POLITECNICO
DI MILANO

CISIS
Centro Interregionale
per i Sistemi informatici, geografici e statistici
Comitato permanente
per i Sistemi Informativi Geografici

Guida alla lettura di uno Schema GeoUML

1 febbraio 2012

SpatialDBgroup

SpatialDBgroup@polimi.it

http://SpatialDBgroup.polimi.it

Autori	Politecnico di Milano – Spatial DB Group	Giuseppe Pelagatti (coordinatore), Alberto Belussi, Jody Marca, Mauro Negri
Coordinamento delle attività	CISIS – CPSG Comitato di Progetto	Maurizio De Gennaro (Regione del Veneto – coordinatore tecnico), Massimo Attias (CISIS-referente area geografica e progetti) Stefano Olivucci (Regione Emilia-Romagna), Raffaella Gelleti, Marco Lunardis, Massimo Zia (Regione Friuli Venezia Giulia), Massimiliano Basso, Alessandra Chiarandini (INSIEL-Regione FVG), Simone Patella (Regione Lazio), Gianbartolomeo Siletto (Regione Piemonte), Mauro Vasone (CSI Piemonte), Marco Guiducci, Andrea Peri (Regione Toscana), Gianfranco Amadio, Domenico Bertoldi, Gianluca Riscaio, Sandra Togni (Regione Umbria), David Freppaz (Regione Valle d'Aosta), Virgilio Cima, Umberto Trivelloni (Regione del Veneto), Leonardo Donnaloia, Claudio Mazzi, Pierpaolo Milan (CISIS)
	CISIS –CPSG Struttura di supporto interna	Massimo Attias, (coordinatore struttura), Leonardo Donnaloia, Claudio Mazzi, Pierpaolo Milan, Antonio Rotundo (CISIS)

Indice

1 Introduzione

Questo documento è una *guida alla lettura* di uno Schema (Concettuale) GeoUML, cioè di uno schema definito tramite il *modello GeoUML*. Questa guida fornisce una definizione informale degli elementi del modello GeoUML, sufficiente per molti usi pratici, ma non può sostiuire la specifica formale del modello, contenuta nel documento ***"Il modello GeoUML – Regole di Interpretazione delle Specifiche di contenuto per i Database Geotopografici"***, alla quale si rimanda per risolvere eventuali ambiguità di interpretazione.

1.1 Modello GeoUML e Schema Concettuale

Il modello GeoUML viene utilizzato per definire la parte strutturata, detta *Schema Concettuale*, di una Specifica di Contenuto.
Una Specifica di Contenuto contiene infatti delle porzioni scritte in testo libero e delle porzioni scritte seguendo delle precise regole di strutturazione. La parte strutturata o Schema Concettuale delle Specifiche di Contenuto è la parte delle specifiche che si ottiene eliminando tutte le porzioni formulate in testo libero.
Una Specifica di Contenuto in generale e il suo Schema Concettuale in particolare definiscono il contenuto che un Data Product deve possedere a livello concettuale, cioè *in maniera indipendente dalla tecnologia utilizzata per materializzarlo*.
Dato uno Schema Concettuale è possibile definire un insieme di regole che permettono di materializzare un Data Product che rappresenta i contenuti richiesti dallo Schema Concettuale su una particolare struttura fisica. Tali regole costituiscono un *Modello Implementativo*.

Il modello GeoUML è composto da un insieme di costrutti che consentono di definire lo schema concettuale di una specifica. I costrutti sono suddivisi in due categorie:

- gli ***Elementi Informativi***, che costituiscono tutti i componenti utilizzabili per definire la struttura dei contenuti informativi della specifica, e
- i ***Vincoli di Integrità***, che si applicano agli elementi informativi e definiscono le proprietà che i dati, contenuti in un qualsiasi Data Product conforme alla specifica, dovranno soddisfare.

1.2 Fonte degli Esempi

Gli esempi contenuti in questo documento sono estratti dal documento ufficiale "**CATALOGO DEI DATI TERRITORIALI - Specifiche di contenuto per i DB Geotopografici",** versione del 27 Aprile 2010, indicato nel seguito come [CDT2010].

1.3 Sintassi del linguaggio GeoUML

La sintassi (cioè la forma di rappresentazione dei concetti) del linguaggio GeoUML spiegata in questo testo è quella "tabellare", prodotta nei documenti di specifica dallo strumento GeoUML Catalogue e utilizzata dal documento ufficiale citato sopra; gli stessi concetti sono rappresentati diversamente in altri contesti (ad esempio, se si visualizza lo schema utilizzando lo strumento GeoUML Catalogue).
Inoltre, come già detto, in una Specifica di Contenuto lo Schema Concettuale vero e proprio è mescolato a porzioni descrittive di testo libero che devono essere sempre ben riconoscibili e separabili dallo Schema Concettuale.
Anche se la sintassi tabellare è quella principale, nel senso che uno Schema Concettuale deve essere completamente definito utilizzando tale sintassi, è possibile integrare tale schema con alcuni diagrammi in forma grafica. Valgono su questo aspetto le seguenti regole e limitazioni:

1. I diagrammi sono considerati non convenienti per rappresentare la struttura interna delle classi (attributi, domini, ecc...), ma costituiscono un utile complemento alla forma testuale nella rappresentazione dei legami che sussistono tra classi diverse, cioè associazioni tra classi, gerarchie di ereditarietà tra classi e vincoli di integrità tra classi;

2. La forma grafica del GeoUML è pertanto definita solamente per queste componenti: classi, associazioni, gerarchie e vincoli;
3. Nel caso in cui, per errore, vi sia un'inconsistenza tra i diagrammi e le definizioni date nella parte testuale, *prevale sempre la forma tabellare*.

2 Elementi Informativi di base del GeoUML

2.1 Nomi, codici numerici e codici alfanumerici

Tutti i costrutti di base del modello GeoUML possiedono le seguenti proprietà:

- **Nome applicativo** (obbligatorio): è la parola (o insieme di parole) che identifica il costrutto nel contesto applicativo a cui la specifica si riferisce.
- **Codice** (obbligatorio ad eccezione dei vincoli): è un codice univoco alfanumerico che identifica il costrutto.
- **Codice alfanumerico** (obbligatorio per le classi, ma opzionale per gli altri costrutti): è un'abbreviazione del nome applicativo.

2.2 Classe, Strato e Tema

La nozione fondamentale dello Schema Concettuale è la ***Classe***. Una classe definisce un insieme di oggetti omogenei per quanto riguarda la struttura del loro contenuto informativo. Tale struttura di contenuto è rappresentata in primo luogo dall'insieme degli ***attributi descrittivi*** e degli ***attributi geometrici*** (o ***componenti spaziali***) della classe.
Un oggetto appartenente a una classe è chiamato ***istanza***; ogni oggetto è dotato implicitamente di un ***identificatore*** (OID) che non viene esplicitamente dichiarato. L'insieme delle istanze presenti in un certo contesto (ad esempio, un Data Product) è detto ***popolazione*** (della classe in quel contesto).

ESEMPIO 2.1
In Figura 2.1 è mostrata la definizione della classe "Albero isolato" estratta da [CDT2010]. Tra parentesi dopo il nome sono indicati il codice alfanumerico e il codice numerico - (ALBERO - 060403). Nel seguito nella discussione di tutti gli esempi utilizzeremo sempre il codice alfanumerico, perché più breve del nome e più espressivo del codice numerico.

Le 2 scritte successive, *Classe con istanze monoscala* e *Popolamento della classe* non le consideriamo per il momento; la spiegazione verrà fornita nel capitolo relativo al problema del popolamento diversificato. Per lo stesso motivo trascuriamo le 2 colonne di destra, titolate NC1 e NC5.

Successivamente la definizione della classe contiene la sezione relativa agli Attributi della classe (attributi descrittivi); per ogni attributo descrittivo sono elencati:

- il codice: ***06040301***, del quale si può osservare che è costituito dalle 6 cifre della classe seguite da 2 cifre relative all'attributo (deve essere univoco nell'ambito della specifica)
- il codice alfanumerico: ***ALBERO_TY*** (opzionale)
- il nome: ***tipo*** (deve essere univoco nell'ambito della classe)
- il tipo: ***enum***, seguito da una sottotabella intestata Dominio, che verrà spiegata più avanti

Dopo la sezione relativa agli attributi descrittivi c'è la sezione relativa alle Componenti Spaziali della classe. Questa classe possiede una sola componente spaziale, caratterizzata da

- il codice: ***06040301***
- il codice alfanumerico: ***ALBERO_POS***
- il nome: ***Posizione***
- il tipo - ***GU_Point3D*** – intuitivamente si tratta di un punto tridimensionale (la spiegazione dettagliata dei tipi geometrici è fornita nel relativo capitolo)

CLASSE: Albero isolato (ALBERO - 060403)

Classe con istanze monoscala

	NC1	NC5
Popolamento della classe	P	

Attributi						
Attributi della classe					NC1	NC5
06040301	ALBERO_TY	tipo		Enum	P	
	Dominio (Tipo)				NC1	NC5
	01	monumentale			P	
	95	altro			P	

Componenti spaziali della classe				NC1	NC5
060403101	ALBERO_POS	Posizione	GU_Point3D - Point 3D	P	

Figura 2.1

Le classi sono raggruppate per comodità in ***Strati*** e ***Temi***, formando una gerarchia nella quale diverse classi appartengono a un unico Tema e diversi Temi appartengono a un unico Strato. In pratica gli Strati e i Temi svolgono 2 funzioni:

1. costituiscono una strutturazione in capitoli e sottocapitoli della specifica
2. possono essere utilizzati, come in [CDT2010], nella costruzione del codice delle classi, perche le prime 2 cifre del codice di una classe derivano dal codice dello strato e le seconde due dal codice del tema.

ESEMPIO 2.2

Nel documento [CDT2010] la definizione della classe ALBERO è contenuto nel sottocapitolo (tema) Verde Urbano del capitolo (strato) Vegetazione.
Il codice 060403 della classe ALBERO deriva dal codice 06 dello strato Vegetazione e dal codice (06)04 del tema Verde Urbano.

2.3 Domini di base degli attributi descrittivi

I domini o tipi degli attributi descrittivi descrivono l'insieme dei valori che tali attributi possono assumere; i domini possono essere:

- domini di base (descritti in questa sezione)
- dominio enumerato (descritto più avanti)
- dominio enumerato gerarchico (descritto più avanti)
- dominio DataType (descritto più avanti)

I domini di base sono: *String*, *NumericString*, *Integer*, *Real*, *Boolean*, *Date*, *Time*, *DateTime*.

- *String* rappresenta una sequenza di caratteri di lunghezza finita,

- *NumericString* rappresenta una sequenza di cifre di lunghezza finita,
- *Integer* rappresenta i numeri interi,
- *Real* i numeri reali in virgola mobile,
- *Boolean* i valori di verità vero e falso,
- *Date* raccoglie i valori di tipo data nel formato: gg/mm/aaaa,
- *Time* i valori di tipo ora nel formato: hh:mm:ss,
- *DateTime* individua valori di timestamp formati da una data e un'ora nel formato: gg/mm/aaaa hh:mm:ss.

Per motivi legati all'implementazione e all'interoperabilità è necessario indicare, nei tipi *String* e *NumericString* un parametro che indica la lunghezza massima delle stringhe rappresentate. Quindi nella specifica tali tipi assumo la forma: String(N) e NumericString(N), dove N rappresenta la lunghezza massima.

2.4 Dominio enumerato

Un ***dominio enumerato*** è un dominio finito i cui valori sono predefiniti ed elencati nello schema.
La definizione di un attributo enumerato ha due forme: i valori possono essere elencati direttamente nella definizione dell'attributo, in tal caso si parla di *"dominio embedded"* e tale dominio è strettamente legato all'attributo e viene cancellato quando si cancella l'attributo, oppure essere elencati in una definizione separata del dominio, in tal caso si parla di "*dominio dichiarato separatamente*".
In Figura 2.1 abbiamo visto un esempio di dominio enumerato embedded; i valori possibili sono elencati all'interno della dichiarazione dell'attributo della classe e sono costituiti da un codice numerico e da un valore alfanumerico.
I domini dichiarati separatamente sono elencati in una sezione apposita dello schema concettuale; ogni dominio dichiarato ha il codice, il nome e può avere il codice alfanumerico. In figura 2.2 è mostrata la dichiarazione del dominio *Lingua (0200)*.

DOMINIO: Lingua (0200)

Valori del dominio			**NC1**	**NC5**
01	**bulgaro – bul**		**P**	**P**
02	**ceco – cze**		**P**	**P**
03	**danese – dan**		**P**	**P**
04	**estone – est**		**P**	**P**
05	**finlandese – fin**		**P**	**P**
06	**francese – fre**		**P**	**P**
07	**greco – gre**		**P**	**P**
08	**inglese – eng**		**P**	**P**
09	**irlandese – gle**		**P**	**P**
10	**italiano – ita**		**P**	**P**
altri valori …	…	…	**P**	**P**

Figura 2.2

2.5 Dominio enumerato gerarchico

In alcuni casi è necessario rappresentare attributi i cui valori enumerati 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 un ulteriore attributo di dominio enumerato (che rappresenta un livello aggiuntivo della gerarchia) e uno o più attributi aggiuntivi monovalore di tipo base.
Anche il dominio enumerato gerarchico può essere embedded oppure a dichiarazione separata.

ESEMPIO 2.3

In figura 2.3 è mostrata la definizione del dominio gerarchico embedded dell'attributo struttura della classe Ponte. La struttura gerarchica è ricostruibile dai codici: i valori da 0101 a 0106 sono specializzazioni del valore 01, i valori da 0601 a 0604 sono specializzazioni del valore 06. Si noti che si può nidificare questo tipo di attributo anche più profondamente.

Attributi					
…					
02030103	PONTE_STRU	struttura	Enum	P	P
	Dominio (Struttura)			NC1	NC5
	01	fisso		P	P
	0101	ad arco			
	0102	a sbalzo			
	0103	di barche		P	
	0104	a trave, struttura reticolare o piena			
	0106	altro			
	06	mobile		P	P
	0601	a sollevamento verticale o scorrevole			
	0602	levatoio			
	0603	girevole			
	0604	girevole/scorrevole			
… altri attributi					

Figura 2.3

2.6 Il dominio DataType

Il dominio *DataType* definisce un insieme di valori "strutturati", cioè composti da una combinazione di valori elementari. I valori elementari (componenti) possono appartenere ai domini di base, enumerati e enumerati gerarchici, ma non ai DataType e ai tipi geometrici.

ESEMPIO 2.4

In figura 2.4 è mostrata la definizione del Datatype Multilinguismo; costituisce in pratica un'abbreviazione per indicare la coppia di elementi nome e lingua. Tale abbreviazione è utile quando lo stesso Datatype viene utilizzato per diversi attributi presenti in diverse classi, come è tipicamente il caso del Multilinguismo, che caratterizza molti nomi definiti nella specifica.

DATATYPE: Multilinguismo (MULTILING - 80)

Attributi del Datatype					NC1	NC5
	01	NOME	nome	String(100)	P	P
	02	LINGUA	lingua	Enum (Lingua)	P	P

Figura 2.4

2.7 Cardinalità degli attributi (attributo multivalore, attributo opzionale, valori nulli)

La cardinalità degli attributi stabilisce il numero minimo e massimo di valori che possono essere presenti per ogni oggetto della classe.
La cardinalità può essere applicata a tutti i tipi di attributi (descrittivi e geometrici) ed è espressa dalla notazione [min..max] con le seguenti combinazioni [0..1], [1..1], [0..*] e [1..*].
Nel caso in cui sia omessa è assunto il valore di default [1..1]; questo è il caso nell'esempio di figura 2.1
Il valore "*" della cardinalità massima indica che il valore dell'attributo può essere costituito da un insieme di valori senza duplicati.
L'opzionalità del valore dell'attributo (cardinalità minima uguale a zero) significa che è possibile assegnare a tale attributo il valore *nullo*; tale valore costituisce un valore speciale del GeoUML la cui implementazione in diversi Modelli Implementativi può variare.
Si noti che non sono ammessi la stringa vuota o l'attributo multivalore vuoto (insieme vuoto) come sostituzione del valore nullo negli attributi opzionali (e tantomeno negli attributi obbligatori).
Dato che il valore nullo può avere origini e significati diversi, è possibile associare ad ogni valore nullo un'etichetta presa dal dominio "*Null_Interpretation (D_NI)*". Tale dominio deve essere configurato per ogni specifica di contenuto (in assenza di configurazione tale dominio è vuoto).

ESEMPIO 2.5

Nello Schema del National Core sono previsti i seguenti casi di valore nullo

	CODICE	VALORE	DESCRIZIONE
Incompletezza dell'informazione	**91**	**Non conosciuto**	Valore supposto esistente ma non conosciuto in fase di raccolta dati
	93	**Non definito**	Valore non assegnato perché non esiste o non è stato definito nell'universo reale (è il caso di una denominazione od una codifica)
	94	**Non applicabile**	Valore previsto dalla specifica non applicabile all'istanza

Cardinalità di attributi con dominio DataType

Un attributo con dominio DataType ha a sua volta una cardinalità, se ne evidenziano le implicazioni:

- gli attributi che costituiscono il Datatype possono essere opzionali o obbligatori,ma non possono essere multivalore;
- nel caso in cui l'attributo con dominio DataType abbia cardinalità minima 1, cioè sia obbligatorio, deve esistere almeno un record e in ogni record deve esistere almeno un componente con valore diverso da nullo.

2.8 Associazione (binaria) senza attributi

Un'associazione rappresenta un legame tra gli oggetti di due classi. L'associazione viene rappresentata in ognuna delle due classi da un ruolo, che è simile a una attributo i cui valori sono oggetti dell'altra classe. Analogamente agli attributi un ruolo ha una cardinalità, con le stesse convenzioni degli attributi.

ESEMPIO
Consideriamo la definizione di ruoli di figura 2.5, estratta dalla definizione della classe Unità Volumetrica (UV). Il ruolo ***Cediuv*** può essere letto come un attributo particolare i cui valori sono gli identificatori dei corpi edificati (CR_EDF) ai quali una istanza di UV è collegata. La notazione *Cediuv [1] : CR_EDF* indica che data una unità volumetrica deve essere presente uno e un solo CR_EDF associato.
La parte successiva della stessa riga *inverso Uvdice [0..*]* indica che nella classe CR_EDF è presente un ruolo inverso, che, data un'istanza di CR_EDF gli associa un insieme eventualmente vuoto di UV.

Ruoli

Cediuv

definisce di quale corpo edificato è parte la specifica unità volumetrica. Non possono esistere unità volumetriche che non siano associate ad alcun corpo edificato.

Cediuv [1] : **CR_EDF** inverso **Uvdice** [0..*]

Figura 2.5

Talvolta un'associazione possiede attributi propri. 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*.

Nel documento [CDT2010] non sono presenti associazioni con attributi.

2.9 Ereditarietà tra classi

L'ereditarietà tra classi istituisce una relazione di sottotipo tra una o più classi (dette ***sottoclassi***) e un'altra classe (detta ***superclasse***), con le seguenti implicazioni:

- tutti gli oggetti della sottoclasse appartengono anche alla superclasse, ma non necessariamente viceversa
- la sottoclasse eredita gli attributi, i ruoli (associazioni), le loro cardinalità e i vincoli nei quali la superclasse è vincolata;
- la sottoclasse può aggiungere alle proprietà ereditate propri attributi, ruoli e vincoli; in questo caso il nome assegnato ai propri attributi e ruoli deve essere univoco nella classe considerando anche le proprietà ereditate.

Una gerarchia può essere ***complete*** (ogni oggetto della superclasse appartiene almeno ad una sottoclasse) oppure ***incomplete***.
Una gerarchia può essere ***disjoint*** se un oggetto della superclasse non può appartenere contemporaneamente a più sottoclassi, ***overlapping*** in caso contrario.

Poiché una superclasse può essere a sua volta sottoclasse di un'altra classe è possibile generare una ***gerarchia di ereditarietà a più livelli***. In tal caso un oggetto di una classe appartiene anche a tutte le classi antenate dirette o indirette nella gerarchia e la classe eredita le proprietà di tutte le classi antenate della gerarchia. Ciò significa che un'associazione o un vincolo tra due classi coinvolgerà gli oggetti delle due classi e gli oggetti di tutte le classi definite nella porzione di gerarchia di ereditarietà della quale le due classi specificate ne rappresentano la radice.

Una classe può essere ***astratta (ABSTRACT)***. Una classe astratta non può avere istanze dirette, cioè le sue uniche istanze sono quelle appartenenti alle sue sottoclassi e pertanto può essere definita solo come superclasse di una gerarchia completa di classi al fine di fattorizzare la rappresentazione di proprietà comuni a più sottoclassi.

Commento

> Una gerarchia di ereditarietà coinvolge spesso classi astratte al fine di definire proprietà che le classi concrete metteranno poi a disposizione; per questo motivo esse sono spesso collocate alla radice della gerarchia. Una classe astratta è sempre superclasse di una gerarchia "complete" in quanto essa non può avere istanze proprie.

ESEMPIO 2.6

Consideriamo la gerarchia costituita dalla classe astratta Corpo Edificato (superclasse) e dalle due sottoclassi Edificio Minore (EDI_MIN) e Edificio (EDIFC). La gerarchia è dichiarata *"disjoint, complete"* (vedi figura 2.6).
Nella figura sono mostrate parzialmente la definizione della classe astratta corpo edificato (CR_EDF) e della sottoclasse Edificio minore (EDI_MIN), che eredita tutti gli attributi e i ruoli di CR_EDF. Per leggibilità, sono state eliminate alcune dichiarazioni di attributi e di domini irrilevanti. La porzione evidenziata nella classe EDI_MIN è costituita dagli elementi ereditati dalla classe CR_EDF.

CLASSE <<ABSTRACT>>: Corpo edificato (CR_EDF - 020181)

SUPERCLASSE Disjoint complete DI [EDI_MIN, EDIFC]

Componenti spaziali della classe							**NC1**	**NC5**
020181101	**CR_EDF_IS**	**Ingombro al suolo**	**GU_CXSurfaceB3D - Complex Surface Boundary 3D**				**P**	**P**
Attributi di questa componente spaziale							**NC1**	**NC5**
02018101	**CR_EDF_TYC**	**Tipo di contorno [0..1]**	**Enum**	aTratti sul contorno 3D su	Ingombro al suolo			
020181102	**CR_EDF_ME**	**Max_estensione**	**GU_CPSurface2D - Composite Surface 2D**				**P**	
02018102	**CR_EDF_POR**	**Tipo di porzione**	**Enum**	aSottoaree su	Max_estensione		**P**	

Ruoli **Uvdice**

Uvdice [0..*] : **UN_VOL** inverso **Cediuv** [1]

Cpdice

Cpdice [0..*] : **ELE_CP** inverso **Cedicp** [0..1]

CLASSE: Edificio minore (EDI_MIN - 020106)

SOTTOCLASSE DI : CR_EDF

Attributi								
Attributi della classe							**NC1**	**NC5**
02010601	**EDI_MIN_TY**	**tipologia edilizia**	**Enum**				**P**	**P**
	Dominio (Tipologia edilizia)						**NC1**	**NC5**
ecc…								
02010602	**EDI_MIN_PR**	**struttura precaria**	**Boolean**				**P**	**P**
020181101	**CR_EDF_IS**	**Ingombro al suolo**	**GU_CXSurfaceB3D - Complex Surface Boundary 3D**				**P**	**P**
Attributi di questa componente spaziale							**NC1**	**NC5**
02018101	**CR_EDF_TYC**	**Tipo di contorno [0..1]**	**Enum**	aTratti sul contorno 3D su	Ingombro al suolo			
020181102	**CR_EDF_ME**	**Max_estensione**	**GU_CPSurface2D - Composite Surface 2D**				**P**	
02018102	**CR_EDF_POR**	**Tipo di porzione**	**Enum**	aSottoaree su	Max_estensione		**P**	

Ruoli **Uvdice**

Uvdice [0..*] : **UN_VOL** inverso **Cediuv** [1]

Cpdice

Cpdice [0..*] : **ELE_CP** inverso **Cedicp** [0..1]

2.10 Attributo di attributo geometrico

Gli attributi geometrici possono essere ulteriormente descritti da attributi che precisano alcune caratteristiche della geometria che viene rappresentata nell'attributo geometrico.
Un attributo di attributo geometrico può essere definito su qualsiasi tipo di attributo geometrico, ha un nome univoco nell'ambito degli attributi della componente spaziale sulla quale è definito, ha un codice, un codice alfanumerico opzionale e può avere la cardinalità come gli attributi normali; infine, il suo dominio può essere qualsiasi dominio applicabile agli attributi normali della classe.

2.11 Chiave primaria

Tutti gli oggetti sono dotati di un identificatore automatico, chiamato OID (object identifier), che li identifica in tutto il Data Product.
Oltre a questo identificatore possono essere definiti ulteriori identificatori, chiamati ***chiave primaria***, al fine di soddisfare la caratteristica di visibilità esterna.
Una chiave primaria è un identificatore definito all'interno di una classe ed è costituita da un insieme di attributi e/o ruoli di associazioni (senza attributi) che ha le seguenti proprietà:
- il dominio degli attributi deve essere uno di quelli di base, un dominio enumerato o un dominio gerarchico;
- la cardinalità degli attributi e/o ruoli è [1..1].

2.12 Strato topologico

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

- nei dati può esistere un solo oggetto di tale classe (classe mono-istanza)
- ha un unico attributo geometrico monovalore di nome ***geometria***, il cui tipo può essere solamente uno dei seguenti: GU_CXCurve2D, GU_CXCurve3D, GU_CXSurface2D (questi tipi sono definiti nel prossimo capitolo)

Gli strati topologici vengono spesso utilizzati per imporre condizioni più stringenti sulla strutturazione di un gruppo di oggetti geometrici, ad esempio la definizione di coperture del suolo.
Nota Bene: la nozione di Strato Topologico, qui definita, non ha nessun legame con la nozione di "Strato" utilizzato per la organizzazione delle classi. La condivisione del termine è solo la conseguenza dell'esigenza di mantenere continuità con terminologie utilizzate in precedenza.

3 Il Modello Geometrico di GeoUML

3.1 Caratteristiche generali degli oggetti e dei tipi geometrici

Il modello geometrico definisce un insieme di tipi che descrivono le possibili geometrie degli attributi geometrici. Esistono fondamentalmente due categorie di oggetti geometrici:

- le **geometrie primitive:** valori geometrici atomici non ulteriormente divisibili composti da un singolo, connesso ed omogeneo elemento dello spazio: ***Point***, CPC***urve*** e CPS***urface***;
- le **collezioni geometriche:** insiemi di geometrie elementari, suddivisi a loro volta in:
  - collezioni omogenee di punti, di curve, di superfici: ***CXPoint***, ***CXCurve*** e ***CXSurface***
  - collezioni eterogenee: ***Aggregate***

(I nomi utilizzati per i tipi geometrici del GeoUML hanno una motivazione storica, legata a precedenti definizioni del modello geometrico, basate sui complessi dello standard 19107, e non sono necessariamente intuitivi)

Questi tipi fondamentali a loro volta sono suddivisi in base al numero di coordinate:

- tipi definiti nello spazio 2D;
- tipi definiti nello spazio 3D.

L'indicazione della dimensionalità dello spazio di riferimento è indicata in coda al tipo; considerando che i tipi hanno un prefisso GU per indicare che sono tipi GeoUML, otteniamo il seguente insieme di tipi fondamentali:

- **GU_Point2D**: punti singoli bidimensionali
- **GU_CPCurve2D**: curve bidimensionali, con le seguenti specializzazioni:
  - **GU_CPSimpleCurve2D**, per le curve semplici
  - **GU_CPRing2D**, per gli anelli chiusi
- **GU_CPSurface2D**: superfici bidimensionali
- **GU_CXPoint2D**: insiemi di punti singoli bidimensionali
- **GU_CXCurve2D**: insiemi di curve bidimensionali, con le seguenti specializzazioni:
  - **GU_CXRing2D**, insiemi di anelli
  - **GU_CNCurve2D**, insieme di curve connessi (grafo connesso)
- **GU_CXSurface2D**: insiemi di superfici bidimensionali
- **GU_Aggregate2D**: insiemi di punti, curve e superfici bidimensionali
- **GU_Point3D**: punti singoli tridimensionali
- **GU_CPCurve 3D**: curve tridimensionali, con le seguenti specializzazioni:
  - **GU_CPSimpleCurve3D**, per le curve semplici
  - **GU_CPRing3D**, per gli anelli chiusi
- **GU_CPSurfaceB3D**: superfici bidimensionali con frontiera tridimensionale
- **GU_CXPoint3D**: insiemi di punti singoli tridimensionali
- **GU_CXCurve3D**: insiemi di curve tridimensionali, con le seguenti specializzazioni:
  - **GU_CXRing3D**, insiemi di anelli
  - **GU_CNCurve3D**, insieme di curve connessi (grafo connesso)
- **GU_CXSurfaceB3D**: insiemi di superfici bidimensionali con frontiera tridimensionale
- **GU_Aggregate3D**: insiemi di punti, curve e superfici, di cui almeno uno tridimensionale

Nel seguito, per semplificare il riferimento a più tipi geometrici contemporaneamente si utilizza il carattere * ; ad esempio C*curve*D sta per CPcurve2D oppure CPcurve3D oppure CXcurve2D oppure CXcurve3D.

Si noti che rappresentazioni specifiche dei tipi geometrici, quale quella vettoriale, e i metodi di interpolazione, non sono considerate dal GeoUML, ma dai Modelli Implementativi; ad esempio, il tipo concettuale GU_CPCurve è rappresentato nel Modello Implementativo SQL dal tipo Linestring.

3.2 Caratteristiche dei tipi geometrici del GeoUML

In questa sezione si indicano le principali caratteristiche dei tipi elencati sopra.

3.2.1 Proprietà comuni a tutti i tipi o a insiemi di tipi

Tutti gli oggetti geometrici sono definiti in un sistema di riferimento di coordinate.
Dal punto di vista matematico un oggetto geometrico generico è un insieme infinito di punti (ad eccezione dei tipi che descrivono punti isolati):

1. definito in uno **spazio euclideo** $\Re^2$ (oggetti 2D) oppure $\Re^3$ (oggetti 3D), dove la coordinata Z è tipicamente usata per rappresentare l'altitudine;
2. **topologicamente chiuso**, ossia l'insieme di punti che l'oggetto rappresenta include anche i punti che costituiscono la frontiera dell'insieme;
3. **regolare,** ossia l'unione della parte interna dell'insieme di punti e della sua frontiera coincide con l'insieme stesso; quest'ultima proprietà impedisce oggetti anomali come ad esempio, poligoni che abbiano dei tagli nella parte interna o buchi composti da un solo punto.

Le seguenti funzioni si applicano a tutti gli oggetti GeoUML:

- **boundary():** Restituisce la frontiera dell'oggetto geometrico – la definizione di quale sia la frontiera è descritta separatamente per ogni tipo
- **isCycle()**: Ritorna TRUE se l'oggetto geometrico è ciclico (il termine ciclico è spesso sostituito dal concetto di "chiuso su se stesso" quando non esiste la possibilità di confondere quest'ultimo termine col concetto di topologicamente chiuso). Un oggetto ciclico non ha frontiera.
- **isSimple()**: Ritorna TRUE se l'oggetto geometrico è semplice, ossia se non possiede punti di autointersezione o autotangenza.
- **planar()**: Restituisce un oggetto geometrico nello spazio 2D che descrive l'insieme di punti ottenuti dalla proiezione nello spazio 2D dell'insieme di punti rappresentato dall'oggetto. Il tipo di oggetto restituito dipende dallo specifico sottotipo considerato.

Operazioni Insiemistiche: Le operazioni insiemistiche (unione, intersezione, differenza) possono essere applicate su tutti gli oggetti, tuttavia risulta complessa l'interpretazione del risultato in termini di tipi. Ad esempio, l'unione di due curve non è necessariamente una curva, ma può esserlo in casi particolari. Per questo motivo le operazioni insiemistiche ammesse sugli oggetti geometrici vengono definite dopo aver trattato i tipi geometrici.

3.2.2 I tipi GU_Point2D e GU_Point3D (Point)

Un oggetto geometrico dei tipi *GU_Point2D* e *GU_Point3D* è un oggetto zero-dimensionale chiamato "punto" che rappresenta una posizione in uno spazio di coordinate 2D e 3D rispettivamente.

Ridefinizione delle funzioni generali

- **boundary()** - la frontiera di un punto è sempre vuota
- **isCycle()** - è sempre vera
- **isSimple()** - è sempre vera
- **planar()** - se il punto è 3D restituisce il corrispondente 2D, altrimenti è il punto stesso

3.2.3 I tipi GU_CPCurve2D e GU_CPCurve3D

I tipi *GU_CPCurve2D* e *GU_CPCurve3D* definiscono intuitivamente una curva elementare continua ottenuta "muovendo" con continuità un punto nello spazio, dove quindi non sono ammesse biforcazioni e punti di rottura della continuità.

Inoltre ***non sono ammesse autointersezioni su infiniti insiemi di punti*** (cioè sovrapposizioni di porzioni di curva), ma sono ammesse autointersezioni di singoli punti. Esempi di curve elementari corrette sono riportati in Figura 3.2, mentre in Figura 3.3 sono riportati esempi scorretti di curva.

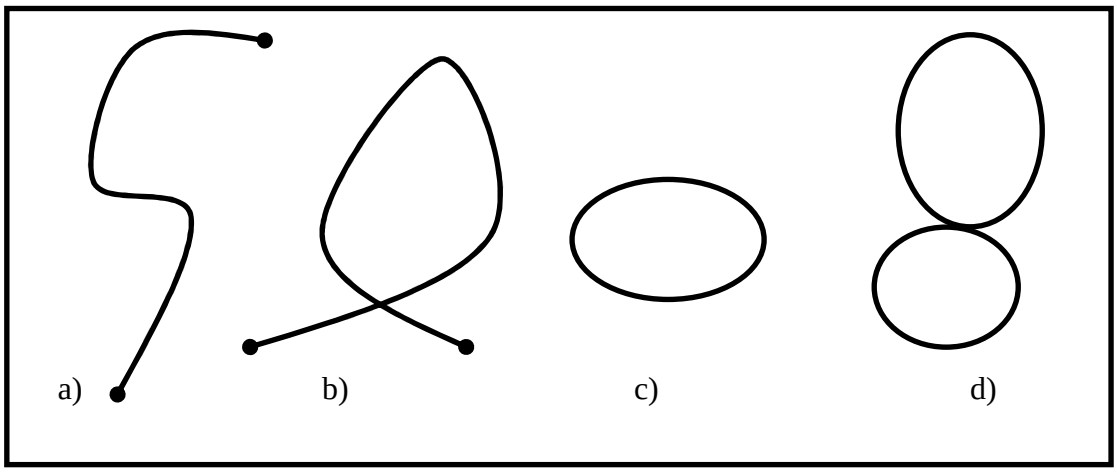

Figura 3.2 - Esempi di curve elementari (*GU_CPCurve2D*).

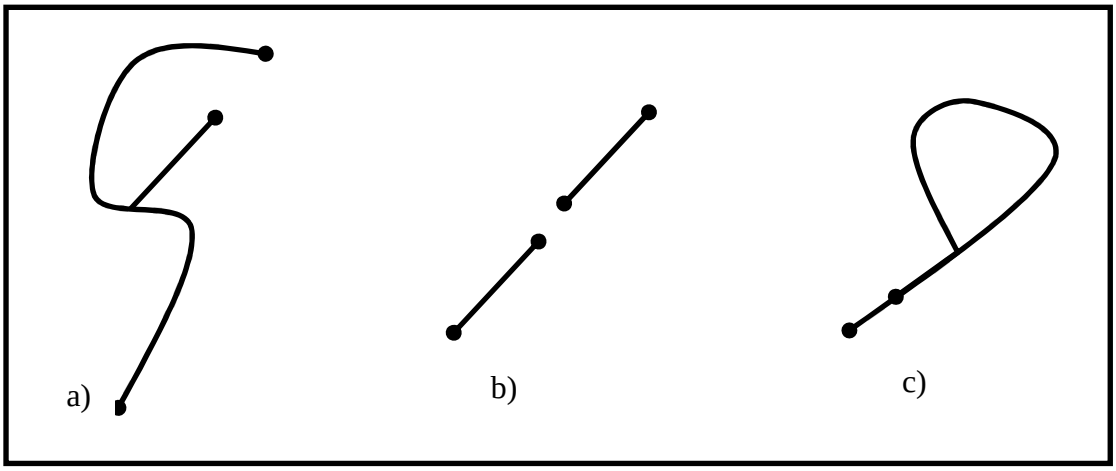

Figura 3.3 – Esempi di oggetti non descrivibili come curve elementari.

Ridefinizione delle funzioni generali

- **boundary()**
  La frontiera di una curva coincide con i due punti estremi nelle curve aperte (casi a) e b) di Figura 3.2), mentre la frontiera non esiste se la curva si chiude su se stessa (casi c) e d) di Figura 3.2).
- **isCycle()**
  Restituisce TRUE se la curva si richiude su se stessa (casi (c) e (d) di Figura 3.2, FALSE altrimenti
- **isSimple()**
  Restituisce TRUE se la curva non passa due volte dallo stesso punto (caso (a) di Figura 3.2) oppure tale punto coincide con i soli estremi della curva (caso (c) di Figura 3.2). Si noti che una curva non semplice può comunque intersecarsi solo in un numero discreto di punti (casi (b) e (d) di Figura 3.2).
- **planar()**
  La proiezione nello spazio 2D di una curva primitiva 3D in generale genera una curva primitiva dello stesso tipo della curva originale, ma in alcuni casi la curva proiettata genera un oggetto di tipo diverso, ad esempio: una curva verticale composta da vertici nei quali cambia solo la coordinata Z genera un

punto nel piano, un anello nel piano XZ genera una curva semplice nel piano, una curva semplice con segmenti che si sovrappongono o si intersecano nella proiezione genera un aggregato di curve per descrivere il grafo nel piano.

3.2.4 I tipi GU_CPSimpleCurve2D e GU_CPSimpleCurve3D

I tipi *GU_CPSimpleCurve2D* e *GU_CPSimpleCurve3D* definiscono una curva semplice (casi (a) e (c) di Figura 3.2).

3.2.5 I tipi GU_CPRing2D e GU_CPRing3D (Composite Ring)

I tipi *GU_CPRing2D* e *GU_CPRing3D* definiscono una curva semplice e chiusa su se stessa, corrispondente al concetto intuitivo di anello (caso (c) di Figura 3.2).

3.2.6 Il tipo GU_CPSurface2D

Un oggetto geometrico di questo tipo corrisponde ad una superficie bidimensionale elementare definita nello spazio 2D. Una superficie elementare è definita da un insieme di anelli di tipo *GU_CPRing2D*: un anello `fe` che rappresenta la frontiera esterna della superficie e un insieme di zero o più anelli `Fi` = $\{fi_1, \ldots, fi_n\}$ che rappresentano le frontiere interne che delimitano gli eventuali buchi della superficie; si noti che poiché un anello non si autointerseca una frontiera non può possedere asole e la superficie non può degenerare ad una curva aperta (la frontiera esterna composta da un solo segmento percorso in un senso e poi in quello inverso).
Una superficie `S` descritta dall'anello esterno `fe` e dall'insieme degli anelli interni `Fi` risulta costituita dall'insieme di punti dello spazio 2D che soddisfano le seguenti proprietà:

1. La superficie `S` è descritta dai punti interni alla frontiera esterna ed esterni ad ogni frontiera interna, e include le frontiere per garantire la chiusura topologica della superficie
2. Tutti i buchi devono essere contenuti nella regione interna definita dalla frontiera esterna e ogni frontiera interna può toccare la frontiera esterna al più in un solo punto;
3. Un buco non può essere contenuto in un altro buco o sovrapporsi ad esso. Inoltre due buchi possono toccarsi al più in un punto
4. La superficie `S` deve avere la sua parte interna connessa, ossia tale che due qualsiasi punti della superficie `S` (escluse le frontiere) sono connessi da una curva che non attraversa la frontiera.

Nelle Figure 3.4 e 3.5 sono mostrati rispettivamente esempi di superfici corrette e non corrette.

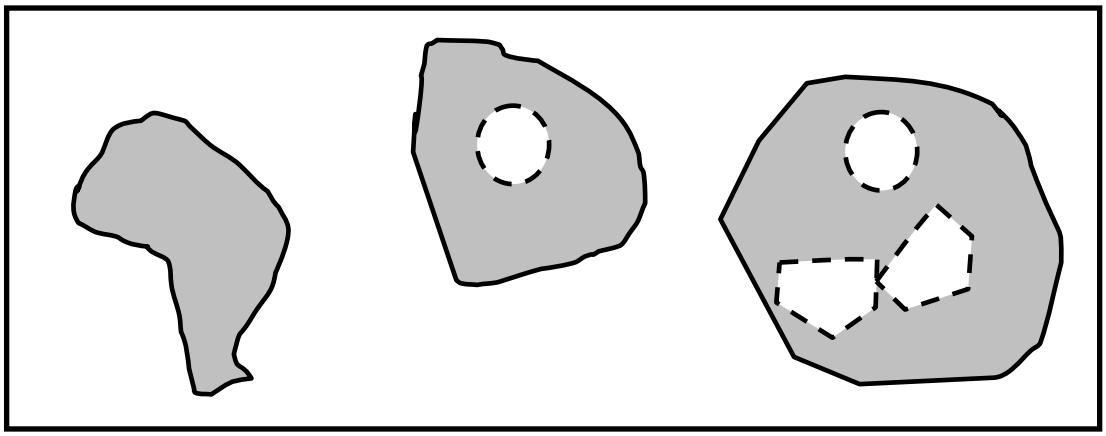

Figura 3.4 – Esempi di superfici (*GU_CPSurface2D*). Le curve tratteggiate rappresentano frontiere interne.

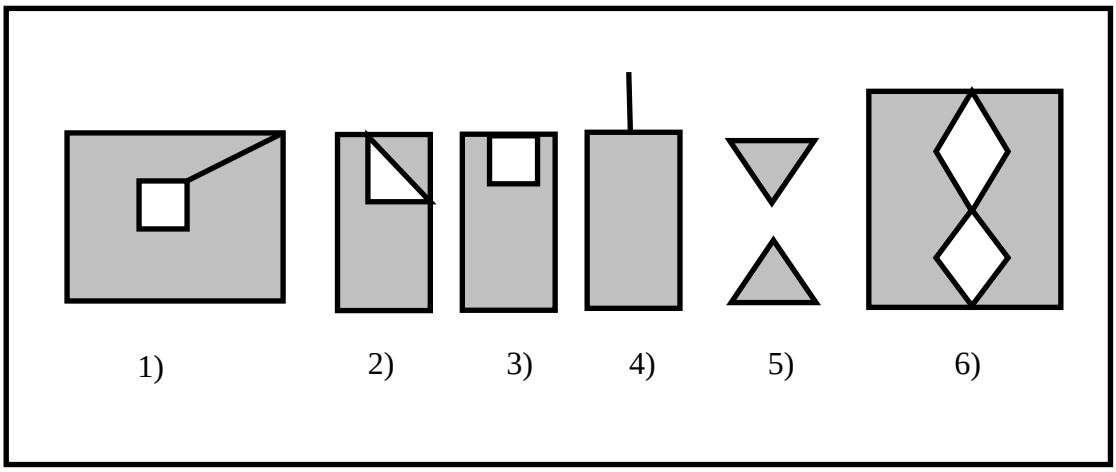

Figura 3.5 – Esempi di geometrie non descrivibili come oggetti del tipo *GU_CPSurface2D*.

Si noti che i poligoni 2, 5 e 6 di Figura 3.5 sono descrivibili come aggregati di due superfici, mentre gli altri richiedono la rimozione di un segmento lineare per essere considerate superfici ammesse.

Ridefinizione delle funzioni generali

- **boundary()**
  restituisce un aggregato del tipo *GU_CXRing2D* i cui componenti sono gli anelli che rappresentano le frontiere esterna e interne della superficie.
- **isCycle()**
  è sempre FALSE, perché una superficie non si può chiudere su se stessa nello spazio 2D.
- **isSimple()**
  è sempre FALSE, perché una superficie non può autointersecarsi in uno spazio 2D

3.2.7 I tipi aggregati generici GU_Aggregate2D e GU_Aggregate3D

I tipi *GU_Aggregate2D* e *GU_Aggregate3D* permettono la definizione di un aggregato, nello spazio 2D e 3D rispettivamente, composto da una collezione di zero o più oggetti geometrici primitivi (anche di tipi diversi) che condividono lo stesso sistema di riferimento che rappresenta quello dell'aggregato. Non sono ammessi aggregati di aggregati. Infine un aggregato generico non impone vincoli alle geometrie dei componenti (possono sovrapporsi e anche coincidere).

L'aggregato generico può contenere oggetti di diversa dimensione e quindi non è possibile associare la dimensione staticamente al tipo come avviene per i suoi sottotipi e pertanto la dimensione dell'aggregato è determinata dall'oggetto di dimensione più grande.

Ridefinizione delle funzioni generali

- **boundary(), isSimple(), isCycle() NON sono definite per gli aggregati** – questa limitazione implica molte restrizioni sulla usabilità degli aggregati
- **planar() -** restituisce un oggetto di tipo *GU_Aggregate2D*

3.2.8 I tipi GU_CXPoint2D e GU_CXPoint3D

Un oggetto geometrico dei tipi *GU_CXPoint2D* e *GU_CXPoint3D* è un aggregato di zero o più punti appartenenti tutti al tipo *GU_Point2D* e *GU_Point3D* rispettivamente.

Ridefinizione delle proprietà ereditate

- **boundary()**
  `self.boundary() = ∅`
- **isCycle()**
  `self.isCycle() = true`
- **isSimple()**
  un insieme di punti è semplice se sono tutti geometricamente disgiunti tra loro.

3.2.9 I tipi GU_CXCurve2D e GU_CXCurve3D (Complex Curve)

Un oggetto dei tipi *GU_CXCurve2D* e *GU_CXCurve3D* è un aggregato mono-dimensionale costituito da una collezione di zero o più curve del tipo *GU_CPCurve2D* e *GU_CPCurve3D* rispettivamente che non devono sovrapporsi tra di loro in modo parziale o completo (duplicazione) al fine di mantenere invariante la proprietà di frontiera dell'aggregato.
Questo tipo definisce oggetti lineari dove sono ammesse biforcazioni e che possono non essere connessi.

Ridefinizione delle funzioni generali

- **boundary()**
  la frontiera di una curva complessa contiene i punti della curva che appartengono alla frontiera di un numero dispari di curve componenti dell'aggregato (regola ***"mod 2 union rule"*** dello standard ISO 19125; per la motivazione di questa regola apparentemente strana si veda lo standard).

  La frontiera della curva di Figura 3.6 a) è costituita dai 4 punti estremi anche nel caso in cui l'aggregato è composto da 4 curve semplici convergenti nel punto di intersezione, viceversa la frontiera della curva di Figura 3.6 b) è costituita dai 3 estremi e dal punto di intersezione interna sia nel caso in cui il punto interno sia frontiera di una sola curva o di tre curve. Per analogia al caso a) di Figura 3.6 anche la frontiera della curva di Figura 3.6 c) è costituita dai soli estremi delle curve componenti, mentre nel caso d) la frontiera è vuota poiché tutti i componenti sono cycle.
- **isCycle()**
  la curva complessa è ciclica se tutte le curve component sono cicliche
- **isSimple()**
  Una CXCurve è semplice se ogni curva componente è semplice e se le curve si toccano tra di loro al più sui punti di frontiera; si noti che questo vincolo impedisce la sovrapposizione della parte interna di due curve componenti, ma anche che una curva tocchi con un proprio punto di frontiera la parte interna di un'altra curva componente.

La Figura 3.6 mostra una CXCurve semplice (caso c), CXCurve non semplici (i casi a e d) con componenti semplici; la CXCurve di Figura 3.6 (caso b) è semplice se è composta da 3 curve che si toccano nel punto interno, mentre non sarà considerato semplice se contiene 2 curve con una che tocca con la propria frontiera la parte interna dell'altra.

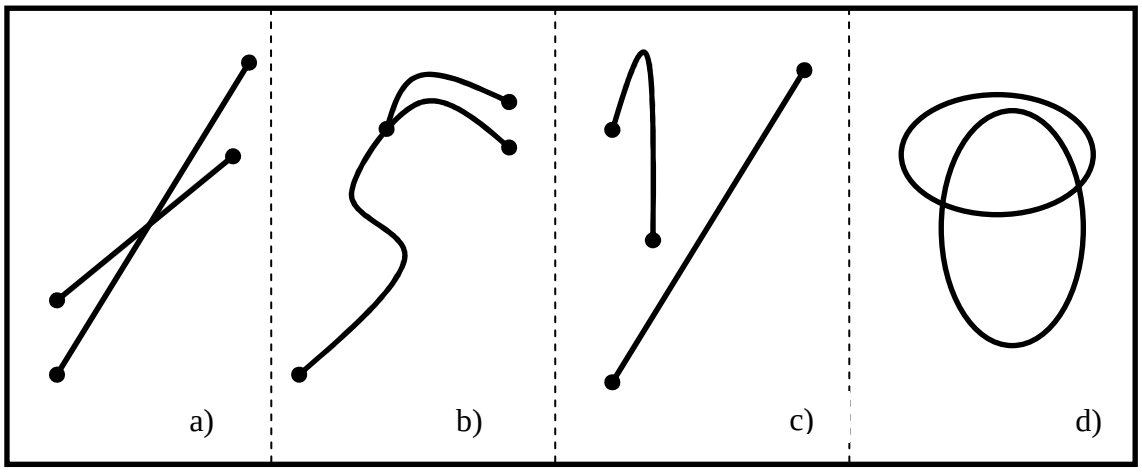

Figura 3.6 – Esempi di aggregati di curve (*GU_CXurve2D*).

Commento

Si noti che NON esiste una corrispondenza biunivoca tra un aggregato di curve inteso come collezione di oggetti geometrici primitivi e l'insieme di punti che lo rappresenta nello spazio poiché lo stesso insieme di punti può corrispondere ad aggregati differenti di oggetti; ad esempio, l'aggregato di figura 3.6.b può essere composto da 2, 3 o più curve primitive. La definizione di frontiera della curva complessa si basa sull'insieme di punti descritto dall'aggregato ed è quindi invariante rispetto alle diverse composizioni di oggetti dell'aggregato che corrispondono allo stesso insieme di punti dello spazio considerato.

3.2.10 I tipi GU_CXRing2D e GU_CXRing3D (Complex Ring)

I tipi *GU_CXRing2D* e *GU_CXRing3D* sono una specializzazione dei tipi *GU_CXCurve2D e GU_CXCurve3D* rispettivamente. Essi permettono la definizione di un aggregato mono-dimensionale costituito da una collezione di zero o più anelli del tipo *GU_CPRing2D* e *GU_CPRing3D* rispettivamente. Si noti che eredita il divieto di sovrapposizione parziale o totale dalla classe *GU_CXCurve* della corrispondente dimensione, ma non vincola ulteriormente le relazioni topologiche possibili tra i componenti.

Ridefinizione delle proprietà ereditate

- **boundary()**
  la frontiera è sempre vuota
- **isCycle()**
  sempre TRUE

3.2.11 I tipi GU_CNCurve2D e GU_CNCurve3D (Connected Curve)

Definizione dei valori possibili

I tipi *GU_CNCurve2D* e *GU_CNCurve3D* sono specializzazioni dei tipi *GU_CXCurve2D* e *GU_CXCurve3D* rispettivamente che impongono alla curva complessa la proprietà di connessione della parte interna: due qualsiasi punti della curva complessa sono connessi da una curva elementare contenuta nella curva complessa.

3.2.12 **Il tipo GU_CXSurface2D (Complex Surface)**

Un oggetto del tipo *GU_CXSurface2D* è una superficie complessa costituita da una collezione di zero o più superfici di tipo *GU_CPSurface2D* che sono disgiunte o che al più possono toccarsi solo in alcuni punti singoli della frontiera (la superficie complessa è quindi in generale un oggetto non connesso). Si noti che l'adiacenza su un tratto della frontiera non è ammesso poiché le due superfici sarebbero rappresentabili con un'unica superficie di tipo *GU_CPSurface2D*.

Ridefinizione delle proprietà ereditate

- **boundary()**
  restituisce un aggregato del tipo *GU_CXRing2D* i cui componenti sono gli anelli che rappresentano le frontiere esterna e interne di tutte le superfici componenti dell'aggregato.
- **isCycle**
  è sempre FALSE
- **isSimple**
  Le singole superfici componenti sono per definizione semplici e inoltre la definizione dei vincoli imposti dal tipo garantiscono la proprietà di semplicità dell'aggregato.

La Figura 3.7 mostra superfici complesse composte da due superfici elementari disgiunte (caso a)), adiacenti in un punto (caso b)) e in due punti (caso(c)). La Figura 3.8 mostra due superfici non rappresentabili come superfici complesse, ma come superficie elementare (caso b)) e come aggregato generico (caso a)) nel quale il tratto lineare è una curva distinta dalle due superfici.

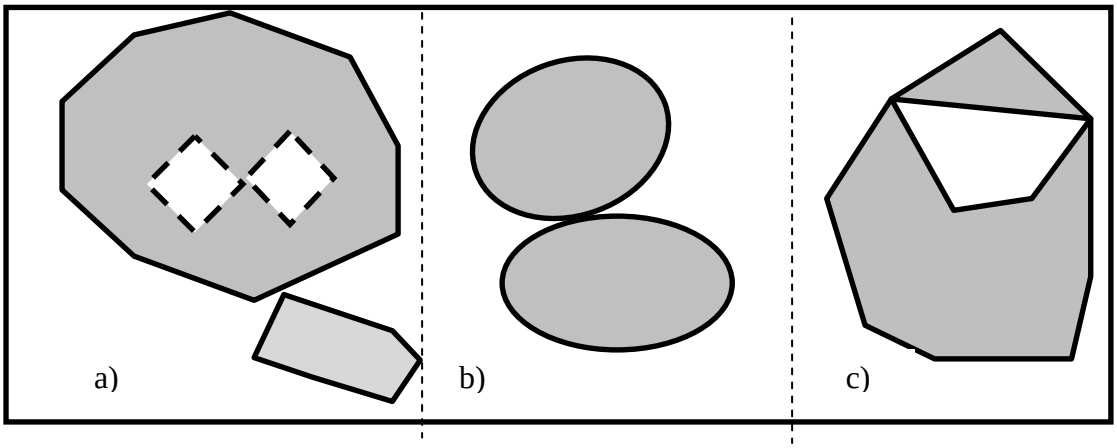

Figura 3.7 – Esempi di superfici complesse (*GU_CXSurface2D*) composte da due superfici elementari

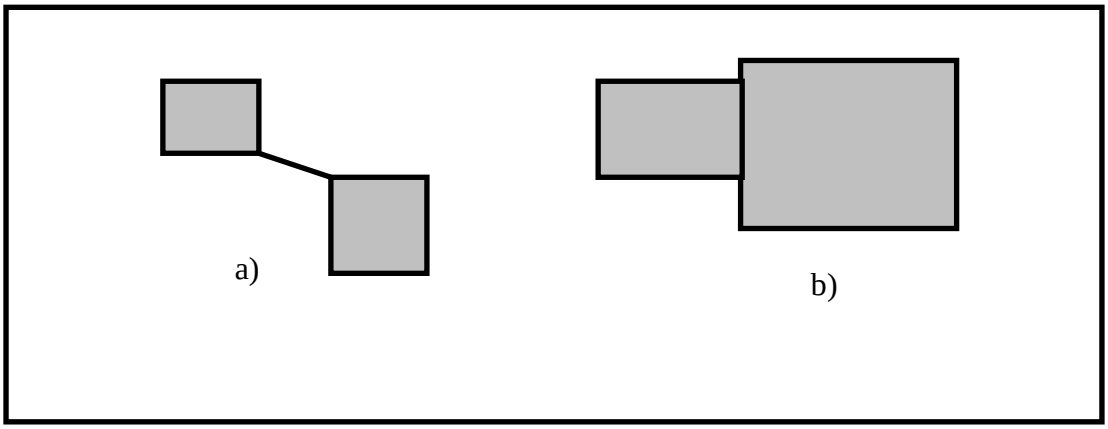

Figura 3.8 – Esempi di geometrie non descrivibili come GU_CXSurface2D

3.2.13 I tipi GU_CPSurfaceB3D/GU_CXSurfaceB3D (Superfici con frontiera 3D)

Questi due tipi descrivono un oggetto geometrico costituito da due attributi geometrici legati tra loro da un vincolo (vedi figura 3.9):

- l'attributo "B3D" che in entrambi i tipi descrive la frontiera reale della superficie nello spazio 3D; tale frontiera può essere composta da più anelli ed è definita tramite un aggregato di anelli del tipo *GU_CXRing3D;*
- l'attributo "superficie" che descrive la proiezione planare della superficie nello spazio 2D tramite una superficie primitiva *GU_CPSurface2D* nel tipo *GU_CPSurfaceB3D* e un aggregato di superfici di tipo *GU_CXSurface2D* nel tipo *GU_CXSurfaceB3D.*

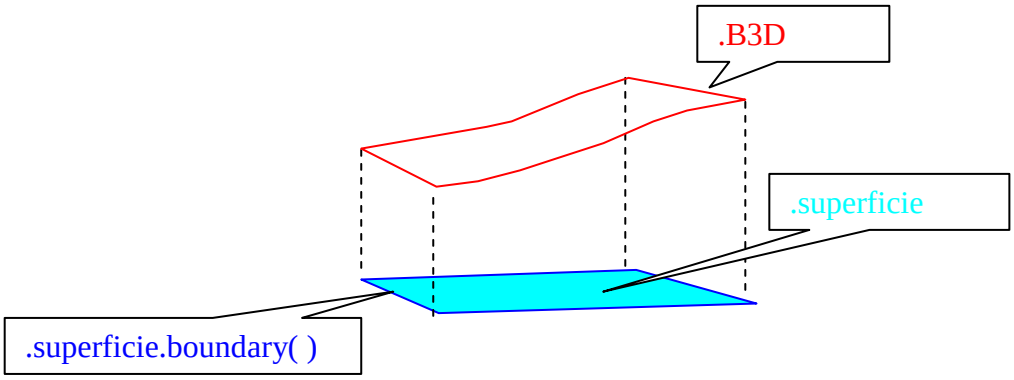

Figura 3.9

Il vincolo seguente lega i due attributi: la frontiera della superficie (.superficie.boundary in figura) deve coincidere con la proiezione planare della frontiera 3D.

Le superfici B3D trovano molte possibili applicazioni, perché permettono di vedere gli oggetti areali considerandoli nello spazio tridimensionale come semplici anelli (cioè senza la determinazione esatta della superficie tridimensionale 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.
Si noti che il tipo è una composizione di altri tipi e quindi non sono associate proprietà al tipo nel suo complesso. Nelle relazioni spaziali e in tutte le espressioni di vincoli è necessario far riferimento agli attributi componenti. Questo implica che quando ci si riferisce ad un attributo di tipo *GU_C*SurfaceB3D* va aggiunto `.superficie` oppure `.B3D` a seconda di quale delle due componenti geometriche si voglia considerare.

Osservazione finale: si ricorda che il tipo "superficieB3D" è definito a livello concettuale e quindi possono esistere Modelli Implementativi che non richiedono di rappresentare esplicitamente le due componenti "superficie" e "B3D".

3.3 Le relazioni topologiche sugli oggetti geometrici.

Per descrivere le relazioni spaziali tra gli oggetti, in particolare quando si vogliono specificare i vincoli di integrità di natura geometrica in uno schema GeoUML, è necessario utilizzare un insieme di relazioni topologiche.
L'insieme fondamentale di relazioni topologiche utilizzato in GeoUML, chiamato REL_{topo}, è composto dalle relazioni Disjoint (DJ), Touch (TC), In (IN), Contains (CT), Equals (EQ) e Overlaps (OV). Questo insieme 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 si applicano ad oggetti dello stesso tipo o di tipi differenti

Le relazioni dell'insieme REL_{topo} non specificano la dimensione del risultato e sono applicabili a tutti gli oggetti geometrici del GeoUML ad eccezione dell'aggregato generico (perchè su questo non è definito il concetto di frontiera); inoltre, nel caso delle superfici B3D devono essere applicate specificando uno degli attributi componenti il tipo. Se le relazioni topologiche sono applicate a tipi aggregati, esse non coinvolgono singolarmente i componenti dell'aggregato, ma si applicano all'insieme di punti dell'aggregato interpretato come il risultato dell'unione degli insiemi di punti dei componenti.
Inoltre le relazioni dell'insieme REL_{topo} sono valutabili solamente tra oggetti definiti nello stesso spazio (2D o 3D); non è ammesso viceversa il confronto tra oggetti definiti in spazi differenti.

Definizione dell'insieme di relazioni REL_{topo}

Le relazioni topologiche sono definite utilizzando i concetti di parte interna, frontiera e parte esterna di un oggetto geometrico; dato un oggetto geometrico a di tipo *GU_Object* si definiscono:

1. pointset di a, PS(a): è l'insieme di punti dell'oggetto
2. parte interna *I(a)*: è l'insieme di punti dell'oggetto che non appartengono alla sua frontiera
3. parte esterna *E(a)*: è l'insieme dei punti dello spazio che non appartengono all'oggetto stesso.

Detti a e b due oggetti geometrici di un qualsiasi tipo ad eccezione dei tipi GU_Aggregate2D, GU_Aggregate3D, GU_CPSurfaceB3D e GU_CXSurfaceB3D, le relazioni topologiche tra a e b sono definite nel modo seguente:

DJ: $a.Disjoint(b) \equiv_{def} (PS(a) \cap PS(b) = \emptyset)$

- la relazione DJ impedisce punti in comune tra oggetti, mentre tutte le altre prevedono che i due oggetti abbiano punti in comune;

TC: $a.Touch(b) \equiv_{def} (I(a) \cap I(b) = \emptyset) \wedge (PS(a) \cap PS(b) \neq \emptyset)$

- se i punti in comune non sono punti interni degli oggetti, allora la relazione è Touch, 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 curva contenuta nella frontiera di una superficie sia un caso possibile per la relazione TC. La relazione TC sarà sempre falsa quando entrambi gli oggetti sono del tipo GU_Point*D;

IN: $a.In(b) \equiv_{def} (PS(a) \cap PS(b) = PS(a)) \wedge (PS(a) \cap PS(b) \neq PS(b)) \wedge (I(a) \cap I(b) \neq \emptyset)$

- il contenimento IN corrisponde intuitivamente al concetto di contenimento insiemistico con l'eccezione del caso in cui un oggetto è contenuto della frontiera dell'altro come citato al precedente punto (relazione Touch) o è uguale all'altro (relazione Equals);

CT: $a.Contains(b) \equiv_{def} b.in(a)$

- è il rovesciamento di IN

EQ: $a.Equals(b) \equiv_{def} (PS(a) \cap PS(b) = PS(a)) \wedge (PS(a) \cap PS(b) = PS(b))$

- richiede l'identità degli insiemi di punti costituenti I due oggetti

OV: $a.Overlaps(b) \equiv_{def} (I(a) \cap I(b) \neq \emptyset) \wedge (PS(a) \cap PS(b) \neq PS(a)) \wedge (PS(a) \cap PS(b) \neq PS(b))$

- nel caso di 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 che hanno in comune). La relazione OV sarà sempre falsa quando uno o entrambi gli oggetti sono punti;

Oltre all'insieme minimo e completo REL_{topo} in GeoUML sono definite anche le relazioni `Intersects` e `Cross` (tra oggetti lineari) derivabili dalle altre ma di uso comune:

INT: $a.Intersects(b) \equiv_{def} \neg\ a.Disjoint(b)$

CR: $a.Cross(b) \equiv_{def} a.Overlaps(b) \wedge (PS(a) \cap PS(b)).dimension()=0$

- la relazione CR è una specializzazione della relazione OV che si applica solo ad oggetti dei tipi GU_C*Curve*D e verifica che la dimensione dell'intersezione sia puntiforme;

Commento

Con riferimento alle relazioni definite si può osservare che:

- le relazioni OV, DJ, CR, EQ, TC sono simmetriche (ad esempio, a.Touch(b) è uguale a b.Touch(a);
- la relazione DJ tra due oggetti è sempre vera se la geometria di almeno uno dei due oggetti è vuota, mentre le altre relazioni topologiche sono sempre false in presenza di almeno una geometria vuota.

Le relazioni dell'insieme REL_{topo} (eccetto Equals e Contains) sono illustrate in Figura 3.10, dove in ogni colonna mostra la stessa relazione topologica applicata a oggetti di tipo diverso e in ogni riga mostra le diverse relazioni si applichino alla stessa coppia di tipi di oggetti.

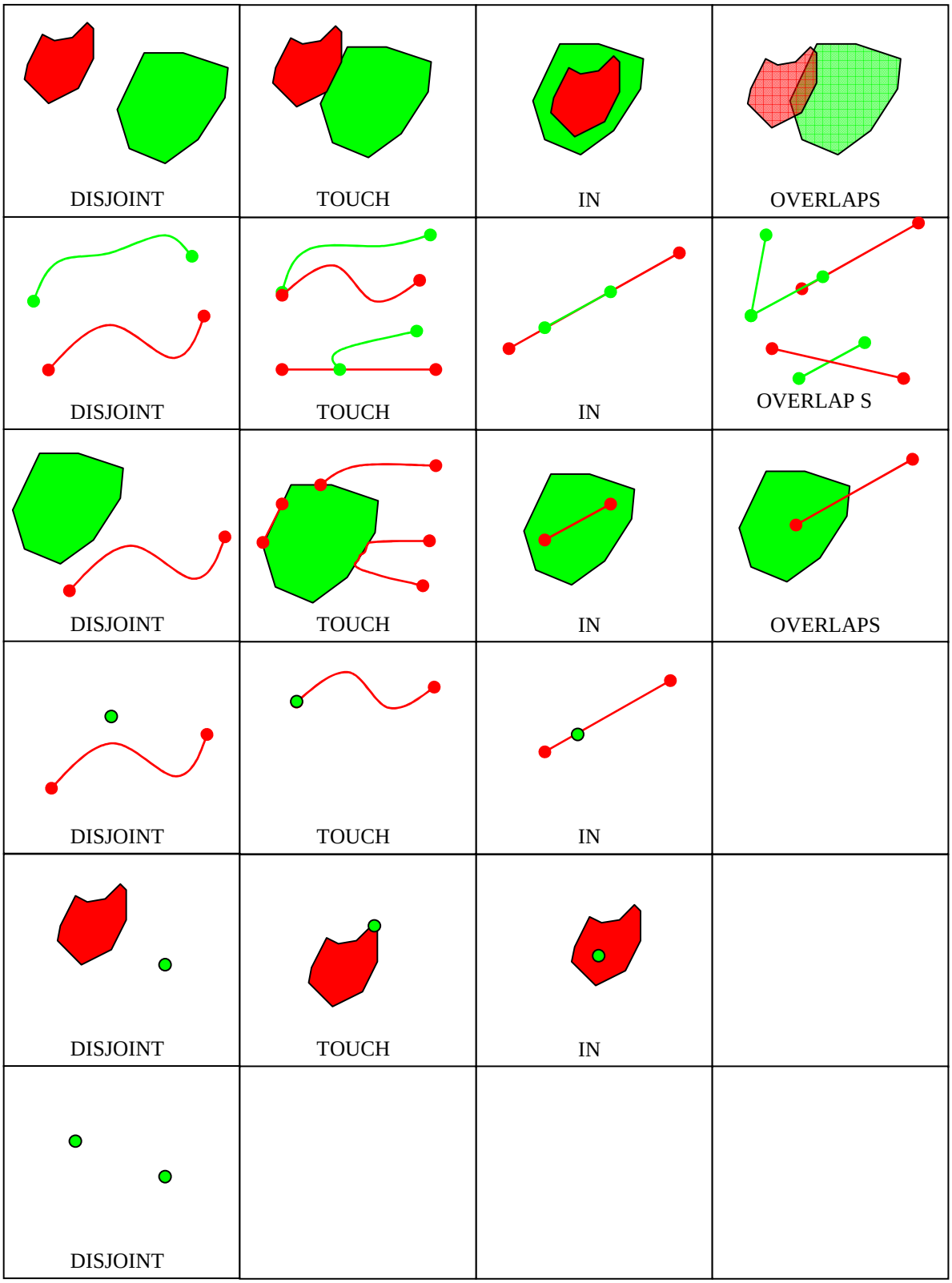

Figura 3.10 – Esempi di relazioni topologiche su diversi tipi di oggetti geometrici.

4 Attributi dipendenti dalle geometrie

4.1 Introduzione

Gli attributi dipendenti dalla geometria sono attributi il cui valore è una funzione dei punti appartenenti ad un attributo geometrico. Esistono tre varianti dell'attributo dipendente dalla geometria: l'attributo a ***tratti***, a ***sottoaree*** e ad ***eventi***. Di seguito si definiscono l'attributo a tratti e a sottoaree dipendenti rispettivamente da una geometria lineare e areale e l'attributo a eventi dipendente da entrambe le geometrie.

Attenzione: gli attributi dipendenti da una componente spaziale devono essere definiti nella stessa classe in cui è definita la componente spaziale stessa; non è quindi possibile definire un attributo dipendente da una componente spaziale ereditata da una superclasse.

4.2 Attributo a tratti

Data una classe contenente un attributo geometrico g di tipo lineare, è possibile definire su g un attributo a tratti che descrive una proprietà che dipende dalla geometria di g.
L'attributo a tratti ha un nome univoco nell'ambito degli attributi della componente spaziale sulla quale è definito, oltre al codice e al codice alfanumerico opzionale.
Il dominio dell'attributo a tratti può essere un dominio di base, un dominio enumerato o un dominio enumerato gerarchico senza attributi aggiuntivi di dominio di base (è escluso il dominio DataType).

ESEMPIO
In figura 4.1 è riportata la definizione della classe elemento stradale (EL_STR), dotata di un attributo geometrico Tracciato di tipo GU_CPCurve3D.
Successivamente sono definiti diversi attributi aTratti su Tracciato. Per ogni attributo a tratti è definito il dominio.
Ad esempio, l'attributo Stato è a tratti su Tracciato e il suo dominio è enumerato embedded, costituito dai valori 01 (esercizio), 02 (in costruzione), 03 (in disuso). Questa definizione significa che in ogni punto del tracciato è definito il valore di questo attributo e che tale valore può variare tra diversi punti del tracciato.

Componenti spaziali della classe						**NC1**	**NC5**
010107101	**EL_STR_TRA**	**Tracciato**	**GU_CPCurve3D - Composite Curve 3D**			**P**	**P**
Attributi di questa componente spaziale						**NC1**	**NC5**
01010701	**EL_STR_TY**	**Tipo**	**Enum**	aTratti su	Tracciato	**P**	**P**
	Dominio (Tipo)					**NC1**	**NC5**
	01	**di tronco carreggiata**				**P**	**P**
	ecc..					**P**	**P**
01010703	**EL_STR_CF**	**Classifica tecnico-funzionale**	**Enum**	aTratti su	Tracciato	**P**	**P**
	Dominio (Classifica tecnico-funzionale)					**NC1**	**NC5**
	01	**autostrada**				**P**	**P**
	02	**strada extraurbana principale**				**P**	**P**
	ecc…					**P**	**P**
01010705	**EL_STR_STA**	**Stato**	**Enum**	aTratti su	Tracciato	**P**	**P**
	Dominio (Stato)					**NC1**	**NC5**
	01	**in esercizio**				**P**	**P**
	02	**in costruzione**				**P**	**P**
	03	**in disuso**				**P**	**P**
01010706	**EL_STR_FON**	**Fondo**	**Enum**	aTratti su	Tracciato	**P**	**P**
	Dominio (Fondo)					**NC1**	**NC5**
	01	**pavimentato**				**P**	**P**
	02	**non pavimentato**				**P**	**P**
01010707	**EL_STR_CL**	**Classe di larghezza**	**Enum**	aTratti su	Tracciato	**P**	**P**
	Dominio (Classe di larghezza)					**NC1**	**NC5**

Figura 4.1

Tratti sul contorno
E' possibile definire un attributo a tratti anche sul contorno di una componente spaziale areale, come nell'esempio seguente.

ESEMPIO
In figura 4.2 è riportata la definizione della classe Area di circolazione veicolare (**AC_VEI**), dotata di un attributo geometrico **Estensione** di tipo GU_SurfaceB3D. L'attributo Tipo_contorno è definito aTratti_sul Contorno3D di estensione e il suo dominio è enumerato con valori 01 (contorno fisico) e 02 (contorno fittizio). Evidentemente questo attributo qualifica i diversi punti della curva 3D che costituisce la frontiera dell'attributo Estensione.

CLASSE: Area di circolazione veicolare (AC_VEI - 010101)

Componenti spaziali della classe							**NC1**	**NC5**
010101101	**AC_VEI_SUP**	**Estensione**	**GU_CPSurfaceB3D - Composite Surface Boundary 3D**				**P**	**P**
Attributi di questa componente spaziale							**NC1**	**NC5**
01010120	**AC_VEI_CON**	**Tipo_contorno**	**Enum**	aTratti sul contorno 3D su	Estensione			
	Dominio (Tipo_contorno)						**NC1**	**NC5**
	01	**contorno fisico**						
	02	**contorno fittizio**						

Figura 4.2

4.3 Attributo a eventi

La definizione di un attributo a eventi segue regole simili a quello a tratti, utilizzando la parola chiave *aEventi su*.
L'attributo a eventi ha un nome univoco nell'ambito degli attributi della componente spaziale sulla quale è definito, oltre al codice e al codice alfanumerico opzionale.
Un evento è un punto della geometria a cui è associato un valore del dominio. Non tutti i punti della geometria hanno associato un evento (cardinalità minima sempre 0) e possono esistere punti a cui sono associati più eventi (cardinalità massima *).

4.4 Attributo a sottoaree

Analogamente all'attributo a tratti, l'attributo a sottoaree associa un valore di un dominio a diverse sottoaree di un attributo geometrico areale.
L'attributo a sottoaree ha un nome univoco nell'ambito degli attributi della componente spaziale sulla quale è definito, oltre al codice e al codice alfanumerico opzionale. Il dominio dell'attributo a sottoaree può essere

un dominio di base, un dominio enumerato o un dominio enumerato gerarchico senza attributi aggiuntivi di dominio base (è escluso qualsiasi dominio DataType).

Da queste definizioni si deduce che:
- una sottoarea corrisponde in generale ad una superficie complessa non connessa;
- nel caso di attributo opzionale può esistere una sottoarea che raccoglie tutti i punti nei quali l'attributo è *nullo*;
- due sottoaree diverse possono sovrapporsi ad esempio, quando una porzione della geometria g condivide due o più valori dell'attributo (solo se la cardinalità massima è *)
- l'unione di tutte le sottoaree definite su g corrisponde alla geometria di g.

Sottoaree e B3D

Nel caso in cui la componente spaziale di un oggetto sia di tipo SurfaceB3D le sottoaree sono definite sulla componente "superficie", e quindi sono delle superfici 2D come rappresentato in figura 4.3

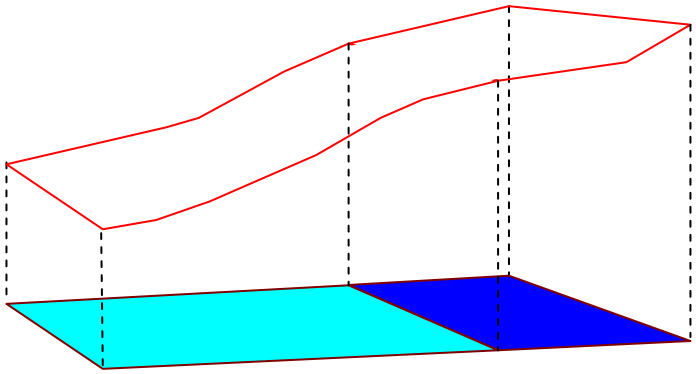

Figura 4.3 – Un oggetto di tipo B3D con 2 sottoaree

ESEMPIO

In Figura 4.4 è riportata la definizione della classe Area di circolazione veicolare (AC_VEI), dotata di un attributo geometrico Estensione di tipo GU_SurfaceB3D e di un attributo Tipo_contorno, come già visto nell'esempio precedente, con aggiunte le definizioni di alcuni attributi aSottoaree su Estensione. Ad esempio, l'attributo a sottoaree Fondo indica che per ogni punto della superficie dell'Estensione è definito se è Pavimentato ppure no.

Si osservi che l'attributo Estensione è di tipo SurfaceB3D, quindi l'attributo a tratti sul contorno di estensione è applicato a una curva 3D, mentre l'attributo a sottoaree di estensione è applicato a una superficie 2D.

CLASSE: Area di circolazione veicolare (AC_VEI - 010101)

							NC1	NC5
Componenti spaziali della classe							**NC1**	**NC5**
010101101	**AC_VEI_SUP**	**Estensione**	**GU_CPSurfaceB3D - Composite Surface Boundary 3D**				**P**	**P**
Attributi di questa componente spaziale							**NC1**	**NC5**
01010120	**AC_VEI_CON**	**Tipo_contorno**	**Enum**	aTratti sul contorno 3D su	Estensione			
	Dominio (Tipo_contorno)						**NC1**	**NC5**
	01	**contorno fisico**						
	02	**contorno fittizio**						
01010101	**AC_VEI_ZON**	**Zona**	**Enum**	aSottoaree su	Estensione		**P**	**P**
	Dominio (Zona)						**NC1**	**NC5**
	01	**tronco carreggiata**					**P**	**P**
	0101	**tronco ordinario**					**P**	**P**
	0102	**rampa/svincolo**					**P**	**P**
	ecc..						**P**	**P**
01010102	**AC_VEI_FON**	**Fondo**	**Enum**	aSottoaree su	Estensione		**P**	
	Dominio (Fondo)						**NC1**	**NC5**
	01	**pavimentato**					**P**	
	02	**non pavimentato**					**P**	
01010103	**AC_VEI_SED**	**Sede**	**Enum**	aSottoaree su	Estensione		**P**	**P**
	Dominio (Sede)						**NC1**	**NC5**
	01	**a raso**					**P**	**P**
	02	**su ponte/viadotto/cavalcavia**					**P**	**P**
	ecc…						**P**	**P**

Figura 4.4

5 Vincoli di integrità spaziale

5.1 Introduzione

I vincoli di integrità spaziale esprimono condizioni che devono essere soddisfatte dalle componenti spaziali delle istanze delle classi alle quali fanno riferimento.
Esistono due famiglie di vincoli di integrità spaziale: i vincoli topologici e i vincoli di composizione.

5.2 Vincoli topologici

I vincoli topologici utilizzano le relazioni topologiche definite su tutti i tipi geometrici GeoUML, esclusi gli aggregati generici, che quindi non possono essere coinvolti nei vincoli.

Esistono 3 categorie di vincoli topologici: quello esistenziale, quello universale e quello su unione.
Per ognuna di queste categorie esistono una versione di base e diverse varianti che permettono di descrivere condizioni più articolate.

5.2.1 Vincolo topologico esistenziale di base

Consideriamo l'esempio di figura 5.1, che mostra la definizione della classe "area a servizio dei trasporti" (SV_TRA). Alla fine di tale dichiarazione è definito un vincolo topologico che richiede che ogni area di questo tipo sia accessibile, cioè sia adiacente a un'area stradale. Formalmente il vincolo è definito nel modo seguente:

SV_TRA.Estensione **(TC)** esiste **AR_STR**.Estensione.*superficie*

Gli aspetti che caratterizzano il vincolo sono:

- la ***classe vincolata*** (SV_TRA) e il suo attributo geometrico (Estensione),
- la ***classe vincolante*** (AR_STR) e il suo attributo geometrico (Estensione),
- la ***relazione topologica*** (TC).

Il vincolo si legge nel modo esguente:

- per ogni istanza della classe vincolata (SV_TRA) deve esistere (*esiste*) almeno una istanza della classe vincolante (AR_STR) tale che
- le corrispondenti componenti spaziali SV_TRA.Estensione e AR_STR.Estensione.superficie
- siano tra loro in relazione topologica di adiacenza (TC).

CLASSE <<ABSTRACT>>: **Area a servizio dei trasporti** **(SV_TRA - 100181)**

SUPERCLASSE Disjoint complete DI [SV_STR, SV_ATR]

	Componenti spaziali della classe				NC1	NC5
	100181101	SV_TRA_EXT	Estensione	GU_CXSurface2D - Complex Surface 2D	P	P

Vincoli

Adiacenza con area stradale

Ogni area a servizio del trasporto deve essere accessibile e quindi risultare adiacente ad un'area stradale

SV_TRA.Estensione **(TC)** esiste **AR_STR**.Estensione.*superficie*

Figura 5.1

5.2.2 Regole generali per la formulazione dei vincoli

Per tutti i vincoli formulabili in GeoUML valgono le seguenti regole generali:

- **Unicità dello spazio per l'applicazione delle relazioni**: i tipi di tutte le geometrie coinvolte in un vincolo devono appartenere allo stesso spazio (2D o 3D) in quanto le relazioni topologiche non sono definite tra oggetti appartenenti a spazi differenti (in alcuni casi per soddifare questa regola vedremo che si può usare la funzione ***planar***() che cambia lo spazio di riferimento di una geometria);
- **Disgiunzione di relazioni topologiche**: la relazione topologica del vincolo può essere sostituita in generale da una disgiunzione di relazioni topologiche elementari, cioè da diverse relazioni poste in OR logico tra loro (ad esempio "DJ oppure TC", indicato come DJ|TC);
- **Gerarchie di ereditarietà:** in presenza di gerarchie di ereditarietà la definizione di un vincolo tra due classi implica che sia implicitamente applicato anche a tutte le sottoclassi (dirette e indirette) della classe vincolata e che per ogni loro oggetto siano coinvolti per la verifica gli oggetti della classe vincolante e quelli di tutte le sue sottoclassi (dirette e indirette). Inoltre la definizione del vincolo può far riferimento alle proprietà dirette delle classi coinvolte o a quelle ereditate dagli antenati delle classi vincolata e vincolanti;
- **Autoanello**: se un vincolo coinvolge una classe sia come classe vincolata che come classe vincolante (autoanello), l'insieme di oggetti vincolanti applicati per soddisfare il vincolo su un certo oggetto O è costituito da tutti gli oggetti della classe ***escluso l'oggetto O stesso***;
- **Superfici con frontiera 3D**: se uno o entrambi gli attributi geometrici coinvolti sono di tipo GU_C*SurfaceB3D è necessario specificare la componente considerata nel vincolo, ossia l'attributo *B3D* o l'attributo *superficie*;
- **Abbreviazioni**: per semplificare la formulazione dei vincoli si può usare il codice alfanumerico della classe al posto del nome completo e inoltre il prefisso della classe può essere omesso davanti ai nomi degli attributi quando la classe è quella del contesto corrente;
- **Limiti nell'uso di attributi**: i vincoli non possono far riferimento agli attributi di un'associazione e agli attributi aggiuntivi di tipo base presenti nel dominio enumerato gerarchico di un attributo.

5.2.3 **Varianti del vincolo topologico esistenziale di base**

Le varianti del vincolo topologico esistenziale definite nel seguito possono essere applicate anche in combinazione tra loro.

5.2.3.1 Vincolo topologico esistenziale con selezioni – uso delle funzioni planar e boundary

Questa variante permette di selezionare gli oggetti delle classi coinvolte nel vincolo.
Una selezione sulla classe vincolata limita gli oggetti che devono soddisfare il vincolo, rendendo il vincolo più debole; una selezione sulla classe vincolante riduce gli oggetti che possono essere utilizzati per soddisfare il vincolo, rendendolo più forte.
Invece di usare direttamente una componente spaziale nel vincolo, si può utilizzare la sua proiezione planare (PLN) oppure la sua frontiera (BND).

ESEMPIO
In figura 5.2 è riportato un vincolo sulla classe "Accesso o passo carraio" (ACC_PC). La struttura generale del vincolo è simile a quella dell'esempio precedente, però la classe vincolata ha una selezione, per cui il vincolo si applica solamente alle istanze di ACC_PC il cui tipo è "accesso esterno diretto".
Il vincolo richiede che la proiezione planare della posizione di un accesso "ACC_PC.Posizione.**PLN"** sia contenuta o adiacente alla componente .*superficie* dell'ingombro al suolo di un corpo edificato (l'ingombro al suolo è infatti di tipo surfaceB3D).

Vincoli

Posizione su contorno edifici per accesso esterno diretto

Per ogni accesso esterno diretto deve esistere un Corpo edificato (Edificio o Edificio minore) tale per cui la proiezione planare della posizione dell'accesso sia contenuta sul boundary o al più all'interno della superficie dell'ingombro al suolo

(tipo = "accesso esterno diretto") **ACC_PC**.Posizione.*PLN* **(IN| TC)** esiste **CR_EDF**.Ingombro al suolo.*superficie*

Figura 5.2

Interpretazione del valore nullo nella valutazione dei vincoli
Il GeoUML ammette l'opzionalità dei valori degli attributi/ruoli di una classe e modella l'assenza di valore attraverso il concetto di ***nullo***;
La definizione dei vincoli implica la possibile selezione di oggetti e/o tratti (eventi, sottoaree) e la possibile applicazione della funzione boundary che possono produrre un insieme vuoto di oggetti e/o di geometrie).
La semantica dei vincoli rispetto a questo problema è governata dalle seguenti regole:

- qualsiasi funzione (ad esempio, boundary()) applicata ad un valore nullo produce un valore nullo come risultato;
- l'interpretazione delle selezioni applicate alle classi (ai tratti,…) sottoposte ad un vincolo fa riferimento alla semantica standard dell'SQL92 anche per l'interpretazione del valore nullo; si ricorda che la presenza di un valore nullo durante la valutazione di una condizione può generare un risultato "unknown" (non è vero e non è falso); in tali casi l'SQL forza a falso la valutazione e non seleziona l'oggetto. La stessa condizione si verifica nella selezione dei tratti (eventi, sottoaree);

- le geometrie (della classe vincolata e di quelle vincolanti) concorrono alla valutazione di un vincolo solo se contengono un valore non nullo; un vincolo stabilisce pertanto una condizione che deve essere soddisfatta dalle sole geometrie realmente disponibili (anche vuote) all'atto della valutazione del vincolo.

5.2.3.2 Vincolo topologico collegato ad una associazione

Con questa variante si considerano, ai fini del soddisfacimento del vincolo, solo gli oggetti della classe vincolante che sono legati all'oggetto della classe vincolata attraverso un'associazione specificata nello schema.

ESEMPIO
In figura 5.3 è mostrato un vincolo relativo alla classe toponimo stradale (TP_STR): il tracciato (in proiezione planare) di un toponimo stradale comunale deve essere contenuto nel territorio (Boundary compreso) del comune di pertinenza (definito in base al ruolo CmdiTp).
Nella formulazione di questo vincolo la variante rispetto ai vincoli precedenti è espressa nella parte relativa alla classe vincolante, la classe Comune in questo caso, che non viene indicata esplicitamente ma riferita attraverso l'espressione "TP_STR.Cmditp", che indica le istanze di Comune collegate al Toponimo con il ruolo Cmditp (in base ai vincoli di cardinalità deve essere una sola istanza, in questo caso).

Contenimento tracciato di toponimo stradale nel proprio territorio comunale

Il tracciato (proiezione planare) di un toponimo stradale comunale deve essere contenuto dal territorio (Boundary compreso) del comune di pertinenza (definito in base al ruolo CmdiTp)

TP_STR.Tracciato.*PLN* **(IN)** esiste **TP_STR**.Cmditp.Estensione

Figura 5.3

5.2.3.3 Vincolo su attributi a tratti o a sottoaree

Il vincolo può anche fare riferimento alla geometria di un attributo a tratti, a tratti sul contorno, a eventi o a sottoaree.
Per far questo nel vincolo si utilizza una delle 3 seguenti funzioni, che possono contenere una condizione di selezione e nella quali A deve essere sostituito da un attributo a tratti, eventi o sottoaree:

- ***TrattiDi_A***(*condizione-di-selezione*): restituisce l'insieme dei tratti di A che soddisfano la condizione
- ***EventiDi_A***(*condizione-di-selezione*): restituisce l'insieme dei punti evento di A che soddisfano la condizione
- ***SottoareeDi_A***(*condizione-di-selezione*): restituisce l'insieme delle sottoaree di A che soddisfano la condizione

ESEMPIO
In figura 5.4 è riportato un vincolo relativo alla classe PONTE contenente sia una normale selezione sulla classe vincolata, sia una selezione sulle sottoaree dell'attributo a sottoaree *Sede* della classe vincolante Area di circolazione veicolare (AC_VEI).

Contenimento sedi aree di circolazione

Ogni sede di ponte con uso stradale-autostradale deve contenere la corrispondente sede di area di circolazione veicolare

(Sup_sede.uso = "autostradale" **OR** Sup_sede.uso = "stradale") **PONTE**.Sup_sede.*superficie* **(CT)** esiste **AC_VEI**.*SottoareeDi*_Sede (Sede = "su ponte/viadotto/cavalcavia")

Figura 5.4

La condizione iniziale seleziona nella classe vincolata PONTE le istanze che hanno la superficie di sede con uso stradale o autostradale *(Sup_sede.uso = "autostradale" OR Sup_sede.uso = "stradale").*
La condizione vincolante fa riferimento all'esistenza, in un'area veicolare, di almeno una sottoarea dell'attributo a sottoaree Sede che soddisfi la condizione voluta; infatti, data una generica AC_VEI, la funzione

SottoareeDi_Sede (Sede = "su ponte/viadotto/cavalcavia")

restituisce le sottoaree che soddisfano la condizione.

5.2.4 **Vincolo topologico su unione**

Il vincolo topologico su unione fa riferimento all'unione geometrica delle componenti spaziali di tutte le istanze della classe vincolante, invece di richiedere l'esistenza di una singola istanza che soddisfi il vincolo.
In altri termini, la relazione topologica viene verificata rispetto alla geometria che si ottiene dall'unione dei valori delle componenti spaziali di tutte le istanze della classe vincolante.
Anche per il vincolo topologico su unione esistono le varianti presentate per il vincolo topologico esistenziale, vale a dire: la versione con selezione, la versione che si riferisce al boundary() e planar(), quella che si riferisce ad un'associazione e quella che si riferisce ai tratti. e sottoaree.

ESEMPIO

La definizione di figura 5.5, relativa alla classe toponimo stradale (TP_STR), indica che la frontiera del tracciato di un toponimo, cioè i suoi 2 punti terminali, devono appartenere (IN) all'unione delle giunzioni stradali.

Delimitazione tracciato analitico con giunzioni stradali

Il boundary del tracciato analitico di ogni toponimo stradale deve coincidere con un insieme di giunzioni stradali

TP_STR.Tracciato.*BND* **(IN)** unione **GZ_STR**.Posizione

Figura 5.5

5.2.5 **Vincolo topologico universale**

Il vincolo topologico universale richiede che la relazione topologica sia presente tra l'oggetto vincolato e ***tutte*** le istanze della classe vincolante.
Esistono le varianti presentate per il vincolo topologico esistenziale, vale a dire: la versione con selezione, con le funzioni boundary e planar, con i tratti e quella che si riferisce ad un'associazione.
Il vincolo topologico universale è usato quasi esclusivamente con la relazione spaziale Disjoint, eventualmente in disgiunzione con la relazione spaziale Touch.
ATTENZIONE: Un caso particolare è costituito dai ***vincoli universali nei quali la classe vincolata coincide con la classe vincolante***: in questi casi nella valutazione di un'istanza O della classe vincolata si deve fare riferimento alle altre istanze della stessa classe, intesa come vincolante, ma ***escludendo l'istanza O stessa***, perché l'unica relazione spaziale di un oggetto con se stesso è Equals.

ESEMPIO

Questo esempio richiama la regola generale appena citata. Un Comune deve essere in relazione DJ|TC con tutte le altre istanze della stessa classe (ma non ovviamente con se stesso).

Disgiunzione-adiacenza dei comuni

Non devono esistere situazioni di sovrapposizione tra i Comuni, ma al più di adiacenza

COMUNE.Estensione **(DJ| TC)** perOgni **COMUNE**.Estensione

Figura 5.6

5.2.6 **Disgiunzione di vincoli topologici**

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

5.3 Vincoli di composizione (vincoli part_whole)

Questa categoria di vincoli è costituita da un vincolo fondamentale, il vincolo di composizione (parola chiave: *compostoDa*), e da un vincolo derivato, il vincolo di partizione (parola chiave: *partizionato*).
Il vincolo di partizione è derivato in quanto esprimibile tramite il vincolo di composizione e alcuni vincoli topologici leggermente modificati. Dato che la combinazione di vincoli topologici modificati ha una sua utilità specifica, viene definito un vincolo ad hoc, detto di appartenenza disgiunta (parole chiave: *dj_IN* e *qdj_IN*), per esprimere tale combinazione.

Anche per i vincoli di composizione sono applicabili alcune delle varianti già introdotte per i vincoli topologici: è possibile utilizzare selezioni, riferire il vincolo alla frontiera o alla proiezione planare del valore geometrico, riferire il vincolo alla geometria di un attributo a tratti, a eventi o a sottoaree e legarlo ad associazioni.

5.3.1 Vincolo di composizione

Il vincolo di composizione stabilisce che per ogni oggetto X della classe vincolata l'attributo geometrico sia *uguale* all'unione degli attributi geometrici *di uno o più oggetti* della classe vincolante.
Nel caso in cui il vincolo sia collegato ad un'associazione il vincolo è più stringente, perché richiede che per ogni oggetto X della classe vincolata l'attributo geometrico sia uguale all'unione degli attributi geometrici *di tutti gli oggetti della classe vincolante collegati a X nell'associazione.*
Il vincolo di composizione è di natura esistenziale: infatti esso richiede che, dato l'attributo geometrico G di un oggetto della classe vincolata (classe composta), esistano nella classe vincolante (classe componente) un numero intero di istanze le cui componenti spaziali, in unione geometrica tra loro, siano uguali a G.

Il vincolo di composizione impone che le geometrie delle classi vincolata e vincolanti siano tutte della stessa dimensione (ad esempio, solo curve o solo superfici) perché non è possibile comporre un oggetto di dimensione diversa da quella dei componenti (ad esempio, una superficie non può essere ottenuta tramite l'unione di curve).

ESEMPIO
Il seguente vincolo (figura 5.7) richiede che per ogni toponimo stradale (TP_STR) esista un insieme di elementi stradali (EL_STR) in modo che l'unione geometrica dei loro tracciati sia uguale al tracciato del toponimo stesso.

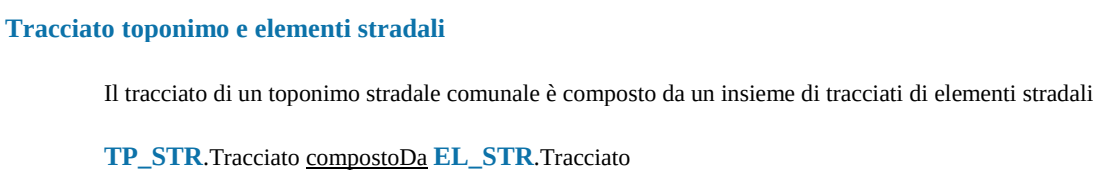

Figura 5.7

5.3.2 Il vincolo di appartenenza

I seguenti due vincoli di appartenenza combinano il contenimento topologico (IN) con la disgiunzione degli elementi nel modo seguente:

- il vincolo di appartenenza con disgiunzione (*dj-IN*) richiede che ogni istanza della classe vincolata sia contenuta geometricamente nella classe vincolante (vincolo topologico IN) e che sussista, tra le istanze della classe vincolata che sono contenute nella stessa istanza di classe vincolante, la relazione

Disjoint oppure di *Touch* ristretta al caso in cui esista solo l'intersezione di frontiera con frontiera (nota: la relazione di *Touch* ammette anche l'intersezione di frontiera e parte interna);

- il vincolo di appartenenza quasi-disgiunta (*qdj-IN*) vale solo per oggetti geometrici di tipo GU_C*Curve*D (incluse le specializzazioni) e consente che tra le istanze della classe vincolata sussista oltre alle relazione *Disjoint* e *Touch* anche la relazione *Cross*.

In realtà questo vincolo è raramente usato direttamente, ma insieme a quello di composizione per generare i vincoli di partizione.

5.3.3 Il vincolo di partizione

Il vincolo di partizione (*partizionato*) esprime il concetto matematico di partizione, cioè un oggetto della classe vincolata deve essere "suddiviso" in oggetti della classe vincolante disgiunti tra loro. Inoltre, non possono esistere oggetti della classe vincolante indipendenti, cioè che non appartengono a nessun oggetto della classe vincolata.
Tale concetto equivale a una combinazione di un vincolo di composizione con un vincolo di appartenenza disgiunta (nelle due versioni *dj* e *qdj*; in quest'ultimo caso il vincolo usa la parola chiave *q-partizionato*)

ESEMPIO
La suddivisione della Estensione (territorio) di una Provincia nelle Estensioni dei Comuni che le appartengono è un esempio classico di partizione (figura 5.8).

CLASSE: Provincia (PROVIN – 090105)

Attributi					
Attributi della classe				**NC1**	**NC5**
Componenti spaziali della classe				**NC1**	**NC5**
090105101	**PROVIN_EXT**	**Estensione**	**GU_CXSurface2D – Complex Surface 2D**	**P**	**P**

Ruoli

Cmdipv

Definisce di quali comuni è composta una specifica provincia

Cmdipv [1..*] : **COMUNE** inverso **Pvdicm** [1]

Vincoli

Partizione del territorio provinciale nei comuni

l territorio della specifica provincia è partizionato nel territorio dei comuni che la compongono, tra loro disgiunti; viceversa ogni territorio comunale deve appartenere al territorio della provincia di cui è parte

PROVIN.Estensione partizionato **PROVIN**.Cmdipv.Estensione

Figura 5.8

Questo esempio inoltre mostra la combinazione del vincolo di partizione con il riferimento ad un'associazione, perché si vuole che una provincia sia partizionata nell'insieme dei comuni che le sono collegati nell'associazione "Cmdipv"; ciò è possibile perchè le varianti introdotte per il vincolo topologico esistenziale possono essere applicate anche ai vincoli di composizione.

5.3.4 Vincoli di composizione con più classi vincolanti

Esiste una variante dei vincoli *compostoDa*, *partizionato* oppure *q-partizionato* per permettere che facciano riferimento all'insieme dei valori di attributi geometrici di diverse classi vincolanti. Le diverse classi con i rispettivi attributi devono in questo caso essere elencate tra parentesi come mostrato nel seguente esempio.

ESEMPIO

Gli strati topologici di copertura del suolo sono un tipico esempio di questi vincoli. La copertura del suolo è definita da una partizione del territorio in oggetti che sono selezionati da diverse classi, come mostrato in figura 5.9. Si noti che su ciascuna classe vincolante si applicano in questo esempio selezioni e riferimenti ad attributi a sottoaree.

STRATO TOPOLOGICO: Copertura del suolo destinata alla mobilita' e ai trasporti (CP_TRA - 800101)

Definizione

Raggruppa tutte le porzioni di "suolo" destinate alle aree di circolazione di varia tipologia (veicolare, pedonale, ciclabile, area di viabilità mista secondaria) e le sedi per il trasporto su ferro

Tipo Geometrico **GU_CXSurface2D - Complex Surface 2D**

Vincoli

Copertura partizionata nelle opportune aree di circolazione

definisce le regole di disgiunzione o al più adiacenza tra le porzioni di suolo destinate alle aree di circolazione ed alle sedi di trasporto su ferro

NB: l'attributo Posizione dell'area di circolazione ciclabile deve diventare di classe e non a sottoaree

CP_TRA.geometria partizionato (**AC_VEI**.*SottoareeDi*_Sede (Sede <> "su ponte/viadotto/cavalcavia" **AND** Sede <> "in galleria") , (posizione = "non in sede stradale") **AC_PED**.*SottoareeDi*_Sede (Sede <> "su ponte/passerella pedonale" **AND** Sede <> "in galleria/sottopassaggio pedonale") , **AC_CIC**.*SottoareeDi*_Posizione (Posizione = "isolata") , **AR_VMS**.*SottoareeDi*_Sede (Sede <> "su guado" **AND** Sede <> "su ponticello" **AND** Sede <> "sotterraneo"))

Figura 5.9

6 Gestione delle superfici collassate

Le componenti geometriche di tipo GU_C*Surface* di alcune classi possono essere collassabili. Ciò significa che si ammette la possibilità che alcune (o tutte le) istanze di una classe abbiano come geometria di un attributo di tipo GU_C*Surface* non solo una superficie, ma una curva, un punto o una combinazione di punti curve e superficie.
La causa del collassamento è legata alle dimensioni dell'oggetto in funzione dell'accuratezza metrica prevista per la scala di rilievo.
Nello Schema Concettuale si indicano le componenti spaziali delle classi che ammettono il collassamento, in modo da poter introdurre a livello di Modello Implementativo meccanismi particolari per la gestione delle geometrie collassate solo quando effettivamente servono.

Si precisa che gli attributi a sottoaree su superfici collassate non assumono valore sulla parte di geometria che collassa a curve o punti. Ovviamente se il collassamento è completo, gli attributi a sottoaree non vengono rappresentati affatto.

In questo capitolo si definisce a livello del modello concettuale quali forme può assumere la geometria di una superficie collassata e si definisce il suo comportamento quando questa partecipa ai vincoli.

6.1 Proprietà e valori ammessi

Ogni superficie collassata **Sc** di tipo GU_CPSurface2D, GU_CXSurface2D, GU_CPSurfaceB3D e GU_CXSurfaceB3D viene rappresentata da tre componenti:

- la porzione degenerata a curva **Sc.curve** di tipo GU_CXCurve2D o GU_CXCurve3D,
- la porzione degenerata a punto **Sc.point** di tipo GU_CXPoint2D o GU_CXPoint3D,
- la porzione non degenerata **Sc.surface** di tipo GU_CXSurface2D.

L'insieme delle 3 componenti eterogenee nel tipo non viene assimilato ad un aggregato generico (privo di frontiera e quindi di relazioni topologiche) e preserva le seguenti proprietà dei rispettivi tipi originali:

- di essere un oggetto regolare, pertanto i buchi interni alla superficie non possono mai collassare a curve o punti (tali collassamenti provocano l'eliminazione del buco), altrimenti si creerebbero dei tagli o punture non ammessi negli oggetti regolari;
- dimension(), isCycle() e IsSimple() conservano i valori della superficie originale anche qualora la superficie collassata sia composta solo da curve e/o punti;
- CoordinateDimension() dei componenti Sc.curve e Sc.point coincidono con quello della componente Sc.surface.

La frontiera e la parte interna di queste superfici vengono ridefinite attraverso una reinterpretazione delle componenti di tipo curva e punto. Considerando l'insieme dei punti che descrive una superficie non collassata si può interpretare il collassamento come una funzione suriettiva tra la superficie non collassata e quella collassata come mostrato dal collassamento della superficie di Figura 6.1.a nella superficie della Figura 6.1.b dove molti punti diversi della superficie non collassata convergono allo stesso punto in quella collassata.

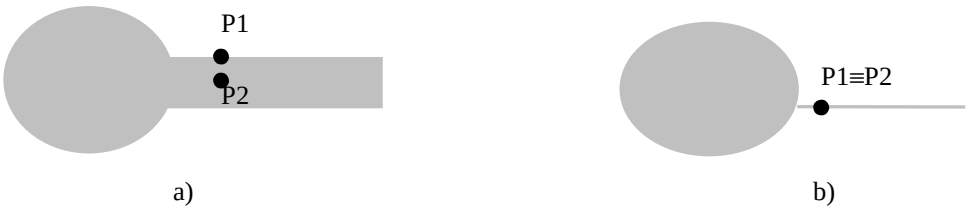

Figura 6.1

Ciò significa che la parte interna e la frontiera nella porzione di superficie collassata diventano indistinguibili e quindi le componenti di tipo curva e punto sono da interpretare come appartenenti contemporaneamente alla frontiera e alla porzione interna della superficie collassata complessiva.

Per i dettagli si rimanda al documento di specifica del GeoUML.

Anche le operazioni insiemistiche richiedono una ridefinizione. Ad esempio, l'operazione insiemistica gUnion applicata a due superfici collassate Sc1 e Sc2 produce in generale una superficie collassata le cui componenti si ottengono nel seguente modo:

- Si esegue l'unione insiemistica di tutte le componenti di Sc1 e di Sc2,
- Si estraggono dall'insieme risultato le tre componenti separatamente.

Si noti che le curve e i punti che intersecano superfici vengono assorbite dalle medesime per effetto dell'unione insiemistica.

Nel seguito si precisano, per ognuno dei tipi che rappresentano superfici in GeoUML, le proprietà aggiuntive che devono soddisfare le tre componenti e che unitamente alle precedenti permettono di identificare quali siano i valori ammessi per una superficie collassata:

GU_CPSurface2D

Proprietà 1 (si esclude la degenerazione parziale a punti): **Sc.point** può essere diverso dal valore nullo solo se sia **Sc.curve** che **Sc.surface** sono nulli e in tal caso essa può essere composta da un solo punto. Questa proprietà permette il mantenimento della connessione della superficie.
Proprietà 2 (si esclude la degenerazione a insieme non connesso): per ogni coppia di punti di S= **Sc.curve**.gUnion(**Sc.surface**) deve esistere una curva di tipo GU_CPCurve* che li collega ed è completamente contenuta in S; la proprietà di connessione di una superficie collassata è quindi più debole di quella definita per una superficie non collassata perché considera anche i punti di frontiera nella valutazione.
Proprietà 3 (si esclude la sovrapposzione lineare tra le componenti): la relazione topologica tra la componente curva e la frontiera della componente superficie deve essere un Touch: **Sc.curve** (TC) **Sc.surface.boundary().**

GU_CXSurface2D

Per le superfici di tipo CX non è richiesta la connessione, mentre è ammessa la degenerazione parziale a punti, quindi rimane solo l'ultima proprietà così ridefinita:
Proprietà 1 (si esclude il self-overlapping tra le componenti): le relazioni ammesse tra le componenti di Sc sono le seguenti:

Disjoint o Touch tra curva e superficie: **Sc.curve** DJ **Sc.surface** OR **Sc.curve TC Sc.surface.boundary()**.
Disjoint tra punto e curva: **Sc.point** DJ **Sc.curve**
Disjoint tra punto e superficie: **Sc.point** DJ **Sc.surface**

GU_CP(CX)SurfaceB3D

Per le superfici di tipo B3D la porzione superficie degenera come definito in precedenza. In aggiunta occorre gestire il collassamento dell'anello 3D che deve essere rappresentato da due componenti

- una componente detta Sc.B3D.curve di tipo GU_CXCurve3D (che non costituisce più in generale un anello) e
- una componente detta Sc.B3D.point di tipo GU_CXPoint3D, che rappresenta la parte del contorno collassata a punti.

Le proprietà che questo insieme di valori deve soddisfare è il seguente:

Il vincolo di uguaglianza del boundary della superficie 2D con la proiezione della B3D deve essere ridefinito come segue:

Sc.B3D.curve.gUnion(**Sc.B3D.point**).planar() =
Sc.point.gUnion(**Sc.surface**.boundary().gUnion(**Sc.curve**))

La figura 6.2 mostra alcuni casi di collassamento ammessi.

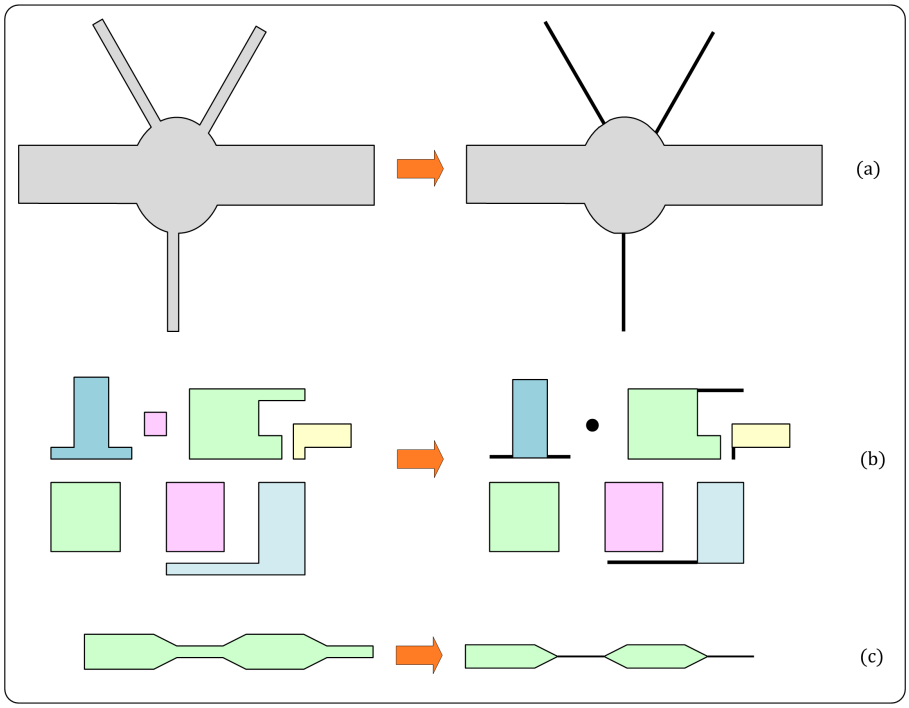

Figura 6.2: Alcuni esempi di collassamenti ammessi.

6.2 Relazioni e vincoli topologici

Le relazioni topologiche che coinvolgono le superfici collassate e i vincoli topologici basati su di esse saranno valutati considerando esclusivamente la componente poligonale delle superfici collassate Sc.surface e ignorando le componenti collassate SC.curve e Sc.point.
Pertanto è possibile che alcuni vincoli risultino violati a causa del collassamento.

7 Popolamento alle diverse scale

Una usuale specifica di contenuto definisce in un unico modo la struttura e le proprietà delle istanze che devono popolare un Data Product conforme alla specifica stessa. Diremo che una specifica di tale tipo è una Specifica Omogenea. In generale, una specifica è interpretata come omogenea ed è implicito che tutti i suoi costrutti siano popolati. In particolare questa è anche l'ipotesi alla base dell'interpretazione di un Application Schema negli standard ISO 19100.

In alcuni casi, tra i quali ricadono anche il Catalogo dei dati territoriali e il National Core, la situazione è più complessa, perché si vuole una specifica nella quale lo stesso elemento può avere proprietà diverse in base alla scala alla quale vengono rilevate le sue istanze. Diremo che una specifica di questo tipo è una Specifica di Contenuto Differenziata.

In una specifica differenziata è possibile definire un certo numero di "livelli di scala" (cioè di raggruppamenti di scale considerate indifferenziate tra loro ai fini della specifica) e dichiarare per ogni elemento informativo della specifica se esso deve essere popolato a ciascuno di tali livelli. In questo modo una specifica differenziata contiene al suo interno diverse specifiche omogenee: la specifica costituita dall'insieme complessivo degli elementi informativi e, per ogni livello di scala LS, la specifica costituita dagli elementi che devono essere popolati a livello LS.

Ad esempio, nel Catalogo dei dati territoriali [CDT2010] sono definiti due livelli di scala:

- NC1, che raggruppa le scale 1000 e 2000
- NC5, che raggruppa le scale 5000 e 10000

e il documento definisce implicitamente 3 specifiche omogenee:

1. L'insieme complessivo di tutti gli elementi che costituiscono il Catalogo dei dati territoriali, che è la specifica più completa e onnicomprensiva
2. Il National Core di livello NC1, che contiene tutti gli elementi che costituiscono la specifica da adottare alle scale 1000 e 2000
3. Il National Core di livello NC5, che contiene tutti gli elementi che costituiscono la specifica da adottare alle scale 5000 e 10000

E' importante osservare che le regole di conformità di un Data Product a una specifica differenziata sono più complesse delle regole di conformità di un Data Product a una specifica omogenea.

Vediamo due esempi con riferimento al National Core:

1) L'attributo Fondo della classe Area Veicolare (AC_VEI) ha nel NC il seguente popolamento alle scale:
"popolato a scala 1000-2000" e "non popolato a scala 5000-10000"
Questo significa che singole istanze della classe AC_VEI hanno una diversa struttura di attributi in base alla scala alla quale sono rilevate. Questa differenza impatta la conformità intrinseca di un Data Product al NC.

2) la classe delle Unità Volumetriche (UN_VOL) ha il seguente popolamento alle scale:
"popolata a scala 1000-2000" e "non popolata a scala 5000-10000"
Questa indicazione significa che le istanze di Unità Volumetriche dovranno essere rilevate solamente laddove la scala di acquisizione è 1000-2000, ma non dove è 5000-10000. Questa differenza impatta la conformità reale, ma non quella intrinseca. Se però esistesse il vincolo che le UN_VOL sono utilizzate per comporre geometricamente altri oggetti, ad esempio gli edifici, allora questo vincolo non potrebbe essere soddisfatto alla scala 5000-10000 e quindi anche la conformità intrinseca ne risulterebbe affetta.

È importante infine osservare che questa differenza di strutturazione della specifica in base alla scala si applica a priori e non altera le regole relative alla soglia di acquisizione di un oggetto; quindi la decisione se un oggetto O appartenente alla classe C deve essere rilevato a una certa scala S applica in sequenza le due regole seguenti:

1. se la classe C non è popolata alla scala S, O non viene rilevato, altrimenti si valuta la successiva regola
2. se le dimensioni di O superano la soglia di acquisizione determinata dall'accuratezza metrica prevista per la scala S, O viene rilevato, altrimenti O non viene rilevato.

7.1 Definizione del popolamento ai diversi livelli di scala

In una specifica differenziata sono definiti i valori di popolamento ai diversi livelli di scala per ognuno dei seguenti costrutti:

- ogni classe,
- ogni attributo,
- ogni valore di dominio enumerato,
- ogni componente spaziale,
- ogni attributo a tratti o sottoaree.

ESEMPIO

In figura 7.1 è riportata la definizione della classe ALBERO già vista in figura 2.1. Le colonne sulla destra indicano il popolamento dei diversi costrutti ai diversi livelli di scala: in questo caso sia la classe, sia i suoi attributi e domini sono popolati solamente al livello di scala NC1, ma non popolati al livello NC 5.

CLASSE: Albero isolato (ALBERO - 060403)

Classe con istanze monoscala

				NC1	NC5
Popolamento della classe				P	

Attributi					
Attributi della classe				NC1	NC5
06040301	ALBERO_TY	Tipo	Enum	P	
	Dominio (Tipo)			NC1	NC5
	01	Monumentale		P	
	95	Altro		P	

	Componenti spaziali della classe				NC1	NC5
060403101	ALBERO_POS	Posizione	GU_Point3D - Point 3D		P	

Figura 7.1

Il significato del popolamento è diverso per le classi e per gli altri costrutti.

Popolamento delle classi

Per quanto riguarda le classi, il popolamento di una classe C a un certo livello di scala LS significa che in un rilievo a una scala S appartenente a LS devono essere rilevate le istanze della classe C presenti nel mondo reale in funzione dell'accuratezza metrica prevista per la scala S.

Si noti che in taluni casi l'insieme di tali istanze può essere vuoto, tuttavia una classe popolata con zero istanze (classe vuota) rappresenta un contenuto informativo diverso da una classe non popolata.

Esempio:

Si consideri la classe ACQ_TER (acque territoriali). E' evidente che in un Data Product di una regione che non si affaccia sul mare tale classe non possiede istanze anche se venisse definita popolata, tuttavia il contenuto informativo della classe vuota dichiara che tale regione non possiede acque territoriali, mentre se la classe è non popolata il contenuto informativo non dice nulla relativamente a questo aspetto.

Il non popolamento ad un livello di scala di una classe che non è superclasse in una gerarchia implica pertanto che non abbia senso definire il popolamento dei suoi attributi, i domini embedded a quel livello di scala.
La caratteristica di popolamento non ha senso per le superclassi astratte poiché per definizione esse non possono avere mai istanze dirette.
Le superclassi astratte oppure non popolate ad uno o più livelli di scala devono invece definire il popolamento dei propri attributi e relativi domini; in tal modo le proprietà di popolamento sono ereditate dalle sottoclassi dirette o indirette della superclasse assieme alle altre caratteristiche (attributi, domini, associazioni, vincoli).

Popolamento degli altri costrutti
I costrutti diversi dalle classi non sono indipendenti, ma esistono esclusivamente all'interno delle classi. Essi rappresentano le proprietà delle classi. Se una di queste proprietà, ad esempio la proprietà P di una classe C, è popolata a un livello di scala LS, significa che nella struttura delle istanze della classe C rilevate alla scala S (appartenente a LS) tale proprietà è prevista, altrimenti no. La conseguenza di questa interpretazione è la seguente: ***se una classe C, popolata a due (o più) livelli di scala LS1 e LS2, possiede delle proprietà che sono popolate diversamente ai due livelli LS1 ed LS2, allora le istanze della classe C avranno una diversa struttura di proprietà ai due livelli di scala.***
Analogamente a quanto detto per il popolamento delle classi, anche per le proprietà di una classe il concetto di non-popolamento è diverso da quello di valore nullo applicabile a proprietà opzionali; se una proprietà P di una classe C non è popolata ad un livello di scala LS questo significa che non si sa nulla relativamente al valore di P in tutte le istanze di C rilevate ad una scala S di LS; invece il fatto che P sia popolata ma abbia un valore nullo in una certa istanza ha un preciso significato.
Il popolamento di una classe ad un livello di scala LS1 impone i seguenti vincoli:

- almeno un attributo proprio della classe o ereditato deve essere popolato in LS1;
- un attributo geometrico non popolato in LS1 implica il non popolamento degli attributi di attributo geometrico, attributi a tratti, eventi e a sottoaree definiti su di esso.
- il dominio embedded di ogni attributo popolato in LS1 deve avere almeno un valore popolato in LS1 e il dominio embedded di ogni attributo non popolato in LS1 non deve avere alcun valore popolato in LS1;

7.2 Determinazione della Scala di Rilievo

In una operazione di rilievo l'interpretazione dei valori di popolamento deve essere supportata da un insieme di **Regole di Determinazione della Scala di Rilievo** che stabiliscono a quale scala si deve rilevare un certo oggetto del territorio.
Esempi di possibili Regole di Determinazione della Scala di Rilievo sono:

- predefinire le caratteristiche del territorio che ne determinano la scala di rilievo, ad esempio urbanizzato a scala 1000-2000 e non urbanizzato a 5000-10000, oppure
- delimitare a priori le aree di rilievo omogenee alle varie scale
- associare la scala di rilievo a singole classi di oggetti da rilevare, ad esempio le strade al 1000, l'uso del suolo al 10000, ecc...

Le **Specifiche di Contenuto non stabiliscono Regole di Determinazione della Scala di Rilievo**, lasciando che siano appunto i capitolati di rilevamento a farlo.
Tuttavia una Specifiche di contenuto differenziata implica il soddisfacimento di alcune condizioni minime, illustrate nel seguito, da parte di un Data Product per essere conforme.

7.3 Classi normali e classi con istanze monoscala

La componente spaziale di una istanza di una classe normale può essere rilevata a diversi livelli di scala (tra quelli ai quali la classe è popolata). Pertanto, nelle classi normali non è possibile parlare della scala di rilievo di una singola istanza. La maggior parte delle classi appartiene a questa categoria.
Ad esempio, l'attributo spaziale "Pertinenza" di una istanza della classe "Estesa Amministrativa" si estende in modo da essere rilevato tipicamente a più scale diverse, e non è possibile quindi definire un valore di scala di rilievo per ogni singola istanza.

Definiamo invece ***classi a istanze monoscala*** quelle classi per le quali la componente spaziale di ogni istanza deve essere rilevata tutta alla stessa scala (se la classe possiede più componenti spaziali, devono essere tutte rilevate alla stessa scala in una singola istanza). Queste classi sono individuate esplicitamente nella specifica di contenuto. Ad esempio, in figura 7.1 la classe ALBERO è dichiarata essere a istanze monoscala.
Per ogni istanza delle classi a istanze monoscala è evidentemente possibile definire una precisa Scala di Rilievo e quindi memorizzare tale informazione in un apposito attributo, che costituisce un metadato di istanza, chiamato ***ScRil***, il cui dominio è un enumerato ***LivScala*** contenente i valori di livello di scala definiti nella specifica stessa.

L'esistenza di classi a istanze monoscala impone evidentemente dei vincoli alle Regole di Determinazione della Scala di Rilievo adottabili dal capitolato di rilievo: ad esempio, se si definiscono delle aree di rilievo a scala omogenea, le istanze delle classi a istanze monoscala non possono essere a cavallo di due diverse aree di rilievo.

7.4 Classi con Specifica omogenea o differenziata

In base ai valori di popolamento le classi presenti in una Specifica differenziata possono essere suddivise in due categorie:

1. Classi a Specifica Omogenea – sono le classi per le quali tutti i costrutti citati al precedente punto 7.1 sono popolati o non popolati in maniera identica ai diversi livelli di scala (la classe ALBERO di figura 7.1 è una classe a specifica omogenea)
2. Classi a Specifica Differenziata – sono tutte le altre classi

Tra la suddivisione delle classi in base all'omogeneità del popolamento e quella trattata precedentemente, in base alla possibilità delle istanze di avere componenti spaziali estese su più scale, esiste solamente il seguente vincolo:

Tutte le classi a Specifica Differenziata devono essere Classi a Istanze Monoscala

Il motivo è evidente: per ogni istanza di una classe a specifica differenziata il Data Product deve contenere una indicazione precisa della scala di rilievo alla quale tale istanza è stata rilevata; infatti, in caso contrario non sarebbe possibile determinare la specifica alla quale tale istanza deve essere conforme. Ma, come mostrato al punto 7.3, la scala di un'istanza è definita solamente per le classi a istanze monoscala.
Dato che il vincolo tra le due suddivisioni delle classi è solamente quello citato, possono esistere 3 combinazioni possibili:

1. classi a specifica omogenea con istanze normali
2. classi a specifica differenziata con istanze monoscala
3. classi a specifica omogenea con istanze monoscala

Le prime due combinazioni corrispondono in maniera diretta alle considerazioni già fatte; la terza costituisce un'opzione applicabile per richiedere ad esempio di spezzare le istanze di certi oggetti sui limiti di zone di rilievo a scale diverse, anche se le proprietà della classe sono omogenee.

7.5 Effetto dei livelli di popolamento delle classi sui ruoli

Il popolamento delle classi ha il seguente impatto sull'interpretazione del vincolo di obbligatorietà (cardinalità minima) dei ruoli di un'associazione:

> data una classe *C* con un ruolo *r* verso *C'* con cardinalità minima 1: se la classe C' è non popolata almeno ad un livello di scala la cardinalità minima viene considerata opzionale.

Commento

Un'associazione semantica non dipende dalle geometrie delle classi, tuttavia il non popolamento di una classe ad almeno uno dei livelli di scala determina una riduzione dell'insieme degli oggetti di una classe, limitando le associazioni possibili.

7.6 Valutazione dei vincoli

In taluni casi il mancato popolamento di alcuni costrutti rende un vincolo non applicabile, quindi la conformità di un Data Product è ottenuta anche senza soddisfare quel vincolo.

7.6.1 Applicabilità del vincolo e popolamento delle classi

Classe vincolata.

Dato che ogni vincolo deve essere soddisfatto per ogni istanza della (selezione sulla) classe vincolata (e di tutte le sue sottoclassi), se la classe vincolata non è popolata il vincolo è automaticamente soddisfatto;

Classi vincolanti.

Il problema fondamentale nella valutazione dei vincoli si pone quando una classe vincolata è popolata ma non sono popolate alcune o tutte le classi vincolanti (includendo anche le loro sottoclassi). Si presentano i casi seguenti:

Vincoli di tipo perOgni (quantificazione universale)

Questi vincoli possono sempre essere applicati anche in assenza di classi vincolanti, perchè in assenza di istanze di classi vincolanti il vincolo è automaticamente soddisfatto.

L'esempio tipico di questo caso è costituito dai vincoli di disgiunzione DJ: se ogni istanza di una classe vincolata C deve essere disgiunta da ogni istanza di una classe vincolante C', allora se C' non è popolata il vincolo è sempre soddisfatto.

Vincoli che implicano una intersezione non nulla tra le parti interne delle classi vincolata e vincolante: Composizione e relazioni topologiche IN, CT, OV, EQ

La regola per queste situazioni è la seguente:

Regola interpretazione vincoli che implicano intersezione non nulla

Si consideri un vincolo del tipo

C vincolataDa V1, V2 ... Vn

Dove C è a istanze monoscala e vincolataDa sta per qualsiasi vincolo di composizione o basato sulle relazioni topologiche IN, CT, OV e EQ.

Un'istanza c della classe C deve soddisfare il vincolo se e solo se in almeno una delle classi V1, V2 ... Vn, la componente spaziale coinvolta nel vincolo è definita popolata al livello di scala di c, indicato dal valore dell'attributo ScRil di c.

Questa regola si basa sulla ipotesi, che costituisce una restrizione sulle modalità di rilievo alle scale definito nei capitolati, che se due istanze di oggetti **appartenenti a classi a istanze monoscala** si sovrappongono nella parte interna allora debbano essere ambedue rilevate alla stessa scala.

Ad esempio, le Unità Volumetriche che compongono un Edificio devono essere rilevate alla stessa scala dell'Edificio, i marciapiedi che compongono un'area stradale devono essere rilevati alla stessa scala dell'area stradale, ecc....

Altri vincoli

Per tutti gli altri vincoli il vincolo deve essere sempre soddisfatto indipendentemente dai livelli di scala.
Nella disgiunzione di vincoli topologici se un vincolo non è applicabile viene eliminato dalla disgiunzione.

7.6.2 **Applicabilità del vincolo e popolamento degli altri costrutti**

Per le classi popolate che partecipano a un vincolo è necessario applicare le seguenti regole:

1. Se una componente spaziale o un attributo a tratti o sottoaree che compare nel vincolo non è popolato a nessun livello di scala, il vincolo non è applicabile.
2. Se un attributo di classe o di attributo geometrico che compare nelle selezioni di un vincolo non risulta popolato a nessun livello di scala, il vincolo non viene considerato applicabile.
3. Gli attributi che compaiono nelle selezioni dei vincoli che sono popolati almeno ad un livello di scala si comportano nel modo seguente:
   - il loro valore deve essere considerato nullo per tutte le istanze rilevate ai livelli di scala ai quali l'attributo non è popolato;
   - la selezione viene valutata in base alle regole generali applicate per il valore nullo