Data Architecture For Location Aware Smarter Water Sensor Visualization

Eli M. Dow
emdow@us.ibm.com

Steven Schmidt
skschmid@us.ibm.com

Thomas Fitzsimmons
tdfitzsi@us.ibm.com

Doug Milvaney
dlmilvan@us.ibm.com

ABSTRACT
The goal behind the Smarter Water Sensor Visualization project is to design and build a common visualization portal for next generation hydrological data. Smarter Water is designed to be extensible and is configured to support meteorological data from multiple sources as well as track any correlative events. The result of the Smarter Water project is a generic foundation for location based sensor data.

General Terms
Measurement, Design, Human Factors, Standardization

Keywords
Smarter Water, ODM, Sensors, Hydrological data, Visualization

1. INTRODUCTION AND MOTIVATION
Building a smarter planet is a corporate initiative of IBM. There are three big ideas driving the building of a smarter planet. The first is to instrument the worlds systems, the second is to interconnect them, and the third is to make them intelligent. Say "water" and relatively few people automatically think about IBM, but when you start talking about instrumentation and interconnected systems, the connection makes sense. As it turns out, IBM is undertaking a variety of water research projects [SMARTER WATER], but the topic of this paper is a next-generation data visualization portal developed by IBM for the Beacon Institute for Rivers and Estuaries (BIRE) [2].

BIRE is headquartered along the Hudson River estuary in the city of Beacon, New York, just a few miles away from several IBM sites, including Poughkeepsie, NY which shares Beacon's view of the Hudson. For quite some time, BIRE and other agencies [7] on the Hudson River have been deploying sensor instrumentation to record and collect data. The key to this project is aggregating the recorded data and sharing it efficiently and effectively so that researchers can retrieve and use the data as easily as possible. This paper describes one effort to aggregate a broad variety of water data, from the Hudson river and elsewhere, into a single, cohesive system for analysis. The contribution of this work is two-fold. The first major contribution is the discussion of enhancements made to the Observations Data Model [4], developed by the Consortium of Universities for the Advancement of Hydrological Science, Inc (CUAHSI), that is necessary for enterprise grade storage and analysis of data. The second contribution of this paper is the discussion of the data aggregation architecture developed by our team along with a discussion of the technical hurdles preventing researchers from easily analyzing existing hydrological data.

2. PROJECT ARCHITECTURE
The architecture of our software can be roughly divided into 3 constituent parts. The first part is the back-end database layer, which is a SQL implementation hosting a schema which is a modification of ODM. Adding support for a new database that supports ANSI SQL syntax is relatively easy. We currently support MySQL [14], Microsoft SQL server [13], and IBM DB2 [9]. We plan to support the open source Postgres database [16] in the near future.

The second constituent part is the business logic which contains the core algorithms for importing and analyzing the data. This portion of the software is written entirely in the JAVA programming language [15]. Java was chosen for its ease of development and enterprise integration. Our implementation consists of a set of libraries and servlets running on either Apache Tomcat [1] or the IBM WebSphere Application Server [10]. The application has not been ported to other containers at this time, such as JBOSS [17], but it should be straightforward to do so given that the application has already been designed to run on two containers from entirely different vendors.

The last component of our software is the user visible elements that make up the front end user interface. This portion of the software was written using a combination of Java Server Pages (JSPs), Javascript/AJAX with jQuery [12], HTML, and CSS.

All of our software was written with the Eclipse Integrated development environment [6] with a version control plug-in to make rapid development effective on a short development time frame [18].

3. OBSERVATIONS DATA MODEL
A key component of the infrastructure used in this implementation is the Observations Data Model. This section of the paper will explain what ODM is, why it was selected, and identify some of the shortcomings that needed to be addressed in order to use ODM in a larger context.

3.1 What is ODM
Observations Data Model (ODM) is a database schema that we have chosen to conform our data to. ODM is designed specifically to hold data about hydrological values as well as where the values have originated from. It was developed by the Consortium of Universities for the Advancement of Hydrological Science, Inc (CUAHSI) with the goal of allowing faster data queries.

3.2 Why We Chose ODM and What Other Choices We Considered

We chose the ODM schema because it allows us to set a standard for hydrological data collection. This means that all data we collect will be manipulated from its original instantiation to conform to this data model. We considered several other choices before deciding such as:

3.2.1 Creating our own Database Schema

The main problem with creating our own database schema is that it would create another standard and that is not what we wanted. We wanted a universal standard that was well established and that others had access to. Another reason we opted not to design our own data model is that ODM has most of the tables required for us and is already well documented.

3.2.2 Net Common Data Form

NetCDF [19] is a set of libraries and self-describing, machine independent formats that support creation, access, and sharing of scientific data. The problem we encountered with NetCDF is that it proved to be too complicated for our needs.

3.3 Problems with ODM

Although ODM has proven useful and has many features we sought in a data model, it was not without its faults. One of the more immediate problem was that ODM was designed for use solely on Microsoft Windows platforms. This meant that there were no creation scripts for other databases such as MySQL and DB2. Also, any tools designed around ODM are Windows only programs which denies use to researchers who wish to use their products on different operating systems.

One of the biggest hurdles to overcome when using ODM is the fact that it is US-centric. This means that the columns in the tables are designed for use specifically with data from US sources such as having columns to hold data for “state” and “county” while there are no columns for country or province. To remedy this, we now use the ISO 3166-1 alpha-2 country codes along with a state or province's two-letter code combined in the state column. This allows ODM to be used globally. For example, New York in the United States would be “US-NY” while British Columbia in Canada would be “CA-BC”. One problem this causes is the extra parsing that must be done when retrieving the data. Another problem that we encountered was the provider USGS stored data for Iraq as Idaho. We solved this by now checking if the site name contains Iraq before inserting into the database.

We have since resolved these shortcomings by implementing schema creation scripts for both MySQL and DB2 database implementations. This allows ODM to be instantiated on low cost open platforms such as Ubuntu Linux with MySQL on Intel class hardware, or on Enterprise grade compute platforms like DB2 version 9 running on AIX 6.1 on IBM System p servers.

We feel the enhancements made to ODM to support other database back-ends are the right approach for the long term. Though we have made no effort to port the various userspace applications which are commonly associated with ODM to the new database platforms, we feel that in the long run, those efforts should be undertaken to support open low cost research platforms as well as those platforms which are used for serious computation and data analysis for hydrological research.

3.4 Extensions to ODM

We have since made several changes to ODM to better suit our own needs. This includes the creation of several tables including: SiteHasData, SiteProvidesVariable, Stats, and a set of tables that keep track of whether or not a site has data for a given week, month, or year. SiteHasData currently stores a list of site's Ids that have at least one entry in the data values table. SiteProvidesVariable stores each variable Id and what site it belongs to. The stats table currently holds two values: sites_with_data which is the count of the total number of distinct sites in SiteHasData, and data_points which is a count of the number of entries in the data values table. We feel these extensions are are necessary for any kind of serious data analysis on data stored in an ODM instance.

Another extension to the original schema was the inclusion of foreign keys to enforce referential integrity. This helps greatly when retrieving data, as we know that if a table references another column, that column value will be present in the database and will not be deleted or missing. We feel that the lack of foreign keys was likely an oversight by the original designers of the schema, and seems to be a non-controversial change which should be adopted upstream.

As well as adding tables, we have created creation scripts for two large, well known databases, MySQL and DB2. This helps people who are not running Windows to set up their own ODM schema.

With our discussion of the data storage format concluded, let us turn our attention to the mechanism by which we populate the database.

4. DATA IMPORTERS

A data importer is a program that retrieves data from a given source, which is to say some entity which provides data values with time stamp information and geolocation data, in the source's own data format. The data is retrieved and then parsed on the fly as it is normalized into the ODM format.

These data importers are each unique in the way they retrieve the data as each source stores their data differently. We will go more in depth about this later. Data importers have been written for individual next generation hydrological sensors and government agencies with copious volumes of data, as well as commercial and non-profit agencies. A data importer can be written, from scratch, in about a single week's time.

4.1 Collecting the Data

In order to populate our own databases, each data importer must establish a connection to the source it wishes to get data from. The data provided is entirely up to the provider of the information. The retrieval methods used range from HTML scraping to downloading CSV files to establishing direct database connections to remote hosts.

Once connected, the importer must get the meta-data information for each site (physical location) the source has data for. This includes the geographic location of the sensor (usually

provided in latitude and longitude) as well as a full list of the variables are measured at the site.

With the list of geographic locations and the variables stored at those locations, the next step is to actually parse or acquire the data values corresponding to those variables. As mentioned earlier, each data importer does this step differently as some handle CSV files while others must scour through HTML. Data importers will usually wait until they have a chunk of data before inserting as to not consistently pressure the database.

A data importer is designed to never stop trying to get data and will soon be configured to acquire the current week's data and then once that task is complete, to start backtracking and reading in historical data. This means data will always be current and eventually all historical data will be present.

4.2 Data Sources

4.2.1 Acoustic Doppler Current Profiler

Our ADCP [8] source provider is responsible for getting data values pertaining to the Hudson river. We retrieve this data by scraping comma separated values from an HTML web page. The unique facet of the ADCP source provider is that it does not present historical data usually over a day or two old. This requires our Data Importer to be continuously running in order to ensure that data is not missed because the data is not archived externally. Our Data Importer is responsible for maintaining a constant connection to the ADCP web page so that we can serve as the database archive of the hydrological data. About 500 lines of provider specific code were written to achieve the data importing of the ADCP source provider.

4.2.2 Beacon Institutes for Rivers and Estuaries

Beacon [2] is a local water research company located in Beacon, NY. We currently only have a small snapshot of data from their database but we are working on establishing credentials to allow a persistent database connection to allow us to pull data directly. Once we are able to get a direct connection to the Beacon database, we will be able to provide a more accurate and complete analysis of the hydrological data for the Hudson River. It will allow us to correlate the recorded information with the ADCP source provider enabling better quality assurance.

4.2.3 Canadian Water Office

The Canadian Water Office [3] is a superb source of hydrological data from across Canada. We acquire our data from the Canadian Water Office by HTML scraping. However, we must first bypass their disclaimer and obtain a cookie saying that we have access to the web page as well as establishing a SOAP connection. Due to these hurdles, the Canadian Water Office Data Importer takes around 1000 lines of provider specific code to access, parse, and record their data.

4.2.4 Husdon River Environmental Conditions Observation System

HRECOS [7] is another Hudson river environmental agency. To retrieve historical data for HRECOS, CSV files are downloaded to a local directory and are manipulated from there. Since HRECOS provides their data in CSV files, we were able to adapt the parsing logic from the ADCP Data Importer to work with these downloaded files. Data from the current year, however, is not available in CSV file format, and must be retrieved from the web service provided by HRECOS. This requires the data importer to dynamically shift between retrieving current data and retrieving historical data using two unique methods of data acquisition. Having more sensors in the Hudson River area enables us to correlate our data with ADCP and Beacon so as to achieve better analysis of the current conditions and to formulate predictions for future hydrological values. The HRECOS Data Importer required approximately 650 lines of provider specific code.

4.2.5 National Oceanic and Atmospheric Agency

NOAA [20] is another large data provider that we pull data from. For NOAA we first must establish a connection to their website which has proven to be fickle and can take several minutes to establish a connection depending on their site's ability to accept SOAP Requests. Once a connection is established, our SOAP Request is processed asking for the available information variables for a specific sensor. We retrieve all of the information for a site through SOAP Requests and SOAP Responses. Due to the nature of SOAP Requests and the various types of available hydrological and meteorological information from NOAA, there are numerous different types of requests that must be sent and received from their site. As a result, the NOAA Data Importer is about 1100 lines of provider specific code.

4.2.6 National Weather Service

NWS [21] is our second largest provider with over 4,000 sites. The data is brought in through HTML scraping as well. This Data Importer was more tedious to write than most of the other providers because of the lack of true site information provided by NWS. We had to write our own API to figure out site information such as the city and state that the site was located in because NWS only displays the latitude and longitude for a site. Other than the location issues, NWS was mainly another application of the HTML scraping we used from the ADCP and Canadian Water Office Data Importers. The NWS Data Importer is comprised of about 600 lines of provider specific code.

4.2.7 United States Geological Survey

USGS [22] is our largest data provider with over 12,200 sites. We currently have a data importer thread responsible for each state and province we are receiving data for to improve our data range over every state. Each individual thread finds all of the available sites in its designated state and proceeds to access the available data for only those sites. This enables parallel importing of hydrological data so that we can see information across the country rather than having to wait as it proceeded one at a time, state by state. USGS is another Data Importer that retrieves data through HTML scraping. This Data Importer is written with about 650 lines of provider specific code.

4.3 Normalizing the Data

All data brought in, whether it be source information, site information, variables or data values, is all broken down and reformatted to be inserted into our ODM database. We have implemented filters which catch a variety of errors in the source data. Though we cannot aim to completely validate and scrub all incoming data, we can check for obvious errors such as temperatures, velocities, or chemical concentrations which fall outside of the realm of values expected in earths hydrological systems.

Once the data is stored in our ODM database, manipulation and observation of the data becomes trivial. The normalization allows comparison between the data of multiple sources, even if the sources themselves store data radically differently from each other. Having the data in one standardized form also allows us to easily visualize the data, as well as release the data online for researchers to download for their own use.

4.4 Gaps In Data

Gaps exist in data for various reasons: a sensor went down for a period of time, data was corrupted, there was no data to collect for that variable, etc. To solve this, we limit our user interface to only show times where data is present and only show variables that have data for the given time period. This ensures that the user will never be able to generate visual data when no data is present and thus prevent the user from becoming frustrated trying to find time periods with data.

5. VISUALIZATION OF THE DATA

The aggregation of vast sums of data is not, by itself a useful endeavor. It is the act of making sense of that data which is the useful activity.

5.1 Standard Time Series

To further that goal, we have opted to implement automatic time series graphs for all data values recorded. Graphs may be drawn as line graphs or scatter-plots and saved directly as PNG or JPEG graphical formats. Users may select the data range (which defaults to a one week interval ending at the current date) by clicking easy to use calendar buttons or by typing the date in textual format.

In addition to allowing users to alter the time range, graph type, and file format, we have opted to provide direct access to the core graphing services as a web-service which is completely documented on the web site. This allows sophisticated users, such as hydrological researchers, to generate graphs of the data pro grammatically for use in their own publications.

Having implemented that service, it quickly became obvious that single value automated time series were insufficient to answer anything but the most basic questions about the data. We therefore implemented arbitrary combinations of data automatically accessible through an easy to use web site, and through the same web service described for single variable time series data graphing.

5.2 Arbitrary Combinations of Data

One key aspect of our data visualization system, is that it is to the best of our knowledge the first such system that allows user defined, arbitrary combinations of data to be graphed concurrently from disparate data sources.

Users are presented with an easy to use web page that lists the agencies providing data and the names of all the geographic site locations with measured data. Once a geographic location is specified, a list of all the measured variables is presented as easy to chose check-boxes. Clicking any combination of check-boxes updates the data visualization graphs in real time. This activity allows users to look into the data and correlate values. One example of such a correlation is comparing the data measurements at the IBM Poughkeepsie ADCP location to the NWS location across the river. There should, generally speaking be a high degree of correlation due to the geographic proximity.

Other examples of the usefulness of this design, include being able to record a value such as salinity at multiple locations throughout the Hudson river. Since the Hudson river is tidal in nature, one can track the salinity over great distances at various times to look for the influence of the tidal rhythms of this data.

Additional combinations which have proven insightful are the combination of meteorological data in conjunction with hydrological data such as air temperature and rain fall to measured hydrological values in bodies of water to look for correlation trends.

6. WEB SERVICES TO OBTAIN DATA

In addition to allowing visual manipulation of our data, we offer our data freely to be downloaded in three different formats in order to spread the ideas of smarter water. Our formats have been selected due to their commonality and how easy the data could be parsed from these formats.

6.1 Why Web Services?

Web services were the obvious choice to display the data as the web is cross platform, no special hardware is required beyond a connection to the Internet, and it does not require top of the line computers to run. Web services also allow easy access to our data by being only a mouse click away with no special downloads.

6.2 NetCDF

NetCDF, also known as Net Common Data Format, is a set of libraries and self-describing, machine independent formats that support creation, access, and sharing of scientific data. We have the ability to export our data as NetCDF. We do this because it is a widely used format and has ties to IBM.

6.3 Water Markup Language

Water Markup Language, better known as WaterML, is a branch of traditional XML developed by CUAHSI. WaterML was created in conjunction with ODM to display. Since we use ODM as our database schema, it made sense to export a format designed to display the contents of ODM. More informtion about WaterML can be found at the CUAHSI web site [5].

6.4 Compact XML Representation

For internal usage between front end AJAX and Java servlets, we use our own version of XML with compacted identifier tags. This is done to speed up transfer of XML across the system because now that the files are significantly shorter, they are smaller and thus easier to transfer. This is done at the expense of readability. The readability however is not an issue as an end-user will never be exposed to the XML generated by these servlets.

In addition to our own compact version, we do offer a full, non-compacted XML version of data values that can be accessed and downloaded by end-users.

6.5 Concluding Thoughts on Data Representation

We can say unequivocally that if each agency we communicated with provided some form of self describing data, ideally a

standard such as WaterML, the process of aggregating and comparing data would be much simpler. There is obviously work to be done in order to persuade the agencies involved to ratify and adopt common standards. Finally, it should be noted that providing additional data output formats as a web service is a relatively trivial endeavor and can generally be accomplished within a week if the output file format is well specified.

7. ANALYTICS

We currently perform basic analysis on our data. This includes processing the mean, median, mode, and standard deviation for a given date range in order to identify outliers.

We hope to later include such analysis as being able to detect outliers as they are added as well as being able to notice invalid data by comparing data values from sites that are relatively near one another. These would greatly help our data sets by removing incorrect values which could skew analysis and cause invalid graphs to be drawn.

In addition, we hope to implement a notification scheme for identifying situations where recorded values seem invalid or trending towards extremes. The applications of such a system include identifying potentially hazardous chemical concentrations on extremely short time scales.

8. FUTURE WORK

The system has been implemented in a robust, easily expandable manner in order to adjust and adapt to future developments in the field of hydrological research. Optimally, this system will see additions over time to adjust for future desired data or new sensor technology. There is also room to expand the options for acquiring and examining the data stored in our databases, such as new graphing formats or more advanced analytics for stored data.

As mentioned above, the system is designed such that new data importers can easily be written for other sources or organizations. This means that as new organizations come to light in this area, we can retrieve their data and store it for observation and analysis along with the organizations that came before them. As a data importer takes about a week to develop to completion, the system can be up to date with new organizations very quickly.

The methods for visualization and analysis can also be expanded and improved. Currently we implement the viewing of data in line graphs or scatter-plots [26, 27], in either JPG or PNG image formats. Due to the use of the standard time series, new graph types can be easily implemented over the existing engine. Future possibilities include the implementation of heat maps, bar graphs, and Box-and-Whisker plots [25, 23, 24]. New image format implementation is trivial, and could be enacted quickly if a request for a new format was made.

As discussed previously, we also wish to implement more advanced analytics into the system and web service. The inclusion of data sanitization will improve the overall quality of data, and also allow us to inform organizations in charge of sensor maintenance when their sensor is malfunctioning.

A loftier, ideological goal of the Smarter Water project is the promotion of a single standard in the hydrological data field. Ideally, Smarter Water will work with government agencies and private institutions to adopt WaterML and ODM as the standards for hydrological data. This would allow the exchange of data between agencies to be much more efficient.

Standardization would also encourage the building of a community around the acquisition and study of hydrological data, where those in charge of sensor maintenance and data visualization will be able to freely and easily communicate with researchers, students, and other parties with interest in studying the data acquired.

9. ACKNOWLEDGMENTS

We would like to thank Dr. Harry Kolar and Dr. Anton Riabov from IBM Research, Dr. Michael Passow from IBM Systems and Technology Group in Fishkill NY, development manager Gary Anderson from IBM Poughkeepsie, and the IBM Poughkeepsie Site Location Executive Michael Desens for their tireless support in these efforts.

10. REFERENCES (TODO)

[1] APACHE SOFTWARE FOUNDATION, 2011. Apache Tomcat. http://tomcat.apache.org/

[2] BIRE, 2011. Beacon Institute of Rivers and Estuaries. http://www.bire.org/home/

[3] CANADIAN WATER OFFICE, 2011. Canadian Weather Office. http://www.wateroffice.ec.gc.ca/

[4] CUAHSI, 2010. Observational Data Model. http://his.cuahsi.org/odmdatabases.html

[5] CUAHSI, 2011. WaterML. http://his.cuahsi.org/wofws.html

[6] ECLIPSE, 2011. Eclipse Interactive Development Environment. http://www.eclipse.org/

[7] HRECOS, 2011. Hudson River Environmental Conditions Observing Systems. http://www.hrecos.org/joomla/

[8] IBM, 2011. Acoustic Doppler Current Profiler. http://spirit109.watson.ibm.com:9020

[9] IBM, 2011. DB2 Database Software. http://www-01.ibm.com/software/data/db2/

[10] IBM, 2011. IBM WebSphere Software. http://www-01.ibm.com/software/websphere/

[11] IBM, 2011. Smarter Water Management. http://www.ibm.com/smarterplanet/us/en/water_management/ideas/index.html?ca=v_water

[12] JQUERY, 2010. jQuery. http://jquery.com/

[13] MICROSOFT, 2011. Microsoft SQL. http://www.microsoft.com/sqlserver/en/us/default.aspx

[14] MYSQL, 2011. MySQL. http://www.mysql.com/

[15] ORACLE 2011. Java. http://www.java.com/

[16] POSTGRESQL, 2011. PostgreSQL http://www.postgresql.org/

[17] REDHAT, 2011. jBoss. http://www.jboss.org/

[18] TIGRIS, 2009. Subclipse. http://subclipse.tigris.org/

[19] UNIDATA, 2011. Network Common Data Form. http://www.unidata.ucar.edu/software/netcdf/

[20] US DEPT OF COMMERCE, 2011. National Oceanic and Atmospheric Administration. http://noaa.gov/

[21] US DEPT OF COMMERCE, 2011. National Weather Service. http://weather.gov/

[22] US DEPT OF INTERIOR, 2011. United States Geological Service. http://usgs.gov/

[23] WIKIPEDIA, 2011. Bar Graph. http://en.wikipedia.org/wiki/Bar_graph

[24] WIKIPEDIA, 2011. Box and Whisker Plot. http://en.wikipedia.org/wiki/Box_and_whisker_plot

[25] WIKIPEDIA, 2011. Heat Map. http://en.wikipedia.org/wiki/Heat_map

[26] WIKIPEDIA, 2011. Line Graph. http://en.wikipedia.org/wiki/Line_graph

[27] WIKIPEDIA, 2011. Scatter Plot. http://en.wikipedia.org/wiki/Scatter_plot