



# **Geodatabase: note di progettazione**

Cesare Gerbino

ArcGIS: Geodatabase

## Sommario

Scopo e uso del documento .....	3
Il processo di progettazione di un geodatabase .....	3
Analisi, sviluppo e gestione documentale dei casi d'uso .....	8
Modello concettuale .....	16
Modello progettuale .....	17
Dalle associazioni alle relazioni .....	19
Organizzazione diagrammi UML .....	21
Riferimenti .....	22

## **SCOPO E USO DEL DOCUMENTO**

Il presente documento ha come scopo quello di riportare alcune note d'interesse nel processo d'analisi e progettazione di un geodatabase ESRI.

## **IL PROCESSO DI PROGETTAZIONE DI UN GEODATABASE**

Il processo di progettazione ha come obiettivo quello di catturare gli aspetti significativi della realtà dell'utente (dominio), in forma di casi d'uso e, applicando diverse tecniche analitiche, decomporli in un insieme di concetti che riflettono la logica di business dei singoli casi d'uso, rifinando poi tali concetti in un insieme di classi e loro relazioni che definiscono lo schema della struttura della base dati.

Nella fase di progettazione si parte sempre da concetti generali scendendo via via in maggior dettaglio. Si tratta quindi di decomporre obiettivi ad alto livello in sottoobiettivi più piccoli espressi in termini di casi d'uso: tali casi d'uso sono i processi di business che supportano gli obiettivi di livello più alto. Ogni caso d'uso va poi decomposto ulteriormente per individuare le "cose" che sono coinvolte, ognuna delle quali diventa un concetto del modello concettuale del singolo caso d'uso. I concetti a loro volta possono poi essere decomposti in classi: è necessario tuttavia eliminare quei concetti che non saranno mai salvati all'interno di un data base (es. l'attore e i prodotti finali), raggruppare concetti simili individuando gerarchie di ereditarietà. Un dato concetto si può decomporre in molte classi, oppure diversi concetti possono aggregarsi in un'unica classe. E' necessario determinare gli attributi di ogni classe, i loro tipi di dato e i domini di valori ammessi, come pure le relazioni tra le singole classi e le loro cardinalità. Il risultato è espresso utilizzando un digramma delle classi UML. Usando degli strumenti CASE è possibile leggere direttamente lo schema della base dati da ArcCatalog (usando lo schema wizard), oppure implementando lo schema a mano direttamente in ArcCatalog.

Il processo di disegno OO è caratterizzato dai seguenti punti:

- **Guidato dai casi d'uso:** i casi d'uso sono i processi di business del dominio dell'utente
- **Guidato dall'analisi dei rischi:** i casi d'uso relativi ai rischi dovrebbero essere affrontati nelle prime fasi del ciclo di vita dell'analisi / progettazione

- **Incrementale:** è possibile isolare dei pezzi del problema e iniziare il loro sviluppo prima che l'intero disegno sia terminato. E' tuttavia necessario che si raggiunga una "stabilità" della porzione di problema analizzato (devono essere stati chiariti tutti gli aspetti relativi alla porzione del problema che è stata individuata, quindi tutte le classi che ne sono coinvolte e le loro relazioni).
- **Iterativo:** non tutti i casi d'uso devono essere sviluppati nelle prime fasi dello sviluppo, così come non tutte le funzionalità devono essere attive e implementate per i casi d'uso sviluppati. Successive iterazioni (cicli di sviluppo), aggiungeranno maggiori casi d'uso e funzionalità (come per il punto precedente valgono anche in questo caso le considerazioni in merito alla "stabilità" degli aspetti che si decide di implementare)

I casi d'uso devono guidare tutte le fasi dello sviluppo, dal disegno della base dati allo sviluppo dell'applicazione compresa la fase di testing: i casi d'uso sono analizzati per individuare gli oggetti che devono essere memorizzati nella base dati, sono usati nella fase di sviluppo per realizzare le applicazioni che supportino direttamente le attività che caratterizzano di casi d'uso, i piani dei test sono costruiti basandosi sui casi d'uso.

I casi d'uso inizialmente sono descritti in linguaggio naturale e da un punto di vista dell'utente (quindi non devono contenere indicazioni di natura tecnica o descrivere una soluzione, devono descrivere "come" l'utente interagisce con il sistema per eseguire un certo compito): in seguito, dalla loro analisi, sarà possibile scendere maggiormente nel dettaglio sino ad arrivare ad ipotizzare le possibili alternative tecnologiche di implementazione.

Nel ciclo di vita si possono individuare diverse fasi e precisamente:

- **Fase iniziale**
- **Fase di elaborazione**
- **Fase di costruzione**
- **Fase di transizione**

#### **Fase iniziale**

In questa fase sono identificati: business case, casi d'uso di alto livello di astrazione, criteri di successo del progetto, valutazione dei rischi, stima delle risorse del progetto (personale, hardware, software, ecc ...), piano di sviluppo (con milestones e metriche).

Gli output di questa fase sono (almeno a livello di impostazione):

- **Descrizione del progetto:** descrizione ad alto livello del progetto
- **Dizionario del progetto:** un vocabolario comune
- **Lista dei rischi:** identificazione dei rischi
- **Lista dei requisiti:** cosa il progetto deve implementare
- **Lista dei prodotti:** cosa il progetto deve restituire
- **Lista dei dati:** cosa il progetto deve trattare
- **Lista degli attori:** chi sono gli attori (utenti)
- **Lista dei casi d'uso:** che cosa fanno gli attori
- **Diagrammi dei casi d'uso**
- **Matrice casi d'uso / dati:** relaziona dati e casi d'uso (cosa viene utilizzato quando)
- **Piano di sviluppo**

### Fase di elaborazione

In questa fase vengono analizzati in maggior dettaglio i casi d'uso allo scopo di identificare le classi necessarie per supportare i singoli casi d'uso.

Gli output di questa fase sono principalmente i casi d'uso analizzati ad un livello maggiormente dettagliato e dei diagrammi UML. A livello di dettaglio si possono avere:

- **Casi d'uso:** partendo dai casi d'uso ad alto livello di astrazione della fase iniziale si possono avere maggiori dettagli sintetizzati in casi d'uso *Business Level* e casi d'uso *Design Level*
- **Modello concettuale dei dati:** un diagramma UML (non tutti i concetti espressi nel modello concettuale diventeranno classi del geodatabase)
- **Modello dei dati:** un diagramma UML che illustra le classi, i loro attributi, le loro relazioni con altre classi

- **Altri documenti:** altri diagrammi UML come diagrammi di attività, diagrammi di collaborazione, ecc ... utili a spiegare le caratteristiche del progetto

### **Fase di costruzione**

In questa fase viene costruito lo schema del geodatabase, alimentati i dati, e scritto il codice dell'applicazione e realizzate le interfacce che permettono di supportare i casi d'uso.

Gli output di questa fase sono:

- **Schema della base dati**
- **Dati per l'alimentazione della base dati**
- **Applicazione e interfacce**

### **Fase di transizione**

In questa fase viene testato il prodotto e allestito quanto serve per la distribuzione del sistema.

Gli output di questa fase sono:

- **Documentazione di test:** test funzionali, test di carico, ecc ...
- **Documentazione di prodotto:** descrizione tecnica, architetturale, help in linea, manuale utente, ecc ...
- **Piani di Formazione**
- **Supporto tecnico**

E' possibile individuare tre tipi di progetti GIS e precisamente:

- **Progetti standalone:** sono i più semplici, destinati a risolvere un problema specifico
- **Progetti dipartimentali:** destinati a supportare diverse applicazioni su una rete locale e con un numero di utenti limitato e del medesimo gruppo

- **Progetti enterprise:** destinati a supportare molte applicazioni per diversi utenti appartenenti a gruppi diversi. Sono caratterizzati da grandi volumi di dati e di accessi.

Per tutti i progetti è possibile utilizzare il medesimo approccio metodologico di analisi e progettazione, secondo le caratteristiche alcune fasi richiederanno maggiore o minore tempo per essere realizzate.

Nella realizzazione di un progetti GIS è possibile individuare diversi *team* che sono in stretta collaborazione tra loro e precisamente:

- **Design team:** si occupa di fare da tramite tra l'utente e i team "tecnici". E' sua responsabilità intervistare l'utente, catturare i suoi requisiti, analizzarli e presentarli chiaramente e in modo non ambiguo ai gruppi tecnici
- **User interface team:** si occupa di definire e realizzare l'interfaccia dell'applicazione
- **Application team:** si occupa di definire e realizzare la componente applicativa del progetto implementando quanto definito nei casi d'uso proposti dal Design team
- **Platform team:** si occupa degli aspetti legati alle componenti hardware e di rete in collaborazione con il Design team e l'Application team
- **Database administration team:** si occupa di creare e gestire le componenti legati alla base dati cercando di garantire la massima efficienza

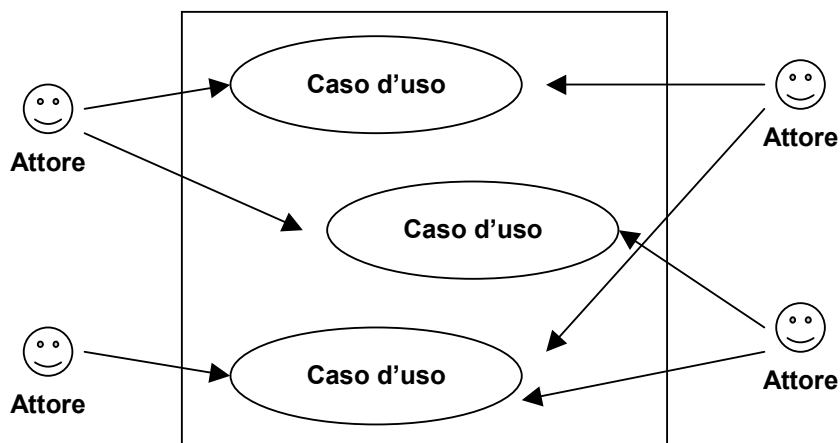
Le varie fasi producono molti documenti che devo essere raccolti in un'area condivisa e raggiungibile da tutti i membri dei diversi team: può essere di aiuto creare dei semplici siti web per raccogliere tutta la documentazione in modo coordinato e coerente.

## ANALISI, SVILUPPO E GESTIONE DOCUMENTALE DEI CASI D'USO

I casi d'uso (almeno quelli ad alto livello e quelli di business), devono essere redatti in stretta collaborazione con gli utenti che conoscono i processi: è necessario che le descrizioni siano espresse in linguaggio naturale evitando termini tecnici non specifici al dominio (es. termini legati alla tecnologia utilizzata per la successiva implementazione), e questi ultimi andrebbero raccolti nel dizionario del progetto.

Parallelamente ai casi d'uso si dovrebbe sviluppare e mantenere aggiornato il digramma concettuale che identifica gli oggetti coinvolti nel caso d'uso (oltre ad eventuali altri documenti collaterali quali digrammi di attività).

L'analisi dei casi d'uso permette di identificare anche gli attori che interagiscono con il sistema: è quindi possibile disegnare uno use case diagram che illustra le relazioni tra attori e casi d'uso.



In casi reali quando gli use case diagrams diventano molto complessi è possibile organizzare le cose su più use case diagram.

A livello di documentazione gli attori vengono descritti in una lista che può essere organizzata nel seguente modello:

<b><u>Progetto ABC</u></b>		
<b>Lista degli attori</b>		
<b>ID</b>	<b>Attore / Area</b>	<b>Descrizione / Casi d'Uso</b>
A1	Manager	Breve descrizione dell'attore e dei suoi compiti
		Casi d'uso: U12 - <Nome caso d'uso> U23 - <Nome casi d'uso>
A2	Analista	Breve descrizione dell'attore e dei suoi compiti
		Casi d'uso: U1 - <Nome caso d'uso> U34 - <Nome caso d'uso>
A3	Operatore	Breve descrizione dell'attore e dei suoi compiti
		Casi d'uso: U17 - <Nome caso d'uso> U22 - <Nome casi d'uso>

E' necessario individuare ogni singolo attore con un identificativo univoco.

Sarebbe opportuno memorizzare il tutto a livello di database e non solo a livello documentale.

Lungo il ciclo di vita i dettagli dei singoli casi d'uso vanno via via aumentando: si possono quindi individuare tre tipologie di casi d'uso

- High level
- Business level
- Design level

A livello di documentazione per i casi d'uso occorrerà gestire la lista dei casi d'uso e la documentazione dei tre livelli sopra citati.

La lista dei casi d'uso può essere organizzata nel seguente modello:

<b><u>Progetto ABC</u></b>			
<b>Lista dei casi d'uso</b>			
<b>ID</b>	<b>Area</b>	<b>Nome</b>	<b>Descrizione</b>
U65	<Codice Area>	<Nome Caso Uso>	Breve Descrizione del caso d'uso
U34	<Codice Area>	<Nome Caso Uso>	Breve Descrizione del caso d'uso
U23	<Codice Area>	<Nome Caso Uso>	Breve Descrizione del caso d'uso

E' necessario individuare ogni singolo caso d'uso con un identificativo univoco.

Sarebbe opportuno memorizzare il tutto a livello di database e non solo a livello documentale.

I casi d'uso di tipo High Level devono fornire una descrizione più dettagliata del compito rispetto a quanto presente nella lista dei casi d'uso. Sostanzialmente si tratta di un documento **non** tecnico che riassume cosa fa un attore per eseguire il compito oggetto del caso d'uso.

La documentazione dei casi d'uso di tipo High Level può essere organizzata nel seguente modello:

<b><u>Progetto ABC</u></b>	
<b>Caso d'uso</b>	<Nome caso d'uso> (<Codice caso d'uso>)
<b>Livello</b>	High Level
<b>Riassunto</b>	Breve riassunto del caso d'uso
<b>Attori</b>	<Elenco attori coinvolti>
<b>Pre-condizioni</b>	Descrizione delle condizioni che devono essere soddisfatte per poter eseguire le azioni relative al caso d'uso (es. l'attore ha ricevuto il nuovo codice di uso del suolo di una certa particella)
<b>Post-condizioni</b>	Descrizione delle condizioni che devono essere soddisfatte dopo l'esecuzione delle azioni del caso d'uso (es. è stata cambiato il codice d'uso della particella ed è stato notificato il cambiamento al proprietario)
<b>Descrizione</b>	Descrizione in linguaggio naturale di cose deve fare un attore per eseguire il compito oggetto del caso d'uso

Sarebbe opportuno memorizzare il tutto a livello di database e non solo a livello documentale.

I casi d'uso di tipo Business Level devono catturare il processo di business come una sequenza temporale di passi, senza mostrare le interazioni attore / computer. Serve per identificare il processo (di utilità per l'Application team), e per gli oggetti di dominio (di utilità per il Design team e per il Database administration team).

A questo livello viene descritto solo lo scenario principale, lasciando i dettagli degli scenari alternativi (o relativi al trattamento delle situazioni di errore o di eccezione), alle fasi successive.

E' importante **NON** introdurre aspetti tecnici in questa fase: ad esempio scrivere "*l'attore fa un click sul bottone Save*" sarebbe sbagliato, occorre esprimere il concetto e lasciare i dettagli implementativi per le fasi successive, quindi più correttamente si dovrebbe scrivere "*l'attore salva il documento*".

Parallelamente a questa fase si dovrebbe mantenere aggiornato e dettagliato il modello concettuale.

La documentazione dei casi d'uso di tipo High Level può essere organizzata nel seguente modello:

<b><u>Progetto ABC</u></b>	
<b>Caso d'uso</b>	<Nome caso d'uso> (<Codice caso d'uso>)
<b>Livello</b>	Business Level
<b>Riassunto</b>	Breve riassunto del caso d'uso
<b>Attori</b>	<Elenco attori coinvolti>
<b>Pre-condizioni</b>	Descrizione delle condizioni che devono essere soddisfatte per poter eseguire le azioni relative al caso d'uso (es. l'attore ha ricevuto il nuovo codice di uso del suolo di una certa particella)
<b>Post-condizioni</b>	Descrizione delle condizioni che devono essere soddisfatte dopo l'esecuzione delle azioni del caso d'uso (es. è stata cambiato il codice d'uso della particella ed è stato notificato il cambiamento al proprietario)
<b>Scenario Principale</b>	

- 1) Azione 1
- 2) Azione 2
- 3) Azione 3
- 4) .....

**Note**

Note utili (es. Significato di alcuni termini propri del dominio da replicare nel dizionario di progetto)

I casi d'uso di tipo Design Level rappresentano la fase di maggior dettaglio e in realtà costituiscono la prima fase della progettazione dell'applicazione, infatti vengono catturate e descritte le interazioni tra gli attori e il sistema.

Vengono inoltre affrontati e descritti gli scenari alternativi e quelli relativi alla gestione delle eccezioni e degli errori.

La documentazione dei casi d'uso di tipo High Level può essere organizzata nel seguente modello:

<b><u>Progetto ABC</u></b>	
<b>Caso d'uso</b>	<Nome caso d'uso> (<Codice caso d'uso>)
<b>Livello</b>	Design Level
<b>Riassunto</b>	Breve riassunto del caso d'uso
<b>Attori</b>	<Elenco attori coinvolti>
<b>Pre-condizioni</b>	Descrizione delle condizioni che devono essere soddisfatte per poter eseguire le azioni relative al caso d'uso (es. l'attore ha ricevuto il nuovo codice di uso del suolo di una certa particella)

<b>Post-condizioni</b>	Descrizione delle condizioni che devono essere soddisfatte dopo l'esecuzione delle azioni del caso d'uso (es. è stata cambiato il codice d'uso della particella ed è stato notificato il cambiamento al proprietario)	
<b>Scenario Principale</b>		
<b>Attore</b>	<b>Sistema</b>	
1) descrizione azione iniziale dell'attore (es. inserimento username)	2) Descrizione risposta sistema (es. accettazione)	
	3) Descrizione operazioni disponibili	
4) Scelta operazione	.....	
.....	.....	
<b>Scenari Alternativi</b>		
Descrizione per punti degli scenari alternativi		
<b>Scenari Errori</b>		
Descrizione per punti degli scenari di errore o di eccezione		
<b>Note</b>		
Note utili (es. Significato di alcuni termini propri del dominio da replicare nel dizionario di progetto)		

Oltre ai documenti sopra citati ci possono essere altri documenti di interesse nell'ambito del progetto ad esempio schemi di interfaccia e esempi o prototipi di layout di reports o mappe, queste ultime di interesse peculiare dei progetti GIS.

14 • Geodatabase: note di progettazione

\_\_\_\_\_

Particolare attenzione deve essere posta agli schemi di interfaccia che possono essere utilizzati nelle fasi di approfondimento dei casi d'uso, perché possono essere fuorvianti per l'utente che potrebbe essere indotto a pensare che rappresentino i prototipi dell'interfaccia definitiva.

In realtà tali schemi di interfaccia devono servire solo per aiutare a chiarire gli aspetti nelle fasi di definizione dei casi d'uso di Business Level e per facilitare la comunicazione con lo User Interface Team. Occorre chiarire queste finalità con l'utente finale ed evitare di entrare troppo nei dettagli, bensì mantenere la concentrazione sul processo sarà poi compito dello User Interface Team definire i dettagli, l'aspetto grafico e la logica dell'interfaccia.

## MODELLO CONCETTUALE

Il modello concettuale sostanzialmente è un'altra modalità di vedere un caso d'uso: nel creare un modello concettuale il caso d'uso viene decomposto in un insieme di concetti relazionati tra loro, e tali concetti e le loro relazioni vengono rappresentate in un diagramma.

Il modello concettuale è uno strumento di analisi e non uno strumento di progettazione: come tale serve a promuovere la comprensione e la comunicazione tra il Design team e l'utente. Il modello concettuale deve essere definito in stretta collaborazione con l'utente allo scopo di assicurarsi di avere ben compreso il problema.

Il modello concettuale non dovrebbe quindi trattare concetti quali database, oggetti software, videate, forms, ecc ..., ma solo concetti di interesse nel dominio del problema.

E' possibile che nel modello concettuale siano presenti concetti che sono importanti per la comprensione del dominio del problema, ma che non diventeranno mai classi e che quindi non saranno presenti nel modello fisico della base dati (modello in cui sono presenti le classi di oggetti che devono essere persistenti nella base dati, e dei quali è necessario definire le caratteristiche, come attributi e cardinalità delle relazioni, in maggior dettaglio).

L'individuazione dei concetti può avvenire in modo diverso ma principalmente analizzando la descrizione testuale del caso d'uso: da una prima lista di concetti "candidati" vengono scartati quei concetti che al di fuori dello *scope*, quelli che possono essere raggruppati, quelli che in realtà sono degli attributi, ecc .. sino ad individuare una lista finale che costituirà l'insieme dei concetti del modello.

In questa fase non è fondamentale individuare gli attributi, in quanto l'enfasi del modello concettuale è comprendere le relazioni strutturali, cioè i concetti e le associazioni che li legano: ovviamente se nell'analisi e nelle determinazioni dei concetti si evidenziano alcuni attributi questi vengono presi in considerazione, ma non è necessario approfondire questo aspetto, anche perché è meglio aspettare che i concetti si stabilizzino e si trasformino in classi prima di scendere nel dettaglio degli attributi che le caratterizzano.

Per quello che riguarda le associazioni occorre tenere presente che alcune delle associazioni individuate, analogamente ai concetti del modello, non compariranno nel modello fisico finale.

L'individuazione avviene anche in questo caso principalmente analizzando la descrizione testuale del caso d'uso senza entrare troppo nel dettaglio, limitandosi a indicare una cardinalità generica (uno a uno, uno a molti, ecc ..), rimandando i raffinamenti alle fasi successive.

## MODELLO PROGETTUALE

Il modello progettuale provvede a dettagliare maggiormente quanto espresso nel modello concettuale e ad eliminare quelle caratteristiche (concetti e associazioni), che erano stati presi in considerazione per ragioni di maggiore comprensione del modello. L'interesse è ora rivolto verso tutti quegli aspetti che sono di interesse e che dovranno essere persistenti nella base dati.

I concetti verranno implementati in termini di *classi*, e le associazioni in termini di *relazioni*.

Nella definizione delle classi occorre fare delle considerazioni anche oltre al singolo caso d'uso per vedere di definire la classe a valenza generale nel contesto del dominio del problema, come pure individuare gerarchie di classe (superclassi e sottoclassi).

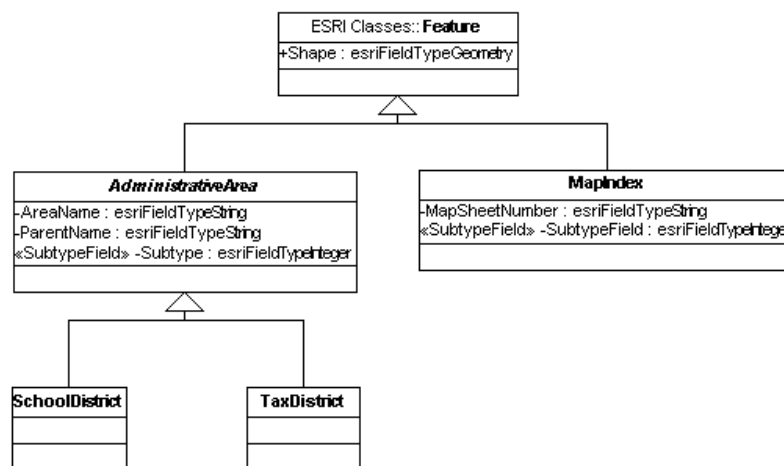
Esistono due tipologie di classe, le classi *astratte* e le classi *concrete*.

Dalle classi astratte non si possono istanziare oggetti, servono per mettere in comune attributi e comportamenti: non hanno rappresentazione dentro il geodatabase.

Nella definizione di un gerarchia di classi, per poter garantire l'integrità, è necessario che:

- le classi foglia siano classi concrete
- le superclassi **NON** siano classi concrete

Ad esempio nella figura seguente



la classe **AdministrativeArea** è una classe astratta (in Viso le classi astratte vengono evidenziate con il nome scritto in corsivo), mentre tutte le classi “foglia” sono classi concrete.

Per ragioni di prestazioni occorre cercare di ridurre al minimo il numero delle sottoclassi: se le sottoclassi individuate si differenziano per pochi attributi e hanno lo stesso comportamento, potrebbe aver senso “collassarle” in un’unica classe utilizzando tutti gli attributi di entrambe. Ovviamente con questa modalità occorre spostare a livello dell’applicazione un po' della logica che nell’organizzazione sarebbe stata implementata a livello della base dati (es. controllare via codice i valori di un attributo che non può più essere associato univocamente ad un dominio, visto che il geodatabase non supporta i domini condizionali).

Per situazioni di questo genere il modello del geodatabase offre il concetto dei *sottotipi* che permettono di raggruppare oggetti, all’interno di un’unica classe, sulla base dei valori di un attributo.

I sottotipi sono una peculiarità del modello del geodatabase, e sono più efficienti delle sottoclassi perché in realtà, operano all’interno di un’unica classe, quindi applicazioni quali ArcMap riescono ad avere maggiori prestazioni perché è minore il numero di classi da aprire e gestire.

In termini di prestazioni occorre poi fare un’altra considerazione: quando ArcMap accede ad una feature class all’interno di un feature dataset, questo viene comunque completamente aperto (cioè vengono considerate tutte le tabelle che lo costituiscono). Essendo le sottoclassi (o i sottotipi), generalmente memorizzati all’interno di un unico feature dataset risulta evidente che il poter usare i sottotipi rispetto alle sottoclassi incide in modo considerevole sulle prestazioni.

## DALLE ASSOCIAZIONI ALLE RELAZIONI

Le associazioni individuate nel modello concettuale possono poi essere implementate nel modello progettuale in modi diversi: come classi di relazione memorizzate permanentemente nel geodatabase (è possibile solo tra classi all'interno dello stesso geodatabase), o come associazioni tra classi del geodatabase e tabelle esterne (tramite join o relates in ArcMap ad esempio).

Nel caso di classi di relazione si ha accesso alla lettura e scrittura alle tabelle relazionate, nel caso di join e relates si ha accesso alle tabelle relazionate in sola lettura.

Nel caso di gerarchie di classi è importante sottolineare che nel modello del geodatabase è possibile stabilire delle relazioni solo tra classi "foglia": questa è una limitazione propria del modello. Occorre tuttavia evidenziare che nel caso si utilizzino i sottotipi, non essendo questi delle classi è possibile stabilire delle relazioni tra sottotipi: in realtà questo significa il raffinare la cardinalità di una relazione che esiste tra le classi generali dei rispettivi sottotipi che deve comunque esistere nel modello.

Nella definizione delle cardinalità delle relazioni occorre prestare attenzione; infatti la specifica generale *uno* corrisponde ad una cardinalità 0..1, mentre la specifica *molti* corrisponde alla cardinalità 0..\*. Specificando quindi genericamente una cardinalità uno-a-molti tra la classe A e la classe B si rischia di avere degli oggetti di classe B "orfani", cioè non collegati a nessun oggetto di classe A: è necessario settare delle regole di relazione che esplicitino "1..1" o "0..\*" per evitare queste situazioni che non garantirebbero l'integrità referenziale. Poiché le regole di relazione si possono stabilire solo utilizzando i sottotipi, è necessario ricorrere a questa metodologia di progettazione se si vogliono controllare questi aspetti.

Nelle tabelle di decodifica vengono utilizzate delle relazioni 1..N: in questi casi è necessario che la tabella con le descrizioni delle codifiche siano le tabelle "origine" della relazione, mentre le tabelle delle classi siano quelle "destinazione", altrimenti si rischia, in caso di cancellazione, di rendere nulli records nelle tabelle di decodifica. La scelta della tabella "origine" controlla la relazione.

Nel caso di relazioni molti-a-molti (N:M), è necessario utilizzare delle tabelle di appoggio (key table), con le quali le tabelle Origine e Destinazione stabiliscono delle relazioni 1:N. Nelle key tables dovranno essere quindi presenti degli attributi che permettano di stabilire tali relazioni (foreign keys), e, se necessario, degli attributi che caratterizzino la relazione stessa. È importante osservare che per le key table:

- non è possibile utilizzare i domini
- non si possono importare o esportare da ArcCatalog
- è necessario popolarle. Esiste uno script in ArcObjects Help che permette di caricare records da una tabella in una key table

Le relazioni di aggregazione non sono supportate nel modello del geodatabase: questo non vuol dire che non si possono utilizzare, vengono utilizzate nel modello concettuale mentre nel modello progettuale vengono ridotte ad associazioni binarie.

Le relazioni di composizione invece sono utilizzate ma occorre tenere presente alcuni vincoli: ad esempio non è possibile avere il comportamento delle parti che si muovono insieme alla componente insieme senza avere la cancellazione a cascata.

Nel caso si utilizzino i sottotipi le relazioni tra una classe origine ed una destinazione possono essere raffinate utilizzando delle “regole di relazione” (relationship rules): ad esempio può esistere una relazione tra la classe Parcel e la classe Building, nel senso che data una particella ci possono essere su di essa diversi edifici, ma nella realtà si ha la necessità di restringere questa generica relazione, infatti non deve essere possibile avere un’industria su una particella residenziale. Le relationship rules forniscono due tipi di vincoli e precisamente:

- stabiliscono i sottotipi origine e destinazione connettendo quindi coppie compatibili
- limitano la cardinalità della relazione (che deve tuttavia essere compatibile con quella stabilita nella relazione “padre”)

Occorre tuttavia sottolineare che stabilita una relationship rule su una relazione questa diventa l’unica relazione valida, quindi se si desiderano stabilire delle relationship rules su una relazione occorre stabilirle tutte.

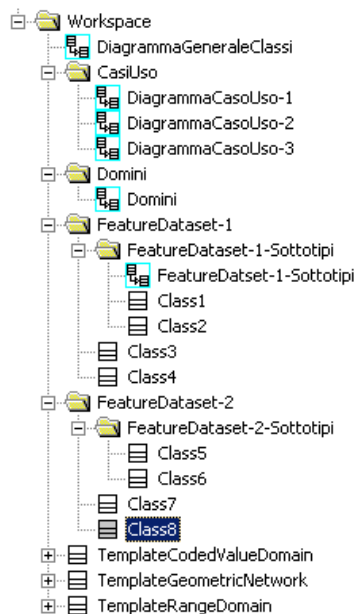
## ORGANIZZAZIONE DIAGRAMMI UML

Per facilitare la lettura del modello è possibile suddividere quest'ultimo in diversi diagrammi, ad esempio:

- creare un diagramma delle classi generale che contenga solo le classi, la loro organizzazione gerarchica e i sottotipi. Per facilitare la lettura non devono essere presenti gli attributi delle classi, i domini, le classi di relazione ecc ...
- creare un diagramma per ogni caso d'uso che faccia vedere solo le classi coinvolte (con il dettaglio degli attributi), le relazioni, ecc ...
- creare uno o più diagrammi con i domini
- creare uno o più diagrammi con le regole di connettività

Quando si debbano effettuare cambiamenti in una classe, questi devono essere registrati nel diagramma generale delle classi: tutti i diagrammi che mostrano la classe in esame mostreranno la classe aggiornata automaticamente. Quando si debba aggiungere una nuova classe la si deve aggiungere nel diagramma generale delle classi e poi farne delle copie nei diagrammi dei singoli casi d'uso in cui questa classe viene utilizzata.

Una proposta di organizzazione della struttura del modello è la seguente:



## RIFERIMENTI

*“Building a Geodatabase”* – Documentazione ufficiale ArcGIS

ArcGIS Data Models – Modelli concettuali ed esempi scaricabili dal sito ufficiale ESRI  
([www.esri.com](http://www.esri.com))