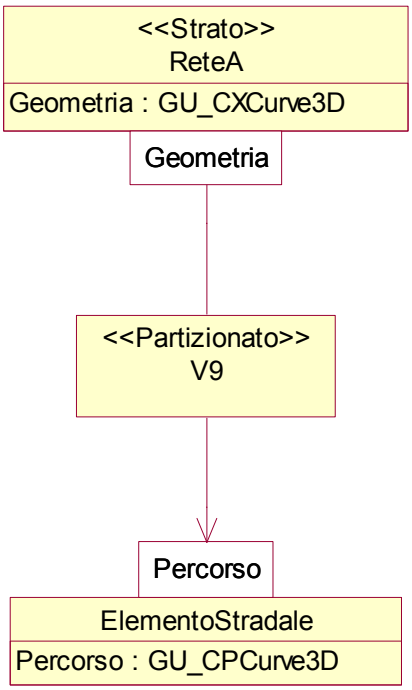

Diagramma 4.9

Si supponga ora di voler aggiungere degli oggetti la cui rappresentazione puntiforme deve appartenere allo strato lineare (ad esempio, delle giunzioni stradali o giunzioni ferroviarie). La classe di questi oggetti e la struttura risultante aggiunte alle precedenti dichiarazioni producono il seguente schema:

strato: ReteA *geometria*: GU_CXCurve
classe: ElementoA
attributi: tracciato: GU_CPCurve
.....
classe: GiunzioneA

attributi: Posizione: GU_Point
...
struttura: ReteA. *geometria* *partizionato* ElementoA.tracciato
struttura : GiunzioneA.Posizione *appartiene* ReteA. *geometria*

Questa dichiarazione esprime solamente il fatto che gli oggetti di tipo GiunzioneA che esistono devono essere dei nodi di ReteA.
Se si vuole esprimere anche il fatto che la frontiera di ogni elemento deve essere composta di giunzioni è necessario aggiungere un vincolo alla classe ElementoA nel modo seguente:

strato: ReteA *geometria*: GU_CXCurve
classe: ElementoA
attributi: tracciato: GU_CPCurve
.....
classe: GiunzioneA
attributi: Posizione: GU_Point
...
struttura: ReteA. *geometria* *partizionato* ElementoA.tracciato
struttura : GiunzioneA.Posizione *appartiene* ReteA. *geometria*
struttura : ElementoA.*bnd* *CompostoDa* GiunzioneA.posizione

E' importante osservare che nello strato esistono nodi che non sono giunzioni, perchè gli elementi sono curve composte e solo i loro nodi terminali sono giunzioni, come mostrato in figura 4.5

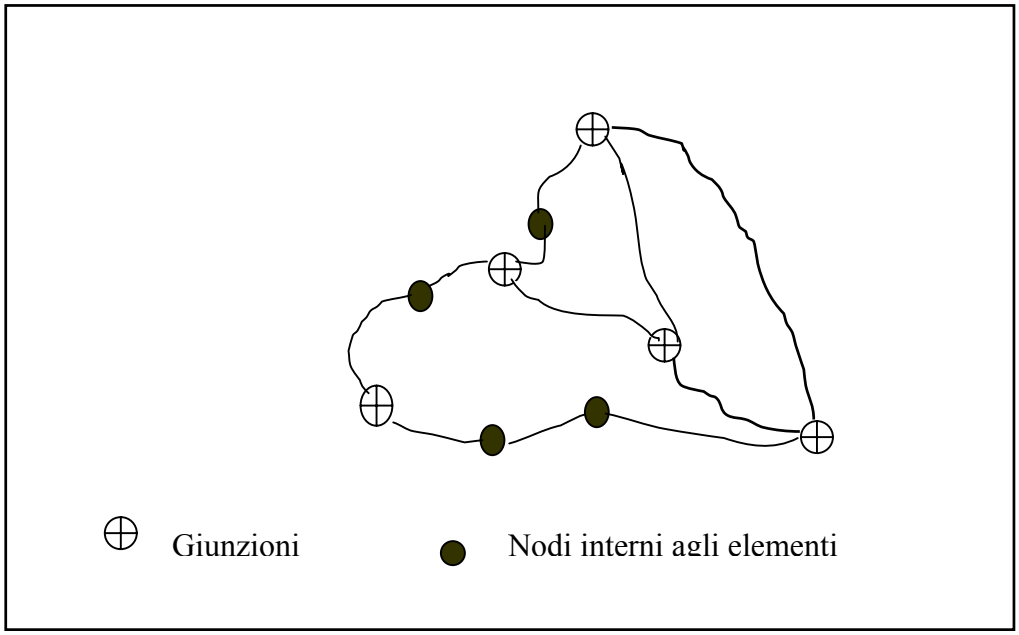

Figura 4.5

Ipotizziamo ora di definire nello stesso modo in cui abbiamo definito ReteA anche un altro strato lineare ReteB costituito da ElementiB e GiunzioniB. Questo strato risulterà totalmente indipendente dal precedente. Se vogliamo costruire le relazioni topologiche tra i 2 strati, possiamo definire uno strato ulteriore ReteAB che contiene sia ReteA che ReteB (figura 4.6) con la seguente dichiarazione

strato: ReteA *geometria*: GU_CXCurve
strato: ReteB *geometria*: GU_CXCurve
struttura: ReteAB. *geometria* *Partizionato*
(ReteA.geometria, ReteB.geometria)

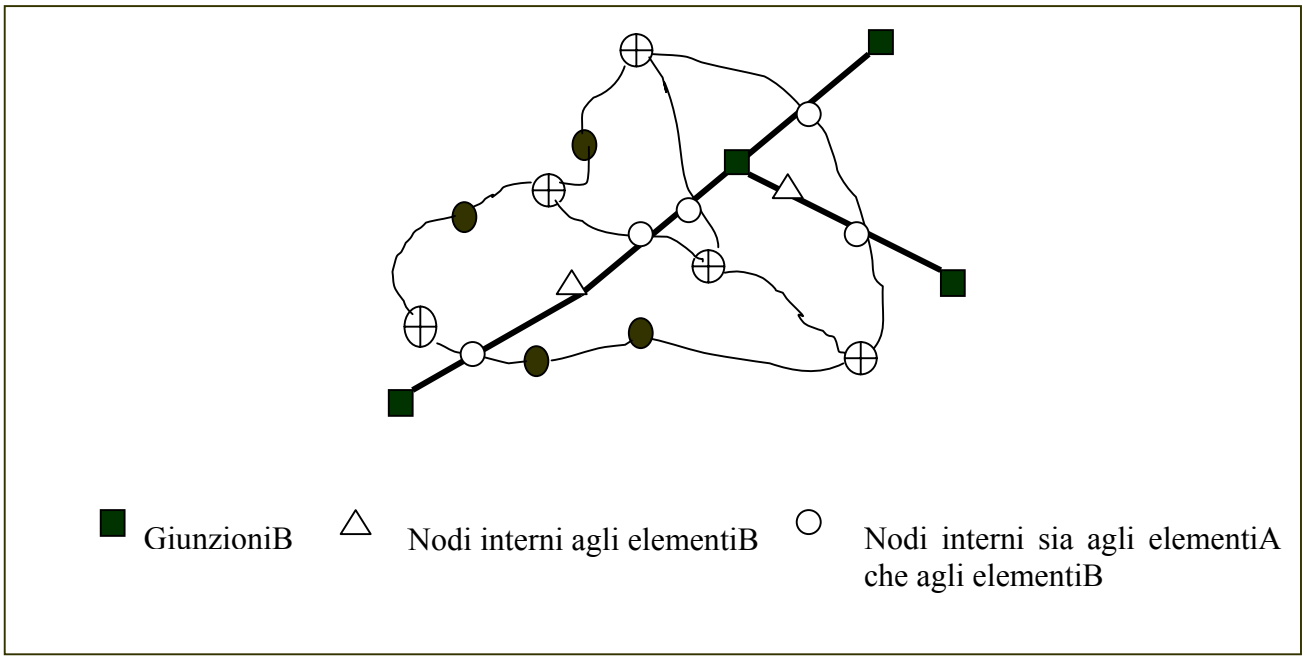

Figura 4.6

4.5 Attributi a tratti e a sottoaree

Gli attributi a tratti e a sottoaree sono attributi dotati di un loro dominio il cui valore non è associato semplicemente a un oggetto applicativo ma è una funzione dei punti che costituiscono un attributo geometrico

Ad esempio, se consideriamo l'attributo "sede" di una strada con un attributo lineare "percorso", il valore di tale attributo non dipende solamente dalla strada considerata ma anche dal punto sul percorso preso in considerazione. I punti del percorso possono essere raggruppati in zone dotate di valore omogeneo della "sede" dette ***tratti***.

Se consideriamo lo stesso esempio immaginando una rappresentazione areale della strada, i punti dotati di valore omogeneo della "sede" costituiscono delle ***sottoaree***.

Gli attributi a tratti possono essere definiti in GeoUML a due diversi livelli di astrazione: un livello più astratto, che non entra nel merito di alcune scelte di rappresentazione, e un livello meno astratto, che implica alcune scelte più definite. I due livelli sono consistenti, e quindi una definizione iniziale più astratta può essere successivamente trasformata in una più concreta.

Al livello meno astratto gli attributi a tratti si distinguono in ***attributi a tratti strutturali***, nei quali i tratti sono rappresentati da elementi geometrici, e ***attributi a tratti dinamici***, nei quali i tratti sono individuati tramite una ***coordinata curvilinea***, cioè tramite un meccanismo di misura lungo la curva.

anche se la rappresentazione astratta può essere sostituita poi da una rappresentazione più concreta, è necessario tenere presente che, se si utilizza la formulazione più astratta non è possibile esprimere alcuni tipi di proprietà aggiuntive che richiedono di fare riferimento alle primitive geometriche che costituiscono i tratti strutturali. D'altra parte, è evidente che, se le caratteristiche dell'informazione richiedono questo tipo di definizione, allora la scelta dei tratti strutturali deve essere fatta dall'inizio e non può essere rinviata.

Per gli attributi a sottoaree invece esiste un unico livello di definizione, in quanto le sottoaree sono sempre rapresentate tramite elementi geometrici, cioè in forma strutturale.

Tratteremo in questo capitolo prima gli attributi a tratti a livello astratto, poi gli attributi a tratti strutturali, poi quelli a sottoaree e infine gli attributi a tratti dinamici.

4.5.1 Attributi a tratti (definizione astratta)

Per definire un attributo a tratti è sufficiente aggiungere, dopo la definizione di un attributo normale, la dizione *aTratti su* (oppure solamente *aTratti*) seguita dal nome dell'attributo geometrico sul quale si vogliono determinare i tratti.

Ad esempio, facendo riferimento all'esempio precedente, la seguente definizione associa alle strade un attributo di sede che è a tratti sul percorso.

classe strada
    attributi: percorso: GU_CPCurve;
        ...altri attributi ...
        sede: TIPO_SEDE *aTratti su* percorso;
dominio TIPO_SEDE (....)

(vedi anche diagramma 4.10)

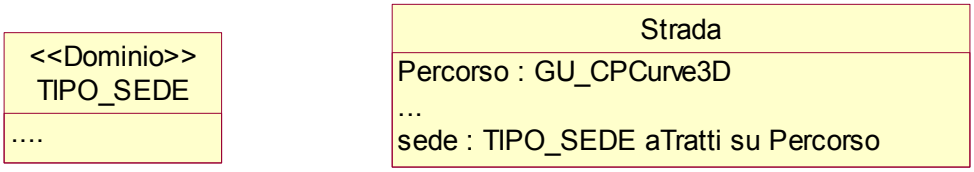

Diagramma 4.10

In alcuni casi l'attributo non è definito su interi tratti lineari ma solamente su singoli punti; in questo caso usiamo la parola chiave *aPunti su* , come nel seguente esempio:

classe strada
attributi: percorso: GU_CPCurve;
...altri attributi ...
evento: TIPO_EVENTO *aPunti su* percorso;
dominio TIPO_EVENTO(...)

4.5.2 Attributi a tratti strutturali

Riprendiamo lo stesso esempio, ma a un livello meno astratto, al quale indichiamo il tipo di implementazione desiderata dei tratti.

Un "tratto" è nella rappresentazione strutturale un sottoinsieme delle primitive geometriche che costituiscono il percorso.

Definizione di Tratto:

> *Dato un valore di un attributo geometrico di tipo GU_CXCurve o GU_CNCurve o GU_CPCurve costituito da un insieme C di curve primitive, un Tratto di questo valore è un sottoinsieme di C al quale viene associato un valore di un attributo A (o una combinazione di valori di un insieme di attributi).*

Nota: l'attributo A può essere multivalore; in tal caso ad ogni sottoarea è associato un preciso insieme di valori.

Si osservi che questa definizione implica che un tratto possieda un valore preciso di attributo, ma non implica che tutte le curve primitive che hanno lo stesso valore di attributo appartengano allo stesso tratto – possono esistere più tratti con lo stesso valore di attributo. Pertanto, con riferimento all'esempio precedente, se indichiamo con P il percorso di una strada (P deve essere di tipo complesso e quindi è un insieme di curve primitive), sono possibili tutte le configurazioni comprese tra i due seguenti estremi:

- tutte le curve primitive di P dotate di uno stesso valore dell'attributo sede costituiscono un unico tratto;
- ogni primitiva di P costituisce un diverso tratto.

La scelta tra queste soluzioni è implementativa e non è quindi definita in GeoUML.

La definizione in GeoUML di un attributo a tratti strutturale si basa sulla definizione di una classe speciale, introdotta dalla parola chiave "*tratto*", dotata di un attributo geometrico di tipo GU_CXCurve che non è necessario dichiarare e di uno o più attributi normali che si vogliono associare al singolo tratto. Normalmente le classi di tratti vengono definite immediatamente dopo la classe alla quale fanno riferimento, ma ciò non è indispensabile.

Con riferimento all'esempio precedente, definiamo la classe strada e la classe tratto nel modo seguente:

classe strada
    attributi: percorso: GU_CPCurve;
        ...altri attributi ...
tratto SedeDiStrada *di* strada.percorso
    attributi: sede: TIPO_SEDE;

(vedi anche diagramma 4.11)

I valori della classe speciale SedeDiStrada sono Tratti nel senso definito sopra, cioè sono dei sottoinsiemi delle curve primitive che compongono strada.percorso aventi lo stesso valore dell'attributo sede.

Se necessario, si può fare riferimento in un'espressione all'attributo geometrico della classe di tratti con il nome standard "*geometria*"; con riferimento all'esempio precedente, la notazione "SedeDiStrada.*geometria*" fa riferimento a questo attributo (che, ricordiamo, è di tipo GU_CXCurve).

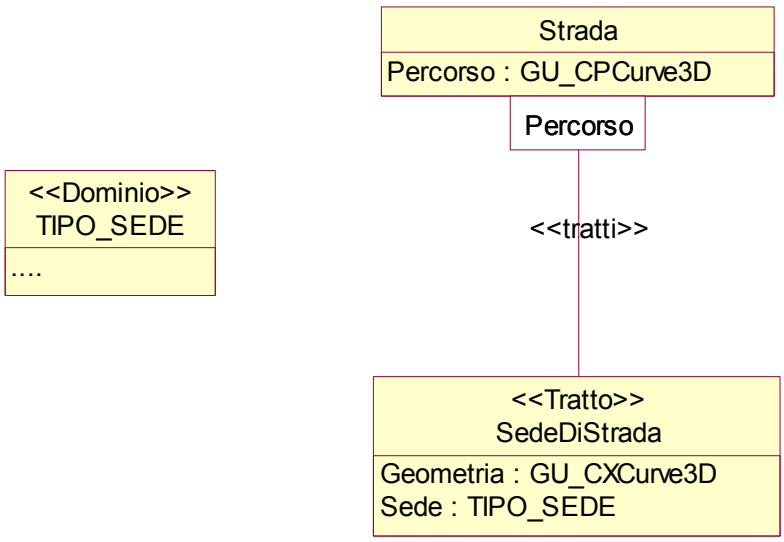

Diagramma 4.11

Osservazioni:

- i tratti hanno un attributo geometrico di tipo CXCurve, perchè sono costituiti da sottoinsiemi arbitrari delle curve primitive dell'attributo di riferimento; quindi, anche se in questo esempio strada.percorso è una CPCurve, un suo tratto può invece essere discontinuo;
- il tipo dei tratti deve essere nello stesso spazio di riferimento (2D o 3D) dell'attributo al quale si riferisce;
- una classe di tratti può avere più di un attributo (ad esempio, la sede e la larghezza); in questo caso ogni tratto deve possedere una unica combinazione di valori di tutti gli attributi;
- si possono definire più classi di tratti di uno stesso attributo geometrico; quindi è ad esempio possibile definire separatamente la classe di tratti "SedeDiStrada" e la classe di tratti "LarghezzaDiStrada" invece di un'unica classe di tratti "SedeLarghezza".

L'uso delle classi di tratti non è indispensabile in GeoUML, perchè è possibile rappresentare lo stesso significato utilizzando le classi normali ed i vincoli di struttura; ad esempio, la seguente definizione risulta equivalente a quella dell'esempio precedente e chiarisce ulteriormente il significato delle definizioni date sopra:

classe strada
attributi: percorso: GU_CPCurve;
...altri attributi ...
classe SedeDiStrada
attributi: sede: enumerato....;
percorsoTratti: GU_CXCurve
ruoli: StradaDiSede[1..1]: strada
inverso TrattiSede[0..*];
struttura: SedeDiStrada. percorsoTratti
qdj-appartiene StradaDiSede::strada. percorso

(vedi anche diagramma 4.12)

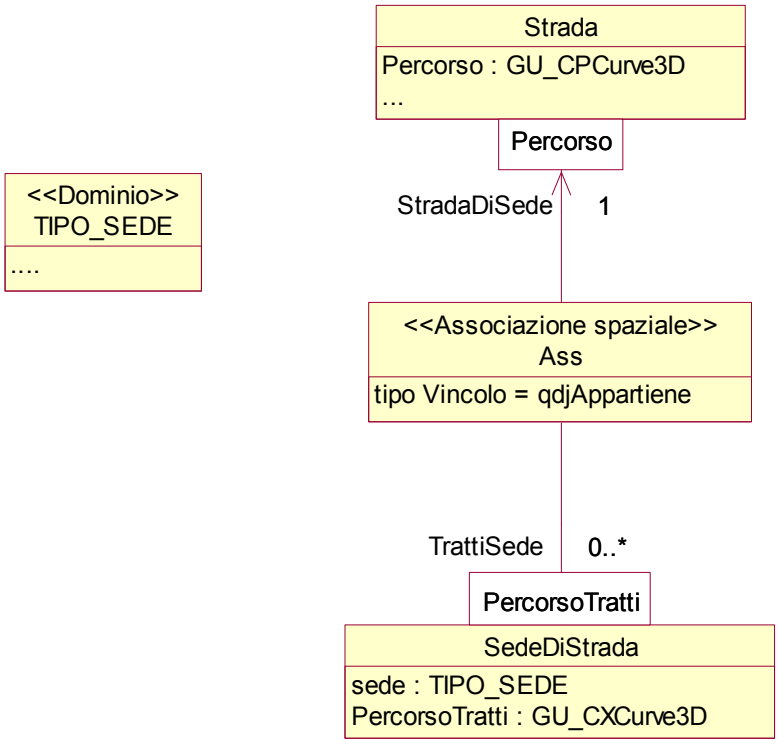

Diagramma 4.12

Cenni all'implementazione: la traduzione appena vista dei tratti in nozioni più elementari del GeoUML permette di definire la relativa implementazione di riferimento.
Volendo definire una implementazione diversa è necessario tenere presente che una curva primitiva può appartenere a diversi complessi, sia dello stesso attributo della stessa classe (esempio: una curva primitiva che appartiene al percorso di due diverse strade) oppure di due attributi diversi (ad esempio, una curva primitiva che appartiene sia al percorso di una diga, sia alla frontiera di un lago). In questi casi la primitiva potrà appartenere a diversi attributi a tratti, e l'implementazione deve garantire l'assenza di ambiguità.

4.5.3 Attributi a Sottoaree

Analogamente all'attributo a tratti strutturale su geometrie lineari, l'attributo a sottoaree associa un valore di un dominio a diverse sottoaree di un attributo geometrico areale.
Ad esempio, in analogia all'esempio precedente, si pensi a una classe "strada" dotata di un attributo geometrico areale "estensione" e all'attributo "sede" che descrive il tipo di sede di tale estensione. In questo esempio potremmo definire un attributo a sottoaree "sede" che associa il valore della sede ai diversi punti dell'estensione della strada; una "sottoarea" è una porzione dall'estensione alla quale è associato un particolare valore dell'attributo.
Definizione di Sottoarea:

> *Dato un valore di un attributo geometrico di tipo GU_CXSurface o GU_CPSurface costituito da un insieme S di superfici primitive, una sottoarea su questo valore è un sottoinsieme di S al quale viene associato un valore di un attributo A (o una combinazione di valori di un insieme di attributi).*

Nota: l'attributo A può essere multivalore; in tal caso ad ogni sottoarea è associato un preciso insieme di valori.
Si noti che, come per i tratti, questa definizione implica che una sottoarea possieda un valore preciso di attributo, ma non implica che tutte le superfici primitive che hanno lo stesso valore di attributo appartengano alla stessa sottoarea – possono esistere più sottoaree con lo stesso valore di attributo. Valgono quindi le stesse considerazioni svolte per i tratti.

La definizione in GeoUML di un attributo a sottoaree si basa sulla definizione di una classe speciale, introdotta dalla parola chiave "*sottoarea*", dotata di un attributo geometrico di tipo GU_CXSurface2D, che non è necessario dichiarare, e di uno o più attributi normali che si vogliono associare al singolo tratto. Normalmente le classi di sottoaree vengono definite immediatamente dopo la classe alla quale fanno riferimento.

Il seguente esempio in GeoUML è perfettamente analogo a quello visto precedentemente, ma lo svolge interpretando la sede come sottoaree di una rappresentazione areale della strada.

classe strada
    attributi: estensione: GU_CPSurface2D;
        ...altri attributi ...
sottoarea SedeDiStrada *di* strada.estensione
    attributi: sede: enumerato....;

(vedi anche diagramma 4.13)

I valori della classe speciale SedeDiStrada sono Sottoaree nel senso definito sopra, cioè sono dei sottoinsiemi delle superfici primitive che compongono strada.estensione aventi lo stesso valore dell'attributo sede.
Se necessario, si può fare riferimento all'attributo geometrico della classe di sottoaree con il nome standard "*geometria*"; con riferimento all'esempio precedente, la notazione "SedeDiStrada.*geometria*" fa riferimento a questo attributo (che, ricordiamo, è di tipo GU_CXSurface2D).

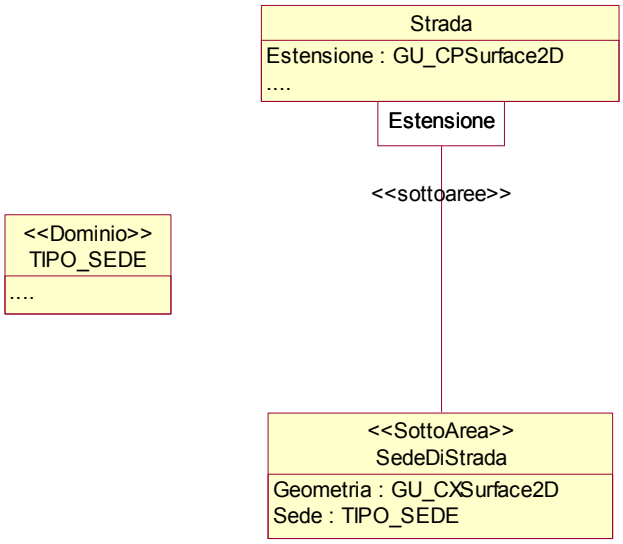

Diagramma 4.13

Come per i tratti, anche per le sottoaree è possibile rappresentare lo stesso significato utilizzando le classi normali ed i vincoli di struttura, e quindi valgono le considerazioni svolte relativamente all'implementazione di riferimento; ad esempio, la seguente definizione risulta equivalente a quella dell'esempio precedente:

classe strada
attributi: estensione: GU_CPSurface2D;
...altri attributi ...
classe SedeDiStrada
attributi: sede: enumerato....;
sottoaree: GU_CXSurface2D
ruoli: StradaDiSede[1..1]: strada
inverso SottoareeSede[0..*];
struttura: SedeDiStrada.sottoaree
qdj-appartiene StradaDiSede::strada. estensione

(vedi anche diagramma 4.14)

Per ragioni di simmetria si definisce in GeoUML anche una sintassi per la rappresentazione più astratta degli attributi a sottoaree, analoga a quella a tratti, come nel seguente esempio:

classe strada
attributi: estensione: GU_CPSurface2D;
...altri attributi ...
sede: TIPO_SEDE *aSottoaree su* estensione;
dominio TIPO_SEDE (....)

Questa forma si traduce solamente nella corrispondente forma strutturale, e quindi non permette di rinviare scelte implementative; è una forma sintatticamente conveniente se si vogliono raggruppare gli attributi a sottoaree dentro la classe di appartenenza, ma non

permette, come per i tratti, di esprimere alcune proprietà che fanno riferimento alle primitive geometriche.

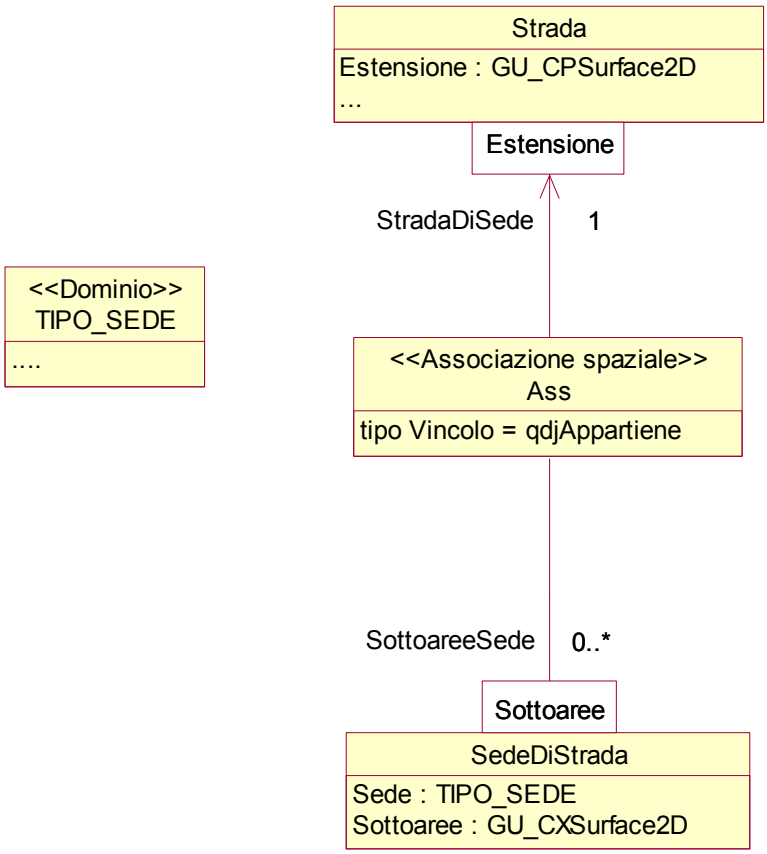

Diagramma 4.14

4.5.4 Attributi a Tratti Dinamici

Gli attributi a tratti dinamici svolgono una funzione simile a quella degli attributi a tratti strutturali, ma individuano un tratto non come un insieme di primitive geometriche ma come una porzione misurata lungo un percorso lineare; questo metodo è generalmente detto "coordinata curvilinea".

La definizione di un tratto dinamico, detto anche "evento lineare" (classe *eventoLineare* in GeoUML), presuppone di avere un insieme ordinato, anche se non necessariamente connesso, di "segmenti di percorso", ognuno di tipo GU_CPCurve, sul quale viene misurata l'ascissa curvilinea del punto iniziale e di quello finale. Tale insieme di segmenti deve essere contenuto nell'attributo geometrico dell'oggetto al quale l'attributo a tratti si riferisce, ma può non coincidere con esso.

Oltre agli eventi lineari è possibile definire "eventi puntiformi" (classe *eventoPuntiforme*), caratterizzati da un'unica misura.

In ambedue i casi la definizione della classe deve includere, oltre all'indicazione della classe e dell'attributo geometrico di riferimento, come per gli attributi a tratti strutturali, anche la dichiarazione di un'unità di misura e di un metodo, che può assumere i valori *assoluto* oppure *relativo*.

Con il metodo assoluto le misure sono prese dall'inizio dell'oggetto geometrico di riferimento, mentre con il metodo relativo le misure possono riferirsi ad altri punti presenti sull'oggetto geometrico.

Ogni evento lineare possiede due attributi standard di tipo real che non è necessario dichiarare, che definiscono i punti iniziale e finale dell'evento; a questi attributi si può fare riferimento nelle espressioni con le parole chiave *MisuraInizio* e *MisuraFine*. Per gli eventi puntiformi esiste un solo attributo real al quale si può fare riferimento con la parola chiave *MisuraPosizione*.

Applicando queste regole allo stesso esempio utilizzato per la dichiarazione dell'attributo a tratti "sede" in forma strutturale, otteniamo la seguente dichiarazione di attributo dinamico o evento lineare con metodo assoluto:

classe strada
    attributi: percorso: GU_CPCurve;
        ...altri attributi ...
eventoLineare SedeDiStrada *di* strada.percorso
        unitàMisura metri *metodo* *assoluto*;
    attributi: sede: enumerato....;

(vedi anche diagramma 4.15)

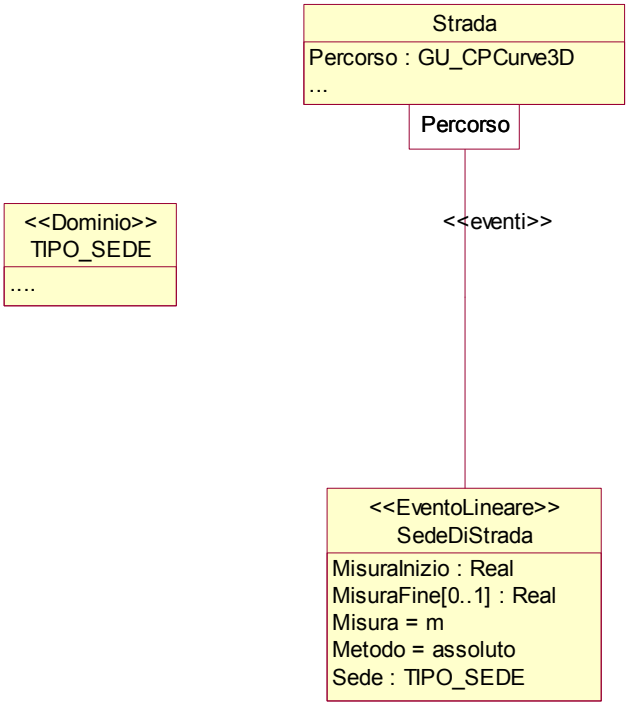

Diagramma 4.15

Come per gli attributi a tratti strutturali, anche per quelli dinamici l'uso delle classi di eventi lineari non è indispensabile in GeoUML, perchè è possibile rappresentare lo stesso significato utilizzando le classi normali ed i vincoli di struttura, ma in modo piuttosto complesso; ad esempio, la seguente definizione risulta equivalente a quella dell'esempio precedente e viene riportata perchè fornisce una ulteriore precisazione della corretta interpretazione da dare alle definizioni precedenti:

classe strada
    attributi: percorso: GU_CPCurve;
        ...altri attributi ...
classe SedeDiStrada
    attributi: sede: enumerato....;
        MisuraInizio: real;
        MisuraFine:real
    ruoli: StradaDiSede[1..1]: strada
            inverso Sede[0..*];
classe SegmentiDiStrada
    attributi: geometria: GU_CPCurve;
        lunghezza: real;
        unitàDiMisura: enumerato;
    ruoli: StradaDiSegmento [1..*] :strada
        inverso SegmentiDiStrada [0..*] *ordinati*;
struttura: SegmentiDiStrada. geometria
        qdj-appartiene StradaDiSegmenti::strada. percorso

(vedi anche diagramma 4.16)

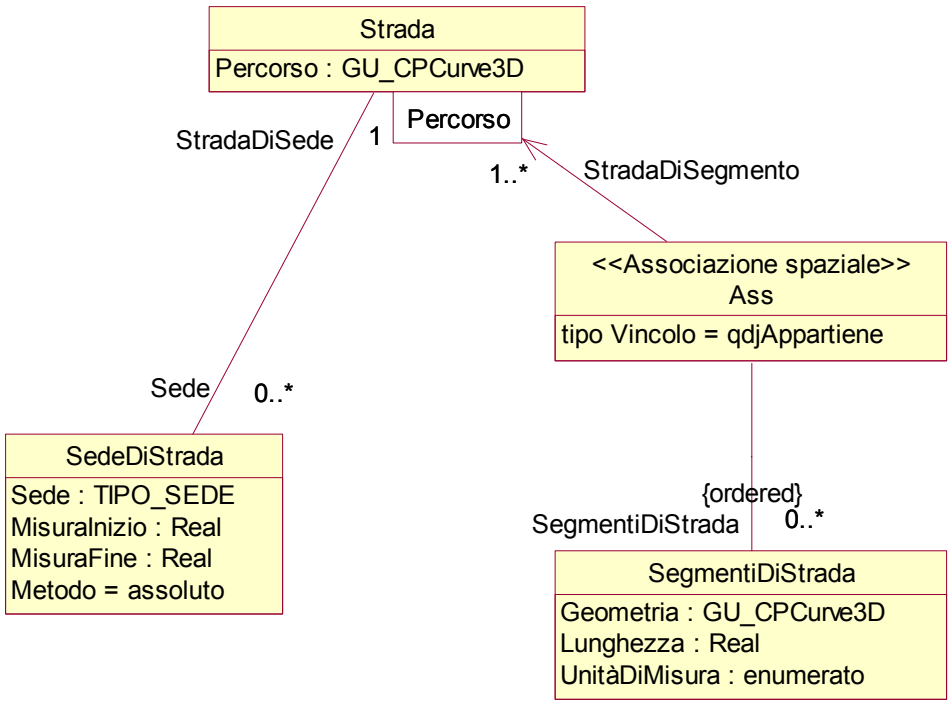

Diagramma 4.16

In questo schema i segmenti di una strada costituiscono un insieme ordinato di curve composte definito nella classe segmenti, mentre la classe SedeDiStrada definisce i parametri di un tipo di evento lineare (sede) con riferimento a tale insieme.

Un esempio di uso del metodo relativo è il seguente: l'evento puntiforme "incidente" viene posizionato in base alla distanza da una istanza di PietreMiliari, che deve avere un attributo puntiforme posizionato correttamente sul percorso della strada.

classe strada
    attributi: percorso: GU_CPCurve3D;
        ...altri attributi ...
eventoPuntiforme incidente *di* strada.percorso
        unitàMisura metri *metodo relativo a* PietreMiliari;
    attributi: tipo: TIPO_INCIDENTE;
classe PietreMiliari
    attributi: luogo: GU_Point3D;
        ...altri attributi ...

E' possibile esprimere anche questa struttura utilizzando classi normali nel modo seguente:

classe strada
attributi: percorso: GU_CPCurve3D;
...altri attributi ...
classe PietreMiliari
attributi: luogo: GU_Point3D;
...altri attributi ...
struttura PietreMiliari.luogo *appartiene* strada.percorso

classe incidente
attributi: tipo: TIPO_INCIDENTE;
MisuraPosizione: real;
ruoli: PietraDiRiferimento[1..1]: PietreMiliari;
classe SegmentiDiStrada
attributi: geometria: GU_CPCurve3D;
lunghezza: real;
unitàDiMisura: enumerato;
ruoli: StradaDiSegmento [1..*] :strada
inverso SegmentiDiStrada [0..*] *ordinati*;
struttura: SegmentiDiStrada. geometria
qdj-appartiene StradaDiSegmenti::strada. percorso

4.6 Superfici con frontiera 3D

Per agevolare la definizione di schemi relativi a oggetti rappresentati geometricamente da superfici bidimensionali dei quali si vuole rappresentare la frontiera in 3D sono definiti due tipi **GU_CPSurfaceB3D** e **GU_CXSurfaceB3D**.
Questi nuovi tipi non sono stati introdotti nel capitolo precedente, perchè dal punto di vista geometrico non definiscono un nuovo tipo di oggetto geometrico, ma una combinazione opportunamente strutturata di due oggetti geometrici dei tipi già trattati; la classe SurfaceB3D consiste di oggetti composti da due componenti, chiamate S e B, di tipo Surface2D e Ring3D rispettivamente, legate dal vincolo che la proiezione planare di B sia uguale alla frontiera di S, cioè contenga gli stessi punti.

Il tipo GU_CPSurfaceB3D può essere definito come una classe dotata di 2 attributi e di un vincolo topologico, la cui espressione potrà essere formulata con le regole che tratteremo nel prossimo capitolo 5:

classe GU_CPSurfaceB3D
attributi: S: GU_CPSurface2D;
B: GU_CXRing3D;
(più un vincolo topologico del tipo seguente:
GU_CPSurfaceB3D.B.planar uguale GU_CPSurfaceB3D.S.boundary)

Analogamente per la classe GU_CXSurfaceB3D:

classe GU_CXSurfaceB3D
attributi: S: GU_CXSurface2D;
B: GU_CXRing3D;
(più un vincolo topologico del tipo seguente:
GU_CXSurfaceB3D.B.planar uguale GU_CXSurfaceB3D.S.boundary)

(vedi anche diagramma 4.17)

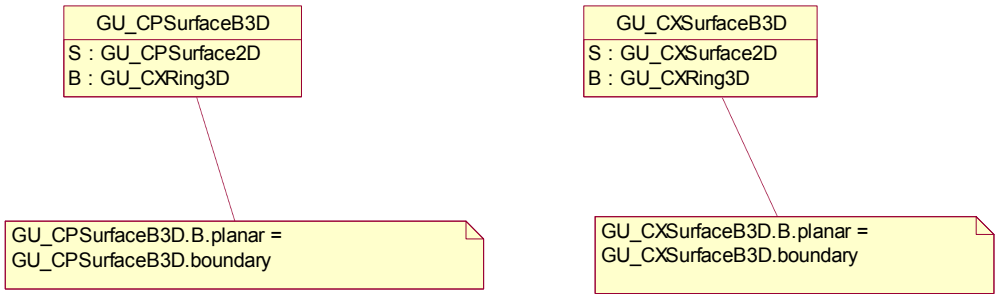

Diagramma 4.17

Dal punto di vista GeoUML queste due nuove classi predefinite sono trattabili come tipi geometrici quando si vuole assegnare questo tipo ad un attributo geometrico, ma per tutte le espressioni di vincoli in cui deve comparire un vero tipo geometrico è necessario esplicitare la componente (S oppure B) alla quale si vuole fare riferimento.

Le superficiB3D trovano molte possibili applicazioni nel contesto di GeoUML, perchè permettono di vedere gli oggetti areali considerandoli nello spazio tridimensionale come semplici anelli (cioè senza la definizione della superficie delimitata dall'anello stesso), ma permettono allo stesso tempo di definire molte proprietà aggiuntive, che richiedono il riferimento a superfici, quali la copertura di un'area, l'adiacenza, il contenimento di altri oggetti geometrici, ecc.., riferendosi alle superfici 2D delimitate dalle proiezioni di tali anelli.

Un esempio di dichiarazione in GeoUML con riferimento a questo tipo è il seguente:

classe Lago
attributi: estensione: GU_CPSurfaceB3D;
...

Specializzazione dei vincoli su superfici B3D
Come già detto, i vincoli definiti in questo capitolo per i tipi geometrici possono essere applicati alle componenti S e B di oggetti B3D, senza bisogno di ulteriori precisazioni, in quanto S e B sono dei normali tipi geometrici.
Esempio

strato AreeIdriche *geometria*: GU_CXSurface2D

classe Lago
attributi: estensione: GU_CPSurfaceB3D;

struttura Lago.estensione.S *dj appartiene* AreeIdriche.*geometria*;

In alcuni casi si vuole però esprimere un vincolo che coinvolge ambedue le componenti S e B, cioè si vuole che due oggetti le cui componenti S sono adiacenti, che condividono quindi una porzione della frontiera planare S.boundary, condividano anche la corrispondente componente lineare 3D (in sostanza, se fossero rilevate le superfici nello spazio 3D, esse sarebbero adiacenti anche in tale spazio).
Questo tipo di vincolo può essere espresso utilizzando i vincoli standard *dj_appartiene* oppure *partizionato* con riferimento all'attributo geometrico B3D senza specificare la componente S oppure B.
Nell'esempio precedente, se sostituiamo il vincolo strutturale con il seguente (nel quale è stato eliminato il riferimento alla componente S)

struttura Lago.estensione *dj appartiene* AreeIdriche.*geometria*

oltre al normale significato del vincolo su S aggiungiamo la proprietà che le componenti B di due laghi adiacenti siano condivise (ovvero: due laghi adiacenti come superfici bidimensionali condividono un'unica linea di separazione in 3D).
Questa forma del vincolo strutturale può essere applicata solo per i vincoli dj_appartiene e partizionato, non può essere utilizzata per gli altri.

5. I vincoli topologici

Oltre alle proprietà strutturali legate ai complessi viste nel capitolo precedente è possibile definire in GeoUML alcuni vincoli topologici più generali, che non si basano sui tipi complessi e sull'appartenenza di diversi oggetti allo stesso complesso.
I vincoli topologici sono condizioni che devono essere vere nel database e che fanno riferimento alle proprietà spaziali degli oggetti, in particolare alle relazioni topologiche tra oggetti. Le relazioni topologiche costituiscono la formalizzazione delle relazioni spaziali intuitive che vengono comunemente usate nell'analisi di una carta geografica da parte dell'utente umano.
A differenza dei vincoli strutturali i vincoli topologici non determinano la struttura dei dati e sono generalmente controllati proceduralmente al momento dell'aggiornamento.
Indipendentemente dal tipo di implementazione, a livello concettuale i vincoli di tutti i tipi costituiscono un importante contributo alla definizione delle proprietà dell'informazione che si intende rappresentare nel database.
Per tener conto della tendenza della informazione geografica a presentare molto più spesso di quella amministrativa condizioni anomale, per i vincoli topologici è prevista la nozione di ***eccezione certificata***, cioè della possibilità che in alcuni casi un oggetto geometrico violi la proprietà definita dal vincolo. A livello concettuale questo significa che le proprietà che vengono definite non devono necessariamente essere vere nella totalità dei casi; a livello implementativo questo significa che il vincolo dovrebbe essere sempre controllato, ma talvolta l'aggiornamento che viola il vincolo potrebbe essere accettato su esplicita indicazione.
La definizione dei vincoli topologici richiede di specificare due tipi di elementi:

1. le ***relazioni spaziali*** utilizzate nel vincolo
2. la ***struttura logica*** del vincolo

La struttura logica dei vincoli topologici spesso prevede la presenza di un ***quantificatore esistenziale*** (∃), cioè il vincolo richiede che data un'istanza c di una classe C *esista* un'istanza c' in un'altra classe C' tale che valga una certa relazione spaziale tra c e c'. A partire da questa forma sono definite numerose varianti.
Il seguito di questo capitolo è organizzato nel modo seguente: prima vengono definite le relazioni topologiche utilizzabili in GeoUML, poi viene descritto il vincolo topologico esistenziale di base, poi vengono descritte alcune varianti di tale vincolo, poi viene descritto un vincolo basato sul quantificatore universale invece che su quello esistenziale e infine viene descritta la possibilità di porre in alternativa diversi vincoli.

5.1 Le relazioni topologiche utilizzabili nell'espressione di vincoli

Per definire i vincoli topologici è necessario individuare un'insieme di relazioni topologiche di riferimento. Ciò non è semplice in quanto valori geometrici di tipo diverso danno vita a relazioni topologiche diverse. L'insieme di relazioni topologiche che viene proposto qui, chiamato REL_{topo}, si basa su lavori presentati in letteratura e sull'esperienza svolta nella definizione di contenuti, e costituisce un compromesso tra un insieme troppo piccolo di relazioni, facile da usare ma poco potente nel caratterizzare le diverse situazioni possibili, e un insieme troppo ricco, di uso e comprensione troppo difficili.
Le relazioni dell'insieme REL_{topo} sono le seguenti: Disjoint (DJ), Touch (TC), In (IN), Contains (CT), Equal (EQ) e Overlap (OV), che risultano applicabili a tutti i valori geometrici del GeoUML dotati di una definizione precisa di frontiera, che, come abbiamo visto, sono i valori di tutti i tipi eccetto GU_Complex e GU_Aggregate e i tipi derivati da quest'ultimo. Per questi ultimi tipi vedremo che si può applicare un insieme meno ricco si relazioni.

L'insieme di relazioni REL_{topo} possiede le seguenti caratteristiche:

- le relazioni che lo compongono sono mutuamente esclusive, cioè se tra due oggetti geometrici vale la relazione R, allora non vale nessuna altra relazione dell'insieme
- l'insieme è completo, cioè, dati due oggetti geometrici in una certa situazione spaziale esiste sempre una relazione dell'insieme che è vera in quella situazione

Le relazioni dell'insieme REL_{topo} (eccetto Equal) sono illustrate in figura 5.1, che mostra, se letta per colonne, come la stessa relazione topologica si presenti tra oggetti di tipo diverso, se letta per righe, come le diverse relazioni si applichino a coppie di oggetti di certi tipi.

La relazione Disjoint (DJ) sussiste tra oggetti che non hanno punti in comune, mentre tutte le altre relazioni prevedono che i due oggetti abbiano punti in comune.
Se i punti in comune non sono punti interni di ambedue gli oggetti, allora la relazione è Touch (TC), o adiacenza. La definizione considera non solo i casi ovvi di adiacenza, in cui solo punti della frontiera sono in comune, ma anche i casi più complessi, nei quali i punti di frontiera di un oggetto sono anche punti interni dell'altro. Questo comporta che, ad esempio, una linea contenuta nella frontiera di un poligono sia definita in Touch.
Il contenimento (IN e la relazione simmetrica CONTAINS) sono abbastanza ovvi, tranne per il fatto che escludono alcuni casi che sono in Touch, come la linea contenuta nella frontiera di un poligono, menzionata sopra.
L'uguaglianza è ovvia.
Nel caso di OVERLAP (OV) i due oggetti hanno punti interni in comune (quindi non sono in Touch), ma non sono in relazione di contenimento o uguaglianza (quindi ambedue gli oggetti hanno una parte che è fuori dalla parte comune).

Per una definizione precisa sono riportate le seguenti definizioni formali, basate sulla seguente notazione, con riferimento a due valori geometrici A e B:

- A è l'oggetto completo, che include la sua frontiera (chiusura di A)
- $A°$ è l'insieme dei punti interni ad A
- $\varnothing$ è l'insieme vuoto
- $A \cap B$ è l'intersezione insiemistica tra i punti di A e i punti di B
- *dim(A)* è una funzione che vale 0 se A è un punto, 1 se A è una linea e 2 se A è un poligono.

(A Disjoint B) $\equiv_{def}$ $(A \cap B = \varnothing)$
I due oggetti non hanno punti in comune.

(A Touch B) $\equiv_{def}$ $(A^\circ \cap B^\circ = \varnothing)$ *and* $(A \cap B \neq \varnothing)$
I due oggetti non hanno punti interni in comune, ma hanno punti in comune, quindi non sono disgiunti (nota che i due oggetti non possono essere ambedue punti, perchè in tal caso non aver punti interni comuni implica di non aver nessun punto in comune).

(A In B) $\equiv_{def}$ $(A \cap B = A)$ *and* $(A \cap B \neq B)$ *and* $(A^\circ \cap B^\circ \neq \varnothing)$
L'intersezione dei due oggetti è uguale all'oggetto contenuto, ma non a quello che lo contiene; inoltre i punti interni non sono disgiunti, quindi si esclude la relazione Touch.

(A Contains B) $\equiv_{def}$ *(B In A)*
Si ottiene per simmetria dalla relazione IN, scambiando i due oggetti.

(A Equal B) $\equiv_{def}$ $(A \cap B = A)$ *and* $(A \cap B = B)$
L'intersezione dei due oggetti è uguale ad ambedue i singoli oggetti

(A Overlap B) $\equiv_{def}$ $(A^\circ \cap B^\circ \neq \varnothing)$ *and* $(A \cap B \neq A)$ *and* $(A \cap B \neq B)$
I due oggetti hanno punti interni in comune, quindi non sono in Touch o Disjoint, e sono esplicitamente esclusi il contenimento e l'uguaglianza.

In aggiunta alle relazioni dell'insieme REL_{topo} sono definite in GeoUML alcune altre relazioni utili.

5.1.1 La relazione Intersects (INT)
La relazione Intersect è una relazione topologica molto generale, non primaria, cioè derivabile dalle altre, definita nel modo seguente:

$$(A \text{ Intersects } B) \equiv_{def} ((A \cap B \neq \varnothing) \text{ and not } (A \text{ Equal } B)$$

5.1.2 La relazione Cross (CR) tra oggetti lineari
La relazione Cross è una specializzazione della relazione Overlap per i soli oggetti di tipo lineare e richiede che la dimensione dell'intersezione sia puntiforme:

$$(A \text{ Cross } B) \equiv_{def} (A \text{ Overlap } B) \text{ and } dim(A \cap B)=0 \text{ and } dim(A)=dim(B)=1$$

5.1.3 Relazioni spaziali valide per oggetti di tipo GU_Complex e GU_Aggregate
Le relazioni dell'insieme REL_{topo} sono ben definite solamente per gli oggetti con frontiera ben definita. Per gli oggetti geometrici di tipo GU_Complex e GU_Aggregate invece si possono utilizzare solamente due delle relazioni dell'insieme REL_{topo} , cioè Equal e Disjoint, e la relazione Intersects.

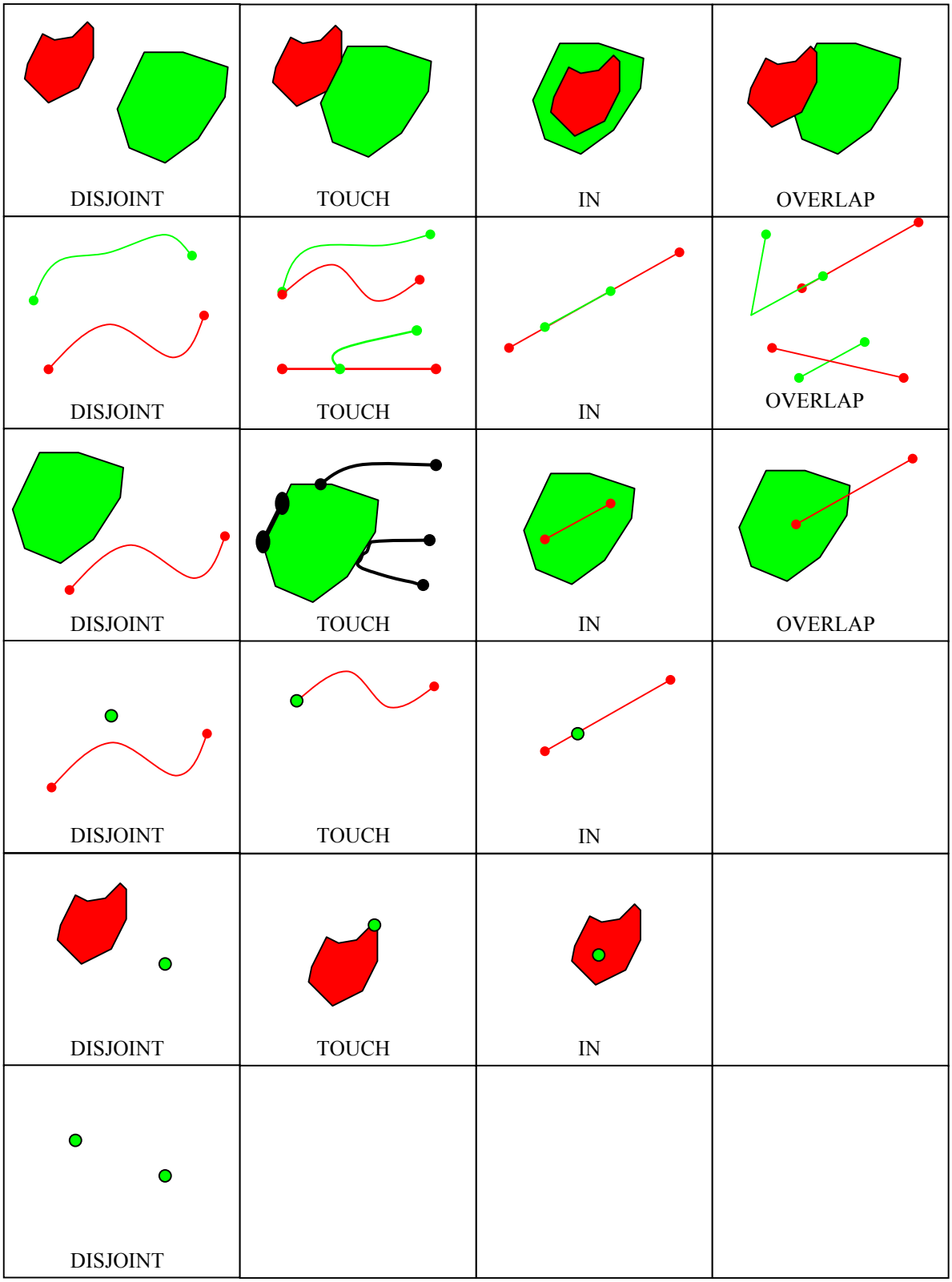
DISJOINT
TOUCH
IN
OVERLAP
DISJOINT
TOUCH
IN
OVERLAP
DISJOINT
TOUCH
IN
OVERLAP
DISJOINT
TOUCH
IN
DISJOINT
TOUCH
IN
DISJOINT

Figura 5.1

5.1.4 Interpretazione delle relazioni strutturali di disgiunzione e quasi-disgiunzione come vincoli topologici

Può essere utile interpretare le proprietà di disgiunzione e quasi-disgiunzione sulle primitive dei complessi, viste al capitolo precedente, in termini di relazioni topologiche, nel modo seguente:

- La disgiunzione (delle primitive interne) tra due oggetti complessi X e Y si traduce in termini di relazioni topologiche nell'espressione
  (*Disjoint* OR *Touch*).
- La quasi-disgiunzione tra due oggetti complessi X e Y di tipo lineare (se i complessi sono superfici nel piano abbiamo visto che la quasi-disgiunzione coincide con la disgiunzione) si traduce in termini di relazioni topologiche nell'espressione
  (*Disjoint* OR *Touch* OR *Cross*).

5.2 Vincolo topologico esistenziale di base

La forma più semplice di vincolo topologico esistenziale è quella che richiede, per ogni istanza c della classe vincolata C, l'esistenza di almeno una istanza c' della classe vincolante C', tale che un attributo geometrico g' di c' si trovi in una particolare relazione topologica con un attributo geometrico g di c.
Gli aspetti che caratterizzano il vincolo sono quindi:

- le due classi C e C' con i rispettivi attributi geometrici g e g'
- l'insieme di relazioni R

Questi elementi sono appunto specificati nelle varie forme di GeoUML.
La definizione testuale è illustrata dal seguente esempio, il cui significato è il seguente: per ogni ElementoStradale (classe vincolata) deve esistere una AreaStradale (classe vincolante) la cui estensione contiene il percorso dell'ElementoStradale stesso; in GeoUML:

classe AreaStradale
    attributi: estensione: GU_CPSurface2D
classe ElementoStradale
    attributi: percorso: GU_CPCurve2D
vtopo ElementoStradale.percorso IN *esiste* AreaStradale.estensione

Rispetto a quanto mostrato dall'esempio si tenga presente che è possibile scrivere una lista di relazioni separate da virgola per indicare che deve valere una di esse.

La rappresentazione grafica in Rose dello stesso esempio è mostrata dal diagramma 5.1, che sostanzialmente contiene gli stessi elementi, ma obbliga a dare un nome al singolo vincolo (in figura è chiamato VT_ElementoStradale)

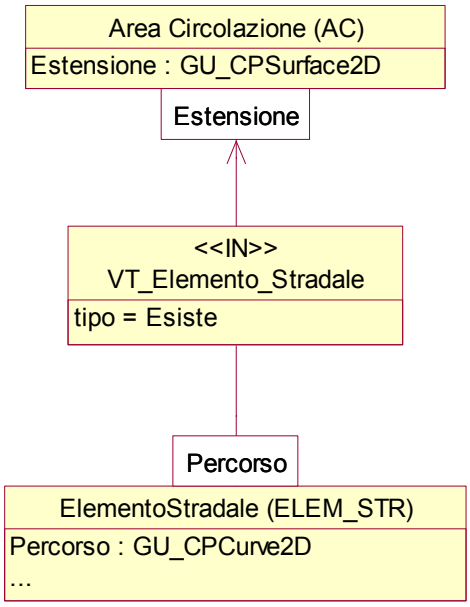

Diagramma 5.1

5.3 Varianti dei vincoli esistenziali

In questo capitolo si presentano 3 varianti del vincolo topologico esistenziale che aumentano la sua capacità di defnire accuratamente certe proprietà dell'informazione spaziale. Queste varianti possono essere applicate anche in combinazione tra loro.

5.3.1 Vincolo topologico esistenziale con selezioni

Una prima variante permette di selezionare le istanze delle classi coinvolte nel vincolo. Una selezione sulla classe vincolata limita le istanze che devono soddisfare il vincolo a quelle che soddisfano la selezione, rendendo il vincolo più debole; una selezione sulla classe vincolante riduce le sitanze che possono essere utilizzate per soddisfare il vincolo, rendendolo più forte.

La forma testuale del vincolo si basa sull'inserimento tra parentesi di una formula di selezione dopo la classe sulla quale si vuole operare la selezione; nel seguente esempio si restringe il vincolo dell'esempio precedente richiedendo che solo gli ElementiStradali di un certo tipo debbano soddisfare il vincolo di essere contenuti in un'AreaStradale.

classe AreaStradale
attributi: estensione: GU_CPSurface2D

classe ElementoStradale
attributi: percorso: GU_CPCurve2D
tipo: TIPO_ELEMENTO

dominio TIPO_ELEMENTO (T1, T2, T3)

vtopo ElementoStradale(tipo=T1).percorso
IN *esiste* AreaStradale.estensione

Per quanto riguarda la formula di selezione si ammettono due casi:

1. selezioni normali: espressioni logiche di predicati semplici del tipo "attributo comparatore valore", dove attributo deve appartenere alla classe alla quale si applica la selezione;
2. selezioni di join: in questa forma si ammettono predicati del tipo "attributo comparatore classeVincolata.attributo", che permettono di legare le istanze della classe vincolante utilizzabili per soddisfare il vincolo all'istanza considerate di classe vincolata.

Il seguente esempio mostra l'utilizzazione di un vincolo con selezione di tipo join per vincolare i tratti lineari di strada ad essere contenuti nelle sottoaree della strada con lo stesso valore di sede (questo è un esempio del motivo per cui gli attributi geometrici dei tratti e delle sottoaree, pur non essendo dichiarati esplicitamente, devono essere riferibili tramite i nomi standard *tratti* e *sottoaree*).

classe ElementoStradale
attributi: percorso: GU_CPCurve;
...altri attributi ...
tratto TrattoSede *di* ElementoStradale.percorso
attributi: sede: TIPO_SEDE;
classe AreaStradale
attributi: estensione: GU_CPSurface2D;
...altri attributi ...
sottoarea AreaSede *di* AreaStradale.estensione
attributi: sede: TIPO_SEDE;
vtopo TrattoSede.*geometria* IN *esiste*
AreaSede(sede=TrattoSede.sede).*geometria*

(vedi anche diagramma 5.2)

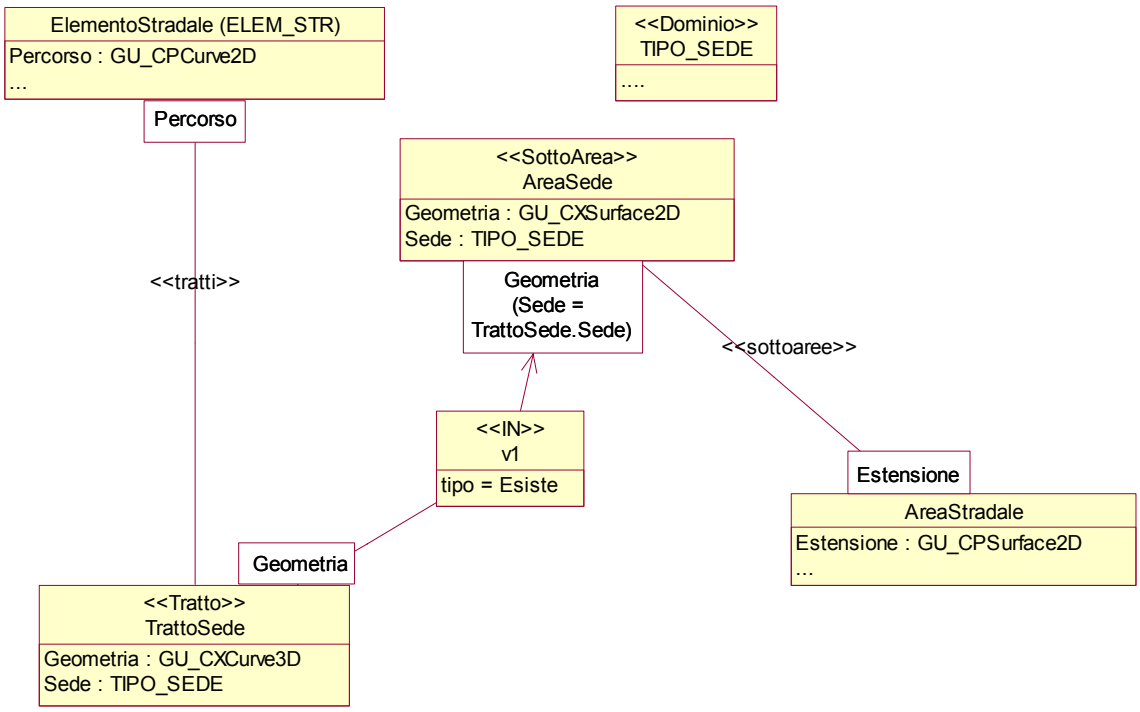

Diagramma 5.2

5.3.2 Vincolo topologico esistenziale sulla frontiera o sulla proiezione

Questa variante permette di applicare le funzioni boundary e/o planar agli oggetti geometrici coinvolti nel vincolo prima di verificare la relazione topologica. In tal modo è possibile esprimere, ad esempio, il contenimento di un oggetto nel boundary di un altro.

Le funzioni boundary e planar possono essere applicate sia alla classe vincolata che a quella vincolante e possono essere anche applicate in cascata, con la notazione:

classe.attributo.*bnd*.*pln*

Come esempio, si consideri l'esempio del paragrafo precedente applicato al caso in cui la classe ElementoStradale abbia geometria in 3D, per cui è necessario applicare la funzione planar nel vincolo:

classe AreaStradale
attributi: estensione: GU_CPSurface2D
classe ElementoStradale
attributi: percorso: GU_CPCurve3D
tipo: TIPO_ELEMENTO
dominio TIPO_ELEMENTO (T1, T2, T3)
vtopo ElementoStradale(ElementoStradale.tipo=T1).percorso.*pln*
IN *esiste* AreaStradale.estensione

Graficamente l'applicazione della funzione boundary (planar) viene indicata con l'inserimento del suffisso ".BND" "PLN" posto sugli attributi geometrici dei quali si voglia considerare la frontiera (o la proiezione).
La rappresentazione grafica dell'esempio precedente è mostrata nel diagramma 5.3

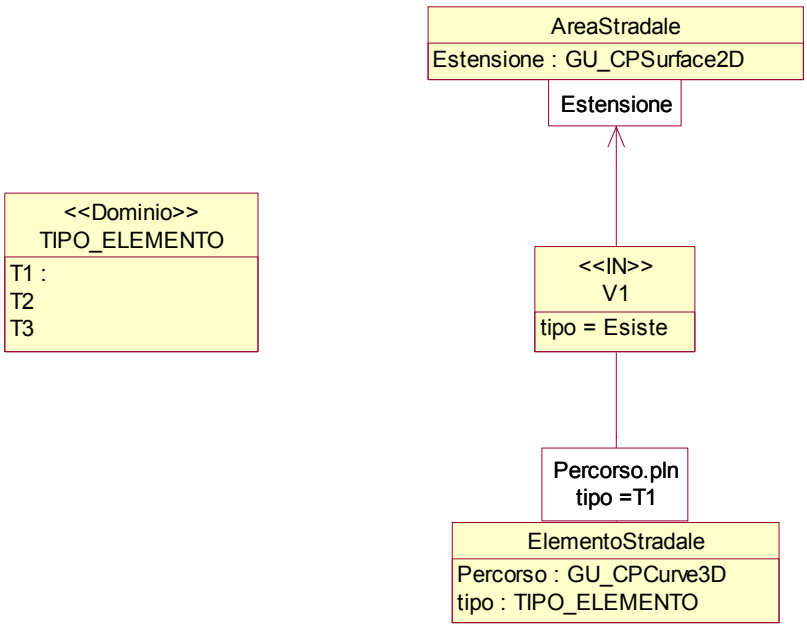

Diagramma 5.3

5.3.3 Vincolo topologico collegato ad una associazione

Con questa variante, che è analoga alla variante dello stesso tipo applicata ai vincoli strutturali, si considerano, ai fini del soddisfacimento del vincolo, solo gli oggetti della classe vincolante che sono legati all'oggetto da verificare attraverso un'associazione specificata nello schema.

La forma testuale di questa variante è illustrata dal seguente esempio, che utilizza un'associazione che svolge anche la funzione di chiave, già esemplificata al capitolo 2, e utilizza la funzione planar, per mostrare come i diversi tipi di vincoli possano combinarsi.

classe Strada
attributi: *PK* codice;
percorso: GU_CPCurve3D
ruoli: *PK* ComuneAppartenenza [1..1] : Comune
inverso StradaDelComune [0..*]
classe Comune
attributi: Estensione: GU_CPSurface2D
ruoli StradaDelComune [0..*]: Strada
inverso ComuneAppartenenza [1..1]
vtopo Strada.percorso.*pln* IN *esiste* Strada.ComuneAppartenenza.Estensione

La rappresentazione grafica dello stesso esempio è mostrata nel diagramma 5.4

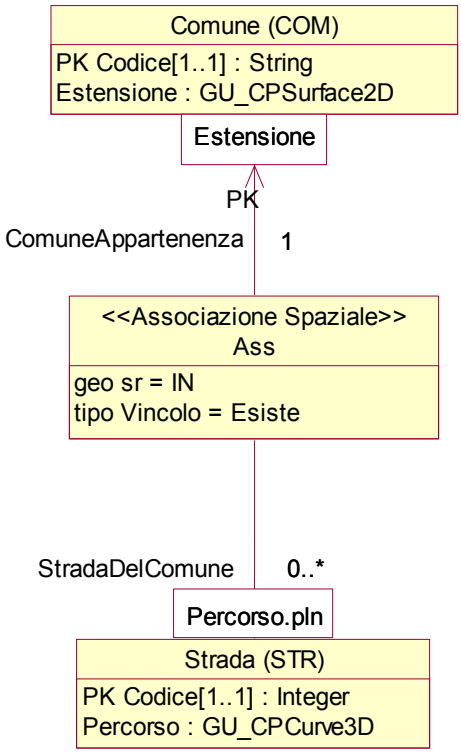

Diagramma 5.4

5.4 Vincolo topologico su unione

Il vincolo topologico su unione fa riferimento all'unione degli oggetti che rappresentano le istanze dell'attributo geometrico della classe vincolante CLY, invece di richiedere l'esistenza di una istanza che soddisfi il vincolo. Ciò significa che la relazione topologica viene verificata rispetto all'oggetto che si ottiene dall'unione dei valori dell'attributo geometrico di tutte le istanze di CLY.

Anche per il vincolo topologico su unione esistono le 3 varianti presentate per il vincolo topologico esistenziale, vale a dire: la versione con selezione, la versione che si riferisce alle funzioni boundary e planar, e quella che si riferisce a un'associazione.

Osservazione: è possibile interpretare il vincolo su unione in forma esistenziale nel modo seguente: deve esistere un sottoinsieme delle istanze della classe vincolante CLY tale che la loro unione soddisfi la relazione spaziale. Però, dato che il sottoinsieme più grande di CLY coincide con l'insieme intero CLY, il vincolo è stato formulato eliminando la parola chiave esiste.

Un esempio della forma testuale del vincolo è il seguente:

classe AreaStradale
attributi: estensione: GU_CPSurface2D
classe ElementoStradale
attributi: percorso: GU_CPCurve2D
...
vtopo ElementoStradale.percorso IN *union* AreaStradale.estensione

Rispetto alla versione esistenziale dello stesso esempio, qui non si richiede per ogni ElementoStradale che esiste una sola AreaStradale che lo contiene, ma che l'unione di tutte le istanze di AreaStradale lo contenga. il vincolo è quindi molto più debole.
La rappresentazione grafica del vincolo è mostrata nel diagramma 5.5

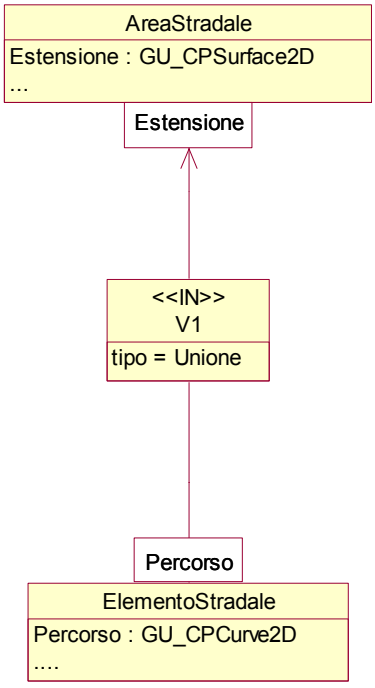

Diagramma 5.5

5.5 Vincolo topologico universale

Sostituendo il quantificatore esistenziale con quello universale otteniamo una nuova versione del vincolo topologico, che richiede il rispetto di una condizione molto più forte rispetto al vincolo esistenziale. Infatti, il vincolo topologico con quantificatore universale richiede che la relazione topologica sia presente tra l'oggetto vincolato e tutti gli oggetti della classe vincolante.

Il vincolo topologico universale è usato quasi esclusivamente con la relazione spaziale Disjoint, eventualmente in disgiunzione con altre relazioni spaziali (tipicamente Disjoint or Touch).

Anche per il vincolo topologico universale esistono le 3 varianti presentate per il vincolo topologico esistenziale, vale a dire: la versione con selezione, la versione che si riferisce alle funzioni boundary e planar, e quella che si riferisce a un'associazione.

Esempio

classe ElementoStradale
attributi: codice;
percorso: GU_CPCurve3D
vtopo ElementoStradale.percorso (DJ,TC) *perOgni* ElementoStradale.percorso

La rappresentazione grafica del vincolo è mostrata nel diagramma 5.6
La direzione della freccia è quella dall'entità vincolata a quella vincolante (cioè la direzione nella quale si leggono le relazioni spaziali). Si noti che nel caso in cui la classe vincolata e quella vincolante sono la stessa classe (come in questo esempio), preso un oggetto O della classe le relazioni spaziali indicate nel vincolo devono valere con tutti gli oggetti della classe stessa tranne O, perchè l'unica relazione spaziale di un oggetto con se stesso è Equal.

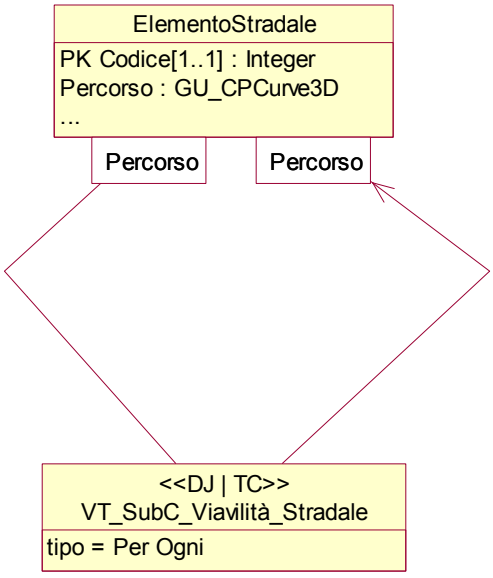

Diagramma 5.6

5.6 Disgiunzione di vincoli topologici

La disgiunzione di vincoli topologici permette di indicare che per ogni elemento di una classe vincolata deve essere soddisfatto almeno uno dei vincoli posti nella disgiunzione.

Nella forma testuale i vincoli in disgiunzione sono separati dalla parola chiave *or*; è necessario fare attenzione che *tutti i vincoli della stessa disgiunzione facciano riferimento alla stessa classe vincolata*.
Un esempio di disgiunzione in forma testuale è il seguente, che richiede che un ElementoStradale sia disgiunto da ogni altro ElementoStradale oppure la sua frontiera appartenga all'insieme delle Giunzioni:

classe ElementoStradale
    attributi: codice;
                percorso: GU_CPCurve3D
classe Giunzione
    attributi: luogo: GU_Point3D

vtopo ElementoStradale.percorso DJ *perOgni* ElementoStradale.percorso
    or ElementoStradale.percorso.*bnd* IN *union* Giunzione.luogo)

Dal punto di vista grafico una disgiunzione di vincoli topologici si rappresenta congiungendo con una linea i vincoli che sono in disgiunzione, come nel diagramma 5.7.

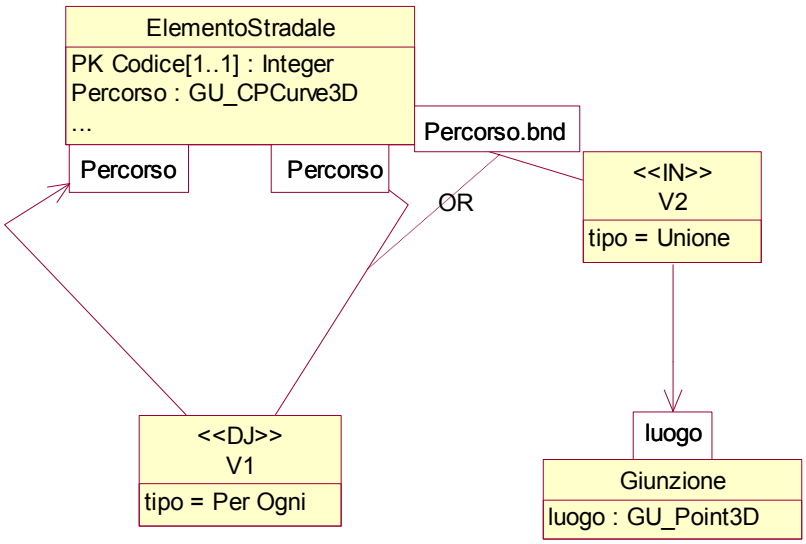

Diagramma 5.7