

Submitted To:



MetroGIS Proximity Finder Prototype Final Report

Submitted By:



December 16, 2010

Summary

This project addresses two related needs identified by MetroGIS participants: listing all the jurisdictions that apply to a particular point, and finding the nearest services based on a particular location, with an option to limit the search based on a jurisdiction polygon. These needs, combined as "Proximity Finder" by a MetroGIS workgroup in January 2009, were addressed by developing a prototype web service that takes url-encoded parameters and returns a response in GeoJSON, KML, and GML. The response is a "nearest to" and/or "within polygon" query result that can be used in a wide variety of client applications. Attributes for point results are returned and could also be used in clients.

In addition to the Finder query, a Loader tool was developed that allows authorized users to upload shape files into the service, where they become generally available as a WMS and WFS (listed in Get Capabilities) as well as being accessed through the query service. Sample clients are included. The prototype shows the feasibility of the service, and also shows the performance limitations of using WFS in a query setting. This could be modified to optimize performance for particular uses in a production web service. All parts of the service are open source and are available for download.

Table of Contents

Introduction	1
Software Design.....	2
Use of Service.....	4
Lessons Learned	7
Recommendations.....	7
Final Deliverables & Conclusion.....	8
Appendixes	
Appendix A. Services Documentation.....	9
Appendix B. Installation Documentation.....	9
Appendix C. GeoMOOSE Client Application.....	9

Introduction

In the new world of web and mobile applications, geospatial information is expected to be current and accessible for a wide variety of uses. Providing data and derived information through web services addresses MetroGIS's core value of "build it once, use many times" by making it easy for application developers with similar needs to get consistent results without having to maintain a local copy of data and code. Prototype applications, especially those readily sharable, can aid developers and help clarify data and service needs.

Concept and Workgroup

The current project addresses two related needs identified by participants at a MetroGIS "Geospatial applications and Web Services Needs Forum" held November 20, 2008: listing all the jurisdictions that apply to a particular point, and finding the nearest government services based on a particular location. These needs were combined as a "Proximity Finder" and a workgroup was set up in January 2009 to define approaches for developing a web service. Original workgroup volunteers included Jessica Fendos (DEED), Joel Koepp (City of Roseville), Pete Henschel (Carver County), Chad Riley (Carver County), John Slusarczyk (Anoka County), Steve Jakala (Scott County), John Carpenter (Excensus), Paul Wickman (Northstar Geographics), and Bob Basques (City of St. Paul). Based on the results of that workgroup, MetroGIS chose to offer funding for a project for further development.

In September 2009 MetroGIS released an RFP to create software that would allow an organization to publish a proximity finder service. SharedGeo, along with their subconsultant Houston Engineering, Inc. was awarded the project, and received notice to proceed on January 13, 2010.

Proximity Finder workgroup members and others that continued to provide guidance during this phase of the project included Jessica Fendos (Chair), Joel Koepp, Pete Henschel, Chad Riley, John Carpenter, Mark Kotz, Matt McGuire, Rick Gelbman and Randy Johnson (Metro Council) and Josh Gumm (Scott County, replacing Steve Jakala). (Members submitting proposals did not participate in funding or evaluation decisions.)

Project Steps

The MetroGIS Proximity Finder Service prototype project consisted of three primary phases:

1. SharedGeo worked with the project workgroup to verify and define the scope for the project and specifications for the service. This was done through a Design Meeting Feb. 27, 2010, with the workgroup, followed by online review and edit, leading to a final technical specification completed in May.
2. The project team developed the prototype software and services to meet the specification. A demonstration of the general service structure and "Nearest" and "What jurisdiction am I in" capabilities was held for the workgroup Aug. 12, followed by a comment/response period. A second demonstration of the prototype data loader as well as refinements to the service was held Oct. 18, and additional comments and suggestions received.
3. Conclusions, observations and supporting information, including relationship to other MetroGIS projects, were compiled in this final report and presentation to MetroGIS, and technical documentation and a downloadable software package compiled. This project has been successfully completed and this report serves as final deliverable for the project with recommendations for future activities.

The majority of this project focused on programming software that would publish a web service to support two different types of use cases: a Jurisdiction Finder and a Nearest Services Finder.

Jurisdiction Finder

At its core, the jurisdiction finder is a simple spatial look-up of polygons that include the point given by the user. The challenge comes in assembling and loading the data about jurisdiction boundaries, and thus it is a natural choice for a web service. Many usable datasets already exist in the MetroGIS Datafinder catalog or from the MN Secretary of State. Given the cost constraints, the initial SharedGeo prototype used existing datasets for this portion of the project from the MnGeo Data Structures project as well as City GIS datasets. Additional datasets can be configured and loaded into the service (see Loader Script, below) and would be important in a production level implementation of the software.

Nearest Services Finder

The Nearest Services Finder presents more challenges both in respect to data availability and in the design of the spatial query. We assumed that the data available would typically be points, with attributes on each point, and the query would be to find the nearest points of a given type, where the type is set primarily by the layer (e.g. "Hospital") or possibly by layer + attribute (e.g. "School", "Primary"). In some cases the "nearest" would need to be limited by a polygon representing the jurisdiction providing the services.

For this prototype we used the datasets assembled for the MnGeo Data Structures project. They included police stations, schools, fire stations and hospitals. Given the cost constraints of this project, it was not possible to do an exhaustive needs assessment and gap analysis for data, especially for the point-attribute data required for Nearest Services. The nearest services finder was constructed and tested using a general search for points within a given layer, with option to limit by polygon. The current design would also support filtering or performing a query by an attribute using WFS standards; a full demonstration of that service capacity and related needs when designing a client were not included due to the complexity of the issue and the specificity of a solution to a particular client type and use case.

Other requirements for this project included a data uploader tool to easily load data into the proximity finder service from a web page. Software used and created had to be freely distributable. This meant leveraging existing open source software packages and OGC standards.

Additional documentation including workgroup meeting summaries and presentations can be found on the project website at <http://proximity.houstoneng.net/webpage/proxfinder.html>.

Software Design

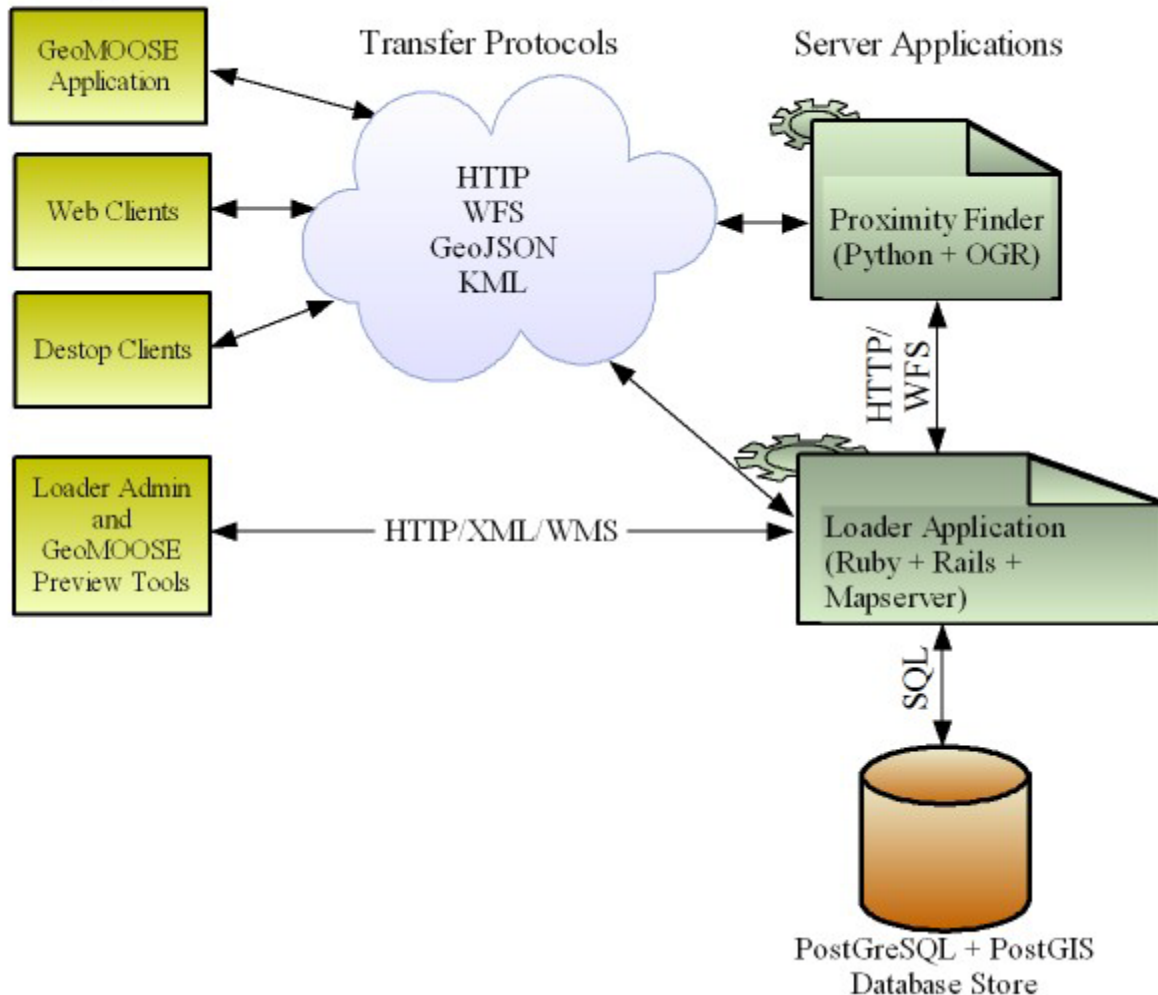
To create the proximity finder web service the project team developed software that leveraged existing open source tools. At a high level the proximity finder project has two core functions:

- it serves out geospatial data as a web service and
- it allows authorized users to upload data to the service.

To gain a better understanding of the software and service the following diagram illustrates the

software components.

Client Applications



The main two components written by the project team include a Python script that uses an existing open source project called OGR (<http://www.gdal.org/ogr/>) to serve as the proximity finder. The proximity finder parses requests from client applications and formats the results to send back to the client. The loader script takes requests from the loader administrative tools to push data into the database and publish a proximity web service. All of the other components are common open source projects that only required configuration files to be made to use them.

Proximity Finder – this is a python script that handles the requests being made to the service. If the request is to the finder service the script will formulate a WFS query using the parameters from the user request and forward that on to MapServer. MapServer is a very robust open source program that can parse the WFS request and make a query to a database. In this case the database contains GIS layers stored in a PostgreSQL/PostGIS database. MapServer then passes the query results back to the proximity finder script in one of 3 formats, either WFS formatted as GML, KML or GeoJSON. This data is then passed back to the client that made the request.

Loader Script – this is a Ruby on Rails application that handles requests that require data to be loaded into the data store. The loader script will do some basic error checking and then push the data into the database via SQL. The script also generates a MapServer mapfile so the layer that was loaded into the database can be made available to the finder service as a WMS and WFS service. Finally the script updates the proximity finder service to show users the GIS dataset is now published as a layer available in the proximity finder service. Currently the loader script only supports ESRI shapefile data format to be loaded.

Samples of Service Use – Two simple sample applications were written to demonstrate the use of the service. First was a GeoMOOSE web application that uses the service in a web map client. The GeoMOOSE application can be used at <http://proximity.houstoneng.net/>. This demonstrates the use cases of finding a nearest feature with the “What’s Near Me?” tool (see screenshots below) and jurisdiction finder with the “What Jurisdiction Am I In?” tool.

Second was a web administrative tool to allow users to load data into the proximity finder service by uploading a shapefile. The loader script then updates a GeoMOOSE application that allows a user to preview the proximity finder layers via WMS. The uploader tools and GeoMOOSE preview application can be tested at <https://www.sharedgeo.org/Proximity-ro/public/geomoose.html>. To load data into the proximity service requires a username and password on the hosted server. Please contact someone on the workgroup to gain access to this level of testing. More information on the loader tools can be read at https://www.sharedgeo.org/datasets/metroGIS_proximity/html/index.html

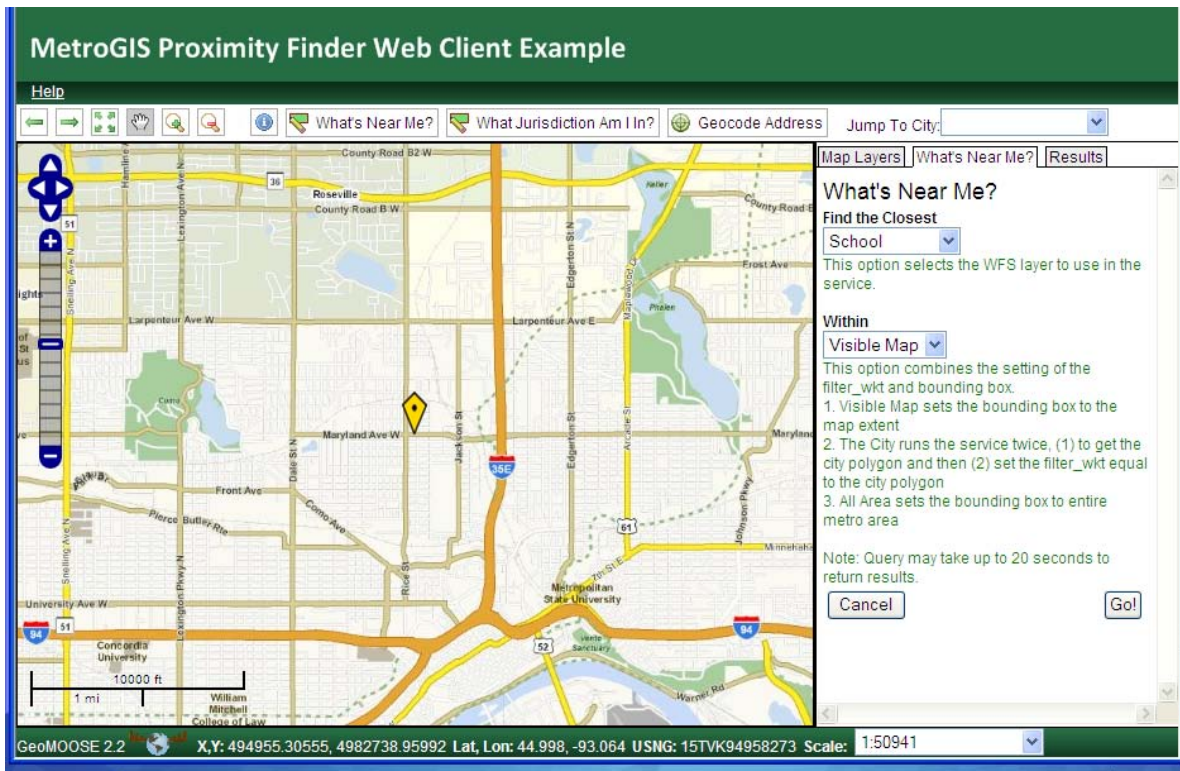
Use of Service

The proximity finder service supports client applications designed to accomplish one of three use cases:

1. Find the closest point to a user-defined point (Nearest Services concept). An example of this could be “Find the nearest school to my address.” An html form example can be tested at <http://proximity.houstoneng.net/webpage/closestpt.html>.
2. Find the closest point to a user-defined point but limit the results to within a defined polygon. An example of this use case could be “Find the nearest fire station to my address within the City of Roseville.” An html form can be tested at <http://proximity.houstoneng.net/webpage/pointinpoly.html>.
3. Find the polygon that a user defined point is located in (Jurisdiction Finder concept). An example of this could be “Find the City I live in.” An html form example can be tested at <http://proximity.houstoneng.net/webpage/pointwithinpoly.html>.

All three of these use cases can also be tested in the GeoMOOSE web client application at <http://proximity.houstoneng.net/>.

Example of a client application that uses the Proximity Finder Web Service.
 (Also uses MetroGIS Geocoder Service to lookup by street address, intersection, or landmark name, uses Metropolitan Council Metro Basemap, MnGeo's MN Structures Collaborative data, and displays location in USNG coordinates)



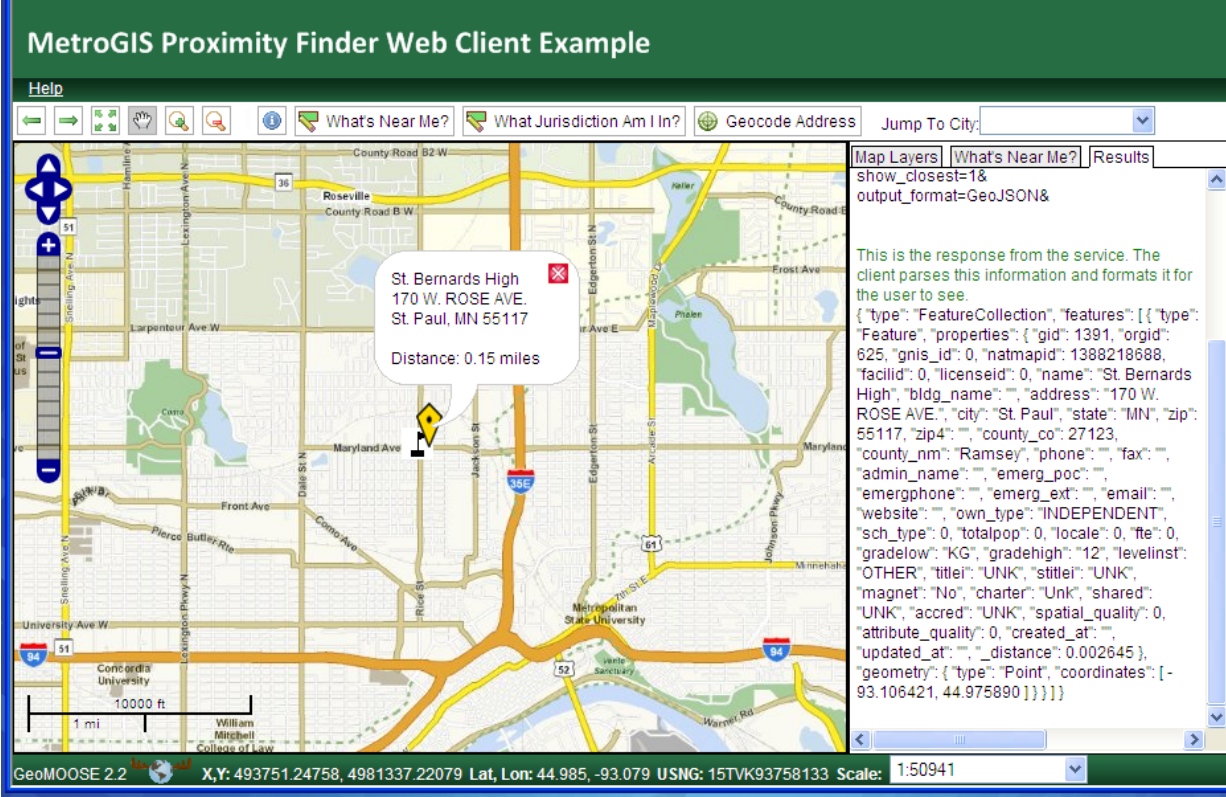
Hitting "Go!" sends the following information to the Service:

```
https://www.sharedgeo.org/MNGEO-
ro/public/ows?SERVICE=wfs&VERSION=1.0.0&REQUEST=getFeature&TYPENAME=layer_1&BBOX=-
93.176005842287,44.940046036,-93.03250675834,45.014592897717&
point_wkt=POINT (-93.1042100000000090 44.9773419999998230)&
filter_wkt=&
show_all=&
show_closest=1&
output_format=GeoJSON&
```

The service responds with information based on the layer chosen in the request. Note that the attributes returned will vary based on the data layer chosen. Results for these attributes could be used in a custom client, for example, a client could use the “show all” option in the service and filter results to returning the closest “PRIMARY” schools. Alternatively, a WFS request could be used for filtering.

The MN/Schools data currently loaded in the prototype are from the MnGeo/EPC Minnesota Structures Collaborative project

<http://www.mngeo.state.mn.us/committee/emprep/structures/index.html>



MetroGIS Proximity Finder Web Client Example

Help

← → ↺ ↻ 🔍 What's Near Me? What Jurisdiction Am I In? Geocode Address Jump To City: [dropdown]

Map Layers | What's Near Me? | Results

show_closest=1&
output_format=GeoJSON&

This is the response from the service. The client parses this information and formats it for the user to see.

```
{ "type": "FeatureCollection", "features": [ { "type": "Feature", "properties": { "gid": "1391", "orgid": "625", "gnis_id": "0", "natmapid": "1388218688", "facilid": "0", "licenseid": "0", "name": "St. Bernards High", "bldg_name": "", "address": "170 W. ROSE AVE.", "city": "St. Paul", "state": "MN", "zip": "55117", "zip4": "", "county_co": "27123", "county_nm": "Ramsey", "phone": "", "fax": "", "admin_name": "", "emerg_poc": "", "emergphone": "", "emerg_ext": "", "email": "", "website": "", "own_type": "INDEPENDENT", "sch_type": "0", "totalpop": "0", "locale": "0", "fte": "0", "gradelow": "KG", "gradehigh": "12", "levelinst": "OTHER", "title": "UNK", "stitle": "UNK", "magnet": "No", "charter": "Unk", "shared": "UNK", "accred": "UNK", "spatial_quality": "0", "attribute_quality": "0", "created_at": "", "updated_at": "", "distance": "0.002645" }, "geometry": { "type": "Point", "coordinates": [ -93.106421, 44.975890 ] } } ] }
```

GeoMOOSE 2.2 X,Y: 493751.24758, 4981337.22079 Lat, Lon: 44.985, -93.079 USNG: 15TVK93758133 Scale: 1:50941

It should be stressed **the proximity finder project created code that publishes and manages a service and is not intended as an end-user client application.** It is expected that developers would use this service as part of other client applications as demonstrated with the above GeoMOOSE web client.

The service could also be used to create a Google Earth display of output by specifying the response as KML.

The Loader application that is part of the proximity finder project is a great tool that could be used by an organization to allow data providers to manage and publish GIS datasets to the proximity finder service. An ancillary use is the application will also publish the layers in OGC WMS and WFS.

The proximity finder service and loader are open source, as required in the RFP, and have no dependencies on commercial software, but have some dependencies on other open source software projects such as MapServer, OGR and PostgreSQL database. Organizations that wish to use the Proximity Finder / Loader software by re-deploying on their own hardware need to have the supporting software installed as well.

Lessons Learned

1. This service can be readily integrated with other MetroGIS and related tools to make straightforward web mapping applications using a variety of client software approaches.
2. The project team recognized that transferring data via the WFS standard is not ideal. While it is an OGC standard and facilitates a structured method for querying and transferring data it is a bit slower for two primary reasons. First the data is not compressed. Second WFS services cannot be indexed in the database to facilitate faster searches. An alternative to WFS could have been to simple query the database directly using Python and not go through MapServer using WFS querying.
3. For best overall performance and reliability, a service could be set up for a specific dataset that has a known set of attributes and geography. Generalizing the software code to publish a generic web service with query capability for potentially any GIS dataset sacrifices performance.
4. The service provides a ready technology platform, but this does not eliminate the need for having data stewards to deploy a trusted service.
5. The Jurisdiction Finder functionality has a number of data sets available that are of adequate quality for general use. However, for the “Nearest” functionality it was challenging for the workgroup to develop concrete use cases during the kickoff meeting. Most ideas that were brought up identified data gaps.

Recommendations

1. The project team suggests the next step is for this concept to be moved from the prototype stage into a production level web service. This would mean the workgroup would need to define a specific business use case and ensure the GIS dataset exists to support the use case. One example that has been mentioned in numerous meetings would be a nearest service finder for things like driver’s licenses. The workgroup would need to define what specific data layers would be published with the service and define who’s responsibility it would be to ensure the data and service are reliable.
2. The project team suggests the ideal hosting organization for the proximity service could vary depending on the business use case developed and who is responsible for the GIS dataset. Successful long term reliability of the service would be most likely if the organization that is the data provider is responsible for data administration. Physical hosting could be done either in-house or through a hosting service such as those provided by Houston Engineering or SharedGeo.

3. If this service were to be made public and put into production the workgroup would need to define authentication roles on who can upload datasets into the service. The workgroup would also need to formalize data custodians for each data layer published in the service.
4. The Loader currently publishes uploaded GIS files in a number of formats (WMS, WFS, KML) usable in open source and other geospatial applications. A nice addition to the proximity finder may be to explore adding GeoServices REST capabilities. ESRI has recently published a GeoServices REST specification (<http://www.esri.com/industries/landing-pages/geoservices/geoservices.html>), so in theory code could be developed to publish GIS datasets uploaded through the loader service into an ESRI REST endpoint. We stress that research would need to be done to determine if this would violate any terms and conditions of the GeoServices REST specification. The current release of this specification is under an Open Web Foundation agreement and is subject to those terms and conditions.
5. The project team recognizes that not every organization involved in MetroGIS uses Open Source software. It may be beneficial to develop an ESRI client application similar to GeoMOOSE to demonstrate the use of the web service with an ESRI client application.

Final Deliverables

The final deliverables for this project include:

- Final report with links to documentation and findings.
- Software and configuration files for complete proximity finder service setup and installation
- Example Client Application – A GeoMOOSE client application that is configured to use the SharedGeo implementation of the proximity finder with GIS layers used in the prototype project.

More information on the final deliverables can be found in the appendices.

Conclusion

Overall the project was successful and demonstrates that software can be developed to publish datasets to a web service that supports both a “Jurisdiction Finder” and “Nearest Finder” use cases. This project serves prototype for a proximity finder service and truly the next steps rely on the workgroup sorting out details of who the data custodians might be for the service. The proximity finder project demonstrates the return on investment of “build it once, use it many times”. The project also demonstrates that the use of Open Source software could save an organization considerable money by avoiding commercial software licensing costs.

Appendix A. Documentation of Services

All of the documentation for the finder and loader services can be found on the SharedGeo project website.

https://www.sharedgeo.org/datasets/metroGIS_proximity/html/index.html

Appendix B. Installation Documentation

Finder Service –

https://www.sharedgeo.org/datasets/metroGIS_proximity/html/proximity-install.html

Loader Service – (same as above)

Appendix C. GeoMOOSE Client Application

The GeoMOOSE client application was created using GeoMOOSE 2.2 with some customizations to the javascript code to handle the custom finder service requests. A custom GeoMOOSE service was also created to allow GeoMOOSE to make calls to the proximity finder web service. The application was also customized to support two user cases which included “What Jurisdiction am I in?” and “What’s Near Me?”. To install the GeoMOOSE code you will need to have a web server running with Apache or IIS. If you are using a Windows operating system you can use the MapServer for Windows installation and add the GeoMOOSE application to it. Follow instructions at http://www.geomoose.org/docs/install_ms4w.html.

After installing and verifying GeoMOOSE is working properly you can replace these files in your setup and you should see the application working as shown at <http://proximity.houstoneng.net/>. The entire GeoMOOSE application code can be downloaded at: <http://proximity.houstoneng.net/proximity.zip>

Replacement files:

- Extensions/proximity.php – Custom proximity service
- Php/addresssearch-geocoder.php – Added links to geocode results for proximity
- Compiled.js – Customizations for vector layer type, spinner, tab names
- Extensions/custom.js – Custom javascript for drawing vector layers, markers
- Mapbook.xml – Several new services for the proximity tools
- Skins/green/green.css – Small tweaks and addition for spinner