Java, GIS e Software libero

di Maurizio Napolitano (ITC-irst)
napo@itc.it

1. Introduzione

In questo articolo verranno presentate le più diffuse soluzioni java che la comunità del software libero offre nelle problematiche relative ai sistemi GIS.

1.1 Cos' è un sistema GIS?

Partendo dall'acronimo è facile intuire quali siano le problematiche risolte da questi sistemi. *G.I.S*. sta per "*Geographical Information Systems*", la cui traduzione italiana più diffusa è "Sistemi Informativi Territoriali", pertanto una applicazione che rientra in questa categoria permette calcoli su rappresentazioni di un territorio (secondo un preciso sistema di riferimento) a cui è associata una base dati alfanumerica.
Le soluzioni così offerte vanno dalla creazione di mappe tematiche (es.: percentuale di determinati tipi di piante presenti su un area o età media degli abitanti delle province d'Italia), alla pianificazione di progetti stradali, passando per l'elaborazione di calcoli complessi come possono essere quelli dell'inquinamento acustico, o delle mappe tridimensionali.
Chiaramente, non sono esclusi dalla lista i *tool* necessari per la creazione dei dati.

1.2 Le tipologie dei dati

Le tipologie di dati utilizzati nei sistemi GIS sono due:

- dati *raster*;
- dati vettoriali.

Nella prima categoria sono incluse tutte le tipologie di dato di

tipo continuo, pertanto foto aeree o carte tecniche, oppure modelli digitali del terreno (matrici che rappresentano griglie che ricoprono una determinata area di terreno, di cui si conoscono le coordinate di ogni singola cella e l'altezza dal livello del mare), a cui sono associate informazioni per recuperare le coordinate geografiche secondo il sistema di riferimento.
La raccolta di questo tipo di dati è molto costosa (si fa uso di altimetri, foto satellitari e foto aeree), inoltre richiede molta potenza di calcolo per correggere gli errori di distorsione dovuti alla necessità di una rappresentazione bidimensionale della Terra (operazione di ortorettificazione).
Nella seconda categoria, invece, sono inclusi dati di tipo discreto che, attraverso delle geometrie (punti, linee e poligoni), rappresentano elementi presenti sul territorio (es.: distributori di benzina, bancomat, fiumi, strade, confini amministrativi, centri urbani, laghi, …); per ciascuno di essi sono poi associati degli attributi (es.: tipologia di distributore di benzina, nome dell'istituto di credito proprietario del bancomat, portata media del fiume, tipologia di strada, nome della regione rappresentata nel confine, quantità di abitanti presenti nel centro urbano, tipologia di fauna ittica presente nel lago, …).
In questo caso la raccolta dei dati può avvenire attraverso apposite strumentazioni (es.: l'uso di GPS) o, più semplicemente partendo dalla base *raster* (digitalizzazione da foto aree o satellitari), o dall'*input* di un utente che fa uso di un software GIS basandosi su una mappa georeferenziata (un'immagine a cui è associato un sistema di coordinate).
Chiaramente, una volta individuate le geometrie, il passo successivo sarà quello di associare ad ogni singolo elemento degli attributi.

1.3 Soluzioni presentate

I *tool* presentati permetteranno ad uno sviluppatore, o ad un semplice utente, l'uso e la creazione di applicazioni in grado di gestire dal semplice al complesso (dalla mappa tematica al *webgis*, dal sistema di ricerca di percorsi a sistemi di simulazione di esplosioni chimiche) sulla base dei dati vettoriali, utilizzando i dati *raster* unicamente come sfondo (pertanto foto aree e carte tecniche georeferenziate). Per quello che riguarda le analisi sulla base di dati *raster* si rimanda a *tool* quali Grass (http://grass.itc.it) e Landserf (http://www.landserf.org), facendo presente che il primo, sviluppato in C/C++, è distribuito su diverse piattaforme sotto licenza GPL, mentre il secondo è in Java, ma non è distribuito con una licenza

libera o analoga che ne permetta l'accesso e modifica dei sorgenti.

2. Gestire i dati

2.1 Procurarsi i dati

Uno dei primi problemi è quello di avere una base dati GIS da cui partire.
Allo stato attuale, in Italia, non esiste una base di dati libera da cui attingere. Ogni pubblica amministrazione regionale, o provinciale, o comunale, dispone di un ufficio S.I.A.T. (Servizio Informativo Ambiente Territorio) dove, se sono raccolti i dati in formato elettronico, questi sono in vendita. Ciò non accade in altri stati come, ad esempio gli USA, dove, invece, ogni tipo di dato prodotto dalla pubblica amministrazione è accessibile a chiunque (eccezion fatta per una lista di stati considerati a loro avversi).
Navigando in rete è possibile recuperare del materiale gratuito. Un buon punto di partenza (ma non solo per i dati) è il portale FREEGIS (http://www.freegis.org), inoltre, aziende quali ESRI, TeleATLAS e NAVTECH (che sono specializzate anche nella vendita di dati costantemente aggiornati) offrono qualche *download* gratuito.
Qualche temerario ogni tanto si lancia nella raccolta di informazioni utilizzando le immagini georeferenziate che si trovano in rete su siti Web che offrono servizi *webgis* per la ricerca di luoghi, o percorsi stradali.
Questa operazione, comunque, non è consigliabile in quanto, prima di agire occorre essere sicuri sia della licenza con cui le mappe sono distribuite su Web, sia che il sistema di riferimento utilizzato sia facilmente ricostruibile.

2.2 Il formato dei dati

Una volta ottenuti i dati si apre il problema dell'accesso al formato con cui sono stati archiviati.
Anche in questo caso esistono: formati liberi, formati non liberi di cui si conoscono le specifiche e formati non liberi di cui non si conoscono le specifiche.
Sui primi, inutile dirlo, il mondo del software libero non ha perso tempo ad implementare librerie di lettura e scrittura per questi formati, unico problema che, forse, si può individuare è che la diffusione di questi formati è di uso recente.
Le motivazioni di questo ritardo trovano spiegazioni storiche: le tecnologie GIS hanno cominciato a diffondersi a partire da sistemi

proprietari la cui logica iniziale è stata di chiusura.
Il problema della definizione degli standard GIS è un discorso aperto dal 1980 che, a partire dal 1994, è riuscito a dare vita all' OGC - *Open Gis Consortium* (http://www.opengis.org). Da questa data in poi sono nati: formati (es.: GML *Geography Markup Language*, un XML definito per rappresentare i dati vettoriali), protocolli (es.: WMS *Web Map Service*, ovvero il protocollo per le applicazioni *webgis*), linguaggi (SFS – *Simple Features* SQL: il linguaggio SQL per operare su dati vettoriali archiviati su un *database server*), ecc…, le cui specifiche sono pubbliche e reperibili sul sito http://www.opengis.org/specs/?page=spec.
A partire da queste specifiche, il mondo del software libero non ha perso tempo a creare librerie di accesso.
Bisogna, però, sottolineare che, data la "giovane età" di molte di queste specifiche, i formati proprietari si sono imposti velocemente.

2.3 Il formato ESRI *Shape file*

Fra i formati proprietari di cui si conoscono le specifiche uno dei più utilizzati è, senza ombra di dubbio, quello proposto da ESRI con il pacchetto software ArcVIEW : il formato ESRI *Shape File*. Questo si compone in realtà di tre *file* che si distinguono, ovviamente, per estensione:.shp, .dbf e .shx.
Si tratta di tre *file* binari che cooperano fra di loro, in modo tale da permettere l'archiviazione e la visualizzazione di dati vettoriali.
Nel *file* con estensione .shp sono presenti le geometrie da rappresentare a video, archiviate secondo il sistema di riferimento utilizzato; quello con estensione .dbf è la classica tabella in formato DBF e contiene gli attributi alfanumerici di ogni singola geometria. Il .shx è un semplice indice che unisce ad ogni geometria rappresentata nel *file* .shp ogni riga relativa della tabella contenuta nel *file* .dbf. Informazioni, librerie e specifiche su questo formato si trovano sul sito http://shapelib.maptools.org.
Per quello che riguarda la programmazione in java, tutti i *package* presentati sono in grado di accedervi. La reale forza di questo formato è la sua diffusione, tale che la maggior parte dei programmi commerciali si impegna a produrne un importatore ed esportatore, divenendo quasi uno *standard de facto*, come spesso accade, anche se non il migliore.

2.4 PostGIS

Negli ultimi anni la tecnologia GIS ha sentito sempre più l'esigenza

di archiviare i dati vettoriali su *database server* da arricchire poi con funzionalità SQL, per ottenere risultati sulle relazioni spaziali fra gli oggetti (es.: calcolo di aree e lunghezze, controlli di intersecazione fra geometrie, ecc.).
Un primo progetto di "*spatial* sql" venne realizzato da Oracle, in collaborazione con ESRI (per poi separarsi rispettivamente in "*Oracle Spatial Server Extensions*" e "*ESRI Spatial Database Engine*"), da quella base ogni azienda produttrice di *database server* ha sviluppato le proprie estensioni. La comunità del software libero non è stata a guardare e, seguendo le specifiche SFS - *Simple Features SQL* proposte dall'OGC (http://www.opengis.org/docs/99-049.pdf), ha prodotto le estensioni GIS per i due *database server* più noti: MySQL e PostgreSQL.
In verità per il primo il supporto non è ancora completo ed è stato introdotto di recente nella versione *alpha* 4.1.
Il secondo, invece, vanta il modulo PostGIS (http://postgis.refractions.net) che, giunto ormai alla versione 0.8.1, è un software maturo e robusto, che ricopre la maggior parte delle specifiche introdotte dall'OGC.
Le specifiche SFS, oltre che a definire una sintassi SQL per calcoli sulle relazioni spaziali fra oggetti geometrici, introducono delle strutture dati relative all'archiviazione di una singola geometria.
Queste strutture si presentano secondo due formati: WKT (*Well Known Text*) o WKB (*Well Known Binary*). Il primo è una rappresentazione testuale della singola geometria in cui viene definito il tipo e le coordinate (es.: per rappresentare un punto che passa per le coordinate 142,2 e 133,67 si usa la sintassi POINT(142.2,133.67)); il secondo ha in comune con il primo la stessa rappresentazione; la differenza è che la sintassi, invece di essere basata su del testo, è espressa con valori in *byte*.
Allo stato attuale PostGIS non è ancora distribuito direttamente con PostgreSQL pertanto, escludendo qualche distribuzione GNU/Linux come Debian o Mandrake, occorre compilare manualmente il modulo come descritto nella documentazione acclusa al pacchetto.
Gli utenti Windows possono contare sul progetto CygWIN PostGIS Installer (http://dcmss.sourceforge.net/postgis_installer.php).
In ogni caso le operazioni da eseguire per la compilazione di PostGIS non sono molto complesse. Requisito di partenza è quello di avere i sorgenti di PostgreSQL.
Per un uso completo di PostGIS è importante aver installato anche le librerie proj4 (http://www.remotesensing.org/proj).

Queste librerie sono alla base della maggior parte dei programmi GIS e risolvono le problematiche relative alle trasformazioni dei sistemi di coordinate.
Per abilitarne l'uso in PostGIS è necessario modificare, prima di compilare il pacchetto, il valore omonimo della variabile presente nel file *Makefile*.
Una volta terminata la compilazione e l'installazione, è necessario popolare la tabella "spatial_ref_sys" (*file* testuale omonimo con estensione .sql contenuto nei sorgenti) che contiene le informazioni necessarie per l'uso completo delle librerie proj4.
Il riempimento delle tabelle con le geometrie da utilizzare in PostgreSQL può avvenire in due modi:

1. attraverso istruzioni SQL di CREATE e INSERT rispettando la struttura *Simple Features* SQL;
2. attraverso l'uso dell'*utility* shp2pgsql distribuita con PostGIS.

Il secondo metodo non è altro che un programma che converte un *file* ESRI Shape in un *file* di testo con le relative istruzioni SQL compatibili con PostGIS.
Nella conversione verrà chiesto di definire un valore per la variabile SRID, che rappresenta l'identificativo del sistema di riferimento spaziale utilizzato dai dati.
I valori che può assumere SRID sono contenuti nella tabella "spatial_ref_sys" di cui sopra.
È altamente consigliabile, una volta inseriti i dati, la costruzione di un indice ad albero (GIST – *Generalized Search Trees*) per la propria tabella contenente le geometrie. L'uso di questo indice aumenta notevolmente i tempi di risposta del *database*.
Una volta riempite le tabelle è possibile utilizzare la sintassi SQL presente in PostGIS per *query* complesse come: il calcolo dell'area di una geometria, la lista delle geometrie contigue con un'altra, l'unione fra geometrie, la distanza fra due punti, il centroide di un poligono, ecc…
Facendo, poi, una JOIN con altre tabelle contenenti dati alfanumerici, o attraverso relazioni, le potenzialità che si ottengono aumentano in maniera esponenziale.
Già a questo livello un qualsiasi sviluppatore, o utente esperto, è in grado di ottenere degli ottimi risultati.

3. Creare/Usare applicazioni - Elaborazioni dei dati e loro visualizzazione

Una volta ottenuti i dati l'utente desidera visualizzare o ottenere calcoli sui dati e, per lo sviluppatore, arriva il momento di "rimboccarsi le maniche".

3.1 JTS – Java *Topology Suite* - Calcoli sui dati geografici senza visualizzazione

Si è visto come, nel caso di PostGIS, calcoli avanzati, senza dover necessariamente visualizzare le geometrie, siano ottenibili attraverso l'uso di specifiche istruzioni SQL. Già a questo livello, uno sviluppatore java, che ha confidenza con lo *spatial* sql può creare ottime applicazioni via JDBC.
Questa è sicuramente un'ottima scelta qualora si senta la necessità di offrire un'architettura *client*/*server*, ma quali sono le soluzioni, allo stesso livello (quindi che non necessitano della visualizzazione di geometrie, ma solo di risposte alfanumeriche) che uno sviluppatore java può utilizzare?
La risposta arriva dallo stesso PostGIS che, recentemente, ha fatto un grande passo avanti inglobando le librerie del progetto GEOS (http://geos.refractions.net).
Queste librerie, che sono il *porting* in C++ delle classi java JTS - *Java Topology Suite* (http://www.vividsolutions.com/jts), distribuite con licenza LGPL dall'azienda canadese VividSolutions, sono un'implementazione delle specifiche *Simple Features* SQL definite dall'OGC.
Il successo e le potenzialità di questo *toolkit* è tale che ormai il 90% delle applicazioni GIS in java, offerte dal mondo del software libero, ne fa uso. L'*input* delle strutture dati delle JTS avviene attraverso la sintassi WKT (*Well Known Text*). Qualsiasi software che ne fa uso è stato progettato in modo tale da poter caricare i dati da un qualsiasi tipo di formato ed essere in grado di riempire queste strutture.
L'uso delle JTS non è, comunque, complesso. Al sito ufficiale è reperibile una ottima documentazione (oltre alla relativa javadoc) che descrive le specifiche tecniche su cui si basa.
La lettura, anche se superficiale, è consigliata.
Di seguito, proponiamo una rapida sintesi.
La classe "madre" di ogni geometria ha nome *Geometry.* Le sottoclassi dirette sono:

- *Point*;
- *LineString*;
- *Poligon*;

che rappresentano, rispettivamente, geometrie come punto, linea e poligono. I metodi della classe *Geometry* permettono, da subito calcoli utili quali: lunghezza (secondo il sistema di riferimento utilizzato), area, informazioni *booleane* sull'intersezione, o sovrapposizione fra due geometrie, la creazione di una nuova geometria data dall'unione fra due, ecc…
Per le relative sottoclassi, esistono metodi specifici (es.: la classe che rappresenta un poligono offre il metodo per il calcolo del centroide).
Una classe importante è *Envelope*.
Con questo termine si definisce una regione rettangolare di coordinate a due dimensioni. Pertanto, il risultato che si ottiene calcolando l'*Envelope* di una geometria, è quello di ottenere le coordinate del rettangolo che la contiene.
La creazione di un oggetto che rappresenta una geometria è possibile o attraverso i relativi costruttori, in cui è necessario specificare le coordinate che lo compongono, o attraverso la sintassi utilizzata per il formato WKT, da sottoporre alla classe WKTReader.

3.2 JUMP – *Unified Mapping Platform* - Applicativo avanzato per la visualizzazione dei dati geografici

Il lavoro fatto da VividSolutions non si è, però, fermato alla creazione delle classi JTS, ma, dall'estate 2003, ha dato vita, tra le altre cose, al software, distribuito con licenza GPL, JUMP – *Unified Mapping Platform* (http://www.vividsolutions.com/jump).
In questo caso si tratta di una vera e propria applicazione pensata per la visualizzazione e l'elaborazione di dati GIS, che non ha nulla da invidiare a software proprietari più blasonati come ArcVIEW.
Inutile nascondere che alla base di questo programma c'è la potenza espressa dagli algoritmi del *toolkit* JTS.
Allo stato attuale (versione 1.1) JUMP è in grado di caricare dati nei formati: GML, ESRI Shapefile e WKT. Tramite una *plugin* scaricabile dal sito di PostGIS, è possibile utilizzare JUMP come interfaccia grafica per i dati archiviati con l'estensione GIS di PostgreSQL. Inoltre, può essere utilizzato anche come *client* di un *server* mappe che fa uso del protocollo WMS, permettendo l'*overlay* (sovrapposizione di livelli/*layer* di dati vettoriali su uno spazio

dedicato alla creazione della mappa) con altre fonti di dati.
Anche nel caso di JUMP la documentazione è molto ricca ed è orientata sia verso lo sviluppatore che vuole creare estensioni al programma, o usarne le API per nuove applicazioni, sia verso l'utente che necessita di analisi di dati vettoriali. La stessa VividSolutions offre la JCS *Conflation Suite* (http://www.vividsolutions.com/jce) che, integrandosi con JUMP, permette correzioni assistite molto complesse dei dati vettoriali.

3.3 GISToolkit - classi *general purpose*

JUMP è un software di ottima qualità che, con il tempo, diverrà sempre più un punto di riferimento per la comunità del software libero interessata allo sviluppo di applicazioni GIS in java.
Allo stato attuale, però, presenta alcuni piccoli vincoli quali:

- lo scarso numero di formati supportati (problema che comunque può essere risolto convertendo i formati in ESRI *Shapefile* o in tabelle PostGIS);
- non gestisce formati *raster* (che possono tornare utili come sfondo dei vari "tematismi" vettoriali che si vogliono rappresentare);
- non è possibile esportare le elaborazioni ottenute in un formato immagine;
- manca anche la gestione di stampa.

Per la verità un utente esperto, dopo aver preso confidenza con le API di JUMP, non farebbe alcuna fatica ad implementare i punti "deboli" esposti sopra e contribuire allo sviluppo di questo progetto.
Nel caso in cui si abbia fretta, una possibile alternativa è espressa da GISToolkit (http://gistoolkit.sourceforge.net).
Questo *toolkit*, distribuito con licenza LGPL, al suo interno fa uso delle classi JTS e, oltre che ad accedere a diversissime fonti di dati vettoriali (ESRI Shapefile, PostGIS, MySQL 4.1, Oracle *Spatial Server*, DB2 Spatial Extender, ESRI SDE,...), è in grado di gestire anche fonti *raster* (TIFF, JPG2000, JPEG, MS Terraserver, ErMapper ECW, ...) complete di relativo *file* di georeferenziazione (es.: TWF - Tiff *World File*, un *file* di testo che contiene le informazioni necessarie per la trasformazione del sistema di coordinate *pixel* nel sistema di coordinate geografiche di riferimento) o delegando all'utente questa operazione.
Per alcune fonti, come ErMapper e ESRI ArcIMS, GISToolkit

si appoggia a librerie proprietarie, distribuite gratuitamente dai rispettivi fornitori.
Nel caso di ErMapper, e del suo interessante formato ECW (di cui purtroppo non sono rilasciate le specifiche), occorre, inoltre, sottolineare un aspetto importante: allo stato attuale l'azienda che ha creato questo formato fornisce per java unicamente un *binding* alle librerie native per MS Windows, vincolando pertanto la scelta del sistema operativo.
Un altro punto a favore di GISToolkit, che lo rende un pacchetto "*general purpose*" per la creazione di applicazioni GIS in java, sono le componenti *server* (*package*: gistoolkit.server.*) che permettono di implementare velocemente un servizio *webgis* (*server* Web che produce mappe) compatibile con il protocollo WMS.
Molto interessanti sono, anche, le componenti *database* che permettono di creare unioni di tabelle (quindi la parte alfanumerica di un vettoriale) fra fonti dati diverse, introducendo, quindi, maggiori attributi sulle geometrie rappresentate.
Unica pecca di questo *tool* è la documentazione: vengono presentate tutte le potenzialità offerte da queste classi, ma in maniera troppo sintetica.
Per farsi una idea di cosa si possa realizzare con GISToolkit si consiglia di eseguire l'applicazione GISEditor (eseguendo il relativo *script* distribuito con il pacchetto) contenuta nel *package*.
Si tratta di una applicazione dimostrativa che può da sola soddisfare le esigenze di un utente a cui interessa creare mappe da stampare, o convertire in formati come jpg, png o svg, e fare qualche piccola operazione come: trovare i valori delle aree o le lunghezze delle geometrie, la conversioni fra sistemi di riferimento diversi, il calcolo delle distanze fra punti, ecc.
Ogni comando corrisponde alla relativa classe ed ai relativi attributi.
L'architettura ruota sulla classe Layer. Quest'ultima rappresenta un insieme di geometrie, o una immagine, che fanno parte di una stessa sorgente dati (es.: un tema che rappresenta tutte le strade di una città o una foto area).
Il caricamento dei dati avviene tramite una classe derivata dall'interfaccia DataSource (le classi che ne derivano sono le implementazioni per l'accesso ai vari formati di dati).
La rappresentazione dei dati alfanumerici avviene nella classe GISDataSet, la quale astrae una tabella, in cui le righe sono i valori e le colonne i campi.

La singola geometria generica è rappresentata dalla classe Shape, i cui metodi derivano dall'architettura offerta da JTS.
Per utenti che vogliono costruire interfacce grafiche personalizzate, viene offerta la classe GISDisplay che, derivando dalla nota classe grafica JPanel, permette velocemente la creazione di applicazioni con una area in cui rappresentare la mappa.
È molto importante notare che per quello che riguarda le operazioni di *zoom* e spostamento della mappa bisogna fare riferimento alla classe Envelope (le cui funzionalità sono le stesse del *toolkit* delle JTS).
Creando un nuovo oggetto Envelope da sottomettere alla classe GISDisplay tramite l'opportuno metodo (setEnvelope), la mappa rappresentata cambierà immagine/posizione.
Come visto precedentemente la creazione di un *webgis* (*server* Web che offre servizi di mappe) con GISToolkit può essere ottenuta attraverso gli appositi *package* che implementano il protocollo WMS.
Anche in questo caso, lo stesso *toolkit* offre un'applicazione demo chiamata GISServer, completa di interfaccia di amministrazione e di due *client*: uno fatto in HTML e l'altro rappresentato da una *applet* java.
Coloro che volessero implementare un proprio *webgis*, con il classico approccio di creazione di immagini PNG o JPG, devono tener presente alcuni accorgimenti.
Prima di tutto, considerare una sessione utente in cui mantenere l'ultimo Envelope visto e gli attributi sugli stili di rappresentazione dei Layer, inoltre fare uso della classe Converter necessaria per convertire in pixel le coordinate dell'immagine inviata al *client*, al sistema di riferimento rappresentato e viceversa.

3.4 GEO*Tools*/GEO*Server* - Applet Java e WebGIS J2EE

Come visto precedentemente, la creazione di un *webgis* è un'operazione complessa, in cui bisogna prendere in considerazione moltissime problematiche fra le quali:

- *performance*;
- peso dei dati;
- sessione utente;
- interfaccia sul *client*.

Quest'ultimo punto presenta, poi, il classico dilemma: *client* leggero

o *client* pesante?
Il primo risolve problematiche di *crossbrowsing* (l'uso di *browser* diversi), ma richiede continue interrogazioni del *server* che deve generare immagini JPG o PNG: ogni operazione fatta sulla mappa necessita di una chiamata al *server*, che deve rigenerare l'immagine, anche per disegnare "solo" un piccolo punto. Bisogna considerare, inoltre, che le interfacce sono solitamente povere e che per arricchirle (e ottimizzarne anche i risultati) si è costretti a ricorrere ad un uso "pesante" di javascript, aprendo, però, problematiche di *crossbrowsing* (risolvibili con l'uso di librerie appositamente pensate, si veda http://www.cross-brosing.com)
Se da un lato l'approccio con *client* "pesante" risolve il problema delle interfacce utente povere, delegando la rappresentazione delle mappe al *client*, dall'altro appesantisce la *workstation* del visitatore e crea problemi di visualizzazione con alcuni *browser*.
In questo caso si ricorre a *plugin* (come Macromedia Flash o Adobe SVG) o ad *applet* java. Per chi volesse fare questo tipo di scelta, considerando, quindi, il *server* Web non come un *webgis*, ma come un "distributore" di *applet* (in quanto i dati vettoriali vengono scaricati sul *client*), un valido strumento è rappresentato dal pacchetto GEOTools (http://www.geotools.org).
Dietro a queste classi (distribuite con licenza LGPL) gravita una grossa comunità di sviluppatori che si sono impegnati anche nel produrre una ricca documentazione piena di esempi e *tutorial*.
Il loro successo è, comunque, limitato all'uso di *applet*, in quanto sono sviluppate secondo il vecchio standard di java 1 (per soluzioni *desktop* si consiglia di ricorrere a GISToolkit, oppure JUMP).
Questo limite non è, comunque, passato inosservato alla comunità di GEOTools, che ha concentrato le proprie forze verso la creazione di un nuovo progetto basato sulla tecnologia Java 2 *Enterprise Edition* (J2EE). Il risultato di questo sforzo ha dato vita al progetto GEOServer (http://geoserver.sourceforge.net), che è stato protetto con la licenza GPL2.
Il risultato è un ottimo *framework*, di facile utilizzo (anche in questo caso la documentazione è ricca di *tutorial*), per la creazione di *web service map* (pertanto è accessibile anche da applicazioni come JUMP), che rispettano le specifiche OGC.
Unica pecca è il mancato supporto per i dati *raster*.

4. Conclusioni
Come visto nell'introduzione del saggio non vi era alcuna

intenzione di presentare informazioni sulla elaborazione dei dati *raster*.
Questo sia a causa dell'inesperienza dell'autore, sia per la povertà di soluzioni offerte dal mondo del software libero orientato a java.
Sicuramente, java non è il linguaggio propriamente indicato per elaborazioni computazionali pesanti, come possono essere l'elaborazione a 3 dimensioni di modelli digitali del terreno (anche se il pacchetto Java3D offerto da SUN è sorprendente).
Allo stato attuale conviene spostarsi verso applicativi creati in C/C++, come GRASS.
Sicuramente, per il futuro, si avranno ottimi risultati.
Il programma LandSerf (http://www.landserf.org) promette nella versione 2.0 (annunciata per l'autunno 2003, ma mai uscita) ottime soluzioni, la pecca è che il software non si sarebbe avvalso di una licenza libera, ma, forse, provando a sollecitare lo sviluppatore di questo programma un cambio potrebbe arrivare in tempi brevi. Per gli sviluppatori più esperti, una valida soluzione è rappresentata dal pacchetto VisAD (http://visad.sourceforge.net).
In realtà si tratta di classi java (con licenza GPL) che implementano complessi algoritmi di calcolo matematico e visualizzazione di dati, ma che possono essere facilmente adattate per la creazione di applicazioni GIS basate su dati *raster*.
Esistono giá applicazioni per calcoli metereologici che fanno uso di questo pacchetto. Si segnala infine un ultimo *framework* java-GIS che fará molto parlare di sé per il futuro: degree (http://degree.sourceforge.net).