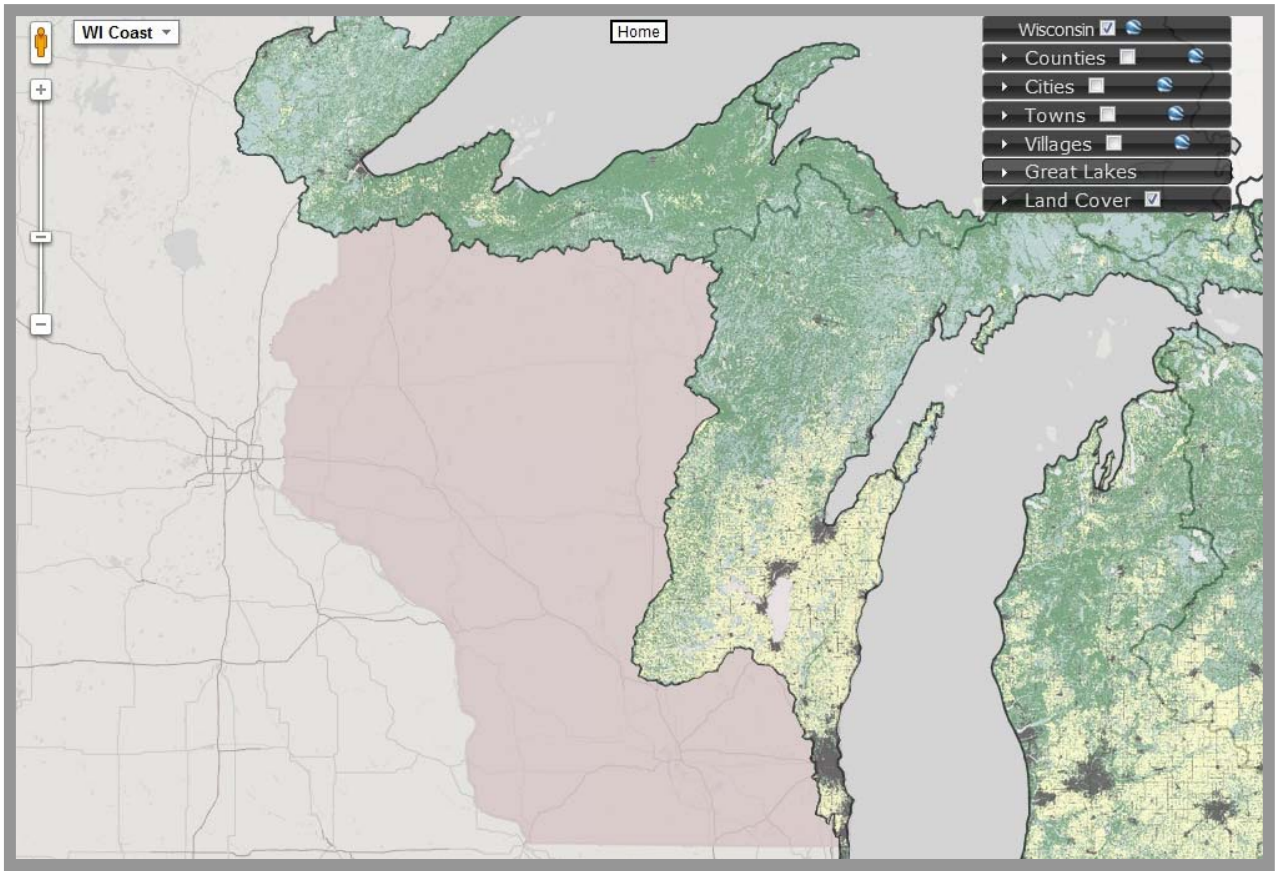
wisconsin coastal atlas white paper

UNPUBLISHED DRAFT



Tools and Best Practices for Coastal Web Maps

Carl Sack and Tim Wallace
UW Sea Grant Institute
September, 2012

Sea Grant
University of Wisconsin

Abstract
Coastal mapping has a long history, but the requirements of cartography for coastal management have changed significantly in recent years. The advent and speedy advancement of internet-based mapping technologies have made possible the collaborative creation, analysis, and sharing of geographic data on a breathtaking new scale. Emerging technologies can benefit coastal management agencies by providing tools for effective use of geographic data by managers and the public. However, they also demand an ongoing investment of work hours to train with, develop with, and maintain in the face of rapid technological change. Open-source software provides the greatest flexibility for in-house customization and the greatest interoperability, while commercial providers distribute out-of-the-box products with a high level of technical support. Essential elements for developing useful coastal web maps include compliance with technical standards set forward by the Open Geospatial Consortium (OGC), attention to cartographic conventions, and interactivity that fits the goals of coastal atlas developers This white paper does not intend to provide an exhaustive survey of the available technologies or exacting development specifications, but rather focus on key examples and how they might best be implemented as part of coastal web atlas development.

I. Introduction

The act of mapping a coast is not wildly different from the act of mapping any other geographic feature. The basic need to refer to appropriate cartographic conventions and methods holds true over land, sea, and coast: projection, generalization, color choices, etc. But the feature itself—the coast—is unique in its diversity of sub- and related features. To further complicate matters, rapidly changing technology in recent years has increased both public accessibility to and the potential complexity of coastal maps. Taken together, these unique circumstances call for developing a set of best practices coastal managers can use in developing coastal web maps. This paper is intended as a contribution to these efforts.

Coastal mapping has a long history. The British Museum houses a Babylonian clay tablet map from ~500BCE that features the land-sea interface (Barber, 2005). The geographic definition of "coastal zone" varies, but certainly includes beaches, shorelines, bluffs, intertidal areas, and portions of the littoral zone (Krishnamurthy, 2008). All of these are unique to the coast and exist as natural sites on which the human-built landscape expands. Themes in modern coastal mapping are both land-based (bluff erosion, tidal zone, storm surge and SLOSH mapping) and sea-based (hazard, reef, shipwreck, and bathymetric mapping) (Committee on FEMA Flood Maps et al., 2009).

Cartographic standards for maritime and land features in the U.S. are set by different Federal agencies. The National Oceanographic and Atmospheric Administration (NOAA) maintains *Chart No. 1*, an exhaustive guide on how and when to implement the thousands of conventional nautical symbols used on their charts (U.S. Department of Commerce and U.S. Department of Defense 2011). Similarly, the United States Geological Survey (USGS) publishes a Product Standard for its US Topo maps, the terrestrial map format replacing the deprecated topographic quadrangle series (Cooley et al., 2011). These differing standards have been created separately because of the common features found in each geographic locale, and both were created with print maps in mind. For web maps, an entirely different set of strategies for conveying information is necessary, one that draws lessons from both traditional cartography and human-computer interaction (HCI) science.

The U.S. Federal Government recognizes the importance of this interaction in its call for a National Spatial Data Infrastructure (NSDI), defined as "the technologies, policies, and people necessary to promote sharing of geospatial data throughout all levels of government, the private and non-profit sectors, and the academic community" (FGDC, 2005). Coastal web atlases in the U.S. play a part in the NSDI by distributing coastal GIS data to the public through services and making that data useful through web maps. Standards for web mapping produced by the Open Geospatial Consortium (OGC) have evolved to insure that these services can be integrated across computing platforms. While they have not yet been universally adopted, conforming to OGC web standards is as important to building a coherent spatial data infrastructure as cartographic standards are to making coherent maps. OGC standards insure the interoperability of a web map as part of the NSDI and a growing worldwide Geospatial Web[1] (Lake and Farley, 2007).

The diversity of features present in coastal mapping makes it perhaps the ideal candidate for implementing cutting-edge web mapping technologies. Web maps can be either *static* (unchanging) or *dynamic* (change in response to changes in underlying data or user controls) and either *view-only* (users can only see the map image) or *interactive* (users can manipulate the map using the mouse or map-related tools) (Kraak and Brown, 2001). Dynamic, interactive maps enable a far greater range of features, processes, and details-on-demand to be presented than in static or paper maps, while affording greater public access and participation. Interactive cartography supports visual thinking, or the generation of new insight through visual representations of the earth, and visual communication, or the distribution of ideas in visual form (Dibiase 1990). Web maps can be used as a visual interface to provide quick access to public geospatial data, on-screen retrieval of data details, and connections to other graphics, multimedia, and web pages. The magic of zooming, panning, layering, and custom symbolization allow web maps to bypass the information limitations of paper maps.

For all of its advantages, web mapping technology also poses an ethical dilemma to which coastal atlas developers should be sensitive. Those who lack access to high-speed internet—often rural or disadvantaged communities—may be unable to equally take advantage of online tools meant for public use and participation, a concept known as the *digital divide* (Crampton, 2010). Coastal resource managers should take steps to include the underserved in participatory decision-making through public workshops in the affected communities, rather than relying solely on the internet.

The purpose of this paper is to give a brief overview of web mapping technology with suggested best practices for coastal alas web maps. It will proceed in logical order from server-side (i.e., providing data and maps to the public) to client-side (i.e., making use of available data and maps to build specialized applications) technologies. This effort is closer to the first word than the last in developing a guide for coastal web mapping.[2] The intention is not to provide an exhaustive list of available technologies, but rather to illustrate how key examples have been or could be implemented for coastal mapping projects.

[1]*Geospatial Web* and *GeoWeb* are terms that refer to the nature of the World Wide Web as a new medium for creating, analyzing and sharing geographic data (Abernathy, 2011).

[2]For more information on coastal atlas uses, general design, and case studies, see Wright, D. et al. (2011), *Coastal Informatics: Web Atlas Design and Implementation*, New York: Information Science Reference.

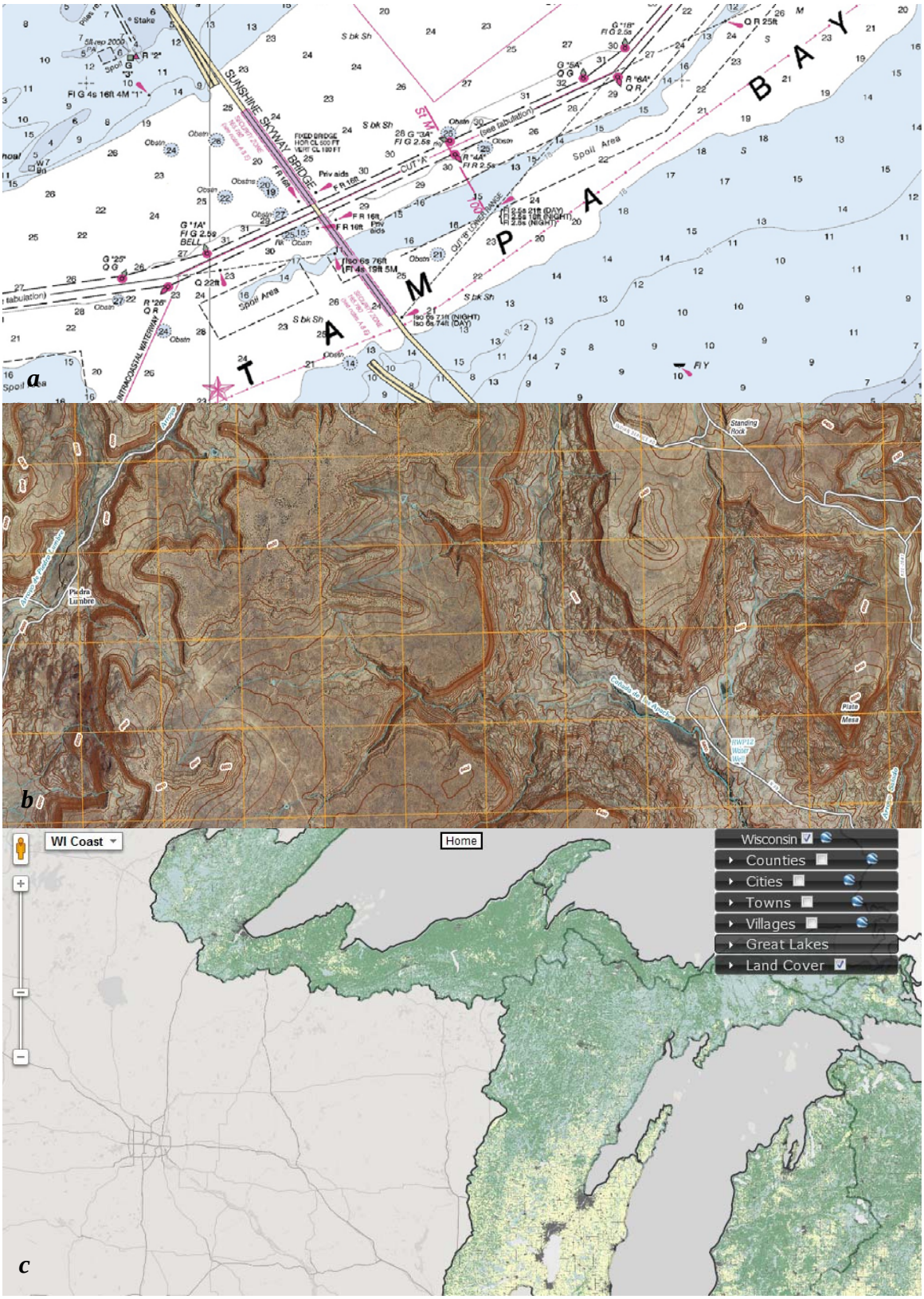


Figure 1: a) A nautical map example from NOAA Chart No. 1, b) a terrestrial map example from USGS US Topo Product Standard, and c) a web map from the Wisconsin Coastal Atlas

II. Data Storage Practices

The Internet can be defined as a global network composed of many smaller networks for transmitting data back and forth between access points. The World Wide Web is the set of hyperlinked documents that are passed back and forth across that network. The web is built on a client-server architecture, wherein one or more powerful computers, or *hosts*, contain software, a *server*, that receives requests for data by an internet user's computer, a *client*, and passes the requested data to the client (Skarlatidou, 2011). Figure 2 illustrates this relationship as it is performed during a Web Map Service (WMS) transaction.

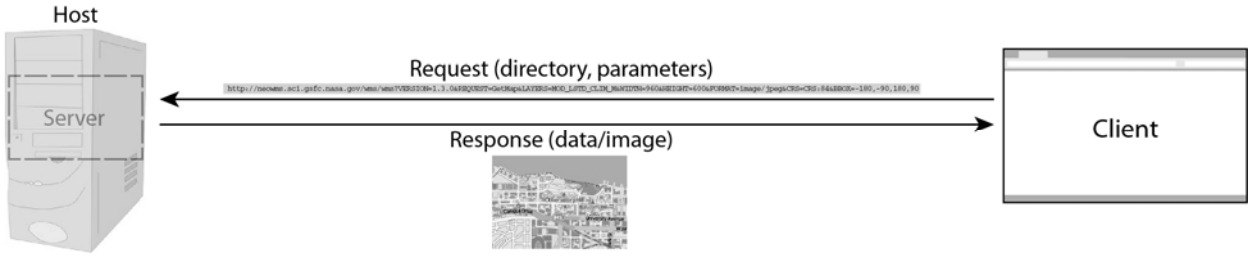


Figure 2: A diagram of server-client architecture for web mapping

It is possible to make highly useful and attractive web maps that access the many free data sources on the web today. Combining disparate data sources into what is known as a *map mashup* can provide helpful references and reveal new data patterns and trends (Crampton 2010). However, a key purpose of a coastal web atlas is to serve as a geospatial data repository, so web maps designed for a coastal atlas should not just present information in a new way, but should also enable greater data exchange (Wright et al., 2011). Therefore, the starting point for a coastal atlas web map is efficiently storing the information to be exchanged. Before any data (e.g., the Map in the illustration above) can be passed from host to client, it must be stored in a way that is easily accessible to the server.

For the purposes of coastal mapping, servers and data both may be stored on host machines owned by the coastal management agency or a cooperating organization. An increasingly popular option, however, is to host server software and data *in the cloud*, which simply means purchasing remote access to storage space and processing power within computer banks owned and operated by a dedicated provider such as Amazon or Esri. By providing vast storage capacity and high processor speeds, these services may allow for faster computing and easier setup and maintenance than local hosting. The sacrifices are local control over hardware and recurring fees (Esri 2011). For the sake of completeness, it will be assumed in this section that all data and software are being hosted on machines owned and controlled by the web map developer.

GIS data are stored in a variety of digital file formats comprising three major types: vector (points, lines, and polygons), raster (pixel-based images), and three-dimensional (TIN or model) data. Although technologies such as Google Sketchup and WebGL are increasing the use of 3D data on the web, this paper focuses on the more common use of two-dimensional vector and raster data sources to create web maps.

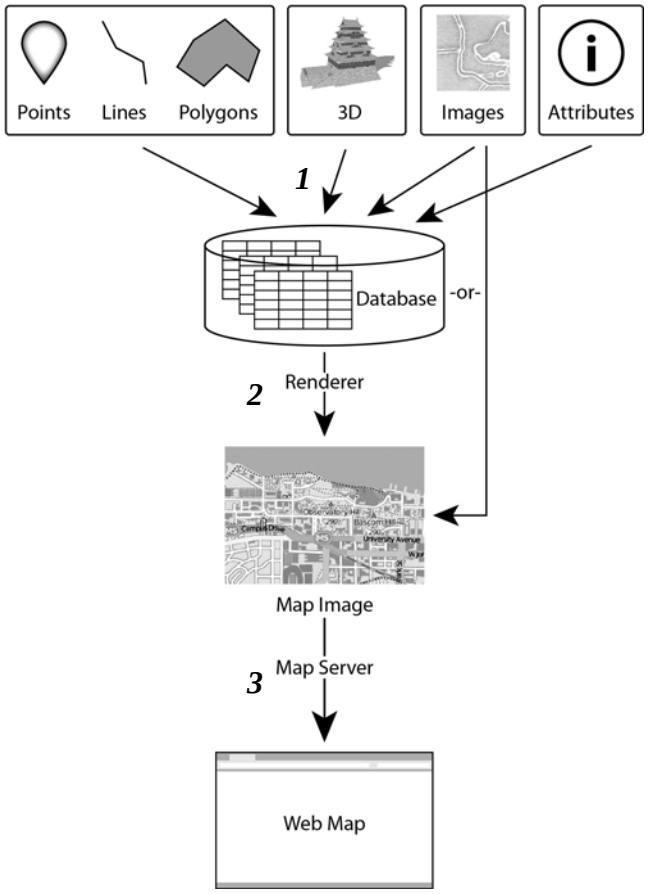


Figure 3: The steps of serving a map: 1) Data is aggregated, typically in a database; 2) a rendering engine is used to transform the data into an image, which can be done either on-the-fly or in advance with the image cached; 3) the image is served through a map server, which may also provide a rendering engine depending on the software used.

Vector data formats store sets of related features as collections of geographic coordinates and attributes. Popular file formats include CSV, Esri Shapefile, KML, GeoJSON, and SVG. Which to use depends on what use is required of the data; for instance, web applications such as GeoCommons[3] allow tabular data to be added as CSV or zipped Shapefiles, while many client-side mapping libaries utilize KML or GeoJSON formats to overlay features on a map.

Because of the tabular nature of vector data, if there are many features that must be rendered on a web map, or if individual features will need to be added, modified, or deleted from the map, it may make sense to store the features in a spatial database. This is a specialized form of relational database that can handle geometric and topological operations (Brinkhoff and Kresse, 2012). A relational database is a collection of tables that can be cross-referenced using a common identifying attribute (column). An index is used to speed up operations with the database. A database management system (DBMS) is typically used to access and write to the database.

For geographic data, the DBMS must include a way to store geographic coordinates and include functions to compute spatial relationships. PostgreSQL with the PostGIS extension is a

[3]http://geocommons.com

popular open-source DBMS for storing spatial data for the web (Davis, 2007). It has been used to store the massive volume of data produced for the crowdsource website OpenStreetMap (OSM)[4] since 2009. There are a number of other geospatially enabled DBMS, including open-source MySQL (used by OSM prior to 2009) and SQLite with Spatialite. Popular commercial products include Microsoft SQL and Oracle.

Unlike tabular vector data, raster data formats represent mosaics of typically square pixels of different color values that are blended together by the human eye to make up an image. Some database formats can store raster data, but there appears to be little computational advantage to this approach over storing image files in a standard directory structure (Davis 2007). For use in web maps, raster files may embed geographic data and metadata, such as in GeoTIFF format, or they may use a file naming and addressing convention that is translatable to geographic coordinates, such as the case of PNG or JPEG tiles in a *slippy map* (an interactive, tile-based map, e.g., Google Maps). Any time the image can be projected accurately as part of a map with a *spatial reference system* (SRS), it is said to be *georeferenced.*

A spatial reference system, or coordinate reference system (CRS), consists of a datum (ellipsoid) with a coordinate system (latitude/longitude grid). A *projected* SRS or CRS uses a set of geometric transformation rules called a *projection* to convert latitude/longitude coordinates to Cartesian points (Kresse et al., 2012). Each SRS is assigned an EPSG (European Petroleum Survey Group) code by the International Association of Oil and Gas Producers (O'Brien and Lott, 2012). All data to be mapped should be stored with the same SRS, one that is compatible with the technologies being used to serve and display the map. Many tile-based technologies require a Web Mercator (EPSG:900913/EPSG:3857) or Plate Carrée (EPSG:4326) projected SRS.

Web Mercator, also known as Spherical Mercator, is currently the most supported SRS for web mapping. However, the temptation to use a Web Mercator projection without careful consideration should be avoided. It is based on a spherical, rather than ellipsoidal, model of the Earth, and thus its accuracy is poor when measuring over long distances (Nielsen, 2008). All Mercator projections emphasize navigational direction at the cost of distorting the shape and size of poleward features, making it cartographically inappropriate for most small-scale maps (Hardy and Field, 2012). Web Mercator became popular with the rise of Google Maps and remains the dominant projection by dint of commercial success rather than ethical or scientific correctness.

The Plate Carrée projection (also known as Equirectangular or Lat/Long) simply converts latitudes and longitudes into Cartesian coordinates with equal values, resulting in less distortion in area, but more distortion in shape at higher latitudes than Mercator. It is more appropriate than Mercator for some thematic web maps (e.g., choropleth) but is not ideal for maps that need to show complex shapes such as coastlines in high latitudes accurately. Fortunately, tile-based mapping technologies are beginning to support other projection options using resources such as the Proj.4 projections library.[5]

The variety of raster and vector file types out there is dizzying, and it is beyond the purview of this paper to review them all. It is recommended to choose the file type(s) for storing the data source based on the needs of the technologies being considered for serving and using the data. Regardless of data type, an important initial task will be processing data from various sources to standardize the attribute fields, spatial reference system, and metadata (Wright et al., 2011).

[4]http://www.openstreetmap.org
[5]http://trac.osgeo.org/proj

For data processing tasks, a pair of open-source tools provided by the Open Source Geospatial Foundation (OSGeo), called GDAL and OGR, are key. GDAL includes scripts to manipulate raster files, while OGR works with vector files. As of this writing, OGR is able to read, create, and modify 25 different georeferenced file types, and GDAL can do the same with 45 raster types, plus many more they can either read or write but not both. Downloading a software package that includes the latest version of these tools is highly recommended. One basic package that works well is FWTools,[6] although this tool is now less frequently updated than larger packages such as OSGeo4W[7] (Davis 2007). Alternatively, the top commercial option for data processing is Esri's ArcGIS 10, which directly supports 65 file types, and can work with another 70 through its data interoperability extension.

Key recommendations for storing geospatial data for the web:

- Choose data storage formats based on the needs of the technologies that are used to meet the objectives of the web mapping project.
- If serving maps with large quantities of vector features, or sets of features that will need to be regularly modified, store data in a relational database using a DBMS. PostgreSQL with the PostGIS extension is recommended.
- Store raster files in a logical file directory structure.
- Standardize metadata and geographic projection when storing data. Use a projection that is as cartographically appropriate as possible given the technologies being used.
- Include GDAL and OGR in the host's software stack.

III. Map Servers and Services

Data are delivered from the data source to clients by a web (or HTTP) server. Although host machines themselves are sometimes referred to as "servers," the technical definition of a server is software that sits on a host computer, relays requests from clients, and returns the requested data to the client. The most popular web servers are open-source Apache and commercial Microsoft IIS. Georeferenced maps are unlike other information delivered over the web because they include spatial relationships that must be maintained, so it is easiest to serve them using specialized software that extends a web server, called a *map server.* Most map servers create *services* that conform to Open Geospatial Consortium (OGC) standards.

The OGC is an international community of universities, government agencies, and private companies that creates specifications for geographic data use through a consensus process. Since the mid-1990s, the OGC has set and updated several standards for serving geographic data over the web. Their goal is to maximize interoperability, such that an agency using one piece of software can tap into services provided by another agency using a completely different software suite, or even the services of an agency from an entirely different country (Buehler and Reed, 2011). Interoperability allows web technology to build upon itself, advancing in stages rather than each new technology having to reinvent the wheel.

OGC standards-compliant services primarily use a RESTful (REpresentational State Transfer) interface, which relies on a Uniform Resource Locator (URL) containing a set of valid parameters to pass a request. Some also specify a SOAP (Simple Object Access Protocol)

[6]http://fwtools.maptools.org;
[7]http://en.wikipedia.org/wiki/FWTools

interface, which relies strictly on XML (eXtensible Markup Language) messages passed back and forth by the server. Most client-side libraries use a RESTful interface for requests. The server responds to the request according to the values provided for each parameter. For example, a Web Map Service (WMS) request may look like this:

```
http://neowms.sci.gsfc.nasa.gov/wms/wms?VERSION=1.3.0&REQUEST=GetMap&
LAYERS=MOD_LSTD_CLIM_M&WIDTH=960&HEIGHT=600&FORMAT=image/jpeg&CRS=CRS
:84&BBOX=-180,-90,180,90
```

The portion of the request above to the left of the "?" represents the server address and directory that holds the stored images, while the parameters required for the request to work are to the right of the "?", shown in all caps (although they are case-insensitive). The request returns the image in Figure 4.

WMS is the most well-developed and well-used OGC web service standard. The standard includes a GetCapabilities request, which returns metadata about the available maps, and a GetMap request, which returns a map image. The client can request any specific portion of the map by applying different bounding box coordinates in the request. The WMS may or may not also support a GetFeatureInfo request, which returns attribute data about individual features (de La Beaujardiere, 2005). Its customizability makes WMS ideal for serving dynamic maps.

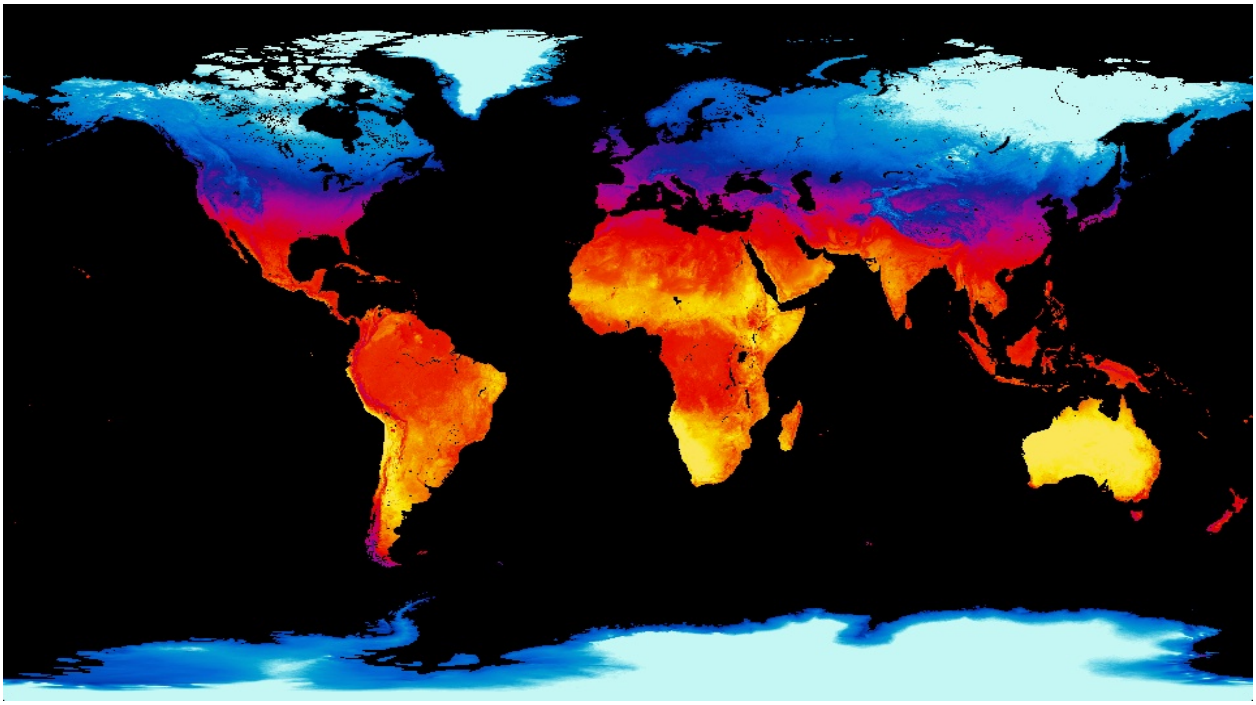


Figure 4: A WMS map from NASA Earth Observatory

Geospatial data is not inherently visible, so the map server must apply a *stylesheet* to generate an image that is returned by the GetMap request. This image is not stored, but is rendered dynamically for each request; hence the custom nature of WMS maps. The standard stylesheet is an XML document called a Styled Layer Descriptor (SLD) that includes rules designating certain colors, line weights, effects, etc. for specific feature types. The OGC

publishes a Symbology Encoding standard that defines the XML syntax used in the SLD. Some map servers support additional methods for styling; MapServer, for example, uses its *.map* file format to style data, but also supports SLD stylesheets, while GeoServer includes an alternative CSS style editor.[8]

WMS alone is functionally limited by the volume of data in each requested map image. Even at high bandwidths, large map images may take a long time to render and load, detracting from the user experience.[9] Loading time decreases if map images are served as a series of smaller (typically 256x256 pixel), edge-matching tiles, with only those tiles within the user's viewing frame sent to the client. Tile sets can provide the ability to zoom to greater levels of detail (data resolutions) at larger scales—termed *semantic zoom*—without resulting in unmanageable file sizes. The tiling approach, combined with AJAX script in the web page, also allows for greater interactivity, with the user able to click and drag to pan the map and quickly zoom in and out.

There are multiple ways to deliver tiles (Figure 5). Some client-side JavaScript libraries (e.g., OpenLayers) *tessellate* WMS maps; that is, they render tiles on the fly by sending multiple WMS requests with different bounding boxes that correspond to each tile's dimensions and edge-match the tiles. This approach is not the fastest, and can generate labeling problems if map labels are placed by the server according to the requested bounding box (Davis 2007). A faster and more reliable approach is to pre-render the tiles using a server-side script and store them in a specialized directory structure. The downside to this approach is less customizability, since the individual tiles are static raster images once they are rendered and stored.

Tiles are typically stored in PNG or JPEG format. The URL for each tile requested by a client's browser does not use parameters, but points to a set of nested directories on the server. The parent directory is named according to a fixed zoom level (*z*), with each child directory representing a column of tiles (*x*), and each image file within that column a separate tile numbered by vertical order (*y*). The URL for a tile is thus represented as */tiles/z/x/y.png* (Switch2OSM, 2012).

Confusingly, there are multiple specifications for tile request formats and addressing schemes. Google was the first user of map tiling and created their own specification for serving tiles, followed by Microsoft, which uses a different schema. The first attempt at a universal specification was the Tile Map Service (TMS) standard proposed by the Open Source Geospatial Foundation (OSGeo), which is used by some map server software (e.g., TileCache). However, TMS is problematic because it is not interoperable with Google's schema and never successfully challenged the dominance achieved by Google.[10] In 2010, the OGC approved the Web Map Tile Service (WMTS) standard, which builds upon the Google schema, but provides enhanced support for tile metadata and multiple geographic projections (whereas Google Maps natively supports only Web Mercator) (Masó et al., 2010). To improve the user experience, it is recommended to serve all static data as a tile layer that can be used in a slippy map, in compliance with the WMTS standard.

---

[8]Software used to pre-render tiles may use other styling methods, such as Mapnik XML and the recently-developed CartoCSS language (Käfer 2011).

[9]Human-computer interaction research indicates that response time must be no more than 0.1 second for a user to feel that the system is reacting instantaneously, no more than 1.0 second for the user to notice a delay but maintain an uninterrupted flow of thought, and no more than 10 seconds to keep the user's attention focused on the dialogue with the system. Feedback to the user is recommended if response time is greater than 1 second or may vary (Nielsen 1993).

[10]A key problem is that Google's order of addressing tiles along the y-axis is reversed from that of TMS.

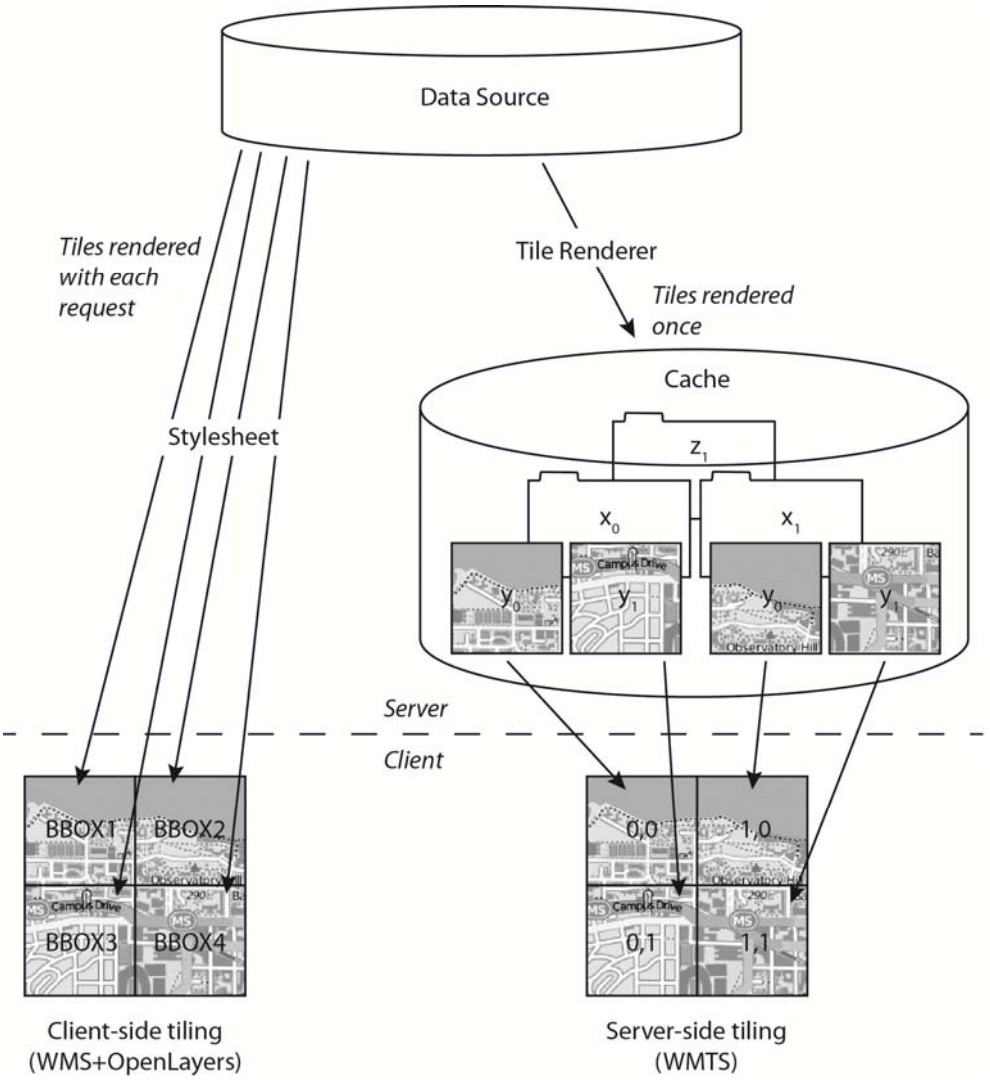


Figure 5: Two different map tiling strategies. The first relies on separate WMS requests to generate each map tile on-the-fly, while the second requests pre-rendered tiles stored in a server-side cache using the WMTS standard.

Tiles are served by a specialized map server extension called a *tile server*. Tile servers may either work with a rendering script to generate tiles on the fly as they are requested by clients, or create a server-side cache to store the tiles for faster delivery. An example of the first approach is the Mod_tile script, which is used alongside the Mapnik tile renderer by OpenStreetMap (OSM) to generate new tiles each time they are requested. This approach serves OSM well for two reasons: because their geographic data is routinely updated, necessitating a dynamic approach, and because they do not have resources to cache the huge volume of raster imagery required to cover the entire planet (OpenStreetMap 2012).[11] Other tile servers cache tiles, either all at once (e.g., TileStache) or on the first request for each tile (e.g., GeoWebCache), and simply send the image from the cache on each subsequent request.

[11]The imagery tiles used by Google Earth, for example, take up 70 terabytes (71,680 GB) (Chitu, 2006).

One increasingly popular, user-friendly tile rendering technology is Tilemill, produced by Mapbox.[12] This desktop application is designed to style and output full tilesets packaged in the company's MBTiles format, which can be served through the MapBox cloud hosting service or locally through TileStache[13], a free open-source tile server. Example applications include the University of Wisconsin Campus Map[14] and web maps by the Chicago Tribune, which has also published some tutorials (e.g., Boyer 2011). Tilemill can produce very aestetically-pleasing thematic and reference maps with limited interactivity using its Carto styling language, a CSS wrapper for Mapnik XML. However, it currently only supports the Web Mercator projection and does not comply with the WMTS or other standards, so is not ideal for coastal atlas use.

The services discussed so far are designed to serve raster images. However, many coastal atlas features start out as vector features, and will no longer be useful for spatial analysis if transformed to raster. Serving vectors has the advantage of allowing individual features not only to be viewed, but queried and modified. The downside to serving vectors is the same as untiled WMS: large data volumes mean slow loading, hampering performance. Some client-side libraries (e.g., OpenLayers and Leaflet) will automatically tile client-generated vector objects, but may need an added workaround to tile vectors from a service.

The OGC Web Feature Service (WFS) standard provides a method of serving and querying individual features stored in a server-side database (Vretanos, 2010). The standard uses scripts written in XML-based Geography Markup Language (GML) to pass feature information back and forth between server and client. *Transactional WFS* (WFS-T) is part of the WFS standard that allows the client to submit feature modifications to the server database as well as create and delete features across the internet (Davis, 2007). If an objective is to allow individual vector features to be queried and/or modified, compliance with WFS is recommended.

Regardless of what type of service is provided, using a server package that supports at least the WMS, WMTS, and WFS standards is recommended to maximize extensibility. There are many commercial and open-source map servers available. Two strong open-source options are MapServer and GeoServer. MapServer dates back to 1994 and has by far the broadest range of available configurations, while GeoServer is easy to set up and includes a user-friendly graphic interface. ArcGIS for Server is a popular commercial option which supports the WMS and WFS standards.

A final OGC service standard worth mentioning is Catalog Service for the Web (CSW), which provides the ability to publish searchable collections of metadata for geospatial data and services. It specifies client request operations, SQL-based query language grammar, a set of core attributes to be included in metadata, and a common record format to be returned by the server (Nebert et al., 2007). Coastal atlases that warehouse a collection of data and services should provide a CSW in compliance with Federal Geographic Data Committee (FGDC) metadata standards. Most existing coastal atlases implement a CSW through a data search function or catalog application (Wright et al., 2011). The Wisconsin Coastal Atlas, for example, uses the free GeoNetwork Open Source,[15] which works with GeoServer. Other commercial products include Intergraph's Geospatial Portal[16] and Esri's Geoportal Server.[17]

[12]http://mapbox.com/tilemill. MapBox is a project of technology consulting firm Development Seed.
[13]http://tilestache.org
[14]http://map.wisc.edu (forthcoming in September, 2012)
[15]http://geonetwork-opensource.org
[16]http://www.intergraph.com/sgi/products/default.aspx
[17]http://www.esri.com/software/arcgis/geoportal

Key recommendations for creating map services:

- Use a map server that supports all OCG standards applicable to the project objectives, preferably WMS, WMTS, and WFS at a minimum. GeoServer is recommended because of its completeness and ease of use.
- If developing a slippy map, render and serve all data that does not require frequent update as cached tiles, in compliance with the WMTS standard. GeoWebCache, which extends GeoServer, is a recommended tile server.
- If serving a single map image or data that requires frequent update but not feature queries, provide a standards-compliant WMS.
- If serving data with individual features that may be queried or changed by the client, provide a standards-compliant WFS.
- If providing multiple data sources and/or services to the public, also provide a CSW with FGDC-compliant metadata for each service. GeoNetwork Open Source, which extends GeoServer, is a recommended tool.

IV. Client-Side Web Map Development

Displaying map services in client browsers requires script to make requests, receive the data from the server, and position it on the web page. HTML (*HyperText Markup Language*) is used to add elements to the web page, and CSS (*Cascading Style Sheets)* is used to position and style elements on the page. Interactivity with the web page is handled using JavaScript.

There are two basic architectures for interactive web maps: *thick client* and *thin client*. (Figure 6). *Thick client* applications run programs within a client's browser. The program is sent as a package from the server to the client and executed through a browser plug-in or HTML5 element. Types of plug-ins used to display web maps include Flash Viewer (from Adobe), Silverlight (from Microsoft), Java Virtual Machine (from Oracle), and Processing (open-source), while newer HTML5 elements of use include Canvas and SVG. These technologies provide robust interactivity, a smooth look and feel, and fast reaction time once loaded, and do not require a stable, high-bandwidth internet connection to run. Their key disadvantage is interoperability, as individual users must have the right browser plug-in installed and enabled, and/or a new enough browser, to take advantage of applications that use them. Examples of thick client web maps include the Capital Regional District Regional Community Atlas,[18] built in Silverlight, and Forbes' American Migration map, built first in Flash[19] and later migrated to SVG.[20]

[18]http://viewer.crdatlas.ca/public#/Home (Vancouver Island, B.C.)

[19]http://www.forbes.com/2010/06/04/migration-moving-wealthy-interactive-counties-map.html

[20]http://www.forbes.com/special-report/2011/migration.html. For many years, Adobe Flash was preferred by cartographers due to its robust development environment and high interactivity and extensibility of its underlying language, ActionScript. However, open-source web technologies continue to improve, while most commercial web mapping services and most mobile platforms have discontinued support for Flash, making its continued use untenable. The Forbes maps provide a unique side-by-side comparison demonstrating this trend.

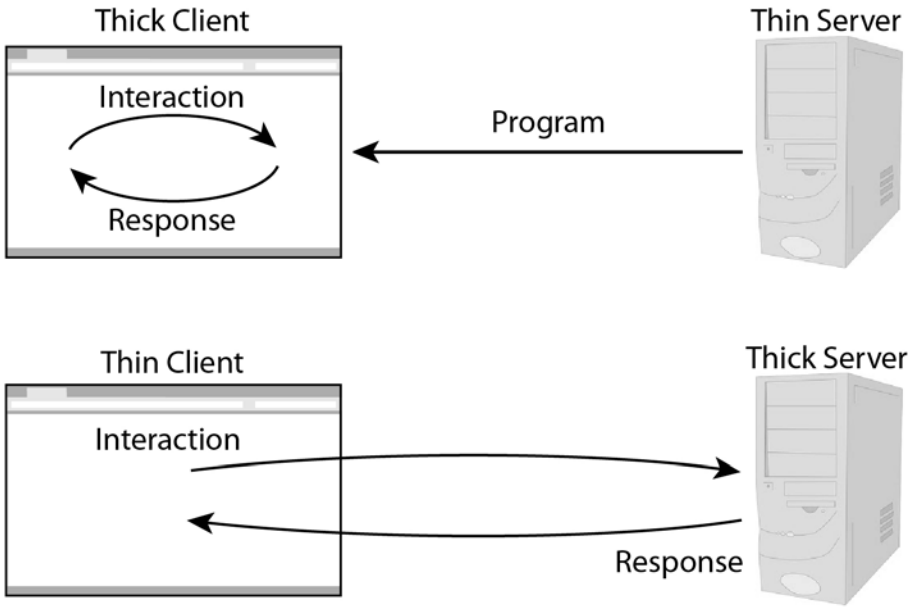


Figure 6: A simplified diagram of thick vs. thin client architecture.

Most thick-client applications rely on plug-ins, but a small number of JavaScript code libraries create web maps using plug-in-free SVG (*scalable vector graphics)* format. SVG allows for robust styling of data and interaction. Kartograph[21] is a thick-client library that includes a Python script to transform data stored as Esri shapefiles into SVG form, and a JavaScript library to display the pure SVG map in the client's browser. A similar approach is used by the Raphaël library,[22] which can transform GeoJSON objects into SVG. This process results in a static map that can be beautifully styled and highly interactive. Polymaps[23] is a base library that mixes thick- and thin-client approaches. It can load vector overlays as tiled SVGs and layer multiple services, such as in Stamen Design's visually-stunning Prettymaps tile services.[24] D3[25] is a useful SVG library for creating data visualizations that can be dynamically linked to the map, such as in Nelson Minar's "Wind History" map.[26] The key drawbacks of SVG are a hefty loading penalty that increases with the number and complexity of features and a lack of support for SVG in older browsers (including Internet Explorer prior to v.9).

With increasing bandwidth available to internet consumers and the continued expansion of interactivity in JavaScript libraries, *thin client* web maps are increasingly the norm. Instead of executing a program on the client, most or all of the business logic sits on the host, with a server handling requests and returning the results of the requested operation. Client requests are handled by a mapping library, usually written in JavaScript and read by the client's browser. This approach does away with the need for browser plug-ins for fast interactivity through the use of AJAX (*asynchronous JavaScript and XML*), which handles data exchange between client and server without causing the background web page to reload. Because JavaScript is an open-source

[21]http://kartograph.org
[22]http://raphaeljs.com
[23]http://polymaps.org
[24]http://prettymaps.stamen.com
[25]http://d3js.org
[26]http://windhistory.com

language, most libraries have high levels of interoperability and extensibility; they can be combined and built on, providing a great deal of creative license to even novice developers. The downside is the reliance on a high-speed internet connection between client and server to achieve adequate response times for user satisfaction. This especially poses a problem for underserved households, particularly in rural areas with less access to high-speed internet, and mobile devices that rely on cellular communication networks (Peterson, 2011).

In the case of complex Web GIS systems, whether to use a thin or thick client architecture for spatial analysis functions may depend on whether the supported operations are generally performed faster by the client's processor (CPU) or by a server passing data back and forth over a high-bandwidth connection. Performance testing indicates that spatial analysis operations should be done on the server when computation takes less time than communication or if the output data is less than the input data. Otherwise, operations should be done on the client through a plugin such as MapGuide[27] (Vatsavai et al., 2012).

Thin-client mapping libraries include commercial *application programming interfaces* (APIs) and open-source *base libraries*. APIs are code libraries provided by commercial outfits such as Google Maps, Bing Maps, and Esri ArcGIS Online to interface with their services; they are controlled by the provider and typically charge fees for their use. Base libraries interact directly with the server. They range from very light-weight, with only the most basic methods for displaying a slippy map (e.g., ModestMaps[28]), to very robust, with methods for adding multiple types of services, vector overlay, and complex user interactions (e.g., OpenLayers[29]). Many functions of both commercial and open-source libraries are interchangeable, while each library may do some things better than others. For example, OpenLayers may be used to layer WMS and Google basemaps, but only the Google Maps API provides a method for client-side styling of Google's Roads basemap.

In general, open-source mapping libraries are more likely to provide native methods for accessing OGC-compliant services than commercial APIs. Google Maps API v3[30] is currently the most popular mapping library due to its strong documentation, ease of use, and momentum as the largest mapping service, but it requires a difficult workaround to display WMS maps. OpenLayers, by contrast, natively supports all OGC services, but uses very strict syntax and can be frustrating for non-expert developers. A strong intermediate choice is Cloudmade's Leaflet,[31] a new and growing open-source library with good documentation that can access multiple commercial and OGC services, although not as many as OpenLayers. Esri provides an ArcGIS API that is designed primarily to interface with ArcGIS REST services, but also natively supports the WMS standard. [32] Through ArcGIS Online, Esri hosts several useful public tilesets composed from a mixture of government and private data.[33] At the time of this writing, use of the ArcGIS Online web application requires a fee, but API access of the tile services is free up to a very high transaction limit for commercial services, and with no limit for government, educational, and non-governmental organizations.[34]

[27]http://mapguide.osgeo.org
[28]http://modestmaps.com
[29]http://openlayers.org
[30]https://developers.google.com/maps
[31]http://leaflet.cloudmade.com
[32]http://help.arcgis.com/en/webapi/javascript/arcgis
[33]http://server.arcgisonline.com/ArcGIS/rest/services
[34]http://www.esri.com/software/arcgis/arcgis-online-map-and-geoservices/common-questions

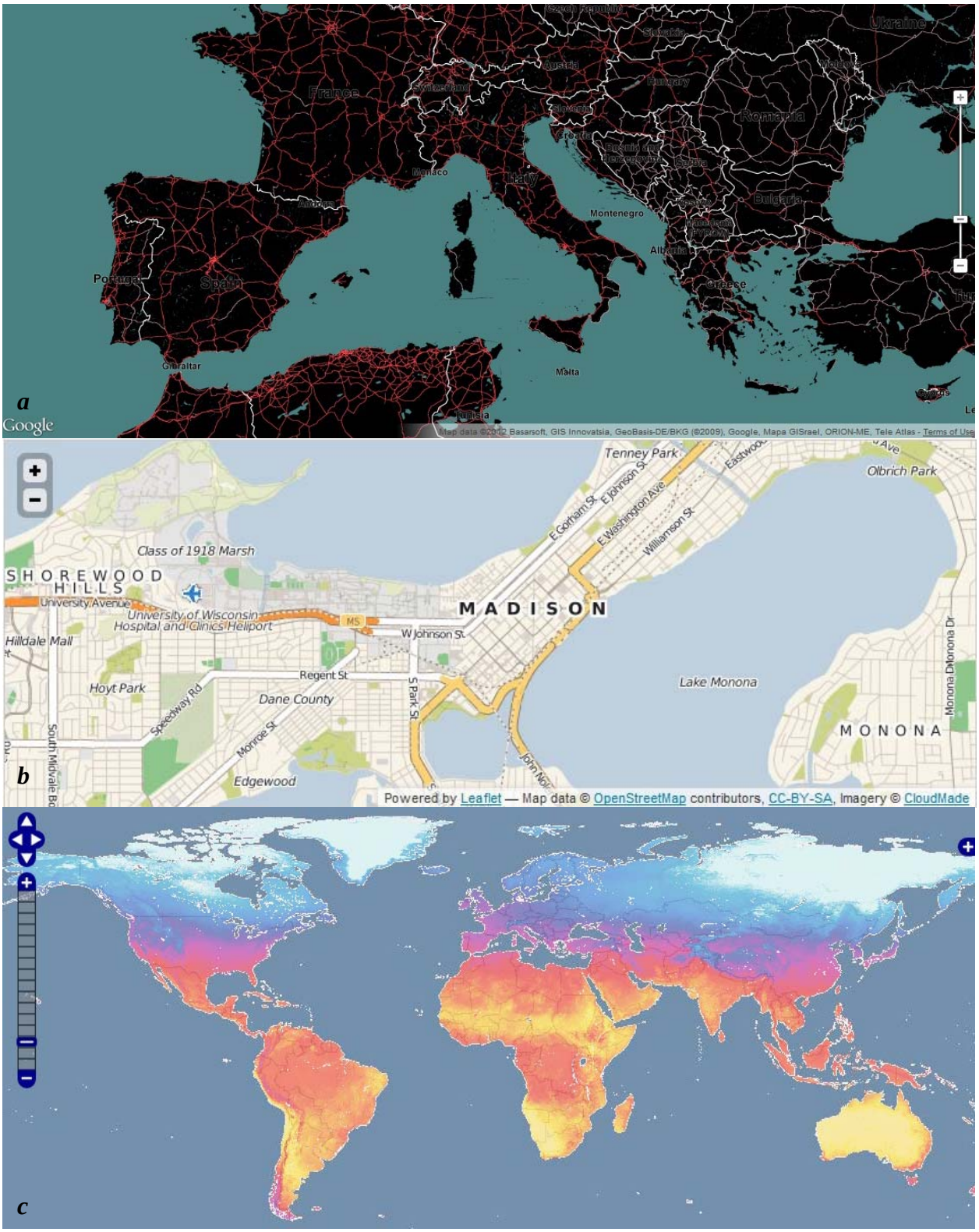


Figure 7: A visual comparison between a) Google API v3 with a styled Google basemap, b) Leaflet with an OSM basemap styled by CloudMade, and c) OpenLayers with the WMS layer from Figure 4.

The mapping library enables not only the display of web maps, but interactions as well. *Interaction operators* (types of interactions) provided by web mapping applications may include reexpression (changing the visual representation of data on the map), sequencing/animation (creating an ordered set of related maps), overlay (adding or removing data layers), resymbolization (changing colors, sizes, etc. of symbols on the map), zooming (changing the map scale and/or resolution), panning (recentering of the map), reprojection (changing the geographic projection/spatial reference system), searching (finding a particular feature on the map), filtering (selecting a subset of map features by user-defined conditions), retrieving (getting specific details about a feature), arranging (moving the map or visualizations related/linked to it), calculation (performing a mathematical operation using the map), importing (adding data to the map), exporting (extracting data from the map), saving (storing the map on the web for future use), editing (changing the feature data displayed on the map), and annotation (adding drawings or notes on the map) (Roth, 2011).

Different libraries natively support different sets of interactions, and none support all of them, although they may be supplemented with additional JavaScript to perform a needed operation. Most support some form of overlay using multiple services and/or vector feature sets (including markers, lines, and polygons) generated by the client and stored as KML or GeoJSON objects. Information retrieval is usually supported through mouse events: dynamic labels (also called *tooltips*) activate when the user hovers over a feature, pop-up info windows activate on mouse click, and other events may be triggered as defined by the developer. Zooming and panning tools and support for click-and-drag panning (the origin of the term 'slippy map') are almost universally included. Support for multiple geographic projections is increasing, although the projection in the client library should match that of the data source, otherwise display problems may arise. Searching and filter queries are sometimes supported, but may require an additional library or API. Other interactions may come with add-ons such as GeoExt, which extends OpenLayers with support for arranging, exporting, printing, and calculation.[35]

The array of library options can be daunting to choose from without knowing which technologies will integrate best while providing the necessary tools and interactions. If the project does not require an innovative design, using a pre-packaged framework may be a good option. There are several out-of-the-box options, both open-source and commercial, which include various configurations of database, map server, and/or client development tools. OpenGeo Suite[36] is a strong package that includes PostgreSQL/PostGIS database, GeoServer, GeoWebCache, OpenLayers, GeoExt, and user-friendly graphic user interfaces. OSGeo4W[37] includes about 150 open-source GIS and web mapping software modules, including GeoServer, but is missing a DBMS. GeoMoose[38] provides an easy-to-set-up package for data presentation built on MapServer, OpenLayers, and Dojo. The Wisconsin State Cartographer's Office's award-winning WHAIFinder application, for example, was built using GeoMoose. Geomajas[39] is very similar, but built entirely in Java. CartoDB[40] is a very user-friendly, robust, and entirely cloud-based framework. It includes PostGIS, Carto map styling, a built-in mapping application, and APIs for accessing data and services from it in other applications.

[35]http://geoext.org
[36]http://opengeo.org
[37]https://trac.osgeo.org/osgeo4w
[38]http://geomoose.org
[39]http://www.geomajas.org
[40]http://cartodb.com

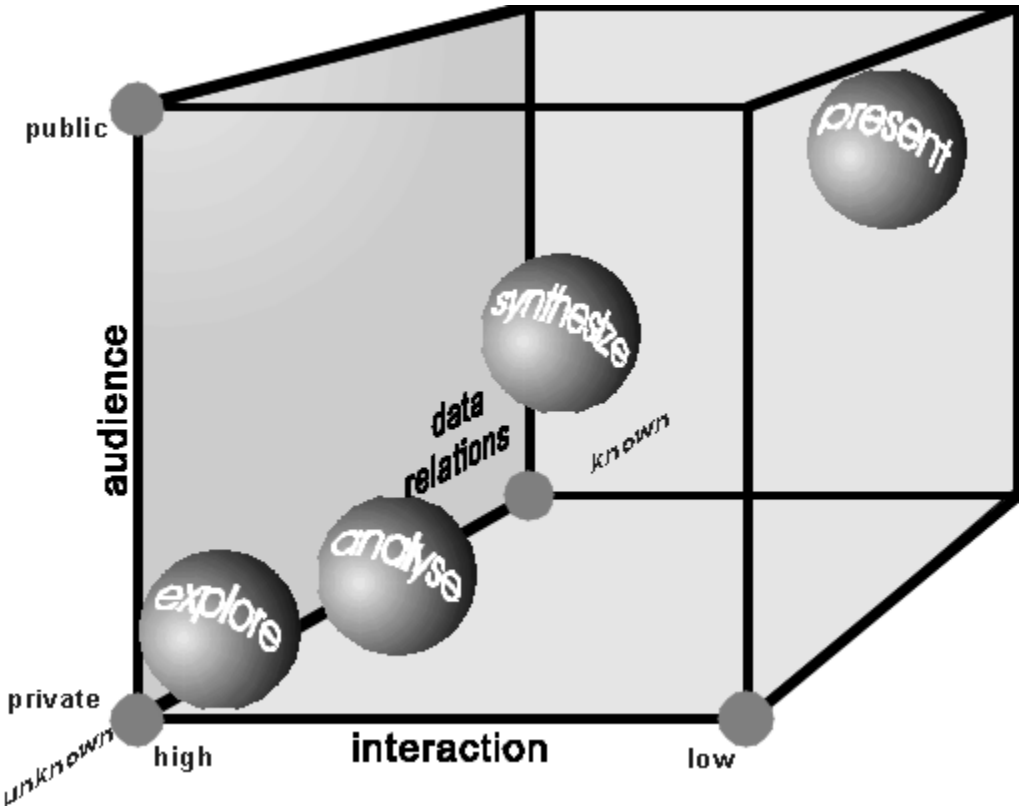


Figure 8: A Cartography[3] diagram showing the sequence of goals that can be facilitated by web maps: 1) data exploration, 2) data analysis, 3) information synthesis, 4) information presentation (from MacEachren and Kraak, 1997).

The library or framework should be chosen largely based on whether it supports the desired human-map interactions. Which interactions, and how much interactivity in general, to include will depend on the purpose of the web map. Coastal atlas web maps may meet a range of goals that constitute stages of scientific visualization, from *data exploration*, to *analysis*, to *synthesis*, to *presentation* (DiBiase, 1990). These map uses move from private visual thinking to public communication of ideas, from revealing unknown relationships to presenting known information, and from high to lower levels of human-map interaction (Figure 8; MacEachren and Kraak, 1997).

Web maps that support *data exploration* fall into the category of *geovisualization* and should include the most interactivity. An example is Trulia's crime maps,[41] which allow users to explore crime data for a given city through two types of maps (point map and heatmap), toggleable layers categorized by crime, a linked bar chart, dynamic labels, and pop-up info windows. Individual crimes are tracked and served dynamically as a point layer, and points are aggregated into point clusters at lower zoom levels. A collapsible panel controls the layers and shows the top five most dangerous intersections in an urban area, linked to dynamic labels on the map, and clickable to recenter the map on the selected intersection and bring up its info window. Other geovisualization applications may include any or all of the interaction operators listed previously.

[41] http://www.trulia.com/crime

Web maps for *data analysis* should also include a great deal of interactivity, but focus more on filtering/querying, spatial calculations, and information retrieval—i.e., a Web GIS system. A robust example is USGS' The National Map Viewer,[42] which provides a large number of services from many government sources as optional data layers, as well as the ability to add data to the map, search for data using CSWs, search for features, select by distance (buffer), create multiple selections, filter by SQL query, draw vectors, and annotate the map. The site uses Esri's ArcGIS API[43] and the Dojo Toolkit library.[44] Most coastal atlases include a Web GIS system generally similar to The National Map, but with fewer available services and analysis tools.

Web maps designed for *information synthesis* should include tools and interactions that allow users to add or create new information on the map. The web environment lends itself to collaboration between multiple users in disparate locations and times, what might be termed *online participatory mapping* (OPM) (MacEachren, 2005; Sack, 2012). In a coastal atlas, the user group may be resource experts, decision-makers, and/or the general public (Wright et al., 2011). Increasingly, stakeholder participation is being sought for coastal resource management decision-making, necessitating systems that are usable by the public to collect, depict, and interpret *crowdsourced* (publicly-contributed) information helpful to this decision-making process (NOAA 2009). This need contrasts with the present state of coastal atlases, which generally support only one-way transfer of information from state agencies to users. *Volunteered geographic information* (VGI), or crowdsourced spatial data, could be a legitimate and useful data source for coastal atlases. Examples of VGI deployment include crisis mapping applications such as the "Faster Than Disaster" map of damage from Hurricane Isaac,[45] and public interest web maps such as OpenStreetMap[46] and Wikimapia[47] (Goodchild, 2007).

Web maps for the purpose of *information presentation* may have a relatively low level of interactivity. The interactions that are provided should focus on tools for viewing the map in different ways (zooming, panning, and optionally overlay) and retrieving information from the map. Most basic mapping libraries natively support these interactions. Examples include the Washington State Coastal Atlas Shoreline Photos viewer,[48] which uses Esri's ArcGIS API, and the Wisconsin Coastal Atlas Coastal Overview,[49] which uses Google Maps API.

Key recommendations for client-side web map development:

- Use a JavaScript web mapping library or API that supports the services and interactions required to meet project objectives and is well-documented.
- If the primary purpose of the application is spatial data exploration, provide the most possible interactivity, including layer controls, linked data visualizations, and multiple methods for retrieving information. GeoExt and D3 are useful libraries.
- If the primary purpose of the application is spatial data analysis, use a Web GIS framework that includes the desired spatial analysis tools. GeoMoose is recommended.

[42]http://viewer.nationalmap.gov/viewer
[43]http://www.esri.com/software/arcgis/web-mapping/javascript
[44]http://dojotoolkit.org
[45]http://isaac.fasterthandisaster.org. This application uses Ushahidi, a framework specialized for crisis mapping. It may be short-lived, but other active examples are available through the company's website, http://ushahidi.com.
[46]http://www.openstreetmap.org
[47]http://wikimapia.org
[48]https://fortress.wa.gov/ecy/coastalatlas/tools/ShorePhotos.aspx
[49]http://www.wicoastalatlas.net/Default.aspx?tabid=79

Conduct analysis operations on the server when computation takes less time than communication or if the output data is less than the input data; otherwise conduct processing on the client.

- If the primary purpose of the application is synthesis of user-contributed data, use a library or framework that includes drawing and data input tools. Leaflet is recommended for custom applications, and GeoMoose or Ushihidi Crowdmap is recommended as an out-of-the-box platform.
- If the primary purpose of the application is presentation of existing information, use an open-source JavaScript library that includes strong documentation and support for the types of services and interactions needed. Leaflet is a recommended library, and GeoMoose is a recommended framework.

V. Conclusion

Web mapping technologies provide a great deal of opportunity for increasing access to agency data by coastal resource managers and the general public. Web maps support visual thinking and communication, simplifying the interface by which users can access data resources and allowing them to make immediate sense of the data. Coastal atlases are taking advantage of some web mapping tools, but could enhance their offerings by incorporating more of the visualization goals in Figure 8 and including support for public participation in information creation. While a growing range of technologies is available to meet these goals, care should be taken to prioritize standards-compliance and good cartographic design. Web map developers should create a conceptual design for each new application to identify what needs it will meet, what services it will provide, and what technologies are most suited to support the project goals. By expanding their web map offerings, coastal atlases can better serve coastal resource decision-makers and play an important role in building and enhancing the GeoWeb.

Sources

Abernathy, D. (2011). “Teaching the Geoweb: Interdisciplinary Undergraduate Research in Wireless Sensor Networks, Web Mapping, and Geospatial Data Management.” *Journal of Geography*, 110(1), 27-31.

Barber, P. E. D. T. (2005). *The Map Book*. St Martins Press.

Boyer, Brian (2011). “Making maps, part 3: Make it pretty with TileMill.” *News Apps Blog.* Chicago Tribune. Accessed August 23, 2012. <http://blog.apps.chicagotribune.com/2011/03/08/making-maps-3>

Brinkhoff, Thomas and Wolfgang Kresse (2012). “Databases.” In Kresse, Wolfgang and David M. Danko (eds.), *Springer Handbook of Geographic Information.* Springer. pp. 61-122.

Buehler, Gred and Carl Reed. 2011. OGC Orientation Slides. Bloomington, IN: Open Geospatial Consortium.

Chitu, Alex (2006). “How Much Data Does Google Store?” *Google Operating System: Unofficial News and Tips About Google blog.* September 10. Accessed August 22, 2012. <http://googlesystem.blogspot.com/2006/09/how-much-data-does-google-store.html>

Committee on FEMA Flood Maps, Board on Earth Sciences and Resources, Mapping Science Committee, National Research Council (2009). *Mapping the Zone: Improving Flood Map Accuracy*. Washington, DC: The National Academies Press.

Cooley, M.J., Davis, L.R., Fishburn, K.A., Lestinsky, Helmut, and Moore, L.R., 2011, US Topo Product Standard: U.S. Geological Survey Techniques and Methods 11–B2, 17 p. pamphlet, 1 sheet, scale 1:24,000.

Crampton, Jeremy W. (2010). *Mapping: A Critical Introduction to Cartography and GIS.* West Sussex, UK: Wiley-Blackwell.

Davis, Scott (2007). *GIS for Web Developers: Adding* Where *to Your Web Applications.* Raleigh, NC: The Pragmatic Bookshelf.

de La Beaujardiere, J. (ed.) 2005. Web Map Service. OGC 04-024, version 1.3. Open Geospatial Consortium Inc.

Dibiase, David (1990). “Visualization in the Earth Sciences.” *Earth and Mineral Sciences* v.59 n.2.

2011. *Estimating the Cost of a GIS in the Amazon Cloud.* Redlands, CAs: Esri.

2005. “The National Spatial Data Infrastructure” fact sheet. Reston, VA: Federal Geographic Data Committee, US Geological Survey.

Goodchild, Michael F. (2007). “Citizens As Sensors: the World of Volunteered Geography.” *GeoJournal* v.69 pp. 211-221.

Hardy, Paul and Kenneth Field (2012). “Portrayal and Cartography.” In Kresse, Wolfgang and David M. Danko (eds.), *Springer Handbook of Geographic Information.* Springer. pp. 323-358.

Käfer, Konstantin (2011). “Introducing Carto: A CSS-like Map Styling Language.” *MapBox blog.* Washington, D.C.: Development Seed.

Kraak, Menno-Jan and Allan Brown (2001). *Web Cartography.* New York: Taylor & Francis.

Kresse, Wolfgang, David M. Danko, and Kian Fadaie (2012). “Standardization.” In Kresse, Wolfgang and David M. Danko (eds.), *Springer Handbook of Geographic Information.* Springer. pp. 393-565.

Krishnamurthy, R. R. (2008). *Integrated coastal zone management.* Singapore: Research Pub. Services.

Lake, Ron and Jim Farley (2007). “Chapter 2: Infrastructure for the Geospatial Web.” In Scharl, Arno and Klaus Tochtermann (eds.), *The Geospatial Web: How Geobrowsers, Social Software, and the Web 2.0 are Shaping the Network Society.* London: Springer.

MacEachren, Alan M. (2005). “Moving Geovisualization toward Support for Group Work.” In Dykes, Jason, Alan M. MachEachren, and Menno-Jan Kraak (eds.), *Exploring Geovisualization.* Elsevier. pp. 445-461.

MacEachren, Alan M. and Menno-Jan Kraak (1997). “Exploratory Cartographic Visualization: Advancing the Agenda.” *Computers and Geosciences* v.23 n.4 pp. 335-343.

Masó, Joan, Keith Pomakis, and Núria Julià. 2010. OpenGIS Web Map Tile Service Implementation Standard. Version 1.0.0. Open Geospatial Consortium Inc.

Nebert, Douglas, Arliss Whiteside, and Panagiotis (Peter) Vretanos (eds.) (2007). *OpenGIS Catalogue Services Specification.* Open Geospatial Consortium, Inc.

Nielsen, Jakob (1993). *Usability Engineering.* San Francisco: Morgan Kaufmann.

Nielsen, Morten (2008). “Spherical/Web Mercator: EPSG Code 3785.” *SharpGIS* blog. Accessed September 6, 2012. <http://sharpgis.net/post/2008/05/SphericalWeb-Mercator-EPSG-code-3785.aspx>.

2009. “Stakeholder Engagement Strategies for Participatory Mapping.” *Social Science Tools for Coastal Programs* series. Charleston, SC: NOAA Coastal Services Center.

O'Brien, C. Douglas and Roger Lott. "Registration of Geospatial Information Elements." In Kresse, Wolfgang and David M. Danko (eds.), *Springer Handbook of Geographic Information.* Springer. pp. 613-630.

2012. "Creating Your Own Tiles." *OpenStreetMap Wiki.* OpenStreetMap. Accessed June 29, 2012. <http://wiki.openstreetmap.org/wiki/Creating_your_own_tiles>

Peterson, Michael P. (2011). "Travel Log: Travels with iPad Maps." *Cartographic Perspectives* n. 68. <http://www.cartographicperspectives.org/index.php/journal/article/view/9/23>

Roth, Robert E. (2011). *Interacting With Maps: The Science and Practice of Cartographic Interaction.* Dissertation. University Park, PA: Pennsylvania State University.

Sack, Carl (2012). *Mapmaking for Change: Online Participatory Mapping Tools for Revealing Landscape Values in the Bad River Watershed.* Thesis proposal. Geography Department, University of Wisconsin-Madison.

Skarlatidou, Artemis (2011). "Web Mapping Applications and HCI Considerations for Their Design." In Hacklay, Muki (ed.), *Interacting With Geospatial Technologies.* Wiley. pp. 244-263.

U.S. Department of Commerce and U.S. Department of Defense (2011). *Chart No. 1: United States of America Nautical Chart Symbols, Abbreviations and Terms.* Eleventh Edition.

Vatsavai, Ranga R., Thomas E. Burk, Steve Lime, Marco Hugentobler, Andreas Neumann, and Christian Stroble (2012). "Open-Source GIS." In Kresse, Wolfgang and David M. Danko (eds.), *Springer Handbook of Geographic Information.* Springer. pp. 939-966.

Vretanos, Panagiotis (Peter) A. 2010. OpenGIS Web Feature Service 2.0 Interface Standard. Version 2.0.0. Open Geospatial Consortium Inc.

Wright, D., Dwyer, N., Cummins, V., eds. 2011. Coastal Infomatics: Web Atlas Design and Implementation. New York: Information Science Reference.

2012. Serving Tiles. *Switch2OSM*. OpenStreetMap and contributors. Accessed June 29, 2012. <http://switch2osm.org/serving-tiles>