~~DEVELOP~~ A PUBLISH-AND-SUBSCRIBE SYSTEM FOR PUBLICIZING EARTH SCIENCE INFORMATION AND SERVICES

by

Chunming Peng
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Earth Systems and Geoinformation Sciences

Committee: [illegible]	Dr. Wenli Yang, Thesis Director
[illegible]	Dr. Ruixin Yang, Committee Member
[illegible]	Dr. Liping Di, Committee Member
[illegible]	Dr. Peggy Agouris, Department Chairperson
[illegible]	Dr. Richard Diecchio, Associate Dean for Academic and Student Affairs, College of Science
[illegible]	Dr. Vikas Chandhoke, Dean, College of Science
Date: 07. 27. 2010	Summer Semester 2010 George Mason University Fairfax, VA

A Publish-and-Subscribe System for Publicizing Earth Science Information and Services

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Chunming Peng
Bachelor of Engineering
Zhejiang University, 2008

Director: Wenli Yang, Assistant Professor
Department of Geography and Geoinformation Science

Summer Semester 2010
George Mason University
Fairfax, VA

Table of Contents

List of Figures

List of Tables

Abstract

A Publish-and-Subscribe System for Publicizing Earth Science Information and Services

Chunming Peng, M.S.

George Mason University, 2010

Thesis Director: Dr. Wenli Yang

Dozens of terabytes of Earth observation data and their derived science products are generated each day. For the tremendous and precious Earth science data and information to be fully explored and utilized, it is required that advanced technologies be developed and deployed for data providers to easily publish and deliver their products and data consumers to easily find and obtain needed data, preferably in customized forms. The purpose of this thesis research is to develop a service- and data-casting architecture for publication and subscription of Earth science data, information, and services. The architecture will not only enable generation of feeds for any updates and changes occurring to Earth Science data and Web Services from a particular provider but also allow aggregation of feeds from other data and service providers, thus significantly enhancing the visibility of available data and services and easing the searching and access of data. A prototypical system based on the architecture has been developed and deployed to publish the data and services provided by GeoBrain, a distributed interoperable web-based geospatial information system. The terms "serviceCasting" and "dataCasting" were coined because of their similarities with the "Podcasting" concept; they all describe the delivery of metadata over the Internet by information feeds based on Really Simple Syndication (RSS) or ATOM standard. GeoBrain data and services are published via ATOM feeds to the cloud and to be syndicated in the to-be-created geographic feed reader. In order to satisfy the special needs of geographers, two namespaces and additional elements are added to enhance geographic features descriptions. The prototypical system demonstrates that syndication and aggregation of feeds can decentralize the repository of earth science data and make information more discoverable and accessible.

1. Introduction

Publish/Subscribe (or pub/sub) is a messaging paradigm where an application in an integration architecture only sends messages to those applications interested in receiving the messages without knowing the identities of the receivers (MSDN, 2008). With this architecture being used for publicizing Earth Science data and services, users subscribe to timely updates from favored websites – for example, NASA GOES data provider – and later aggregate feeds from many sites into one place. Here a web feed is a data format providing users with frequently updated content from a specific website. Among all formats for feeds, Really Simple Syndication (RSS) and ATOM are two most common syndication standards. The primary goal of this thesis is to investigate how pub/sub system can help publicizing the Earth Science data and Web Services, thus facilitating users' accessing to and obtaining of customized data and information.

Among all concepts advocating geographical representation of feeds, two inspired this study the most - "DataCasting" and "ServiceCasting". The former describes the delivery of geographic data over the Internet using XML feeds (Bingham, 2009) while the latter refers to the publication of Web services (for inquiring or retrieving geographic data) through the network using RSS or ATOM feeds (Wilson, 2008). However, the standards can only support dataCasting via RSS, or dataCasting via extended ATOM standard. In this thesis research, we create a unique and complete Feed Publishing Standard that satisfies all dataCasting and serviceCasting needs. We also enable and extend ATOM's geospatial capability by integrating the Atom publishing standard with GeoRSS tags to describe a geographic feature with its shape, spatial co-ordinate values, and other properties.

In order to achieve the primary research goal, three specific objectives need to be accomplished. First, we have to establish a list of elements (under different namespaces) with which dataCasting (see Appendix for Table 2) and serviceCasting feeds can be denoted. Second, we need to create an application that can generate ATOM feeds for any ingested data and services automatically, and a new type of feed reader which aggregates the geographically tagged ATOM feeds. And thirdly, the feed publishing and aggregation tools need to be test-run in order to prove operability of the pub/sub system. With this system working properly from generating feeds, publishing feeds, aggregating feeds to displaying feeds, we will then be able to introduce this server-side tool to other Earth Science data and Web services providers. During this thesis research, a prototypical system is developed to publish the data and Web service available from GeoBrain – an open, web-based interoperable, distributed geospatial information system developed in the Center for Spatial Information Science and Systems (CSISS), George Mason University (Di et al., 2009; Han et al., 2008).

2. Literature Review

The World Wide Web (WWW) is rich soil for revolutions to take place. In the beginning, there was e-mail. E-mail provides users an exciting new way to communicate with others, and has become an absolutely necessary tool for people to interact at work. The next revolutionary change the WWW represented was Web browsers. With Web sites and browsers, the individuals and companies can put information on the Web for anyone to see on demand, mostly for free. However, both of these technologies have their weaknesses (Ayers & Watt, 2005). Indiscriminately e-mail systems suffer from spamming, that nearly 85% of all the email in the world has once had a problem with spam. And while surfing on the Internet is an incredible experience, the flooding of information on the WWW requires users to spend quite some time searching for what they want instead of having the data come to users. Syndication of feeds can be a good remedy for the two problems. An information feed is the XML format metadata of web contents updated regularly, widely used in Web logging and news sites to keep readers informed of what has just happened. Some radio stations and other multi-media providers took a step further to include a download URL in the published feed so that subscribers can get to the viewing or downloading page by a single click when reading the feed. This current trend is called "Podcasting" (Ayers & Watt, 2005).

Compared to receiving frequently published advertisements via emails, web feeds ensure users privacy, safety and convenience since users are not required of their email address when subscribing to a feed and need not send an unsubscribe request in order to stop news. One can stay informed easily by retrieving automatically sorted or filtered feed items in their feed aggregators. From the data provider's perspective, publishing feeds is an effective way to publicize their products and attract potential customers.

The following literature reviews attempt to demonstrate the emergence of feeds, the difference between RSS and ATOM feeds, the different designs of geographic extensions used by dataCasting and serviceCasting, and the insufficiencies of these studies.

2.1 What is Web Syndication?

As a form of syndication in which Web content is made available to multiple other sites, Web Syndication most commonly refers to making web feeds available from one site in order to provide other people with a summary of the website's recently added or changed content. The syndicated content, or feed, can be anything from just headlines and links to stories, to the entire content of the site with its layout stripped off. The technology to this ranges from RDF or RSS 1.* branch, which includes RSS 0.90, RSS 1.0, and RSS 1.1, to RSS 2.* branch including RSS 0.91 through RSS 0.94 and RSS 2.0.1, and all the way to the "well-formed" ATOM standards which covers ATOM 0.3 through ATOM 1.0 (Libby, 1999; RSS, 2008, 2009, 2010; Nottingham and Sayer, 2005; Gregorio and de hOra, 2007). Web Syndication allows users to experience a site on multiple devices and be notified of updates over a variety of services, ranging from a simple list of links sent from site to site, to the complicated Semantic Web.

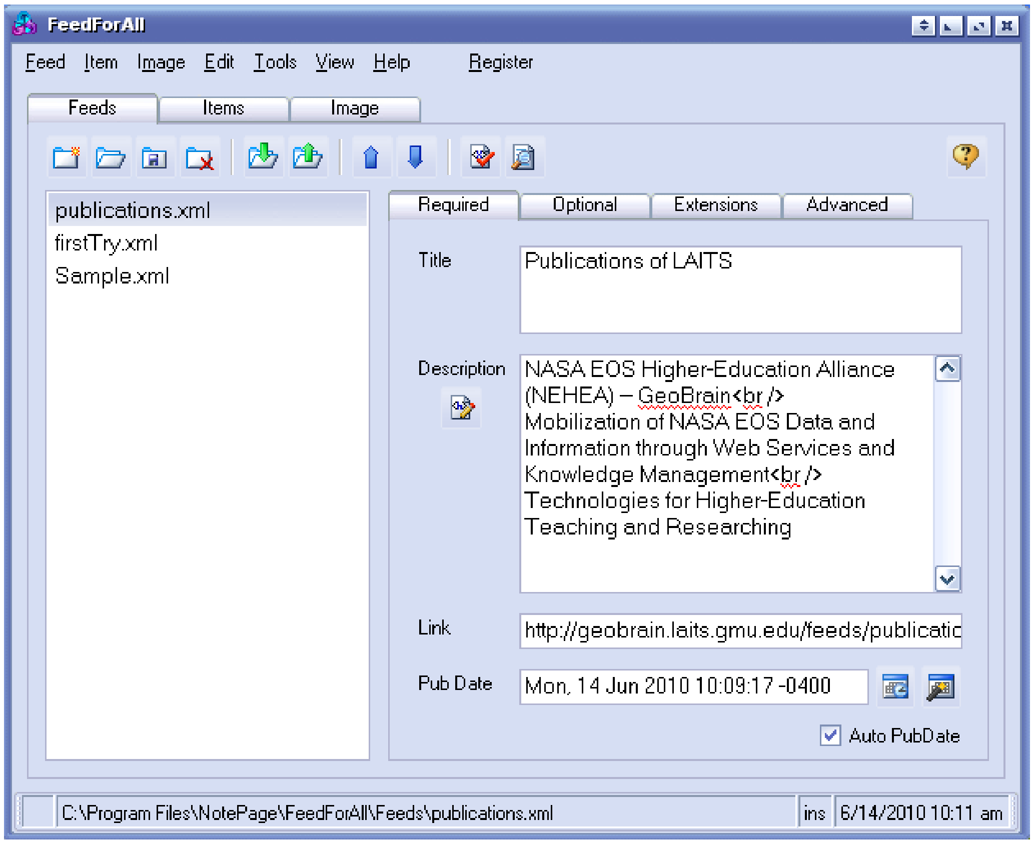
The advantage of receiving other websites' feeds are obvious -- Web syndication is an effective way of adding greater depth and immediacy of information to our own Web pages, and thus attracting more users. But what about supplying feeds of your own? Ben Hammerseley (2003) gives us at least eight reasons in his book "Content Syndication with RSS". Some of the reasons are listed below.

- It increases traffic to your site.
- It builds brand awareness for your site.
- It can help with search engine rankings.
- It improves the site/user relationship.
- It makes the Internet an altogether richer place pushing semantic technology along.

Although the format of syndication feeds can be anything transportable over HTTP, such as HTML or JavaScript, a more commonly adopted format is XML (eXtensible Markup Language). Within the XML families of Web Syndication formats, RSS and ATOM standards have the largest market shares nowadays. Some more details about these two syndication formats are discussed in the following paragraphs.

2.2 What is an information feed?

According to RSS Specifications, RSS or ATOM, is “a defined standard based on XML with specific purpose of delivering updates to web-based content”, which can be used by webmasters to “provide headlines and fresh content in a succinct manner”. Thus, consumers use feed readers or news aggregators to “collect and monitor their favorite feeds in one centralized program or location”. The content consumers view in the feed reader is known as information feed. There are three easy ways to create information feeds for one’s website -- using desktop software, simple text editor, or online feed creation tools. Figure 1 shows an extremely easy feed creation tool, FeedForAll, that allows webmasters to create, edit and publish RSS or ATOM feeds.



publications.xml
firstTry.xml
Sample.xml

Title: Publications of LAITS

Description: NASA EOS Higher-Education Alliance (NEHEA) – GeoBrain<br />
Mobilization of NASA EOS Data and Information through Web Services and Knowledge Management<br />
Technologies for Higher-Education Teaching and Researching

Link: http://geobrain.laits.gmu.edu/feeds/publicatic

Pub Date: Mon, 14 Jun 2010 10:09:17 -0400

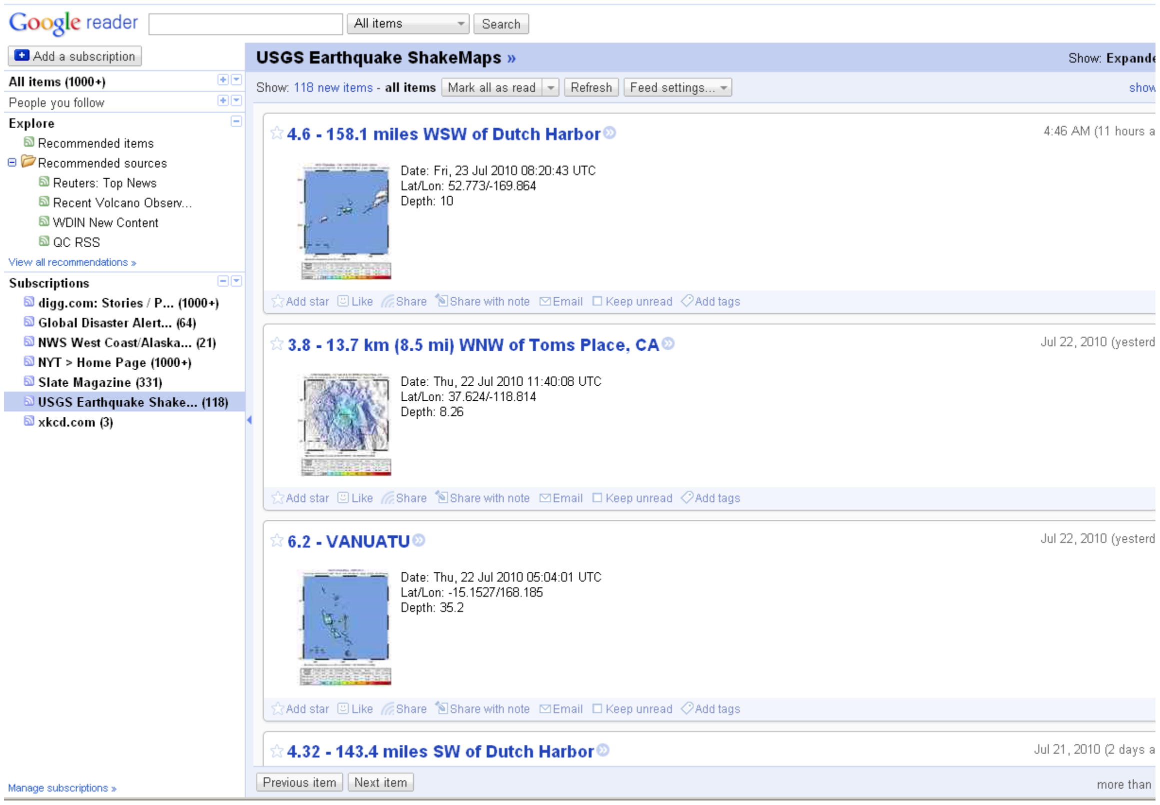
[✓] Auto PubDate

C:\Program Files\NotePage\FeedForAll\Feeds\publications.xml | ins | 6/14/2010 10:11 am

Figure 1 -- Creating a simple information feed using FeedForAll .

After a feed has been created, the webmaster has to submit the feed to other major websites such as Syndic8, Feedster, or professional websites. The webmaster is responsible to provide a discovery-enabled portal for web page browsers to access the feed URL, so that browsers can locate the URL address of the Web feed that they wish to subscribe and add it to their subscription list of their feed reader.

Normally people view their subscribed feeds in three different places – desktop aggregators, online feed readers (i.e. Google Reader) and web browsers such as Firefox Mozilla. Figure 2 shows a Google Reader that displays multiple entries from a single feed hosted by USGS Earthquake Center.



4.6 - 158.1 miles WSW of Dutch Harbor
4:46 AM (11 hours a
Date: Fri, 23 Jul 2010 08:20:43 UTC
Lat/Lon: 52.773/-169.864
Depth: 10

3.8 - 13.7 km (8.5 mi) WNW of Toms Place, CA
Jul 22, 2010 (yesterd
Date: Thu, 22 Jul 2010 11:40:08 UTC
Lat/Lon: 37.624/-118.814
Depth: 8.26

6.2 - VANUATU
Jul 22, 2010 (yesterd
Date: Thu, 22 Jul 2010 05:04:01 UTC
Lat/Lon: -15.1527/168.185
Depth: 35.2

4.32 - 143.4 miles SW of Dutch Harbor
Jul 21, 2010 (2 days a

Figure 2 -- Google Reader displaying subscribed feed.

2.3 Structuring the RSS and ATOM Feeds

A simplest RSS feed consists of a channel, with its own attributes, an image, and a number of items contained within the channel, each with their own individual attributes, such as this:

- Channel (title, description, URL, creation date, etc.)
- Image (usually a RSS icon linking to the feed URL)
    - Item (title, description, URL, etc.)
    - Item (title, description, URL, etc.)
    - Item (title, description, URL, etc.)

These items inside an RSS feed are just links to other resources, with customized description associated with each item.

An ATOM feed has a similar XML-tree structure, except that "channel" and "item" elements are substituted by "feed" and "entry" elements. A simple ATOM feed looks like this:

- Feed (title, subtitle, link, updated, etc.)
    - Entry (title, summary, link, etc.)
    - Entry (title, summary, link, etc.)
    - Entry (title, summary, link, etc.)

Because of their similar structures when only the required elements are used, a basic transformation between these kinds of RSS and ATOM feeds are easily realized. However, with four major differences between RSS and ATOM – content model, date format, internationalization, and modularity, RSS and ATOM feeds are in many cases not convertible, especially when most of their optional elements are put to use.

2.4 RSS vs. ATOM

The result of the syndication standard battle between RSS 2.0 (RSS Advisory Board, 2009) and ATOM 1.0 (Nottingham and Sayre, 2005) is notoriously to tell. ATOM advocates believe “that RSS has limitations and flaws – such as lack of on-going innovations and its necessity to remain backward compatible” (Wiki, 2008). On the other hand, some insist that “RSS outgrows its usefulness”. Some popular applications encourage people to share location information with each other, an increasing need there exists to describe locations in an interoperable manner so that applications can request, aggregate, and map geographically tagged feeds. Of which the two syndication standards can be a better choice, in the sense of extending the existing feeds with geographic information, remains an open debate.

Back in July, 2005, RSS 2.0 was widely deployed while ATOM 1.0 only by a few early adopters. In January, 2006, ATOM 1.0 was widely supported in its basic forms though many aggregators failed on feeds that did not conform to RSS standards closely. Four months later, major consumer feed-reading applications supported ATOM 1.0 except Bloglines. Another month passed, Bloglines rolled out its new ATOM 1.0 parser. By that time, ATOM 1.0 finally won the ticket to battle with RSS 2.0 (ATOM Wiki, 2008).

ATOM 1.0 defines 21 elements, 9 missing from 30 elements defined by RSS 2.0. Those missing elements are either never widely implemented in practice, or whose capabilities are provided in other ways. For instance, ATOM 1.0 uses XML's built-in "xml:lang" attribute while RSS 2.0 has its own <language> element for identification of the language used in feeds. Besides, RSS 2.0 has <enclosure> or <source> elements, which are replaced by <link rel="enclosure"> and <link rel="via"> in ATOM 1.0 (ATOM Wiki, 2008). Speaking of ATOM's extensible family of "rel" values used after "link" element, Wilson (2008) took advantage of the extensibility and thus created different means of interface for web services under namespace "scast" (see Table 3 in Appendix).

The struggle between RSS and ATOM feeds has hindered the progress of an online syndication standard (RSS specifications, 2010). While there are issues in dealing with dual standards, this thesis explore the advantages of both syndication standards to better serve the needs for Earth science data and service publications.

2.5 DataCasting Feed

According to Bingham et al. (2009), data providers such as NASA, are responsible for collecting and publishing vast quantities of instrument data related to the state of the Earth on a daily and real-time basis. This information is presented through the Internet in various forms—eyewitness news reports, in situ measurements, scientific analysis (created manually or automatically), photographs and web-cams, predictions and forecasts, and personal blogs. Take the satellite data as an example. Satellites monitoring the Earth collect vast quantities of data on a daily, often real-time basis, which would be made available on the Internet by numerous data providers. Note that the NASA EOS missions alone have collected over 4.3 petabytes (1 PB = 1,000 TB) of data to year 2008 (Bingham, et al., 2009). The use of feeds can save user’s trouble of checking on each website to see whether a change has taken place since last visit, and of visiting numerous data provider websites to cross-reference or supplement. Bingham et al. (2008) described that DataCasting solution would empower Earth Science data consumers with the ability to acquire data based on a pre-defined need. They took the concept of RSS feeds, for delivering regularly changing web content, and extending this to represent a stream of data granules and deliver regularly changing Earth Science data content. “Datacasting” was coined in this paper (Bingham et al., 2009) because of its similarities with “Podcasting” concept, whose mission was to add extensions to RSS usage that will then enable data consumers to download data granules from Earth Science data providers as soon as the data is made available.

Datacasting is built upon a subscribe-publish architecture, similar to ordinary RSS feeds, where a data consumer subscribes to a HTTP-served XML (eXtensible Markup Language) feed so that whenever a data provider broadcasts the availability of new data files (i.e. data granules) via an XML feed (i.e. RSS) the data consumer can learn about the recently updated products simultaneously. The authors then take a step further by suggesting a solution for filtering on the metadata of a feed in order to identify granules of interest based on user-defined criteria. In this way, the feed reader only needs to initiate data pull for those metadata satisfying some criteria, which would save users from reading all incoming feeds and directly lead their focus to the objects of interest.

Current commercial feeds, written with RSS or the ATOMPub standards, are not capable of displaying temporal and spatial attributes of a certain geographic feature in the first place. Without looking further into the text description or even linking to the original web page, users cannot grasp where and when the geographic event is. The same thing is true in searching and filtering feeds by the temporal and spatial parameters. Only when the geographic feature is expressed by primary tags of a feed will it be convenient enough for the feed reader to perform simple search on points, lines and polygons. Bingham et al. (2009) created a namespace “dcast” for dataCasting. Some elements associated with namespace “dcast” can be seen in the Appendix Table 2.

Also in Bingham et al. (2009) extensions from the GeoRSS namespace were used to add specific geographic details to the Earth Science data. The “georss:where” element is used to describe the location of the bounding region of the data granule. This element can be applied to arbitrary polygon-shaped regions as well as rectangles. An example using “georss:where” can be seen in Table 1.

2.6 ServiceCasting Feed

The dataCasting architecture and scheme described in Bingham et al. (2009) seems suitable to Earth science data’s subscribe-and-publish paradigm, the difficult in attaching multiple enclosures and providing multiple service interfaces in RSS makes RSS a less than desired option for serviceCasting.

By gathering serviceCasts from many other providers, in which links pointed to callable interface (WSDL or WADL), server endpoint, and human-readable documents are included, the Publish-and-Subscribe system is capable to offer users the routinely or periodically updated advertisements that they desire to keep up with (Wilson, 2008). Table 3 illustrates the diverse interfaces ATOM formats support under namespace “scast” (for serviceCasting). While research and some degree of implementation have been reported for dataCasting and serviceCasting, the operational deployment of such publish-and-subscribe systems for both Earth science data and services are rare.

In this thesis research, we investigate the feasibility of defining geographic elements in ATOM standard, and deploying the publish-and-subscribe system in an operational geospatial information and service system, namely the Geobrain system.

2.7 From SOA to ROA

Service Orientated Architecture (SOA) is an architectural style of providing processes as services, which does not limit the use of verbs and nouns, or say, what you do and whom you do it to. On the other hand, Representational State Transfer (REST) defines a uniform interface and hence limits the use of verbs, so that one can only use POST, GET, PUT and DELETE with REST.

In principle, REST is often more about nouns since it limits the number of verbs, whereas SOA is equally about verbs and nouns. And the Pub/Sub system is a different type of SOA, called Resource Oriented Architecture (ROA), which uses a REST approach. The relationship between these three can be seen in Figure 3.

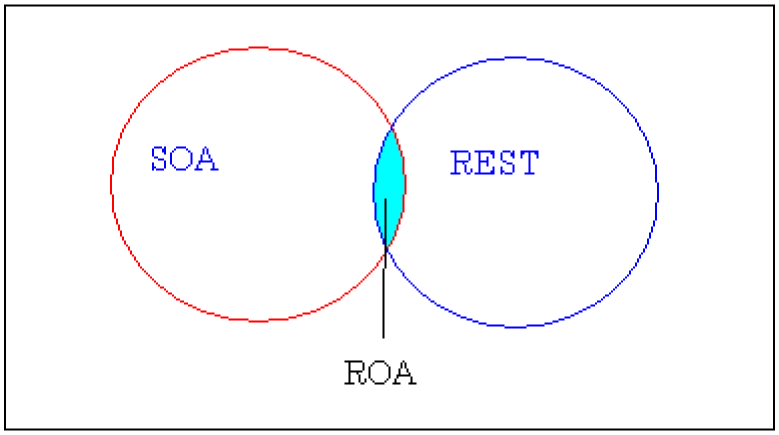
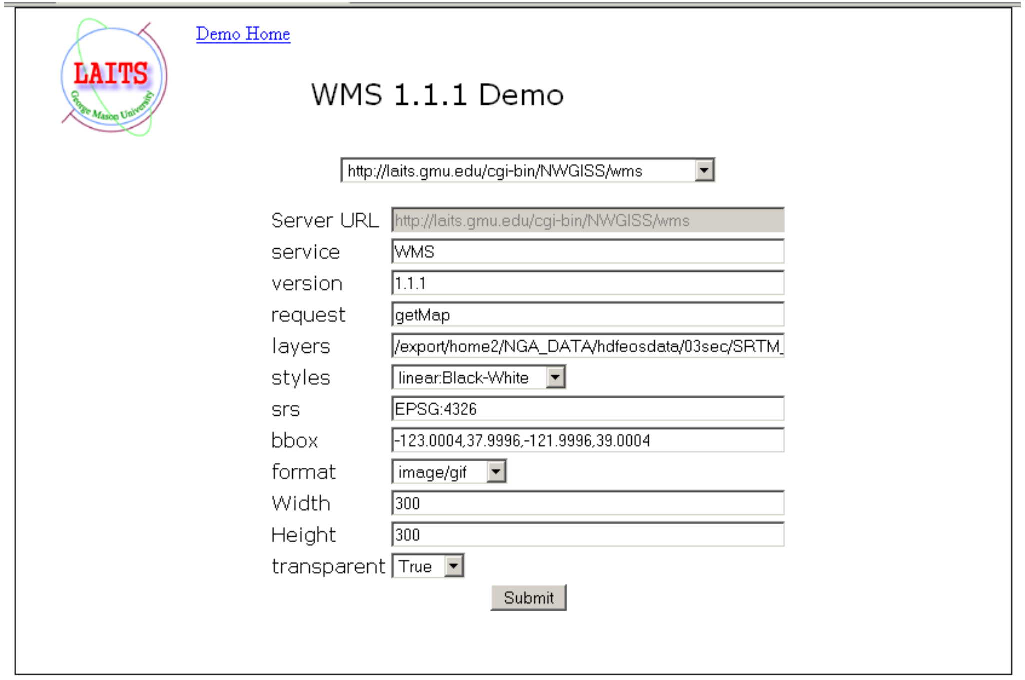


Figure 3 -- SOA VS. REST

How do we know that Pub/Sub is not just SOA? That’s because, according to ATOM Publishing Protocol, a feed publisher only uses POST, PUT, and DELETE commands to manage entries in the feed. The command POST is for creating new entries, by sending requests to the collection. The command PUT is to update existing entries, or to overwrite the existing ones. And a publisher can use DELETE command to delete entries from the collection. These three simple commands can be powerful tools to maintain a routinely or periodically changed feed and to provide the public the most synchronous news. Hence, we can tell that at least the ATOM Protocol is using a RESTful approach.

3. Data

In order to test run the feed generator and the feed reader, two kinds of data are required for basic input as we generate RSS and ATOM feeds for GeoBrain website – geographic data (e.g. satellite data) and Web service interfaces. Since OGC WMS, WCS, and WFS are widely used for retrieving Earth Science Data and information, they are also an important test indicator in our research. In Figure 4 and 5, the demonstrations of WMS and WCS Web services are provided by GeoBrain. Representing these OGC Web services via ATOM feeds and later aggregate these feeds become the first goal of our research.

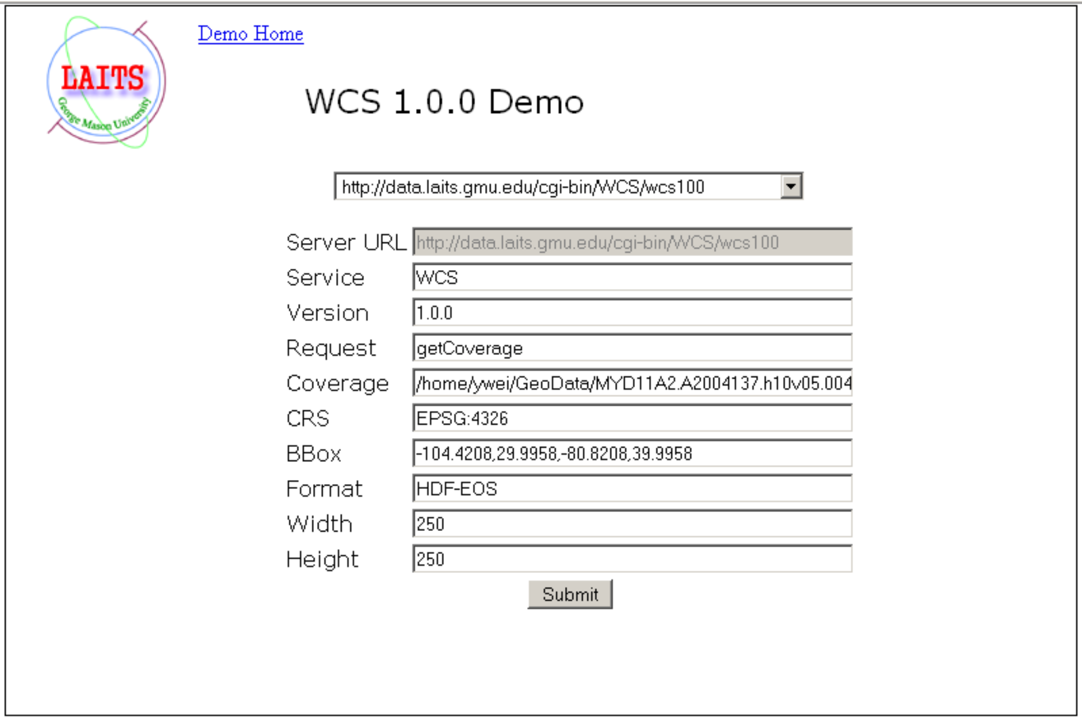


WMS 1.1.1 Demo

http://laits.gmu.edu/cgi-bin/NWGISS/wms

Server URL	http://laits.gmu.edu/cgi-bin/NWGISS/wms
service	WMS
version	1.1.1
request	getMap
layers	/export/home2/NGA_DATA/hdfeosdata/03sec/SRTM_
styles	linear:Black-White
srs	EPSG:4326
bbox	-123.0004,37.9996,-121.9996,39.0004
format	image/gif
Width	300
Height	300
transparent	True

Figure 4 – OGC WMS Demo from GeoBrain Web Site.



WCS 1.0.0 Demo

http://data.laits.gmu.edu/cgi-bin/WCS/wcs100

Server URL	http://data.laits.gmu.edu/cgi-bin/WCS/wcs100
Service	WCS
Version	1.0.0
Request	getCoverage
Coverage	/home/ywei/GeoData/MYD11A2.A2004137.h10v05.004
CRS	EPSG:4326
BBox	-104.4208,29.9958,-80.8208,39.9958
Format	HDF-EOS
Width	250
Height	250

Figure 5 -- OGC WCS service hosted by GeoBrain.

In this thesis, the overall mechanism for implementing the architecture uses a message format known as ATOM+GeoRSS, which extends the popular ATOM specification (for encoding timely event information) to support simple geographic features with GeoRSS. GeoRSS is a logical step forward with a few tags added to RSS that hold spatial data, which may save a user much trouble if he/she is considering creating feeds for a weather forecast (e.g. the temperature in Fairfax, VA will be 84 degrees with partly cloudy skies). With GeoRSS, the user could include that temperature information and the coordinates of Fairfax, VA in a standard form.

Nowadays, this combined feed content has been popularized by Google News and can be easily overlaid on a map using Google Maps. There are two encodings of GeoRSS:

GeoRSS Simple and GeoRSS GML. Both encodings support the encoding of basic geometries – point, line, polygons and box. It falls out of our expectation that GeoRSS Simple geometries may only support one coordinate reference system – WGS-84 latitude/longitude in decimal degrees. In this case, GeoRSS GML is a better option since it is a formal GML Application Profile, and allows the coordinate reference system to be specified with any GML feature (see Table 4 and Table 5). Both formats are designed for use with ATOM 1.0, RSS 2.0 and RSS 1.0 (OGC 08-001).

4. Methods

Aiming to install an operational publish-and-subscribe feed system on GeoBrain server, we use a feed reader made up of Python scripts and a feed aggregator implemented by Java codes. Details are listed below.

4.1 Publish and Subscribe

Franklin P. Jones once said, “The trouble with being punctual is that there’s nobody there to appreciate it.” The same situation comes to pulling feeds from the publisher’s websites; there is always going to be a balance between the need to be up-to-date and the need to refrain from the publisher’ server by requesting the feed every few seconds. Hammersley (2003) mentioned in his book “Content Syndication with RSS” that, there were three stages of work for the whole publish and subscribe approach to take place. Before anything happens, the user shall subscribe to a system that watches the feed continuously. This system has been publishing notifications to all users when the feed changes. The users are then able to update their copies of the feed, ordinarily by requesting it from the server. So basically there are three characters to watch between the scenes: the user, the feed, and the Publish and Subscribe system (better known as the cloud).

The process of Publish and Subscribe within RSS 0.92 and 2.0 is comprised of several steps as shown in Figure 6 (Hammersley, 2003). First, a message via XML-RPC, SOAP, or HTTP-POST, is sent to the cloud from the user's system to subscribe. Second, the cloud returns true if the registration is successful; or else, the registration is failed. After registration, the cloud will detect or be informed of a change in the feed at some time. The cloud will send message to the user using the protocol requested, giving the URL of the changed feed. In the fifth step, the user requests the freshly updated feed from its server and does whatever can be done with feeds (i.e. to view the feed via feed reader, or to update the relevant content in CSW).

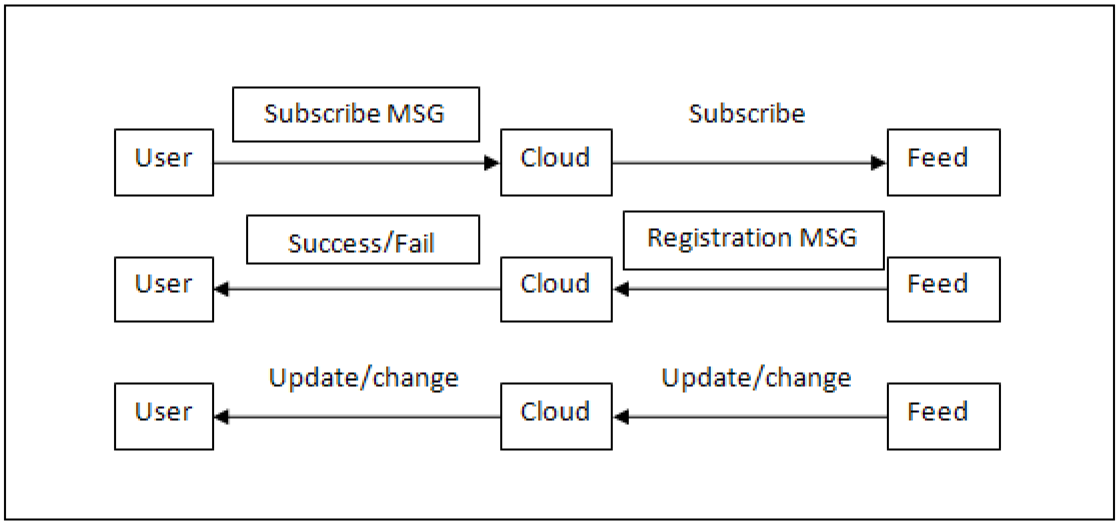


Figure 6 -- Three Stages in Publish & Subscribe System.

An important feature of RSS 0.92 Publish and Subscribe system is that subscriptions expire after 25 hours, forcing users to renew subscriptions on these feeds every day. This weakness will not be seen in the operations of ATOM feeds. The RSS 0.92 Publish and Subscribe system is just like other web services, except that the user's system not only passes a message and waits for a reply, but also continually listens for the cloud and/or the server trying to talk to it.

4.2 Information Polling

How does a client application know when new data is available? It's certainly possible for a server to notify clients when new information is available, broadcasting it as an event to any system that happens to be listening. But this event-driven approach does not fit well with the HTTP Web, where clients initiate the communication and the server providing data is relatively inactive until a client comes along. So the approach adopted by the syndication clients is to determine periodically whether a particular information source has changed.

It is easy to see the inefficiency of checking continuously onto a long list of feed addresses, since the client would devote a lot of processor time and network bandwidth to checking feeds that might only change once a week or even longer. And also, if a lot of clients had their aggregators subscribed to a particular feed, then each of them would be checking on a single Web address and the host of the feed would be under a barrage of requests. So to avoid astronomical service bills for the server, or collapsing of the service system under strain, aggregators generally pull the address of interest only occasionally. There is a loose agreement among developers that the minimum time to wait between requests is one hour. The aggregator or feed reader is responsible to keep track of the time elapsed since it last looked at the feed. For example, suppose a feed was last checked at 2:30 pm, and within the feed there were items A, B, C. Having the delay set to be an hour, another HTTP Get request is sent to the feed's host at 3:30 pm. Now the feed, say, contains items B, C, D. Then what the users are viewing in feed readers would be items A, B, C, and D for the last two hours (See Figure 7). Usually the aggregator will record the contents of the feed, in other words its data model will have a state corresponding to the XML feed polled last time (Hammersley, 2003).

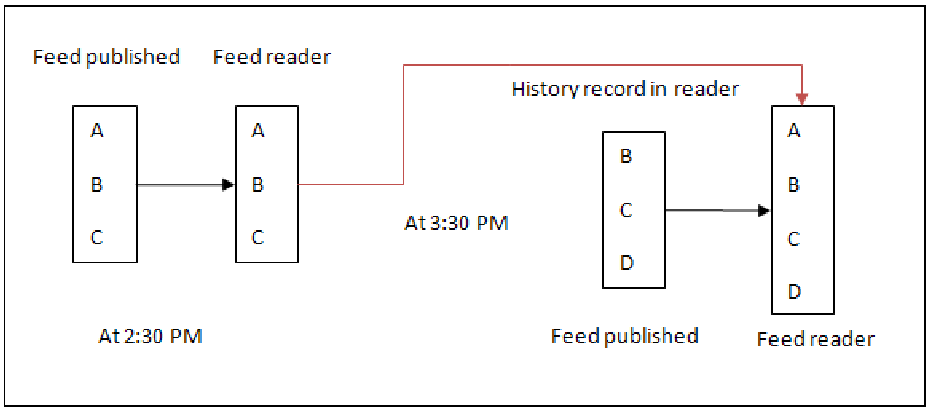


Figure 7 -- A history record of the feed is kept by the feed reader.

4.3 Feed generator

The key of choosing the best practice for producing feed data is to go along with the manner in which one is already managing the data. After all, the syndication with RSS or ATOM feeds has its root in content management, and the technologies involved are intimately related. DOM (Document Object Model) and SAX (Simple API for XML) are two common approaches dealing with XML content. The former allows users to access the information stored in the XML document as a hierarchical object model (or say, interacting with all tree nodes) while the latter chooses to access XML document as a sequence of events instead of creating an object model on top of all XML documents. In this research, I experiment with two methods – one with DOM inclination (I) and the other with SAX inclination (II). Overview of method (I) can be seen in Figure 8.

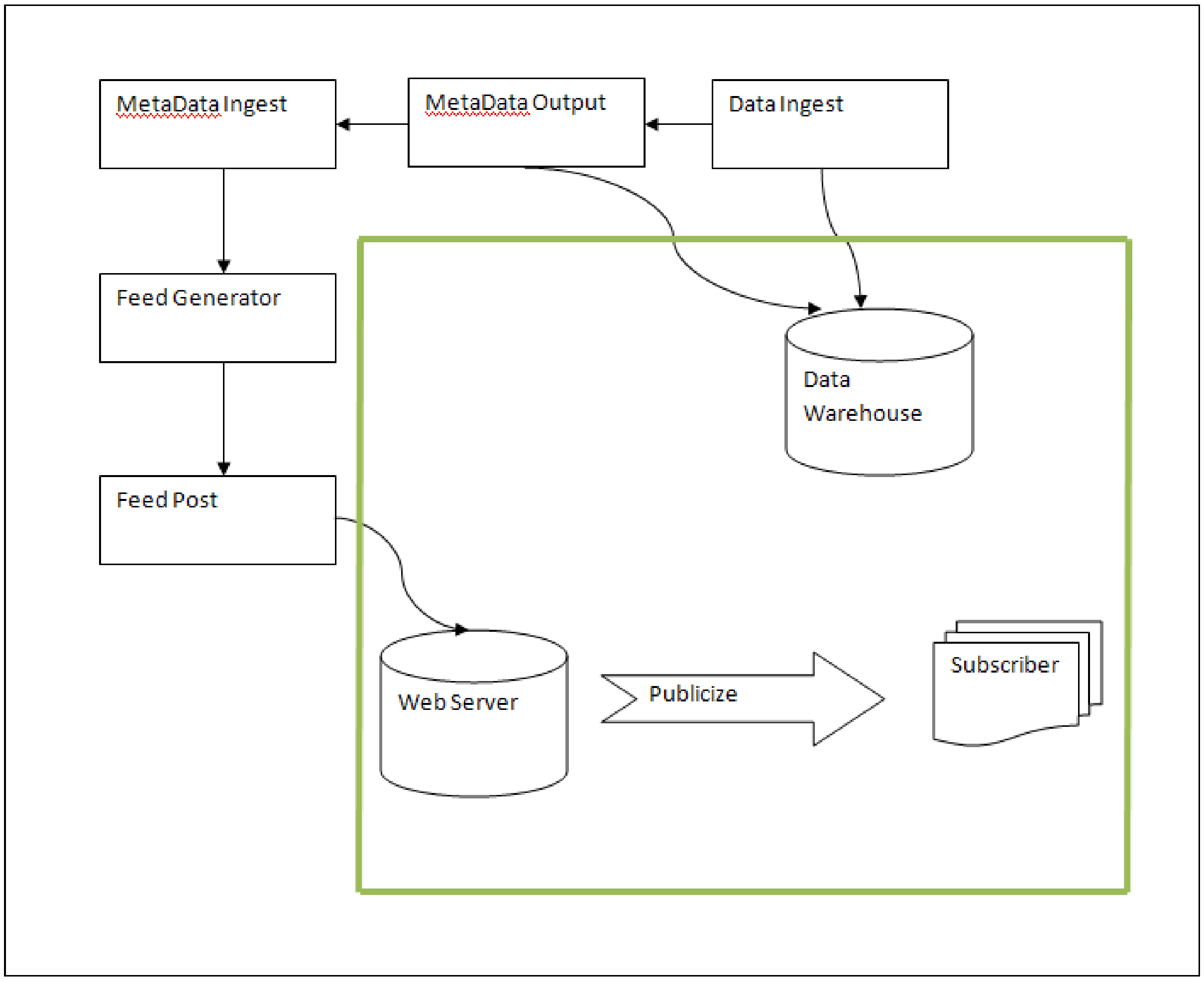


Figure 8 -- Scenario for publishing DataCasting Feed.

There are five major steps for method (I) – Data Ingest, MetaData Output, MetaData Ingest, Feed Generator and Feed Publish. Step four, the Feed Generator, is the most important of all since raw data contents are converted into RSS or ATOM standard XML files via a Python script named PyATOM. It is a set of classes that automatically manage the XML tree structure for the user. The top level of an ATOM feed is an XML "Feed" element with a “<feed>” tag; the Feed element has other elements nested inside it that describe the feed, and then it has zero or more Entry elements, each of which has elements that describe the entry.

To create an XML document with a feed in it, the user does this:

```
xmldoc = XMLDoc( )
feed = Feed( )
xmldoc.root_element = feed
```

To assign an entry to a feed, the user just does this:

```
feed.entries.append(entry)
```

This adds “entry” to the internal list that keeps track of entries. “entry” is now nested inside “feed”, which is nested inside “xmldoc”. Later, when the user wants to save the XML in a file, the user can just do this:

```
f = open(“ATOM.xml”,”w”)
s = str(xmldoc)
f.write(s)
```

To make the string from xmldoc, the XMLDoc class walks through the XML elements nested inside xmldoc, asking each one to return its string. Each element that has other elements nested inside does the same thing. The whole tree is recursively walked, and the tags all return strings that are indented properly for their level in the tree.

Method (II) is to produce information feeds from a simple content management system (CMS), whose data store will be a SQL database with management logic built in using any programming language (see Figure 9 for its scenario). There are three primary interfaces built in HTTP protocol, one for the system administrators, another for content providers, and a third for the target audience. And here I use PHP (Hypertext Preprocessor) as the CMS platform and embed these codes in HTML.

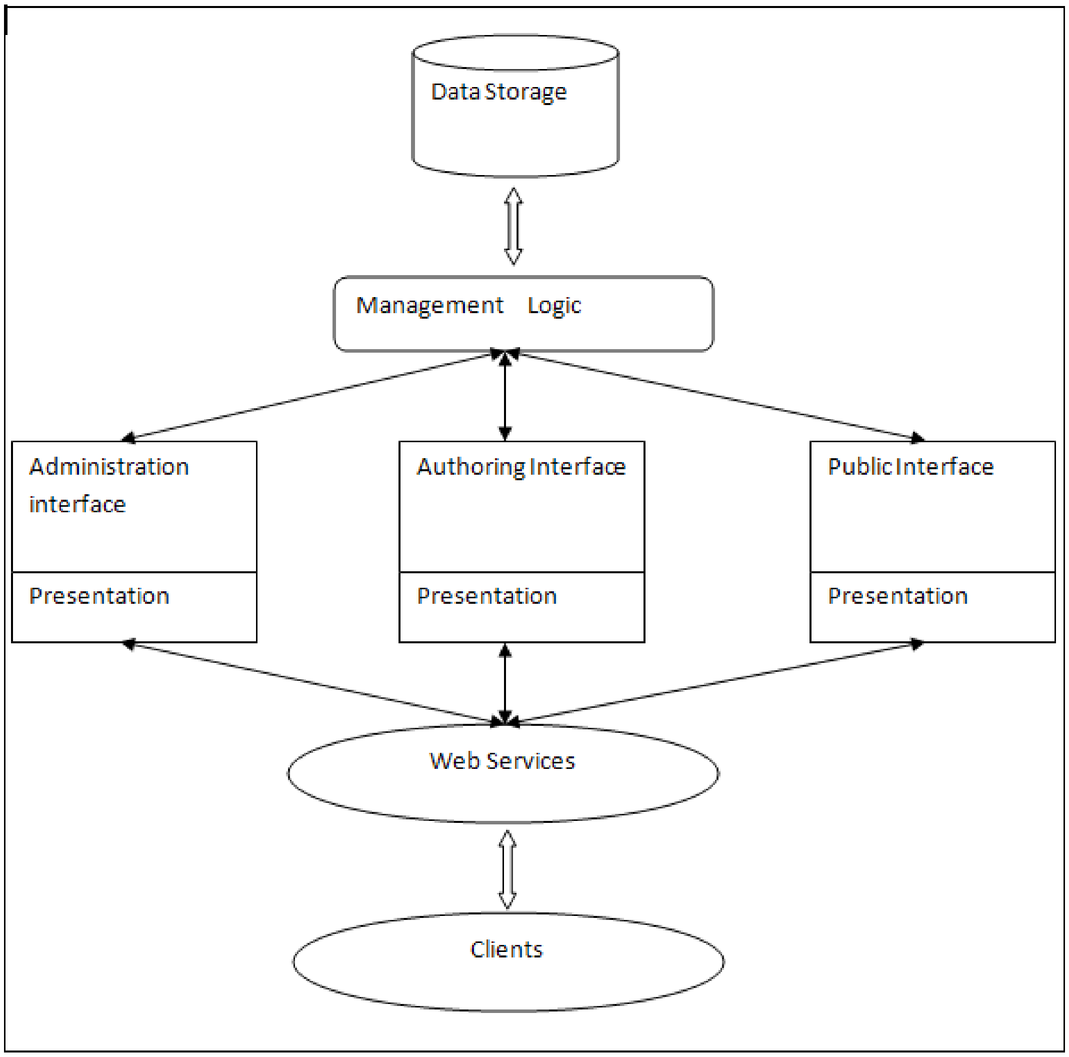
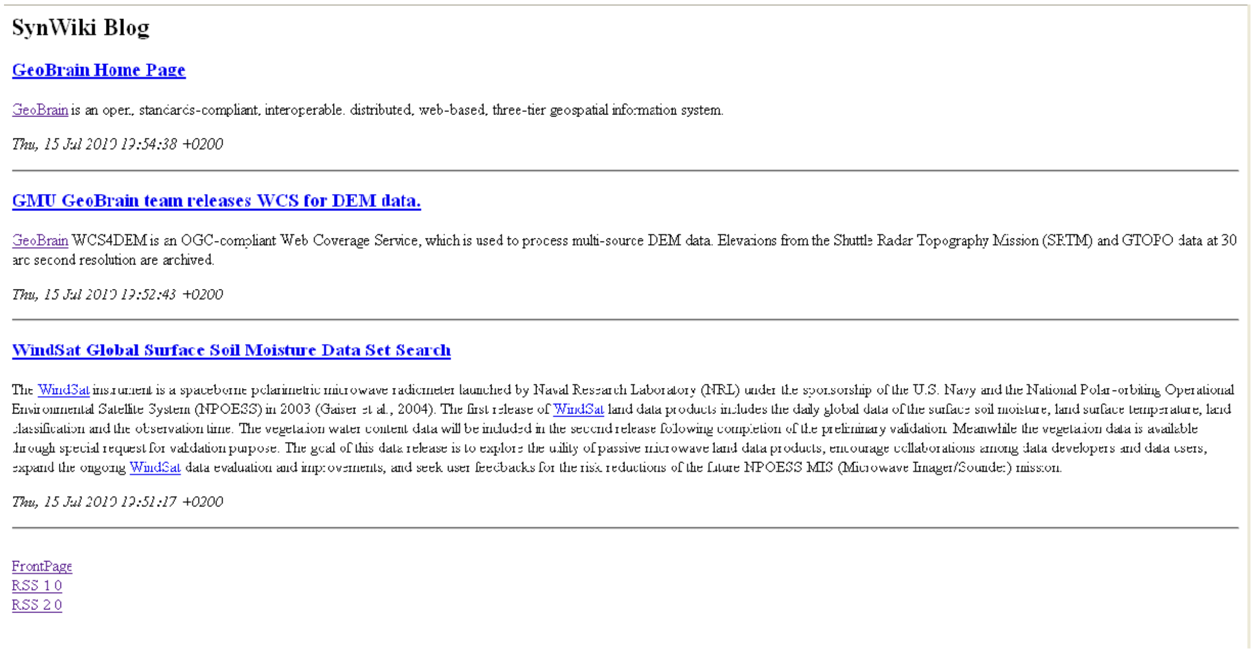


Figure 9 -- Scenario for SAX method of producing feed.

Basically a module in this CMS system can just run a few SQL queries and formats the result into an HTML page. For instance, embedding the following codes:

$dispalyDate = convertDateW3CtoRFC2822($row["date"]);
$content = $row["content"];
$title = $row["title"];
…

yields to a fashioned web blog (see Figure 10).



SynWiki Blog

GeoBrain Home Page

GeoBrain is an open, standards-compliant, interoperable, distributed, web-based, three-tier geospatial information system.

Thu, 15 Jul 2010 19:54:38 +0200

GMU GeoBrain team releases WCS for DEM data.

GeoBrain WCS4DEM is an OGC-compliant Web Coverage Service, which is used to process multi-source DEM data. Elevations from the Shuttle Radar Topography Mission (SRTM) and GTOPO data at 30 arc second resolution are archived.

Thu, 15 Jul 2010 19:52:43 +0200

WindSat Global Surface Soil Moisture Data Set Search

The WindSat instrument is a spaceborne polarimetric microwave radiometer launched by Naval Research Laboratory (NRL) under the sponsorship of the U.S. Navy and the National Polar-orbiting Operational Environmental Satellite System (NPOESS) in 2003 (Gaiser et al., 2004). The first release of WindSat land data products includes the daily global data of the surface soil moisture, land surface temperature, land classification and the observation time. The vegetation water content data will be included in the second release following completion of the preliminary validation. Meanwhile the vegetation data is available through special request for validation purpose. The goal of this data release is to explore the utility of passive microwave land data products, encourage collaborations among data developers and data users, expand the ongoing WindSat data evaluation and improvements, and seek user feedbacks for the risk reductions of the future NPOESS MIS (Microwave Imager/Sounder) mission.

Thu, 15 Jul 2010 19:51:17 +0200

FrontPage
RSS 1.0
RSS 2.0

Figure 10 -- Web blog made with PHP codes.

4.4 Feed reader/aggregator

An ordinary feed reader or aggregator can be used in many ways. First, a user can identify a feed of interest and enter the feed's URL into the feed reader. The feed reader then subscribes to the feed, stores its information and metadata on the local machine. That accomplishes the initial downloads of the feed. Its second functionality is to periodically update the feed with the remote server and show the user any newly updated items. The frequency for re-downloads of the feed from any publisher is determined by the feed reader, typically once every fifteen minutes. Sometimes the feed might not be subjected to any changes in days, so users can customize the updating speed of this specific feed in their feed reader, or to set the download mode to be automatic and scalable. Now the user can subscribe to multiple feeds regardless of their providers or sources, and will be notified of any new data products or Web services automatically, thereby reducing users' efforts of checking each provider websites individually for updates. Third, a feed reader shall allow display, filtering, and other user actions to be performed on the XML tags. Yet, commercial aggregator is not effective for feeds carrying Earth Science data since traditionally only title, author, date and text descriptions get to be loaded into a feed reader. In our aggregator, the parser application recognizes geographic extended elements and extracts the coordinates for later displays of the feed.

What's worth mentioning here is the approach to fetch dataCasting or serviceCasting feeds from provider websites. As we know, feed is published on the Web, and getting hold of data from the Web is generally carried out by one operation – the HTTP GET method. However, the requirements of XML being delivered over HTTP and the additional demand made by subscribers when they poll servers for data make the simple GET method complicated. We shall pay extra attention to syndication so as to ensure our aggregators are on their best behavior in public. To become a reasonably client, we perform conditional GET, which is to check if the subscribed feed has changed since the last visit and quit with only the header downloaded when unchanged.

The aggregation starts with the HTTPConnector class – used to get feed data from the Web (top-left of Figure 11). Then this general-purpose client is greeted by a more syndication-oriented interface, FeedFetcher. Every individual feed will have its own HTTPConnector, FeedFetcher, and an Interpreter. Their purpose is more to simplify the communications rather than carry data themselves. After a new feed is fetched over, the feed data is passed to its Interpreter instance which will parse or process the data. Then the DownloadInterpreter continues to save the feed data as a file. The last step of this cycle is to run refreshAll(), which will call every FeedFetcher instance in turn to ask them to refresh themselves.

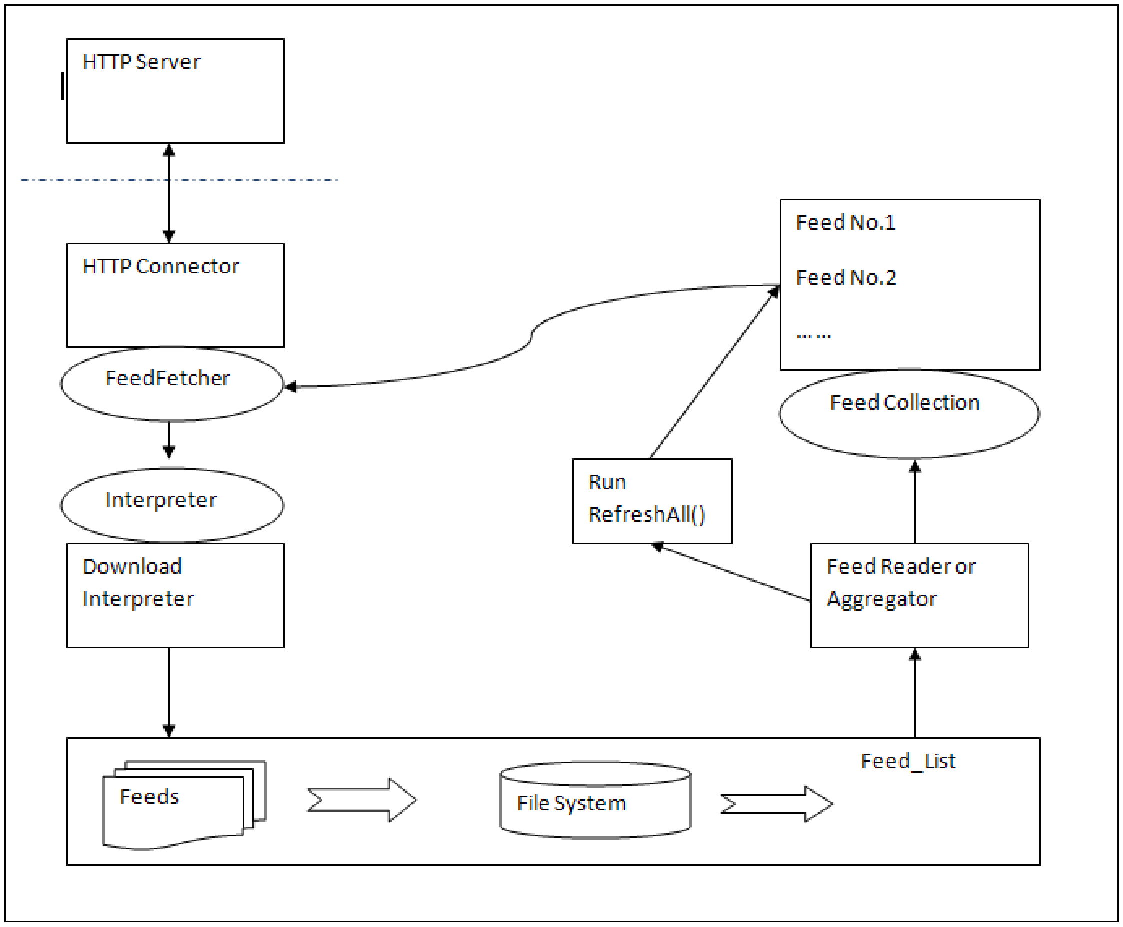


Figure 11 -- Overview of Aggregation initialization.

5. Results

5.1 Namespace Created

One of the primary motivations for defining an XML namespace is to avoid naming conflicts when using or re-using multiple vocabularies. In order to distinguish between duplicate element types and attribute names dataCasting and serviceCasting feeds/entries might have, we create two namespace – “dcast” and “scast” declared as:

```
<feed  xmlns= “http://www.w3.org/2005/ATOM”
       xml:lang= “en”
       xmlns:scast= “http://lester.arl.noaa.gov/2010v1/scast”
       xmlns:dcast= “http://lester.arl.noaa.gov/2010v1/dcast”>
```

The URI references used here, as XML namespace names, are not guaranteed to point to schemas, metadata of the namespace or anything else – they are just identifiers that we do not have any reason to resolve them.

Traditional feed elements are far from enough in the sense of representing dataCasting items. Subscribers may need to see the instrument or platform with which the data was collected, or the start time and end time of the data acquisition. With the flexibility of ATOM feeds, we expanded the list of elements for our dataCasting feeds (see Table 3). Among all “dcast” elements, only “dcast:UID” is required for each feed since it is a unique identifier used in forming directory name for data storage; others are only optional.

Before we dive into the definition of "scast", it is necessary to determine the types of web services GeoBrain provides and the description languages of the corresponding web services. Basically there are two main categories of Web Services – REST and SOAP. Representational State Transfer or REST means that each URL represents some object that one might get its contents using an HTTP GET, or use a POST, PUT or DELETE to modify the object. On the other hand, SOAP (Simple Object Access Protocol) is a solid solution in situations when data needs to be securely transferred that parameters cannot be sent via URIs, or there are large amounts of data waiting for transfer that may even go out of bound within a URI. Hence, the best practice of keeping applications simple and accessible is to try REST first and resort to SOAP only when necessary. As a language describing Web Services in conjunction with SOAP 1.1, HTTP POST and GET, and MIME, WSDL (Web Services Description Language) 1.1 is a standard used by SOAP applications since 2001. In the next few years, W3C issued WSDL 1.2, and 2.0 along with the rise of RESTful web services. However, another specification fills this need in an arguably better way: WADL (Web Application Description Language). Thus, we define "serviceProtocol" and "interfaceDescription" elements under "scast" namespace in order to distinguish between SOAP/WSDL, REST/WADL and other web services (see Table 3).

5.2 Sample ServiceCasting Feeds Created

In order to event-trigger the entire test process of the Pub/Sub system, we have to first insert some metadata of the Web services that GeoBrain supports into the syndication database “syn_db”. The two demonstrations for OGC WMS and WCS will be the first two test cases. The new ingestion of these two groups of metadata triggers the feed creation process, and the next step directly goes to the output of feed content (see Appendix for Table 6 & 7).

5.3 Pub/Sub System Testing

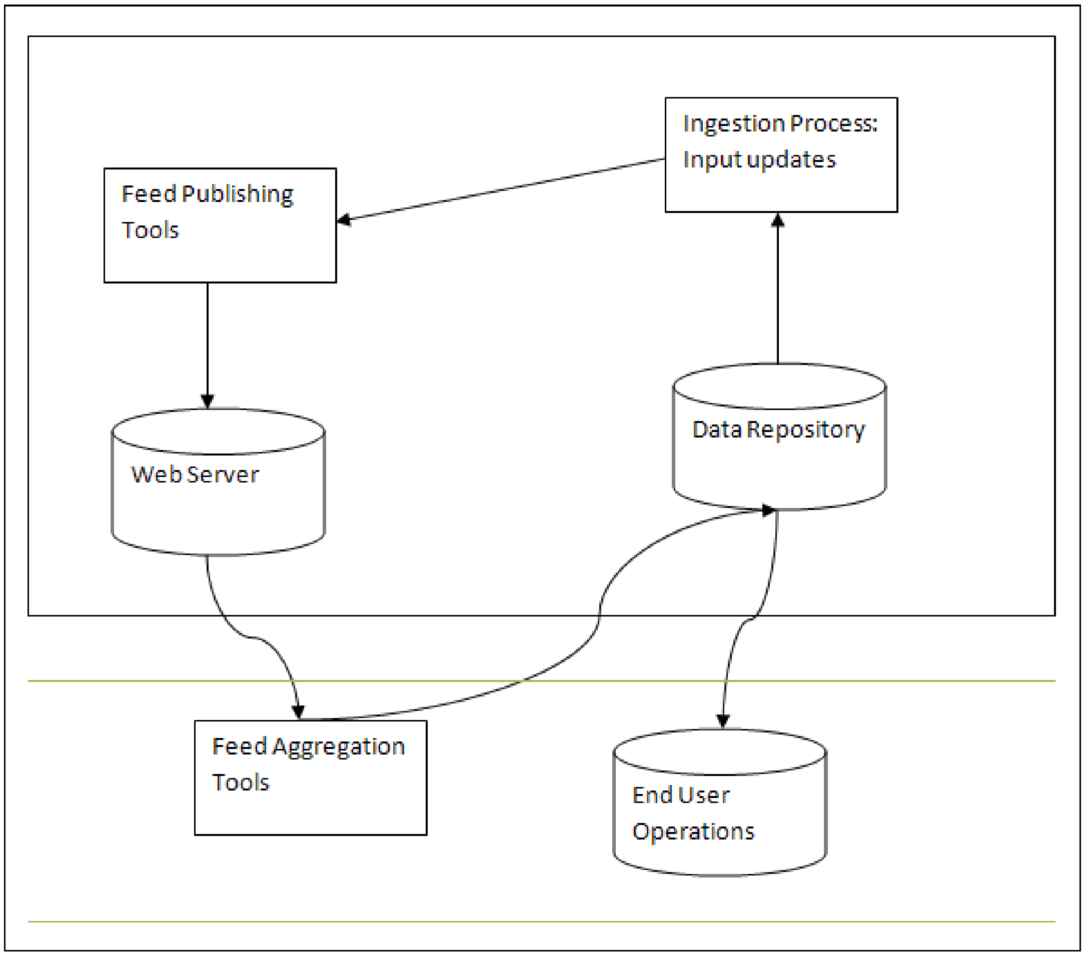
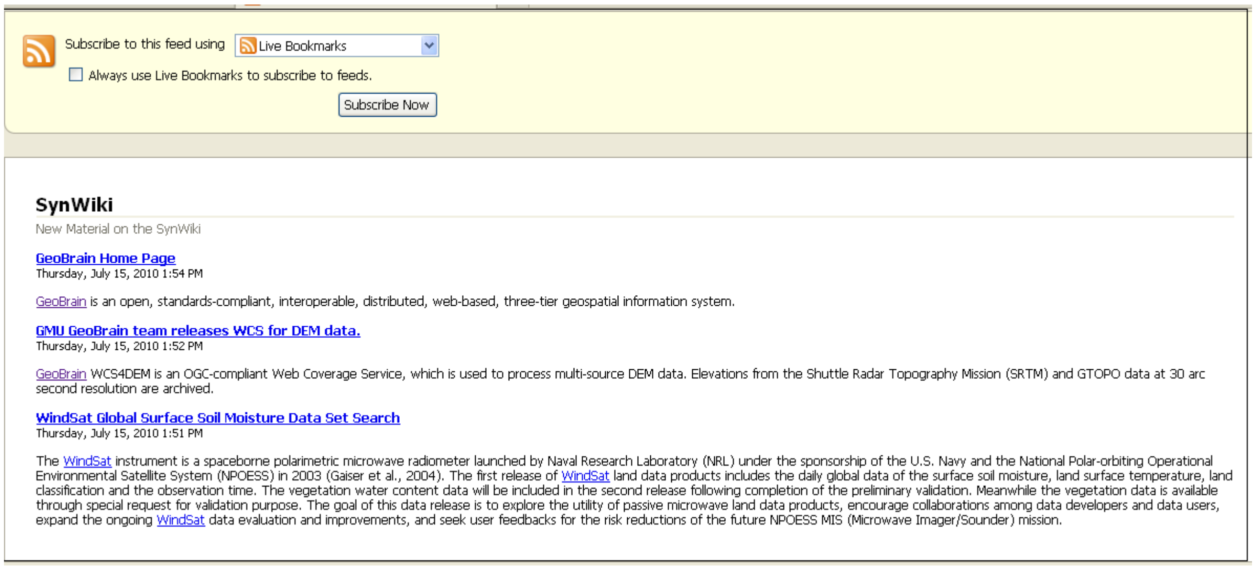


Figure 12 -- Overview of the Pub/Sub System

The advantage of having the feed generator and the feed aggregator on the same server is that feeds generated randomly from any web services or data provided on the server can be ingested into the aggregator directly avoiding to be processed via the cloud. This process can be a primary step in testing whether the feed generator and aggregator are working stably and effectively. Only if the feed aggregator can read the newly-generated feeds into fine formats will these two applications prove to be workable.

Now we have the information feed of GeoBrain websites ready (see figure below).



Subscribe to this feed using
Live Bookmarks
Always use Live Bookmarks to subscribe to feeds.
Subscribe Now

SynWiki

New Material on the SynWiki

GeoBrain Home Page
Thursday, July 15, 2010 1:54 PM

GeoBrain is an open, standards-compliant, interoperable, distributed, web-based, three-tier geospatial information system.

GMU GeoBrain team releases WCS for DEM data.
Thursday, July 15, 2010 1:52 PM

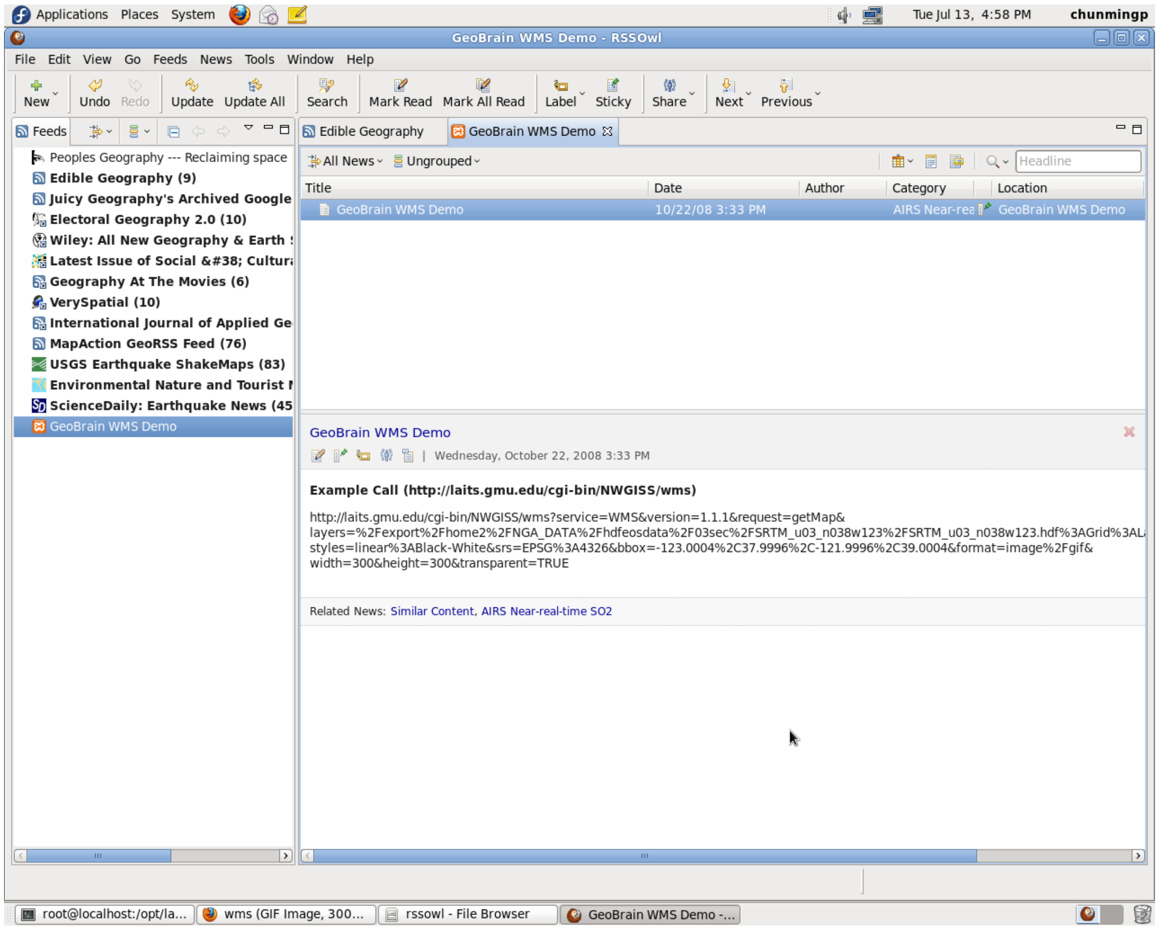
GeoBrain WCS4DEM is an OGC-compliant Web Coverage Service, which is used to process multi-source DEM data. Elevations from the Shuttle Radar Topography Mission (SRTM) and GTOPO data at 30 arc second resolution are archived.

WindSat Global Surface Soil Moisture Data Set Search
Thursday, July 15, 2010 1:51 PM

The WindSat instrument is a spaceborne polarimetric microwave radiometer launched by Naval Research Laboratory (NRL) under the sponsorship of the U.S. Navy and the National Polar-orbiting Operational Environmental Satellite System (NPOESS) in 2003 (Gaiser et al., 2004). The first release of WindSat land data products includes the daily global data of the surface soil moisture, land surface temperature, land classification and the observation time. The vegetation water content data will be included in the second release following completion of the preliminary validation. Meanwhile the vegetation data is available through special request for validation purpose. The goal of this data release is to explore the utility of passive microwave land data products, encourage collaborations among data developers and data users, expand the ongoing WindSat data evaluation and improvements, and seek user feedbacks for the risk reductions of the future NPOESS MIS (Microwave Imager/Sounder) mission.

Figure 13 -- GeoBrain Information feed.

The final step is to subscribe to this feed and see if entries are displayed properly in the aggregator. Here for the sake of visibility in graphic interface, I show the result via an open-source desktop aggregator – RssOwl (see Figure 14).


Applications Places System
Tue Jul 13, 4:58 PM
chunmingp
GeoBrain WMS Demo - RSSOwl
File Edit View Go Feeds News Tools Window Help
New
Undo
Redo
Update
Update All
Search
Mark Read
Mark All Read
Label
Sticky
Share
Next
Previous
Feeds
Peoples Geography --- Reclaiming space
Edible Geography (9)
Juicy Geography's Archived Google
Electoral Geography 2.0 (10)
Wiley: All New Geography & Earth
Latest Issue of Social &#38; Cultur
Geography At The Movies (6)
VerySpatial (10)
International Journal of Applied Ge
MapAction GeoRSS Feed (76)
USGS Earthquake ShakeMaps (83)
Environmental Nature and Tourist
ScienceDaily: Earthquake News (45
GeoBrain WMS Demo
Edible Geography
GeoBrain WMS Demo
All News
Ungrouped
Headline
root@localhost:/opt/la...
wms (GIF Image, 300...
rssowl - File Browser
GeoBrain WMS Demo -...

Title	Date	Author	Category	Location
GeoBrain WMS Demo	10/22/08 3:33 PM		AIRS Near-rea	GeoBrain WMS Demo

GeoBrain WMS Demo

Wednesday, October 22, 2008 3:33 PM

Example Call (http://laits.gmu.edu/cgi-bin/NWGISS/wms)

http://laits.gmu.edu/cgi-bin/NWGISS/wms?service=WMS&version=1.1.1&request=getMap&
layers=%2Fexport%2Fhome2%2FNGA_DATA%2Fhdfeosdata%2F03sec%2FSRTM_u03_n038w123%2FSRTM_u03_n038w123.hdf%3AGrid%3AL
styles=linear%3ABlack-White&srs=EPSG%3A4326&bbox=-123.0004%2C37.9996%2C-121.9996%2C39.0004&format=image%2Fgif&
width=300&height=300&transparent=TRUE

Related News: Similar Content, AIRS Near-real-time SO2

Figure 14 -- GeoBrain feed displayed in RSSOwl.

6. Discussion

Today users and organizations are collaborating over the creation and on-going maintenance of geospatial data sets and web services, so it is of critical importance for all partners to be able to share geographic information in real time, and make the costs associated with sharing negligible, both in labor and IT resources (OGC 08-001). Some may say that this requirement can be approached from the database perspective, collaborating organizations using database replication technology for their geographic data sharing needs. Yet this strategy cannot work mostly because every organization needs to work with a different information model.

The adoption of the Publish-and-Subscribe system will reduce many efforts a service provider used to put on advertising its products, decentralize repository with auto aggregation and enable auto-invocation with different typed links. The aggregation and display of feeds are so simple, since there are already various free feed readers in the market. The best part of this system is that user, service provider and broker are taking a partial responsibility here. A service provider should provide feed for its newly updated or changed web contents and place the feed in the cloud. A user should subscribe to a feed via the feed reader at first, and then every time the feed reader is opened, the user shall be able to see updated feed synchronously. The broker, or the cloud, is responsible for notifying feed readers, which are listening the whole time, that there is a new feed from their subscribed website.

Since there are currently no uniformed standards when publishing feeds for Earth Science data and Web services, our own feed creation and aggregation system, which is able to handle dataCasting and serviceCasting at the same time, based on GeoRSS extended ATOM standards, will doubtlessly provide an easy way out for those feed publishers and consumers who are troubled by how to represent and display Earth Science data and services.

The proposed architecture follows the SOA model in a loosely coupled environment (OGC 08-001). The service allows data/services providers to publicize changes to their geographic databases in a machine-readable manner, enabling geo-synchronization clients to automatically perform synchronization activities.

7. Conclusions and Future Work

In this thesis research, we investigated the syndication and aggregation of geospatial metadata for Earth science data, information, and services and designed and developed a publish-and-subscribe (syndication/aggregation) system in connection with the operational Geobrain system. The publish-and-subscribe makes Web-accessible geospatial data and the services be more easily discoverable, accessible, and usable by increased user communities. The enhancement to the Geobrain system with the publish-and-subscribe system takes advantages of both SOA (service oriented architecture) and ROA (resource oriented architecture). The ATOMPub is places a RESTful protocol on top of the usage of ATOM feeds. With this feed generator and aggregator deployed on GeoBrain, users can subscribe to and receive updates from the GeoBrain system, including both data/information products and Web services. The results from this thesis research show that the publish-and-subscribe system is feasible in an operational geospatial data, information, and service system. Some additional enhancements are identified, among which two could be achieved in the near term. First, subscribing to other sites' information feeds can be a great way to synchronize our own data store with the external sources yet some sites with no published feeds will be ignored. It might be integrate the web scraping with the feed aggregation process so that our server can get the newest updates from the web sites that do not actively push (i.e., feed) their contents.

Second, our feed aggregator is capable of consuming and parsing feeds and later transforming them into HTML codes but we currently do not provide GUI (Graphic User Interface) for public viewers. We will enhance our aggregator to provide more visualization capabilities with GUI and to integrate the GUI with other Geobrain Web service GUIs. Ultimately we will develop an online Earth science data/information visualization and analysis system that will allow users to use the data and services hosted by Geobrain but also by any other systems whose contents can be harvested by our aggregator.

List of Reference

List of Reference

ATOM Wiki, (2008) “ATOM 1.0 and RSS 2.0 Compared”, http://intertwingly.net/, (accessed on July 6, 2010)

Ayers, D., & Watt, A., (2005) “Beginning RSS and ATOM Programming”, Wiley Publishing, Inc.

Bingham, A., et al., (2009) “Earth Science Datacasting: Informed Pull and Information Integration”, IEEE Transactions on Geoscience and Remote Sensing, 47(10): 3570- 3580

Di, L., P. Zhao, W. Han, X. Li, M. Deng, 2009. The Implementation of Geospatial Web Services and Workflows in GeoBrain. In Proceedings of 2009 IEEE International Geoscience and Remote Sensing Symposium, July 12-17, 2009, Cape Town, South Africa.(Extended abstract).

Gregorio, J., and de hOra, B, (2007) “The Atom Publishing Protocol, http://tools.ietf.org/html/rfc5023” (accessed on April 13, 2010)

Libby, D., 1999, RSS 0.91 Spec, revision 3, http://web.archive.org/web/20001204093600/my.netscape.com/publish/formats/rss-spec-0.91.html, (accessed on July 6, 2010)

Hammersley, B., (2003) “Content Syndication with RSS”, O’Reilly & Associates, 1- 15

Han, W. , L. Di, P. Zhao, Y. Wei, X. Li, 2008: Design and Implementation of GeoBrain Online Analysis System (GeOnAS), In Proceedings of 8th International Symposium on Web and Wireless Geographical Information System, Michela Bertolotto, Cyril Ray, Xiang Li (Eds):Lecture Notes in Computer Science (LNCS), 5373, pp. 27-36.

MSDN, (2010) “Messaging Patterns in Service Oriented Architecture”, http://msdn.microsoft.com/en-us/library/aa480027.aspx, (accessed on July 6, 2010)

Nottingham, M., and Sayer, R., (2005) “The Atom Syndication Format, http://tools.ietf.org/html/rfc4287”, (accessed on April 13, 2010)

OGC 08-001, (2008) “Loosely Coupled Synchronization of Geographic Database”, Open Geospatial Consortium, Inc.

RFC 5023, (2007) “The ATOM Publishing Protocol”, http://tools.ietf.org/html/rfc5023, (accessed on July 6, 2010)

RSS Specifications, (2010) “RSS won the syndication standards battle”, http://www.rss-specifications.com/rss-won.htm, (accessed on April 13, 2010)

RSS-DEV Working Group, (2008) “RDF Site Summary (RSS) 1.0”, http://web.resource.org/rss/1.0/spec, (accessed on July 6, 2010)

RSS Advisory Board, (2009) “RSS 2.0 Specification”, http://www.rssboard.org/rss-specification, (accessed on May 12, 2010)

Wikipedia, (2010) “Atom (standard)”, http://en.wikipedia.org/wiki/Atom_(standard) (accessed on April 13, 2010)

Wilson, B., (2008) “Service Casting: A Proposal for Advertising Web Services”, http://esdswg.eosdis.nasa.gov/WG/TIppt/ESDSWG_M7S4_Service_Casting.ppt (accessed on April 13, 2010).

Appendix

Table 1 - An example using "georss:where" to specify locations

```
<georss:where>
  <gml:Envelop>
    <gml:lowerCorner> 3.100000 6.900000</gml:lowerCorner>
    <gml:upperCorner> 63.100000 106.900000</gml:upperCorner>
  </gml:Envelop>
</georss:where>
```

Table 2 - Elements in "dcast" namespace

Element	Description	Example
dcast:UID	A string which uniquely identifies the data stream and is used in forming the data store directory name.	< dcast:UID>AVHRR-night-airquality-4KM</ dcast:UID>
dcast:dataSource	A string indicating the name of the instrument or platform, with which the data was collected and processed.	< dcast:dataSource > AVHRR </ dcast:dataSource>
dcast:fileName	A filename the feed reader will use when downloading and saving the enclosure.	< dcast:filename>foo-bar-20090801.nc </ dcast:filename>
dcast:startTime	Date and time of the data granule collection starts at the instrument.	< dcast:startTime>Tue, 20 Apr 2010 13:58:01 GMT</ dcast:startTime>
dcast:endTime	Date and time of the data granule collection ends at the instrument.	< dcast:endTime>Tue, 20 Apr 2010 19:30:59 GMT</ dcast:endTime>
dcast:preview	The URL of a thumbnail image in JPG, PNG or GIF formats.	< dcast:preview> http://geobrain.laits.gmu.edu/.../s0456789-sst.jpg</ dcast:preview>
dcast:format	The data file format in ESML.	< dcast:format>HDF-EOS</ dcast:format>
dcast:analyst	The email address of an analyst who is capable to add custom elements to the feed.	< dcast:analyst>cpeng1@gmu.edu </ dcast:analyst>
dcast:customEltDef	Used to declare any custom metadata items.	<dcast:customEltDef name="rainDepth" type="float" units="cm" />

dcast:customElement	Any tags associated with a previously declared datacasting customEltDef above.	<dcast:customElement name="rainDepth" value= "8.12" />

Table 3 – Elements in "scast" namespace

Element	usage	example
scast:InterfaceDescription	The URL is pointing to WSDL, WADL, or OGC WXC GetCapabilities Call.	<link rel= "scast:InterfaceDescription" href= "http://phrase1/phrase2/phrase3" />
scast:ServiceEndpoint	The URL is pointing to where the service can be called.	<link rel= "scast:ServiceEndpoint" href= "http://phrase1/phrase2/phrase3" />
scast:ServiceDocumentation	The URL is pointing to the human-readable documentation	<link rel= "scast:ServiceDocumentation" href= "http://phrase1/phrase2/phrase3" />
scast:serviceSemantics	Possible service semantic types – OGC.WXS, AdHoc, Simple, Human, OpenSearch	<scast:serviceSemantics> OGC.WMS </scast:serviceSemantics>
scast:serviceProtocol	Possible service protocols – SOAP, HTTP, HTTPPOST, REST, AJAX, etc.	< scast:serviceProtocol> HTTP </ scast:serviceProtocol>

Table 4 -- An actual feature to be updated using GeoRSS GML.

```
<georss:where>
  <gml:Point srsName="urn:ogc:def:crs:EPSG:6.11.2:4326">
    <gml:pos>43.928958 -78.933283</gml:pos>
  </gml:Point>
</georss:where>
<georss:featureOfInterest>
  <gml:FeatureCollection
   xmlns:gb="http://www.geobase.ca/interop-pilot-2007">
    <gml:boundedBy>
       <gml:boundedBy>
          <gml:Envelope
           srsName="urn:ogc:def:crs:EPSG:6.11.2:4326">
            <gml:lowerCorner>43.92566 -78.93541</gml:lowerCorner>
            <gml:upperCorner>43.92895 -78.93328</gml:upperCorner>
          </gml:Envelope>
       </gml:boundedBy>
    </gml:boundedBy>
    <gml:featureMember>
    <gb:RoadSegment>
      <gb:cgdiId>urn:uuid:cebc5f9b-8bcb-47d6-a51c-
0753d5170475</gb:cgdiId>
      <gb:planimetricDataAccuracy>0
      </gb:planimetricDataAccuracy>
      <gb:acquisitionTechnique>n.a.
      </gb:acquisitionTechnique>
      <gb:acquisitionProvider>n.a.</gb:acquisitionProvider>
      <gb:acquisitionOrRevisionDate>2007-11-06-05:00
      </gb:acquisitionOrRevisionDate>
      <gb:nationalRoadClass>Arterial</gb:nationalRoadClass>
      <gb:numberOfLanes>0</gb:numberOfLanes>
      <gb:pavementStatus>n.a.</gb:pavementStatus>
      <gb:pavedRoadSurfaceType>n.a.</gb:pavedRoadSurfaceType>
      <gb:unpavedRoadSurfaceType>n.a.</gb:unpavedRoadSurfaceType>
      <gb:roadSegmentID>cebc5f9b-8bcb-47d6-a51c-0753d5170475
      </gb:roadSegmentID>
      <gb:segment>
        <gml:LineString srsName="urn:x-
ogc:def:crs:EPSG:6.11.2:4326">
          <gml:posList>43.927222 -78.93475 43.928612 -78.935393

… etc …

                     43.926257 -78.9333834 43.9259504 -78.9332889
          </gml:posList>
        </gml:LineString>
      </gb:segment>
    </gb:RoadSegment>
    </gml:featureMember>

… etc …

  </gml:FeatureCollection>
</georss:featureOfInterest>
```

Table 5 -- Another actual feature to be updated using GeoRSS GML.

```
<georss:where>
  <gml:Point srsName="urn:ogc:def:crs:EPSG:6.11.2:4326">
    <gml:pos>50.74993437116105 -100.6120574479606</gml:pos>
  </gml:Point>
</georss:where>
<georss:featureOfInterest>
  <gml:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
```

```
xmlns="urn:oasis:names:tc:emergency:cap:1.1"
xmlns:gml="http://www.opengis.net/gml">
    <gml:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.11.2:4326">
        <gml:lowerCorner>50.06164 -101.18588</gml:lowerCorner>
        <gml:upperCorner>51.01049 -99.71660</gml:upperCorner>
      </gml:Envelope>
    </gml:boundedBy>
    <gml:featureMember>
      <PlumeAffectedArea fid="PT1"> <id>1</id>
      <elapsedHours>1</elapsedHours>
      <areaAffected>
        <gml:Polygon srsName="urn:ogc:def:crs:EPSG:6.11.2:4326">
          <gml:exterior>
            <gml:LinearRing>
              <gml:posList> 50.73637 -100.796 50.79564 -100.726
50.80338 -100.656 50.79447 -100.610 50.77536 -100.572 50.74900 -100.549
50.71873 -100.542 50.68659 -100.555 50.66018 -100.586 50.64414 -100.632
50.64092 -100.683 50.65117 -100.733 50.67305 -100.772 50.70323 -100.794
50.73637 -100.796
              </gml:posList>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </areaAffected>
      </PlumeAffectedArea>
    </gml:featureMember>
    <gml:featureMember>
      <PlumeAffectedArea fid="PT2"> <id>2</id>
      <elapsedHours>2</elapsedHours>
      <areaAffected>
        <gml:Polygon srsName="urn:ogc:def:crs:EPSG:6.11.2:4326">
          <gml:exterior>
            <gml:LinearRing>
              <gml:posList> 50.78288 -100.803 50.84021 -100.716
50.84655 -100.696 50.85718 -100.623 50.85131 -100.549 50.82967 -100.482
50.79489 -100.431 50.75121 -100.402 50.70389 -100.399 50.67471 -100.410
50.64772 -100.430 50.62421 -100.459 50.60528 -100.495 50.59181 -100.537
50.58445 -100.582 50.58911 -100.674 50.60091 -100.717 50.61839 -100.755
50.64072 -100.787 50.66686 -100.810 50.69557 -100.823 50.72549 -100.827
50.75518 -100.820 50.78288 -100.803
              </gml:posList>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </areaAffected>
      </PlumeAffectedArea>
    </gml:featureMember>
  </gml:FeatureCollection>
</georss:featureOfInterest>
```

Table 6 -- Created ServiceCasting Feed for GeoBrain WMS

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/ATOM"
     xml:lang="en"
     xmlns:scast="http://lester.arl.noaa.gov/2010v1/scast">
  <title>WMS DEMO</title>
  <link rel="self" href="http://lester.arl.noaa.gov/WMS_DEMO.xml"/>
  <updated>2010-06-27T19:33:59.368863</updated>
  <author>
    <name>Chunming Peng</name>
    <email>cpeng1@gmu.edu</email>
  </author>
  <id>uri:http://laits.gmu.edu/cgi-bin/NWGISS/wms</id>

  <entry>
    <title>WMS DEMO</title>
    <id>uri:http://laits.gmu.edu/cgi-bin/NWGISS/wms</id>
    <updated>2010-06-22T19:33:59.368863</updated>
    <summary>Demonstrating how to launch OGC WMS Web service at GeoBrain.</summary>
    <scast:serviceSemantics>OGC.WMS</scast:serviceSemantics>
    <scast:serviceProtocol>HTTP</scast:serviceProtocol>
    <category schema="scast" term="WMS DEMO" />
    <georss:where>
        <gml:Envelop>
            <gml:lowerCorner> -123.100000 37.990000</gml:lowerCorner>
            <gml:upperCorner> -120.100000 39.900000</gml:upperCorner>
        </gml:Envelop>
    </georss:where>
    <link type="application/xml"
        title="Server endpoint"
        href="http://laits.gmu.edu/cgi-bin/NWGISS/wms" />
    <link rel="scast:serviceInterface" type="application/xml"
        title="Service interface description"
        href="http://laits.gmu.edu/cgi-bin/NWGISS/wms?service=WMS&version=1.1.1
&request=GETCAPABILITIES" />
    <link rel="scast:serviceEndpoint" type="application/xml"
        title="Server endpoint"
        href="http://laits.gmu.edu/cgi-bin/NWGISS/wms" />
    <link rel="scast:serviceDocumentation" type="text/html" xml:lang="en-us"
        title="Service documentation"
        href="http://geobrain.laits.gmu.edu/products.htm" />
    <link rel="alternate" type="text/html" xml:lang="en-us"
        title="Service documentation"
        href="http://geobrain.laits.gmu.edu/products.htm" />
    <content type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml">
  <p>
```

```
    <b>Example Call (http://laits.gmu.edu/cgi-bin/NWGISS/wms)</b>
  </p>
  <p>
<![CDATA[
http://laits.gmu.edu/cgi-bin/NWGISS/wms?service=WMS&version=1.1.1&request=getMap
&layers=%2Fexport%2Fhome2%2FNGA_DATA%2Fhdfeosdata%2F03sec%2FSRTM_u03
_n038w123%2FSRTM_u03_n038w123.hdf%3AGrid%3ALandset0%3ABandn0%3A1
&styles=linear%3ABlack-White&srs=EPSG%3A4326&bbox=-123.0004%2C37.9996%2C-
121.9996%2C39.0004&format=image%2Fgif&width=300&height=300&transparent=TRUE
]]>
  </p>
</div>
    </content>
  </entry>
</feed>
```

Table 7 -- Created ServiceCasting Feed for DEM WCS.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/ATOM"
      xml:lang="en"
      xmlns:scast="http://lester.arl.noaa.gov/2010v1/scast ">
  <title>GeoBrain WCS for DEM</title>
  <link rel="self" href=" http://lester.arl.noaa.gov/DEM_WCS.xml "/>
  <updated>2010-04-01T16:25:16.368863</updated>
  <author>
    <name>Chunming Peng</name>
    <email>cpeng1@gmu.edu</email>
  </author>
  <id>uri:http://geobrain.laits.gmu.edu/cgi-bin/gbwcs-dem</id>

  <entry>
    <title>DEM WCS (SRTM 90m, SRTM 30m, GTOPO30arc)</title>
    <id>uri:http://geobrain.laits.gmu.edu/cgi-bin/gbwcs-dem</id>
    <updated>2010-04-01T16:50:50.368863</updated>
   <summary>Get DEM data (SRTM 90m, SRTM 30m, GTOPO30arc) through OGC WCS
requests.
   GetCapabilities,  DescribeCoverage,  GetCoverage
    </summary>
    <scast:serviceSemantics>OGC.WCS</scast:serviceSemantics>
    <scast:serviceProtocol>HTTP</scast:serviceProtocol>
   <category schema="scast" term="DEM WCS" />
    <georss:where>
         <gml:Envelop>
                <gml:lowerCorner> -90.000000 38.000000</gml:lowerCorner>
                <gml:upperCorner> -89.000000 39.000000</gml:upperCorner>
         </gml:Envelop>
    </georss:where>
   <link type="application/xml"
         title="Server endpoint"
         href="http://geobrain.laits.gmu.edu/cgi-bin/gbwcs-
dem?service=wcs&version=1.0.0&request=describecoverage&coverage=SRTM_90m_Global,
SRTM_30m_USA,GTOPO_30arc_Global" />
   <link rel="scast:serviceInterface" type="application/xml"
         title="Service interface description"
         href="http://geobrain.laits.gmu.edu/cgi-bin/gbwcs-
dem?service=wcs&version=1.0.0&request=getcapabilities" />
   <link rel="scast:serviceEndpoint" type="application/xml"
         title="Server endpoint"
         href="http://geobrain.laits.gmu.edu/cgi-bin/gbwcs-
dem?service=wcs&version=1.0.0&request=getcoverage&coverage=SRTM_90m_Global
&bbox=-90,38,-89,39&crs=epsg:4326&format=GTiff&store=true" />
   <link rel="scast:serviceDocumentation" type="text/html" xml:lang="en-us"
         title="Service documentation"
         href="http://geobrain.laits.gmu.edu/products.htm" />
   <link rel="alternate" type="text/html" xml:lang="en-us"
         title="Service documentation"
         href="http://geobrain.laits.gmu.edu/products.htm" />
   <content type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml">
  <p>
   <b>Example Call (SRTM_90m)</b>
  </p>
  <p>
<![CDATA[
http://geobrain.laits.gmu.edu/cgi-bin/gbwcs-
dem?service=wcs&version=1.0.0&request=getcoverage&coverage=SRTM_90m_Global
&bbox=-90,38,-89,39&crs=epsg:4326&format=GTiff&store=true
]]>
  </p>
</div>
   </content>
  </entry>
</feed>
```

CURRICULUM VITAE

Chunming Peng received her Bachelor of Engineering from Zhejiang University (Hangzhou, China) in 2008. She is later enrolled into PhD program in Earth System and Geographic Sciences and now expected to get her Master of Science (secondary degree) in Geography and Geoinformation Science from George Mason University in Aug, 2010.