Architettura Open Source per le applicazioni della WFP Spatial Data Infrastructure

E. Agosto(*), S. Dalmasso(*), P. Pasquali(*)

(*) Ithaca - Information Technology for Humanitarian Assistance, Cooperation and Action
Via P.C. Boggio 61, 10138, Torino
e-mail: eros.agosto@ithaca.polito.it, simone.dalmasso@ithaca.polito.it, paolo.pasquali@ithaca.polito.it

Riassunto
ITHACA (*Information Technology for Humanitarian Assistance, Cooperation and Action – www.ithacaweb.org*) sviluppa e mantiene una SDI (*Spatial Data Infrastructure*) per conto del *World Food Programme*, la più grande agenzia delle Nazioni Unite. Il progetto si sviluppa all interno delle direttive dello UNGIWG (*United Nations Geographic Information Working Group*) che promuove l interoperabilità e l uso di tecnologie *Open Source*. Secondo questi principi ITHACA sviluppa progetti totalmente *Open Source* per la divulgazione dei dati e lo sviluppo di applicazioni *web GIS* avanzate.
La costruzione di applicazioni complesse richiede la cooperazione di numerosi componenti che si occupano di particolari aspetti, dalle elaborazioni lato *server* alle interfacce sul *browser* dell utente. Recentemente ITHACA ha modificato l architettura lato server introducendo l uso del paradigma MCV (*Model Controller View*) grazie anche alla migrazione delle sue applicazioni su un *framework* di sviluppo (Django) e all uso esclusivo del linguaggio Python.
Inoltre la ricerca di interfacce di più facile utilizzo e accattivanti per l utente ha mostrato la necessità di potenziare quanto offerto dal noto *client* OpenLayers che si pone come libreria di più basso livello.
Ciò ha condotto alla modifica di alcune classi javascript di OpenLayers e all integrazione dello stesso con le librerie Ext e GeoExt per la costruzione di funzionalità avanzate e per aumentare il livello di interattività con la mappa. Anche per la simbolizzazione dei dati vettoriali sono stati utilizzati gli standard dell'*Open Geospatial Consortium* - OGC - quali l SLD (*Styled Layer Descriptor*) per lo stile di ogni singolo livello che deve rispondere in modo diverso alle varie scale di rappresentazione e alla rapidità di fruizione degli stessi grazie all uso del protocollo WMS (*Web Mapping Service*) unito ad un substrato di pre-memorizzazione dei dati.

Abstract
Ithaca (Information Technology for Humanitarian Assistance, Cooperation and Action – www.ithacaweb.org) develops and manteins a GSDI (Global Spatial Data Infrastructure) for UN (United Nations) agencies like WFP (World Food Programme). Ithaca uses pure FOSS (Free and Open Source Software) tools to distribute GSDI based services on the web. In order to make web-GIS application manteinance easier, Ithaca changed the logics behind its applications, both on hardware and software side, both on data model, algorithms and style. The adoption of the MCV (Model Controller View) pattern in the architecture and the first released web services are here described. In particoluar the capabilities of popular OpenLayers client have been enhanced by intagrating it with Javascript libraries such as Ext and GeoExt.

Introduzione

Ithaca (*Information Technology for Humanitarian Assistance, Cooperation and Action – www.ithacaweb.org*) sviluppa un'infrastruttura di dati geospaziali a scala globale (GSDI) a supporto della propria attività e di quella di agenzie delle Nazioni Unite come il World Food Programme (WFP) per cui opera dal 2007 [Ajmar et alii, 2007].

Data la scala globale e distribuita degli utenti, importante è rendere possibile la fruizione dei dati presenti nella SDI, oltre che l'estrazione di informazione da essa, tramite la rete. ITHACA, seguendo le indicazioni del UNGIWG, sviluppa l'infrastruttura per la pubblicazione dei dati della SDI con strumenti FOSS (*Free and Open Source Software*). Tale scelta è motivata anche da ragioni di sostenibilità, manutenibilità, replicabilità, sperimentazione e personalizzazione delle applicazioni, oltre che dall'attenzione all'interoperabilità che il mondo FOSS riserva tramite l'adesione a standard condivisi come quelli OGC [Agosto et alii, 2007].

Dal suo inizio, ITHACA ha avviato un numero crescente di progetti che utilizzano e contribuiscono ad arricchire i dati presenti nella SDI; ciascuno di essi prevede una parte di fruizione ed accesso tramite la rete. Di conseguenza applicazioni specifiche sono state nel tempo progettate. Data la natura di sperimentazione che le ha sempre accompagnate, tali applicazioni si sono configurate in maniera differente per quanto riguarda la logica costruttiva e l'architettura software sulla quale sono basate. Al fine di uniformare le applicazioni esistenti, di agevolarne la manutenzione ed estensione, ITHACA ha modificato sensibilmente l'impostazione della propria architettura applicativa.

ITHACA ha adottato in maniera estensiva il modello MCV nella progettazione delle proprie applicazioni, coinvolgendo in tale ottica sia la componente hardware che software; la filosofia MCV ha investito sia la progettazione della base dati, che la scelta degli strumenti software, oltre che la programmazione effettuata.

Il modello MCV

Il Modello MCV è un paradigma architetturale usato in ingegneria del software; tale struttura isola i tre principali livelli logici dell'applicazione:

- il modello dati (*data logic*) e gestione del database
- la *business logic* o *controller* che implementa la logica lato server
- la *presentation logic* che si occupa di rendere disponibili i dati sul *client*.

Questo sistema garantisce la possibilità di sviluppo, manutenzione e test in maniera indipendente per ogni componente. In un progetto basato su logica MCV ad un singolo modello possono essere associati uno o più *controller* (o viste).

La migrazione dell'architettura di Ithaca al modello MCV nasce dall'esigenza di poter razionalizzare e migliorare la struttura applicativa in modo da poter creare varie versioni di ogni singola applicazione in maniera tale da rispondere rapidamente ad una specifica esigenza.

Per una maggiore flessibilità e separazione dei livelli logici, il modello stesso è stato applicato sia alla componente hardware che software sia a livello server che a livello client.

Architettura hardware

L'architettura è basata su una SAN (*Storage Area Network)* alla quale sono collegati i web server. Il database e i dati su file system risiedono sulla SAN a cui sono demandate le funzioni di sicurezza e accessibilità. Tutta la parte applicativa invece è installata sui web server attuando la separazione tra dato e logica propria del modello MCV.

Architettura software lato server

A livello software la necessità era di trovare un *framework* di sviluppo open source che adottasse il paradigma MCV, che fosse basato sul linguaggio di programmazione Python (diffusamente adottato in Ithaca in quanto fornisce accesso alle API – *Application Programming Interface* di numerosi GIS) e potesse gestire nativamente dati georeferiti. La scelta è ricaduta su Django, un *web framework Open Source* in rapido sviluppo che nell'ultimo anno ha visto la nascita di un apposito

modulo (denominato GeoDjango) che mette a disposizione API e modelli per lavorare con dati spaziali.
Come i più moderni *framework* di sviluppo, Django facilita l'interazione col database attraverso apposite API Python. L'architettura di Django consente di disaccoppiare completamente il modello dati dal livello di logica e dal livello di presentazione dei dati. Django è inoltre strettamente aderente al principio DRY (*Don't Repeat Yourself*) secondo il quale "all'interno di un sistema, ogni pezzo di conoscenza deve avere una singola, non ambigua ed autoritativa rappresentazione". In questo senso facilita la semplicità di manutenzione e refactoring del codice.
I dati geografici sono contenuti in PostgreSQL/PostGIS, un database Open Source object-relational.
Il database è installato come *cluster* sulla *storage area*. Il *cluster* è un insieme di database gestito da un'unica istanza di un database server ed è un sistema di replica sincrono della composizione *multimaster* per PostgreSQL. Ha principalmente le funzioni di distribuzione del carico e di *high availability*. Per visualizzare (ed editare) i dati spaziali viene utilizzato il software server *Open Source* GeoServer. La gestione delle richieste http è demandata ad Apache a cui è stato affiancato Tomcat per consentire il funzionamento della *webapp* Java di GeoServer. Per far coesistere i due server, Apache è stato configurato in modo tale da fare da *proxy* verso le *webapp* Java.
L'utilizzo di GeoServer ha i seguenti vantaggi: è basato su Geotools, un *toolkit* GIS *Open Source* Java; utilizza gli standard OGC; implementa gli standard WMS che consentono la creazione di mappe in diversi formati e Web Feature Service (WFS) che permettono di condividere ed editare i dati usati per generare le mappe, oppure permette di far incorporare i dati in altri siti e applicazioni; integra OpenLayers, una libreria FOSS, per una veloce pubblicazione della mappa; permette di visualizzare dati provenienti da altre applicazioni quali Google Maps, Google Earth, Yahoo Maps, Microsoft Virtual Earth oppure ESRI ArcGIS.
I dati cartografici vengono simbolizzati tramite SLD, uno standard encoding OGC e per velocizzare la visualizzazione sul client è stato adottato un sistema di cache (GeoWebCache) che si occupa di memorizzare in anticipo le immagini generate da GeoServer in maniera tale da rispondere in modo immediato alle richieste.

Architettura lato client
Nei moderni web-GIS *Open Source* il client senz'altro più utilizzato è OpenLayers; tale libreria è dotata di API che assicurano estrema flessibilità ed è in costante aggiornamento da parte della comunità. OpenLayers si pone però essenzialmente come livello di logica sul client e per questo motivo non è ottimizzato per la creazione di interfacce utente gradevoli e funzionali.
A tale scopo invece, una delle librerie più promettenti è Ext.js: essa è infatti in grado di generare un elevato grado di interattività con l'utente attraverso l'uso di finestre, pulsanti e layout complessi, risultando, per contro, piuttosto complessa e di grandi dimensioni.
Nell'ultimo anno è iniziato lo sviluppo di GeoExt, una libreria che si interpone tra OpenLayers ed Ext.js, permettendo di sviluppare complesse interfacce per applicazioni web geografiche; GeoExt ha attualmente limitate funzionalità, tuttavia è in fase di rapido sviluppo.
L'insieme di queste librerie crea un'architettura lato client che disaccoppia la logica (dialogo col server delle mappe per il reperimento dei dati e loro gestione con OpenLayers) dalla presentazione degli stessi con Ext, GeoExt.
L'utilizzo di questi strumenti obbliga ad un uso intensivo del Javascript, linguaggio di scripting nato per essere inserito nelle pagine web, che in questo caso invece deve costituire parte della logica di funzionamento della pagina: occorre pertanto una struttura ad oggetti ed un estremo rigore per mantenere il codice pulito, funzionante e mantenibile.

Architettura logica
Come già accennato l'intera architettura basata sul modello MCV consente la razionalizzazione del codice; l'utilizzo di Django, basato sul principio DRY, consente di evitare la duplicazione di qualsiasi porzione di codice a vantaggio della manutenibilità.

Nella struttura implementata la logica di *background* è condivisa tra le varie applicazioni come anche lo stile generale dei siti. I CSS, alla base dell'aspetto grafico delle applicazioni, permettono un disaccoppiamento tra stile e logica, e sono condivisi in modo tale da poter modificare i dettagli grafici di ogni sito con una sola operazione.
La separazione funzionale viene seguita anche nella struttura di cartelle che divide la logica di ogni applicazione dai *templates* (html) e dai file statici (javascript e css). I file deputati alla presentazione delle informazioni (*templates* e file statici) sono condivisi per poter essere ereditati o sovrascritti da ogni applicazione: ad esempio le parti di *header* e *footer* delle pagine web sono uniche e sono inserite nel codice html tramite Python.
Le applicazioni web che fanno parte dell'architettura descritta sono:
- Allerta alluvioni in tempo pseudo reale
- Sito web Ithaca
- Applicazione per l'analisi di dati di siccità in Africa
- Applicazione per il monitoraggio della copertura nevosa nell'area himalayana, in fase di migrazione.

Primi servizi erogati
Applicazione di allerta alluvioni a scala globale
Nata dall'esigenza del *WFP* di avere un sistema di monitoraggio di eventi alluvionali a livello mondiale, l'applicazione *flood monitoring* presenta, in tempo pseudo reale, i bacini fluviali in potenziale allerta.

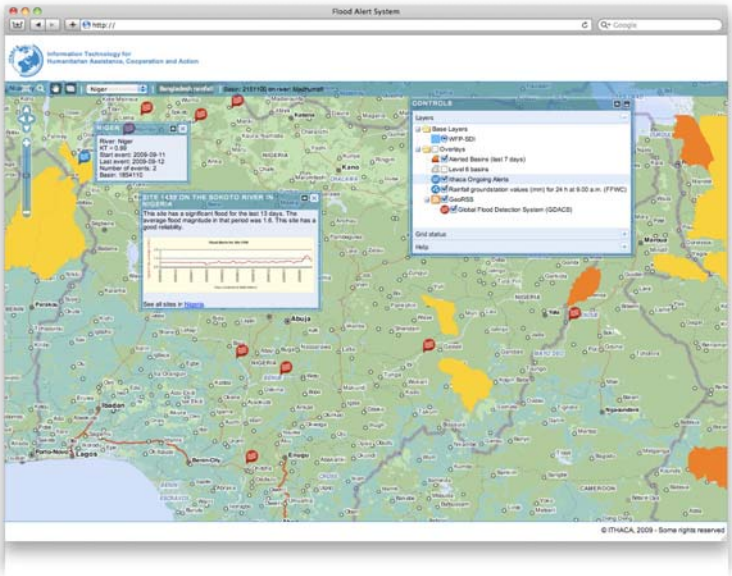

Figura 1 Applicazione di flood monitoring

I dati provenienti dal TRMM (*Tropical Rainfall Monitoring Mission*) vengono scaricati da internet ogni tre ore e vengono processati da un sistema di *grid computing* composto da nove calcolatori in parallelo. Il sistema visualizza i bacini fluviali (di livello 6) per il quali l'indice Kt (parametro che consente di valutare la gravità di un evento) è vicino o superiore a 1. I bacini in allerta vengono rappresentati con una gradazione di colore in relazione con il valore del parametro kt.
Un menu di selezione elenca dinamicamente le nazioni sulle quali sono segnalate allerte; la selezione di una nazione produce il centramento della mappa sulla stessa e crea un'icona su ogni bacino allertato racchiuso entro i confini. Le informazioni del bacino e dell'evento sono contenute

all'interno del *pop-up* associato ad ogni icona cliccabile. È possibile comunque accedere a informazioni sul singolo bacino visibile nella mappa.
È stato inoltre integrato un sistema di monitoraggio in tempo reale dei dati di pioggia misurati da stazioni a terra in varie parti del Bangladesh; anche in questo caso l'elaborazione avviene ogni tre ore.

Applicazione "Snowcover"
L'applicazione WebGIS (Figura 2) rende graficamente disponibili le informazioni sulla copertura nevosa di due aree di interesse (Afghanistan e Nepal) lungo la catena montuosa del Himalaya, e consente di ottenere i tratti di strade potenzialmente interessati.
I dati vettoriali sono il risultato di un'elaborazione automatica di dati satellitari provenienti dal sensore MODIS (*Moderate Resolution Imaging Spectroradiometer*), in particolare del prodotto giornaliero MODIS Snow Cover MOD10_L2, una classificazione (risoluzione spaziale di 500 m - scala1:1000000) del suolo votata alle coperture nevose [Boccardo, 2006].
Rispetto ad applicazioni precedenti [Agosto, 2007, 2008] l'attuale è basata sull'architettura descritta e si pone come *client* WMS per quanto riguarda i dati che mostra. L'interfaccia *client* è attualmente in fase di migrazione.

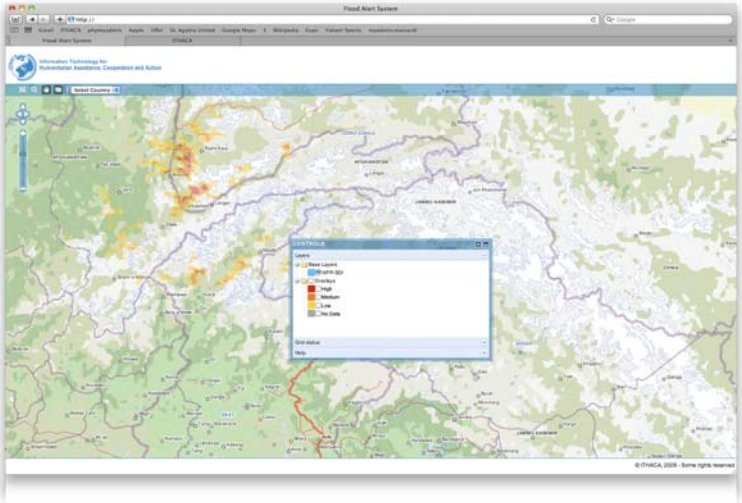

Figura 2: Applicazione snowcover

Applicazione "Drought"
L'applicazione nasce a supporto del progetto di studio e previsione di periodi di siccità in zone test dell'Africa basato su un'analisi geo-statistica di serie storiche di indici NDVI. L'applicazione sviluppata ha lo scopo di fornire via *web* un facile accesso e consultazione ai dati raccolti. Innanzitutto consente di estrarre grafici di previsione elaborati sulla base delle serie storiche, con la possibilità di selezione puntuale, per regione (entità amministrativa) o per area rettangolare definita dall'utente della zona di interesse. L'output fornito consiste in grafici storici sull'andamento dell'indice NDVI con la distribuzione spaziale scelta ed un periodo temporale prefissato o, in futuro, selezionato dall'utente. [Agosto, 2008]
L'applicazione fornisce inoltre la possibilità di sovrapporre ai livelli della GSDI mappe mensili di anomalie degli indici NDVI (25 anni disponibili) e le aree vulnerabili mensili ricavate dall'analisi delle serie storiche.

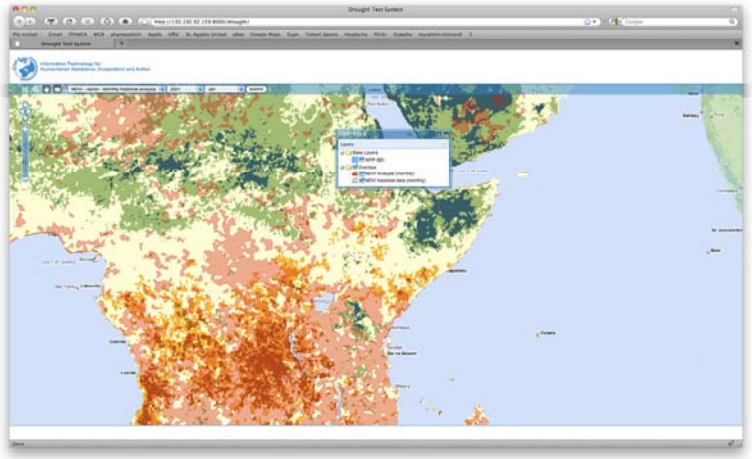

Figura 4: Interfaccia applicazione Drought

Conclusioni e Sviluppi futuri

L'utilizzo di strumenti FOSS ha permesso l'erogazione di geoservizi attraverso i quali consentire l'accesso ai dati ed informazioni contenute nel GSDI che Ithaca ha sviluppato a supporto della propria attività. Il modello MCV facilita lo sviluppo e manutenzione delle applicazioni e verrà ulteriormente perseguito da Ithaca.

Bibliografia

Agosto E., Dalmasso S. (2009), "Ithaca web services on a FOSS framework ", Cartography and Geoinformatics for Early Warning and Emergency Management - C4C Symposium, Praga

Agosto E., Dalmasso S. (2008), "Erogazione di web-services per la fruzione di un geodatabase a scala globale ", Conferenza nazionale ASITA, L'Aquila

Albanese A., Disabato F., Terzo O., Vigna R., Giardino M., Perotti L. (2008), A preliminary approach to flood risk mapping and flood forecasting system for the LDCs. XXI ISPRS Congress, Pechino

Agosto E., Disabato F., Dalmasso S. (2007), "Servizi web per la gestione di emergenze ambientali", Conferenza nazionale ASITA, Torino

Ajmar A., Perez F., Sartori G. (2007), "Sviluppo e implementazione di una Spatial Data Infrastructure per il World Food Programme", Conferenza nazionale ASITA, Torino

Boccardo P., Dequal S., Giulio Tonolo F., Marenchino D. (2006), "ITHACA: un progetto innovativo per la gestione delle emergenze ambientali", Conferenza nazionale ASITA, Bolzano

Webografia

Django, http://www.djangoproject.com/
DRY, http://c2.com/cgi/wiki?DontRepeatYourself
ExtJS, http://www.extjs.com/
GeoExt, http://www.geoext.org/
GeoServer, http://geoserver.org/
GeoWebCache, http://geowebcache.org/
OGR, http://www.gdal.org/ogr/
OpenLayers, http://openlayers.org/
Postgis, http://postgis.refractions.net/
Postgres, http://www.postgresql.org/
Python, http://www.python.it/