UNIVERSITA' DEGLI STUDI DI MILANO

FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI


Corso di Laurea in Informatica

# Integrazione tra GIS ed un algoritmo di ottimizzazione di un sistema dial-a-ride

Relatore:

Prof. Giovanni Righini

Correlatore:

Dott. Matteo Salani

Tesi di Laurea di:

Mammana Francesco

Matr. N. 614277

Anno Accademico 2004/2005

*Ai miei genitori*

# Indice

# Introduzione

Un sistema informativo geografico (GIS) è un insieme di strumenti che, a partire da una rappresentazione del territorio e da un'insieme di informazioni di tipo geografico, associa a queste, secondo specifiche modalità, ulteriori dati provenienti da altri database o da altri sistemi informativi già esistenti. Un sistema dial-a-ride dinamico si riferisce a un contesto in cui vi sono uno o piu' utenti che eseguono una richiesta di trasporto specificando le caratteristiche del servizio e una centrale che cerca di soddisfare queste richieste ottimizzando i mezzi a propria disposizione. Lo scopo della tesi è sviluppare un applicazione che integra le librerie di un sistema informativo geografico (MapObject) con algoritmi per l'ottimizzazione di un sistema a chiamata statico di tipo porta a porta e fermata a fermata (DARIA) . L'applicazione è sviluppata nel linguaggio C# in ambiente Microsoft Visual Studio .NET, con l'utilizzo del provider di dati .NET Framework: OLE DB Microsoft Jet 4.0, per la connessione ai file .mdb (file Access). La tesi è stata svolta in collaborazione con Ala Mobility, società del gruppo AutoGuidovie Italiane S.p.A.

L'elaborato si sviluppa in 4 capitoli:

**Capitolo** 1: Sistemi di trasporto a chiamata

In questa parte analizziamo le due tipologie dei sistemi di trasporto a chiamata: statico e dinamico; inoltre descriviamo DARIA, software implementato per realizzare questo tipo di servizio

**Capitolo** 2: Sistemi informativi geografici (GIS)

Questo capitolo introduce inanzitutto i sistemi informativi geografici analizzandoli da tre punti di vista: Geodatabase, Geovisualizzazione e Geoprocessing, descrive poi le principali tipologie di dati geografici, ed infine presenta le librerie MapObject, utilizzate per la gestione dei dati geografici.

**Capitolo** 3: Integrazione degli strumenti

In questo capitolo descriviamo le due fasi in cui avviene lo scambio di dati tra l 'applicazione e DARIA: nella prima l'applicazione crea delle strutture per la traduzione dei dati geografici nei due formati[1] e invia a DARIA un database contenente il grafo stradale su cui effettuare l'ottimizzazione; nella seconda fase essa salva le richieste di trasporto degli utenti con relativi punti di partenza e di arrivo in un database.

**Capitolo** 4: Interfaccia utente

Questa parte tratta la realizzazione del geocoding[2] e il sistema di visualizzazione dei percorsi e delle fermate.

[1] Il formato TeleAtlas, utilizzato da MapObject, e il formato proprietario di DARIA

[2] Il processo che converte un riferimento spaziale implicito in un riferimento spaziale esplicito

# Capitolo 1

# Sistema di trasporto a chiamata

## 1.1 Sistemi Dial-a-Ride

L' obiettivo di un sistema di trasporto moderno è migliorare il comfort e l'efficienza del servizio offerto garantendo un costo contenuto. Un tale sistema deve determinare la riduzione del traffico, del rumore, dell'inquinamento e contribuire ad una gestione controllata dei costi. Nei tradizionali sistemi di trasporto pubblico basati sull'utilizzo di tram, treni, metropolitane e bus, i veicoli compiono tragitti prestabiliti per raggiungere un insieme di fermate, rispettando dei vincoli temporali. Nei sistemi di trasporto a chiamata, noti come sistemi Dial-a-Ride(DAR), i percorsi dei veicoli non sono fissati a priori e non devono rispettare dei vincoli temporali ma sono stabiliti in base alle richieste degli utenti. Un siffatto sistema di trasporto offre comfort e flessibilità paragonabili ai mezzi privati e ai taxi ad un costo inferiore. Ciò è dovuto alla gestione ottimizzata dei mezzi a disposizione ed anche all'utilizzo di veicoli di capacità maggiore rispetto alle normali automobili. Nei

sistemi di trasporto a chiamata l'utente richiede un servizio specificando le informazioni riguardanti il viaggio, il punto di partenza e d'arrivo, l'orario di partenza desiderato, l'orario d'arrivo accettabile e il numero di persone da servire. Il compito della centrale operativa è di ricevere tutte le chiamate, stabilire se i vincoli richiesti dall'utente sono soddisfacibili, decidere l'assegnazione degli utenti ai diversi veicoli che compongono la flotta e, per ciascun veicolo, costruire la sequenza di carico e scarico degli utenti. I sistemi DAR si possono suddividere in statici e dinamici:

- Un sistema DAR statico è caratterizzato dal fatto che gli utenti richiedono il servizio in anticipo, ad esempio il giorno prima, e l'allocazione e l'instradamento dei veicoli sono calcolati prima che il servizio inizi. In questo caso è possibile dedicare diverse ore al calcolo dell'allocazione ottima e dell'instradamento dei veicoli. L'inconveniente del DAR statico è che non possono essere presi in considerazione gli eventi che accadono durante il periodo di servizio e che richiederebbero una rielaborazione delle chiamate (nuove chiamate, ritardi per ingorghi, utenti che mancano all'appuntamento di carico,...).

- Un sistema DAR dinamico consente agli utenti di richiedere il servizio durante il periodo di attività dei veicoli. La centrale deve allocare la chiamata ad un veicolo ed elaborare un nuovo instradamento per il veicolo interessato al trasporto. I sistemi DAR dinamici sono flessibili perché consentono di ri-ottimizzare le decisioni prese in precedenza, in seguito al verificarsi di eventi imprevisti. La messa in opera di sistemi

DAR dinamici richiede naturalmente anche il continuo monitoraggio dello stato della rete e della flotta di veicoli che si muove su di essa

## 1.2 DARIA

Daria 3.0 e' il software sviluppato per la soluzione di un problema dinamico che si puo' applicare sia al trasporto di persone che al trasporto di merci non pianificato. L'obiettivo e' quello di ottimizzare il servizio offerto all'utenza. DARIA adotta una rappresentazione classica del sistema stradale. Il modello matematico su cui si basa e' un grafo orientato G=(N,A) costituito da |N| vertici e |A| archi,questi ultimi pesati con i tempi di percorrenza. Un punto p nel grafo e' definito da:

$$p = <fp,tp,\lambda p> \qquad fp,tp \epsilon N, \lambda \epsilon [0,1]$$

dove *fp,tp* sono gli estremi dell'arco percorso da *fp* verso *tp* e *p* è la frazione di arco fra *fp* e il punto *p[1].* Il sistema DARIA consente di effettuare le operazioni di carico e scarico in qualunque punto della mappa, sia esso un vertice o un punto intermedio di un arco e dal lato corretto della strada: in questo modo lutente non è costretto ad attraversare la strada, né per salire sul vecolo, né per raggiungere la destinazione quando scende dal veicolo

# Capitolo 2

# GIS

## 2.1 I sistemi informativi geografici (GIS)

Un GIS, acronimo di Geographic Information Systems e traducibile in Sistema Informativo Geografico (o Territoriale), è un sistema per la gestione, l'analisi e la visualizzazione di informazioni con contenuto geografico/spaziale. L'informazione geografica è gestita tramite insiemi di dati (dataset geografici) che costituiscono modelli di fenomeni geografici, cioé riferibili al territorio, utilizzando strutture di dati semplici e generiche. Il GIS è corredato da un insieme completo di strumenti (tool e funzionalità) per lavorare con i dati geografici. Un Sistema Informativo Geografico consente di interagire con l'informazione geografica secondo diversi punti di vista:

1. L'approccio del Geodatabase: un GIS è un database spaziale, ossia un database contenente dataset che comprendono l'informazione geografica. Tramite un modello di dati consente la gestione di elementi vettoriali (features), immagini raster, topologie, reti e così via.

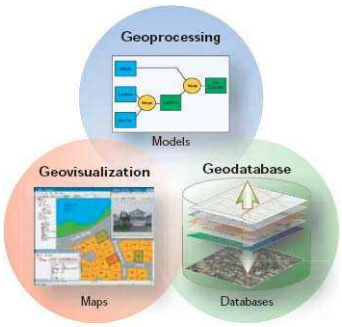


Figura 2.1: Le tre viste di un GIS

2. L'approccio della Geovisualizzazione: un GIS consente di costruire rappresentazioni geografiche complete e complesse (mappe) in cui vengono visualizzati gli elementi (features) e le loro relazioni spaziali sulla superficie terrestre.

3. L'approccio del Geoprocessing: un GIS è un insieme di strumenti operativi per l'analisi geografica e l'elaborazione dell'informazione. Le funzioni di Geoprocessing, a partire da dataset geografici esistenti, consentono di applicare ad essi delle funzioni analitiche e memorizzare i risultati in nuovi dataset[2].

### 2.1.1 Geodatabase

Un GIS è basato su un database tradizionale, e ne estende le funzionalità in modo da descrivere il mondo e la sua realtà in termini geografici/spaziali. I dataset GIS forniscono diverse tipologie di rappresentazione della realtà geografica:

- Collezioni di elementi su base vettoriale (insiemi di punti, linee e poligoni)
- Dataset raster, per rappresentare immagini e modelli
- Insiemi di dati da rilievo topografico
- Altri dati come indirizzi, nomi dei luoghi e informazioni cartografiche

I GIS inoltre consentono di strutturare i dati geografici in livelli tematici (layer ) all'interno di un dataset, ossia di raggruppare logicamente insiemi di oggetti omogenei (per es. il dataset delle Strade può essere strutturato in vari layer quali: rete autostradale, rete viaria principale e secondaria, ponti, tunnel,...). Quando i diversi layer sono georeferenziati, ossia hanno associata una posizione rappresentativa di quella reale sulla superficie terrestre, sono sovrapponibili (overlay) l'uno sull'altro. Molte delle relazioni spaziali tra i vari layer tematici possono essere derivate facilmente attraverso la loro georeferenziazione (es: sovrapposizione, distanza, intersezione, ecc)

### 2.1.2 Geovisualizzazione

Un GIS consente di produrre rappresentazioni geografiche di base ed avanzate (mappe) dei dati contenuti nei database geografici. Le mappe sono il principale strumento per presentare l'informazione geografica agli utenti e consentirne l'interazione. Le mappe sono generate attraverso potenti interfacce utente che consentono di eseguire analisi ed elaborazioni geografiche:

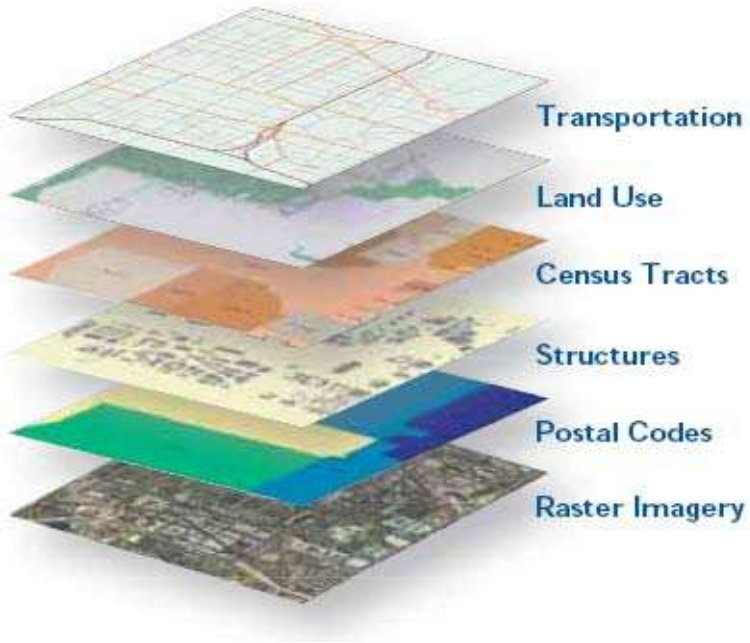


Figura 2.2: Esempio di dataset strutturato in Layer

sono disponibili a diversi livelli a partire dalle semplici interfacce disponibili su palmari (es. GPS) alle più evolute disponibili su WEB, fino alle potenti applicazioni GIS su desktop. Le mappe create con un GIS si differenziano da quelle statiche stampate perchè è possibile interagire con esse. Nella visualizzazione di un layer si possono utilizzare simbologie rappresentative di una qualunque combinazione degli attributi associati agli oggetti del layer stesso. Ad esempio si possono visualizzare strade di colore diverso a secondo dell'importanza, oppure si può specificare la dimensione del simboli dei pozzi sulla base della capacità produttiva.

### 2.1.3 Geoprocessing

I dataset geografici possono rappresentare misurazioni grezze (ad esempio immagini satellitari), informazioni interpretate e compilate dagli analisti (ad esempio strade, costruzioni e tipi di suolo), o informazioni derivate da altre sorgenti di dati usando algoritmi di analisi e di modellazione. Per Geoprocessing si intende l'elaborazione di dataset esistenti e/o la generazione di nuovi dataset, tramite l'applicazione di funzioni analitiche basate sulle relazioni spaziali tra gli oggetti geografici. Un GIS, di norma, include una ricca serie di tools per lavorare ed elaborare le informazioni geografiche; questi possono essere impiegati per operare su tutti gli oggetti di un GIS, dai dataset, agli elementi cartografici relativi alle mappe da stampare, alle colonne delle tabelle di attributi e così via.

## 2.2 Tipologia di dati geografici

Le tipologie di dati geografici principali sono il dato vettoriale e il dato raster. Vediamoli nel dettaglio:

- I dati vettoriali sono costituiti da elementi semplici quali punti, linee e poligoni, codificati e memorizzati sulla base delle loro coordinate. Un punto viene individuato in un Sistema Informativo Geografico attraverso le sue coordinate reali (x1, y1); una linea o un poligono attraverso la posizione dei sui nodi (x1, y1; x2, y2; ...). A ciascun elemento è associato un record del database informativo che contiene tutti gli attributi dell'oggetto rappresentato.

- Il dato raster permette di rappresentare il mondo reale attraverso una matrice di pixel. A ciascun pixel sono associate le informazione relative a ciò che esso rappresenta sul territorio. La dimensione del pixel (pixel size), generalmente espressa nell'unità di misura della carta (metri, chilometri etc.), è strettamente relazionata alla precisione del dato.

I dati vettoriali e i dati raster si adattano ad usi diversi. La cartografia vettoriale è particolarmente adatta alla rappresentazione di dati che variano in modo discreto (ad esempio l'ubicazione dei cassonetti dei rifiuti di una città o la rappresentazione delle strade o una carta dell'uso del suolo), la cartografia raster è più adatta alla rappresentazione di dati con variabilità continua (ad esempio un modello digitale di elevazione)[3].

## 2.3 Map Object 2

Affinché un applicazione possa riprodurre le funzioni tipiche di un sistema GIS, l'Environmental System Research Institute (ESRI) mette a disposizione un set di librerie denominato MapObjects. MapObjects permette ad un'applicazione di gestire una mappa ed implementare una vasta gamma delle funzioni di un GIS fra cui:

- Visualizzare una mappa con layer multipli, quali strade, fiumi e confini geografici
- Disegnare elementi grafici come punti, linee, ellissi, rettangoli e poligoni
- Aggiungere alla mappa descrizioni testuali

- Selezionare elementi lungo linee, o all'interno di aree delimitate
- Selezionare elementi entro una specifica distanza da altri (analisi di prossimità)
- Produrre nuovi file shape in relazione alle elaborazioni svolte
- Aggiungere alla mappa immagini ottenute da fotografie aeree o satellitari
- Produrre mappe tematiche evidenziando le variazioni di una caratteristica mediante opportune tecniche (Value Map Rendering, Dot Density Rendering, Chart Rendering, Class Breaks Rendering)
- Localizzare la posizione geografica di un oggetto conoscendone l'indirizzo (Geocoding)

MapObjects include un controllo ActiveX chiamato map control object, che permette di visualizzare e gestire una mappa usando un set di ulteriori 45 oggetti raggruppabili in diverse categorie[4].

### 2.3.1 Struttura delle mappe

L'insieme dei layers costituenti la mappa viene gestito mediante l'oggetto layers collection che è una collezione ordinata di oggetti di tipo map layer. La mappa viene prodotta sovrapponendo i diversi layers (oggetti map layer) secondo l'ordine in cui essi sono memorizzati nell'oggetto layers collection; la classe layer collection fornisce metodi che permettono di aggiungere od eliminare layer dalla collezione, o modificarne l'ordinamento. Ogni elemento

della collezione (oggetto map layer) fornisce un livello di astrazione per la rappresentazione di uno strato di informazione geografica indipendente dal formato di memorizzazione, sia esso raster oppure vettoriale. Gli oggetti map layer dispongono dei metodi e delle proprietà necessarie per effettuare un collegamento con il database geografico, nel quale sono mantenuti i valori degli attributi di ogni elemento contenuto nel layer stesso. E' possibile effettuare operazioni di selezione di particolari elementi geografici appartenenti al layer usando come parametro la distanza da un determinato punto o elemento, oppure selezionando tutti gli elementi all'interno di una data forma geometrica o in contatto con essa, ecc. Oltre ai layer presenti nella lista layer collection, esiste un ulteriore layer rappresentato dall'oggetto tracking layer, che contiene tutti gli eventi geografici (elementi geografici la cui posizione può variare nel tempo) e tutte le forme geometriche disegnate sulla mappa. Il tracking layer risulta sempre in primo piano rispetto agli altri layer e gli elementi disegnati in esso sono visualizzati in ordine inverso rispetto all'ordine con cui sono disegnati (l'elemento disegnato per ultimo viene posto in primo piano rispetto a tutti gli altri). Gli eventi geografici sono rappresentati per mezzo di oggetti Geo Event; la classe che li implementa espone i metodi necessari per espletare le funzioni di spostamento da un punto all'altro della mappa e gode delle proprietà che permettono di definirne caratteristiche come il simbolo da usare per la rappresentazione dell'evento, il colore , la dimensione e la forma.

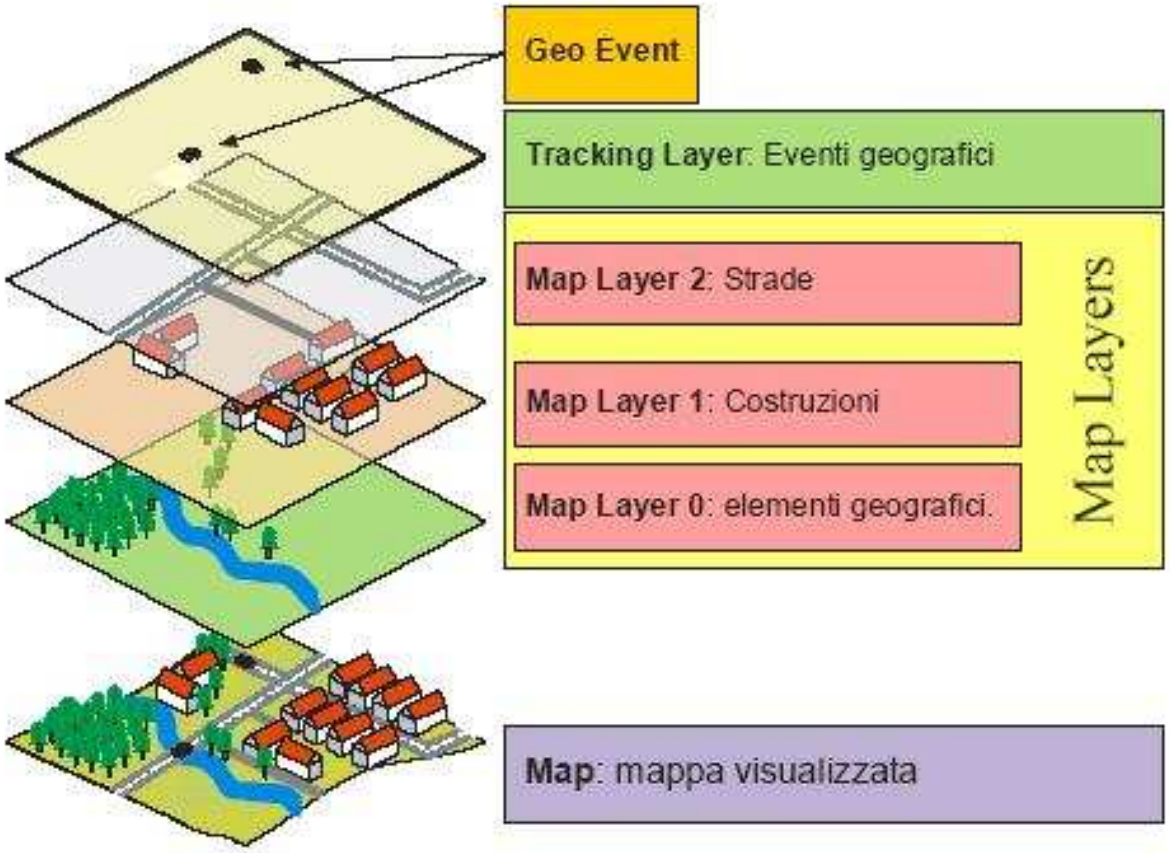


Figura 2.3: Struttura di MapObject

# Capitolo 3

# Integrazione degli strumenti

In questo capitolo descriviamo un'applicazione per la gestione dei dati geografici sviluppata utilizzando le librerie MapObject. Ne descriviamo in primo luogo l'integrazione con DARIA attraverso l'esportazione dei dati geografici dal formato TeleAtlas al formato utilizzato da DARIA e la gestione delle prenotazioni.

## 3.1 Integrazione con DARIA

La gestione di un sistema DIAL-A-RIDE prevede l'interazione tra un operatore esperto, un software di ottimizzazione e un insieme di dati geografici. Le fasi in cui avviene lo scambio di dati tra l 'applicazione e DARIA sono due: nella prima l'applicazione crea delle strutture per la traduzione dei dati geografici nei due formati e invia a DARIA un database contenente il grafo stradale su cui effettuare l'ottimizzazione; nella seconda fase salva le richieste di trasporto degli utenti con relativi punti di partenza e di arrivo in un

database.

### 3.1.1 Esportazione e traduzione dal formato TeleAtlas

Per apprendere la topologia della rete il software DARIA richiede in input un file di testo. Quest'ultimo contiene il numero totale dei vertici seguito da tutti gli archi (nodo iniziale, nodo finale), ed il tempo di percorrenza di ogni tratto. I nodi dei tronchi stradali nel formato TeleAtlas riportati nel file utilizzato da DARIA sono rinominati, e salvati in opportune strutture dati per mantenere la corrispondenza tra i due formati. MapToDaria è utilizzata per trasformare il formato TeleAtlas nel formato di DARIA e DariaToMap esegue la conversione opposta. MapToDaria è implementato mediante un dizonario, mentre DariaToMap è implementato mediante array; riportiamo il codice di inizializzazione delle strutture:

```
esri.Recordset recd = Layer.Records; //estratti tutti i record
while (recd.EOF==false)
{
 //lettura dei nodi dai dati geografici
 ControlloF=recd.Fields.Item(NomeNodoF).ValueAsString;
 ControlloT=recd.Fields.Item(NomeNodoT).ValueAsString;
 if ( MapToDaria.Contains(ControlloF)==false )
    {
     MapToDaria.Add(ControlloF,i);
     DariaToMap[i]=ControlloF;
    }
```

```
        if ( MapToDaria.Contains(ControlloT)==false )
        {
          MapToDaria.Add(ControlloT,i);
          DariaToMap[i]=ControlloT;
        }
        recd.MoveNext();
      }
```

Il formato TeleAtlas non prevede un campo con il tempo di percorrenza di ciascun tratto; per calcolarlo utilizziamo la formula del moto rettilineo uniforme, t=s/v. La velocita' media utilizzata varia in base al campo FRC, che identifica il tipo di strada. Il valore di questo campo varia da 0 a 8, dove 0 sono le strade più importanti, 8 quelle di minore importanza.

L'insieme dei nodi tradotti viene successivamente salvato in un file di testo. Il file TeleAtlas utilizzato come sorgente dei dati geografici è nw.shp poichè contiene tutti i dati topologici necessari alle applicazioni inerenti all'instradamento dei veicoli. Nel formato DARIA ciascun senso di marcia è rappresentato da un arco mentre nel formato TeleAtlas ogni record rappresenta un tronco stradale a senso unico ovvero a doppio senso. Per esempio ogni tratto di strada a doppio senso rende necessario salvare nel file di testo 2 archi. Il campo che indica il senso di marcia nel formato TeleAtlas è SOL (Side Of Line), i valori possibili sono:

- 0 (doppio senso es. A-B,B-A)

- 1 (da sinistra es. A-B)
- 2 (da destra es. B-A)

Nel codice di seguito la variabile SensoDiMarcia identifica il nome del campo SOL, il controllo viene fatto con 2 if() :

```
//Scrivo sul file DataDaria.txt
//il primo valore e' il numero tot dei nodi
DatabaseDaria.WriteLine(DariaToMap.Count.ToString());
recd.MoveFirst();
while (recd.EOF==false)
{
//nodo di partenza
ValNodoF=MapToDaria[recd.Fields.Item(NomeNodoF).ValueAsString].ToString();
//nodo di arrivo
ValNodoT=MapToDaria[recd.Fields.Item(NomeNodoT).ValueAsString].ToString();
//tipo di strada
ValTipoStrada=(int)recd.Fields.Item(TipoStrada).Value;
switch (ValTipoStrada)
{
     case 0:
            Vel=70;
            break;
     case 1:
            Vel=65;
            break;
     case 2:
```

```
                Vel=60;
                break;
        ...
        case 8:
                  Vel=30;
                  break;

    }
double ValoreMetri = (double)recd.Fields.Item(Metri).Value;
double ValoreMetri = (double)recd.Fields.Item(Metri).Value;
int CalcoloTempo = (int)((ValoreMetri*Vel)/3.6)/60;
//Controllo il senso di marcia
if((int)recd.Fields.Item(SensoDiMarcia).Value==0||
(int)recd.Fields.Item(SensoDiMarcia).Value==1)
{
        DatabaseDaria.WriteLine(ValNodoF + " " + ValNodoT + " " + Tempo );
}
else  if((int)recd.Fields.Item(SensoDiMarcia).Value==0||
(int)recd.Fields.Item(SensoDiMarcia).Value==2)
{
        DatabaseDaria.WriteLine(ValNodoT + " " + ValNodoF + " " + Tempo );
}
  recd.MoveNext();
}
DatabaseDaria.Close();
```

### 3.1.2 Realizzazione di un interfaccia per la prenotazione

I campi del sistema informativo utilizzati per la gestione delle prenotazioni sono :

- NAME: nome della via (gc.shp)
- L_STRUCT e R_STRUCT: rappresenta la struttura del lato sinistro e destro del tronco stradale, rispettivamente. Tal campo assume valore 2 se il lato considerato ha numeri pari, 3 se dispari o 4 se misti (gc.shp)
- L_F_ADD-L_T_ADD o R_F_ADD-R_T_ADD : numero civico iniziale e finale del tronco considerato. Le lettere iniziali L e R indicano il lato della strada (gc.shp)
- ID: rappresenta l'identificativo del tronco stradale. E' una chiave esterna che permette la relazione d'unione tra le tabelle corrispondenti di gc.shp e nw.shp
- F_JNCTID e T_JNCTID: nodo iniziale e finale di un tronco stradale (nw.shp)

L'interfaccia realizzata per le prenotazioni permette all'operatore di inserire l'indirizzo di partenza, l'indirizzo di arrivo, gli orari e la data di servizio e il numero di persone da servire. Per fornire a DARIA i dati richiesti è necessario tradurre gli indirizzi (di partenza e arrivo) nella terna: nodo iniziale, nodo finale, $\lambda$. Ricordiamo che $\lambda$ è la frazione di arco a cui corrisponde il numero civico rispetto all'intero tronco stradale. Le informazioni immesse dall'operatore vengono passate come parametri alla funzione SalvaRichieste(), la quale

a sua volta richiama la funzione TrovaArco() per ottenere l'ID del tratto di strada che contiene l'indirizzo (di arrivo o di partenza), la frazione dell'arco corrispondente e la direzione della strada. Vediamo nel dettaglio come vengono ricavati questi dati:

- Viene effettuata un'interrogazione nel layer gc.shp per ottenere i record che contengono il nome della via presente nell'indirizzo, utilizzando il metodo SearchExpression:

```
Layer = (esri.MapLayer)Map1.Layers.Item("gc.shp");
recd = Layer.SearchExpression("NAME = '"+Via+"'");
```

- L'insieme dei record ottenuti dall'interrogazione viene scandito per verificare l'appartenenza del numero civico richiesto al tronco stradale. A tale scopo viene utilizzata la funzione IsPresent(), che riceve come parametri i due estremi; restituisce il valore logico vero se il numero è compreso nell'intervallo

```
if(Numero%2==0)
{    if((int)recd.Fields.Item("L_STRUCT").Value==2||
(int)recd.Fields.Item("L_STRUCT").Value==4)
        {         if(IsPresent(Numero,recd.Fields.Item("L_F_ADD").Value,
recd.Fields.Item("L_T_ADD").Value))
                {
```

- Una volta trovato il tratto di strada richiesto viene calcolata la frazione dell'arco attraverso la funzione CalcoloLambda(), viene ricavato l'ID e il senso di marcia, in base al quale è calcolata la direzione dell'arco,

rappresentata dalla variabile DirStreet. Quest'ultima può assumere 2 valori: 0, se l'arco va dal nodo di partenza a quello d'arrivo del tronco stradale, e 1, altrimenti. Il codice di seguito prende in esame il caso in cui il numero civico cercato si trova sul lato sinistro della strada:

```
ID = Convert.ToUInt64(recd.Fields.Item("ID").Value);
NumF = recd.Fields.Item("L_F_ADD").Value;
NumT = recd.Fields.Item("L_T_ADD").Value;
Lambda = CalcoloLambda(Numero,NumF,NumT);
int SOL = (int)recd.Fields.Item("SOL").Value;
if (SOL==0 || SOL==1)
{
      DirStreet=1;
}
if(SOL==2)
{
      DirStreet=0;
}
```

Ricavati i dati necessari, per ottenere i nodi viene effettuata una successiva interrogazione nel layer nw.shp utilizzando il valore del campo ID calcolato precedentemente, infine in base alla direzione vengono stabiliti i nodi iniziale e finale dell'arco.

```
recd1 = Layer.SearchExpression("ID = "+ID.ToString());
      recd1.MoveFirst();
      if(DirStreet==0)
```

```
{
   NodoFrom = recd1.Fields.Item("F_JNCTID");
   NodoTo = recd1.Fields.Item("T_JNCTID");
}
else
{
   NodoFrom = recd1.Fields.Item("T_JNCTID");
   NodoTo = recd1.Fields.Item("F_JNCTID");
}
```

La procedura descritta viene ripetuta per l'indirizzo di partenza e per l'indirizzo d'arrivo inseriti dall'utente. Successivamente i nodi in formato TeleAtlas sono convertiti nel formato DARIA attraverso la struttura MapToDaria, i dati ottenuti vengono salvati su un database Access (utilizzando i driver OleDB).

# Capitolo 4

# Interfaccia Utente

In questo capitolo trattiamo l'implementazione del geocoding, e la visualizzazione dei percorsi e delle fermate.

## 4.1 Geocoding

Il geocoding (georeferenziazione per indirizzi) è il processo che converte un riferimento spaziale implicito (indirizzo, codice postale) in un riferimento spaziale esplicito (longitudine e latitudine, coordinate planimetriche) [5]. La qualità dei risultati dei processi di localizzazione per indirizzo è legata principalmente ad un corretto utilizzo delle regole di normalizzazione e ricerca impostate. La struttura della rete stradale italiana è più complessa di quella anglosassone in base alla quale sono stati formalizzati i principali strumenti di geocodifica: un indirizzo nello stradario italiano può essere specificato in modi diversi e presentare suffissi diversi, ad esempio: via andrea del castagno, via andrea castagno del, via a. del castagno, via del castagno, via, viale,

località, piazza, piazzale... Il modulo di geocodifica di ESRI utilizza la cartografia di base e fornisce una misura qualitativa sul risultato dell'operazione di localizzazione. Un'opportuna procedura di inserimento garantisce quindi un adeguata geocodifica. Possiamo dividere l'implementazione del geocoding in due parti principali, la prima consiste nella preparazione del geocoder, l'oggetto che individua la locazione geografica dell'indirizzo, la seconda nella lettura delle coordinate del punto sulla mappa.

Vediamo nel dettaglio i passaggi principali :

- Vengono inizializzati gli oggetti Geocoder e Standardizer, il quale permette la standardizzazione dell'indirizzo secondo le regole contenute nella cartella georules

```
geoc = new esri.GeocoderClass();
stan = new esri.StandardizerClass();
stan.StandardizingRules=path+"\\georules\\us_addr.stn;
```

- Si associa l'oggetto standardizer alla proprietà Standardizer del geocoder, si caricano le regole di ricerca (matching rules) nel geocoder, e si assegna una rete stradale al geocoder settando la proprietà StreetTable con un opportuno geodataset. Un geodataset adeguato può consistere in uno shapefile o un ARC/INFO coverage che contiene i nomi delle strade o informazioni territoriali (nel nostro caso gc.shp)

```
geoc.Standardizer = stan;
geoc.MatchRules = path + "\\georules\\us_addr1.mat";
geoc.StreetTable = gds;
```

- Nel Geocoder sono presenti una serie di variabili chiamate MatchVariable che contengono il riferimento ai campi della StreetTable utilizzati per la localizzazione dell'indirizzo, queste vanno impostate con i nomi dei campi del geodataset utilizzato. Nel metodo set_MatchVariableField() il primo parametro è il nome della MatchVariable, il secondo è il nome del campo del geodataset utilizzato (nel codice vengono usate delle variabili di tipo string inizializzate precedentemente).

```
geoc.set_MatchVariableField("FromLeft",FROM_LEFT);
geoc.set_MatchVariableField("ToLeft",TO_LEFT);
geoc.set_MatchVariableField("FromRight",FROM_RIGHT);
geoc.set_MatchVariableField("ToRight",TO_RIGHT);
geoc.set_MatchVariableField("PreDir",PRE_DIR);
geoc.set_MatchVariableField("PreType",PRE_TYPE);
geoc.set_MatchVariableField("StreetName",STREET_NAME);
geoc.set_MatchVariableField("StreetType",STREET_TYPE);
geoc.set_MatchVariableField("SufDir",SUF_DIR);
geoc.set_MatchVariableField("LeftZone",LEFT_ZONE);
geoc.set_MatchVariableField("RightZone",RIGHT_ZONE);
```

- Un geocoder deve avere un indice valido prima di poter essere usato per la localizzazione di indirizzi, con il metodo AddIndex() si specificano gli indici che si vogliono utilizzare per la StreetTable, l'ultimo parametro (mgIndexTypeSoundex o mgIndexTypeNormal) indica se l'indice pu essere soggetto a errori di sintassi (mgIndexTypeSoundex), come gli indirizzi, o è un codice numerico normale (mgIndexTypeNormal), come i codici postali o gli ZIP code.

```
if (geoc.IndexStatus() != esri.IndexStatusConstants.mgIndexExists)
 {
    geoc.AddIndex(STREET_NAME,"",esri.IndexTypeConstants.mgIndexTypeSoundex);
    geoc.AddIndex(LEFT_ZONE,RIGHT_ZONE,esri.IndexTypeConstants.mgIndexTypeNormal);
 }
```

- L'ultimo componente da configurare è la proprietà SearchQueries che rappresenta la query di ricerca all'interno della StreetTable, la ricerca avviene solo in base all'indirizzo immesso (nome della strada e numero civico). In alternativa la stringa "SN? & ZN" codifica la ricerca per nome, numero civico e ZIP code:

```
esri.Strings strcQueries = new esri.StringsClass();
strcQueries.Add ("SN?");
geoc.SearchQueries=strcQueries;
```

La parte descritta sopra viene eseguita al caricamento del layer gc.shp, la seconda parte in cui avviene la geocodifica è stata implementata con la funzione Geocoding() che riceve come parametro l'indirizzo da cercare e restituisce un oggetto esri.Point che identifica un punto sulla mappa TeleAtlas, con relative coordinate:

- Per effettuare un'operazione di geocodifica vengono utilizzate 3 variabili: punto, valore di ritorno della funzione, addLoc (address location) che rappresenta l'indirizzo restituito dal geocoder, e matchStatus che dirà se il geocoding ha avuto successo.

```
esri.Point Punto=null;
esri.AddressLocation addLoc;
esri.GeocodeSuccessConstants matchStatus;
```

- Il metodo StandadizerAddress() standardizza l'indirizzo e carica il risultato nelle MatchVariables specificate nelle Standardizing rules, il metodo GenerateCandidates() genera i risultati dalle MatchVariables, li carica in un array di AddressLocation, restituisce una esri.GeocodeSuccessConstant che ci permette di valutare il successo o meno del geocoding

```
stan.StandardizeAddress(Indirizzo);
matchStatus = geoc.GenerateCandidates();
```

- Per conoscere il risultato della geocodifica analizziamo i possibili valori assunti dalla variabile matchStatus, ne ricaviamo il punto oppure restituiamo un valore nullo se non è stata trovata la locazione sulla mappa. Il parametro 0 passato al metodo geoc.LocateCandidate() rappresenta la locazione che più si avvicina (o corrisponde perfettamente nel caso di GeocodeSuccessSingleBest) all'indirizzo cercato.

```
switch (matchStatus)
{
   case esri.GeocodeSuccessConstants.mgGeocodeSuccessSingleBest:
       addLoc = geoc.LocateCandidate(0);
       Punto = addLoc.location;
       break;
   case esri.GeocodeSuccessConstants.mgGeocodeSuccessMultipleBest:
       addLoc = geoc.LocateCandidate(0);
```

```
            Punto = addLoc.location;
            break;
        case esri.GeocodeSuccessConstants.mgGeocodeSuccessPartial:
            addLoc = geoc.LocateCandidate(0);
            Punto = addLoc.location;
            break;
        case esri.GeocodeSuccessConstants.mgGeocodeFailed:
            Punto = null;
            return Punto;
            break;
    }
```

La procedura descritta viene utilizzata dalle procedure di visualizzazione descritte nei paragrafi successivi.

## 4.2 Visualizzazione dei percorsi e delle fermate

Come abbiamo visto, a seguito della creazione di un sistema di prenotazione per le fermate è stato sviluppato un database nel quale sono state salvate le richieste degli utenti. DARIA elabora le informazioni in esso contenute rendendo disponibili i risultati attraverso un file di testo nel quale sono descritti il numero di veicoli utilizzati per garantire un livello di servizio ottimale e i percorsi di ognuno di essi. Il file può essere suddiviso in 3 parti :

- La prima riga specifica il numero di veicoli
- La seconda parte descrive le fermate, per ogni veicolo

- La terza parte descrive i percorsi e le sequenze di archi che li compongono.

L'applicativo sviluppato acquisisce i dati contenuti nel file generato da DARIA e visualizza sulla mappa il tracciato dei percorsi, e le relative fermate. Descriviamo in questo paragrafo le due procedure usate nell'implementazione.

### 4.2.1 Visualizzazione delle fermate

Ogni richiesta fatta da un utente, comprende un indirizzo di partenza, un indirizzo d'arrivo, gli orari desiderati di partenza e arrivo e la data in cui si vuole che venga svolto il servizio, le fermate visualizzate sulla mappa sono gli indirizzi di partenza e arrivo degli utenti. I dati forniti da DARIA comprendono un identificativo e l'orario in cui avviene la fermata, l'ID ci permette di selezionare dal database delle prenotazioni la richiesta relativa alla fermate e di identificare una fermata di carico o scarico. La funzione Geocoding() viene usata nuovamente per ottenere gli oggetti geometrici di tipo punto relativi alle fermate da visualizzare.

```
if(Tappa==DAR_Carico)
{
    //Selezionare campo indFrom
     string query = @"SELECT indFrom FROM TB_CALLS WHERE id = "+Tappa.ToString();
     oleDbSelectCommand = new System.Data.OleDb.OleDbCommand(query,conn);
     reader = oleDbSelectCommand1.ExecuteReader();
     {
          esri.Point punto;
```

```
            punto = Geocoding(reader.GetString(0));
        }
    }
    if(Tappa==DAR_Scarico)
    {
       //Selezionare campo indTo
        string query = @"SELECT indTo FROM TB_CALLS WHERE id = "+TappaTo.ToString();
        this.oleDbSelectCommand = new System.Data.OleDb.OleDbCommand(query,conn);
        reader = oleDbSelectCommand1.ExecuteReader();
        while(reader.Read())
        {
            esri.Point punto;
            punto = Geocoding(reader.GetString(0));
        }
    }
```

Le librerie MapObject, come visto nell'introduzione, mettono a disposizione gli oggetti GeoEvent per rappresentare gli eventi geografici, nel nostro caso la fermata del veicolo è stata considerata come un evento geografico. Per visualizzare un evento bisogna aggiungerlo al TrackingLayer, che ricordiamo essere il layer che contiene tutte le forme geometriche e gli eventi disegnati sulla mappa, con il metodo AddEvent(). La visualizzazione di un punto prevede un'altro parametro, di tipo intero, che identifica il simbolo grafico da utilizzare. Vediamo i passaggi fondamentali dell'implementazione fatta:

- Inizialmente viene creato un simbolo nel TrackingLayer di tipo punto, al quale è assegnato un colore di riferimento in base al percorso

```
Map1.TrackingLayer.get_Symbol(i).SymbolType = esri.SymbolTypeConstants.moPointSymb
Map1.TrackingLayer.get_Symbol(i).Color = colore[i];
```

- E' richiamato il metodo AddEvent(), i parametri passati sono punto, il risultato del geocoding, e i, l'indice assegnato al Simbolo impostato nel tracking layer.

```
Map1.TrackingLayer.AddEvent(punto, i);
```

### 4.2.2 Visualizzazione dei percorsi

Le informazioni ricevute da DARIA inerenti ai percorsi sono la sequenza di archi (in formato DARIA) e il numero di nodi, per ogni percorso. La metodologia usata per l'implementazione è stata quella di considerare ogni percorso come un'unica linea formata dalla concatenazione degli archi che lo compongono. Successivamente ogni linea viene disegnata direttamente sul TrackingLayer con il metodo DrawShape(), che permette la rappresentazione sulla mappa di un oggetto geometrico. Di seguito descriviamo le operazioni svolte:

- Per ogni arco si ricerca il tronco stradale corrispondente nel layer gc.shp, utilizzando la codifica dei nodi nel formato TeleAtlas

```
arco = Layer.SearchExpression
("F_JNCTID = " + ValNodoF + " AND T_JNCTID = " + ValNodoT);
```

- Si ottiene dal campo shape di ogni tronco un oggetto linea composto di una o più parti. Ciascuna di esse viene concatenata al percorso.

```
l = (esri.Line)arco.Fields.Item("shape").Value;
for (int c=0;c<l.Parts.Count;c++)
{
    linea[i].Parts.Add((esri.Points)l.Parts.Item(c));
}
```

- Il percorso viene disegnato attraverso il metodo DrawShape(). Come per il metodo AddEvent() è necessario specificare il simbolo utilizzato per la visualizzazione.

```
Map1.DrawShape(linea[i],simboloArco[i]);
```

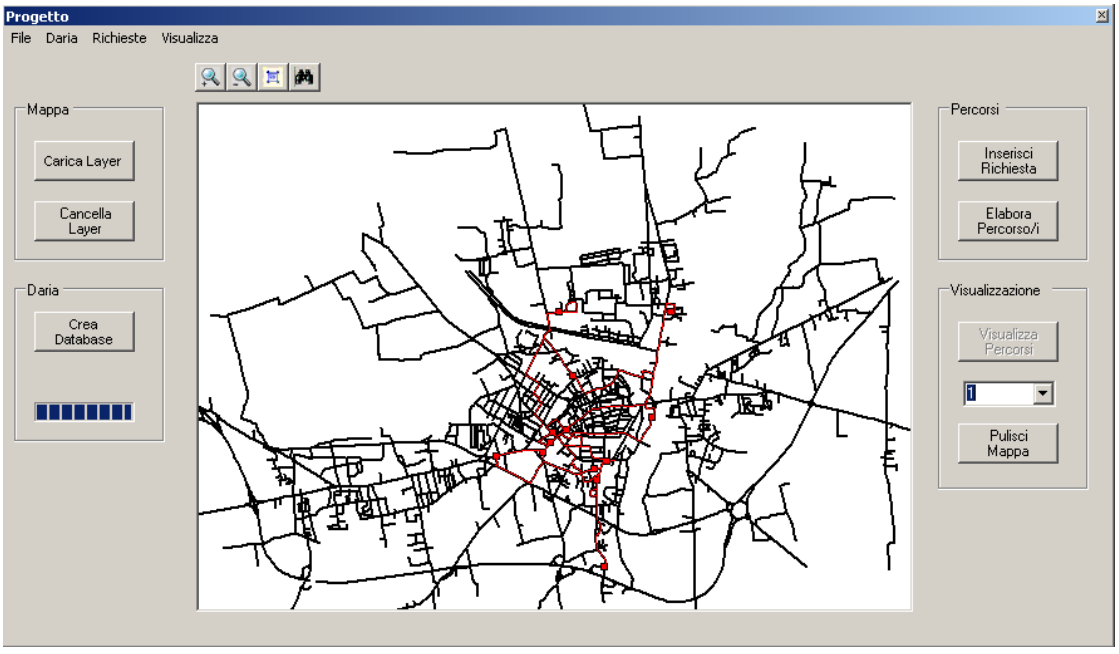


Figura 4.1: Visualizzazione di un percorso con relative fermate

# Conclusioni

Con questa tesi abbiamo raggiunto l'obbiettivo di sviluppare un applicativo che integra le librerie di un sistema informativo geografico (MapObject) con algoritmi per l'ottimizzazione di un sistema a chiamata statico di tipo porta a porta e fermata a fermata (DARIA) .

In una prima fase è stata approfondita la conoscenza dei sistemi informativi geografici e delle loro caratteristiche principali per progettare e realizzare un'interfaccia grafica che utilizza le librerie MapObject per caricare e gestire i dati territoriali (nel formato standard TeleAtlas) e per rendere possibile l'interazione dell' utente con la mappa stessa (zoom, selezione, pan, ecc). In secondo luogo abbiamo analizzato gli algoritmi esistenti per l'esportazione in formato DARIA dei dati relativi alle richieste di servizio pervenute dagli utenti (punti di origine e destinazione dei viaggi, finestre temporali) per integrarle nell'applicativo. Successivamente abbiamo sviluppato le strutture dati utili a mantenere la corrispondenza tra i tronchi stradali in formato TeleAtlas e il formato DARIA. Per ottenere da un indirizzo (via e numero civico) le coordinate geografiche abbiamo studiato il processo di geocoding e la sua implementazione attraverso le librerie MapObject. Infine abbiamo sviluppato un sistema di visualizzazione che consentisse la rappresentazione

dei percorsi e delle fermate elaborati da DARIA. L'applicazione puo' essere utilizzata dalle centrali operative che gestiscono siteme DAR statici per ottenere un'informazione qualitativa sullo stato di utilizzo della rete stradale da parte del sistema. Inoltre l'applicazione puo' essere utilizzata come strumento di pianificazione per il dimensionamento della flotta.

Gli sviluppi dell'applicazione sono legati alla risoluzione delle problematiche relative ai sistemi di trasporto a chiamata dinamici. La messa in opera di sistemi DAR dinamici richiede il continuo monitoraggio dello stato della rete e della flotta di veicoli che si muove su di essa, si rende necessaria quindi un'integrazione con apparecchiature per la rilevazione della posizione dei veicoli (GPS), un sistema di comunicazione tra i veicoli e la centrale operativa, e la visualizzazione sulla mappa delle informazioni ricevute dal rilevatore di posizione installato sui veicoli. Le apparecchiature che si appoggiano alla rete satellitare di posizionamento globale (GPS) sono le più indicate per la localizzazione dei veicoli, essendo in grado di fornire in tempo reale informazioni accurate. Il sistema di comunicazione potrebbe essere basato su un modello *client/server* appoggiato alla rete GPRS (Global System for Mobile communications) che offre una adeguata capacità di canale, tempi di connessione veloci e costi contenuti. La rappresentazione sulla mappa dei veicoli necessita di una traduzione dell'informazione dal sistema di riferimento geografico (lat/long) al sistema di riferimento utilizzato dalla mappa stessa (es. UTM-WGS 1984 ) ed di una estensione del codice in un ambiente *multithread* per la visualizzazione contemporanea dello spostamento di più veicoli[6]. La trasformazione delle coordinate dal sistema di riferimento geografico al sistema di riferimento utilizzato dalla mappa è supportato dalle librerie MapObject, per

questa trasformazione è necessario che nello shapefile utilizzato sia associato un sistema di proiezione delle coordinate, il ProjCoordSys object (PRJ files), che contiene le informazioni necessarie alla trasformazione[7]. Infine è possibile utilizzare la conoscenza acquisita sui sistemi informaztivi geografici per integrare l'applicazione con GIS non commerciali. Ad esempio GRASS (Geographic Resources Analysis Support System) è il GIS open source più diffuso. Distribuito sotto licenza GNU GPL, secondo la quale il codice sorgente è disponibile e modificabile a patto di ridistribuire le modifiche, GRASS è un Sistema Informativo Territoriale estremamente evoluto con funzioni che vanno dall'analisi spaziale alla modellistica ambientale, dalla generazione di mappe tematiche all integrazione con DBMS, dalla visualizzazione 2D e 3D di dati spazialmente distribuiti alla gestione e memorizzazione di dati. Scritto interamente in ANSI-C, supporta le piattaforme Linux, Mac, Sun Solaris, Silicon Graphics Irix, HP-UX, DEC-Alpha, e Windows 95/98/NT/XP[8].

# Bibliografia


[1] Matteo Salani, *"Daria 3.0"*, Tesi di laurea presso il Polo Didattico di Crema 2001

[2] *"Gis: Concetti e requisiti"*, www.esriitalia.it, 2005.

[3] *"Sistema Informativo Geografico"*, http://it.wikipedia.org/wiki/Sistema_informativo_geograf 2005.

[4] *"MapObjects-Windows Edition"*, http://www.esri.com/software/mapobjects/about/overview.l 2005.

[5] *"Using Data in GIS"*, http://www.gis.com/implementing_gis/data/usingdata.html, 2005.

[6] Simone Redondi, *"Sviluppo di interfaccie utente basate su cartografia digitale"*, Tesi di laurea Politecnico di Milano, 2003.

[7] ESRI, http://support.esri.com/, 2005

[8] GRASS Development Team, *"GRASS: Introduction"*, http://www.geo.unipr.it/~grassmirror/intro/general.php, 1999-2004.

[9] ESRI, *"MapObjects 2.3 Help"*, 2005.

[10] Ulrico Hoepli Editore, *"C# Guida per lo sviluppatore"*, 2001.