LA-Plan GUI Development Progress Report

Michael F. Goodchild, Wenwen Li and Nate Royal
11-29-2011
Department of Geography
University of California, Santa Barbara
Santa Barbara, CA, 93106-4060
{good, wenwen, royal}@geog.ucsb.edu

1. Abstract

GIScience plays an increasingly important role in urban economic modeling. This report presents the utilization of advanced methods and technologies in GIScience in support of the LA-Plan. The LA-Plan is a collaborative effort among geographers at UCSB, economist, demographers, and environmental engineers from UCR, and planning and public-policy experts from UC Berkeley. The LA-plan is a multi-year research program funded by the University of California's Multi-campus Research Programs and Initiatives (MRPI). The plan seeks to develop a virtual (online) co-laboratory, aimed at revolutionizing spatial policy analysis for the Greater Los Angeles Region by making it possible for metropolitan planning organizations (MPOs), urban, public policy, and environmental experts to collaborate productively using a best-practice microeconomic simulation model. The core of the project is to provide data for architecture, mapping capabilities, and a human-computer interface for a diverse body of users. Users will be able to interact and collaborate on policy analysis online using the general equilibrium model "RELU-TRAN." The LA-Plan co-laboratory will rely on emerging tools for CyberGIS infrastructure detailed below in the following report. The methods used for data organization, data sharing and data visualization will be introduced in detail in this report.

2. Motivation

The motivation of this Graphic User Interface (GUI) is to provide a convenient online tool for both GIS expert and non-GIS expert to view, manipulate and visualize the geospatial data that are commonly used for urban economic analysis. For non-GIS experts, viewing data and sharing model outputs across the distributed team is difficult as it often requires

the use of professional GIS software to complete the task. Moreover, data available on local computers is hard to share with remotely team members. Therefore, we aim to provide a GUI to facilitate the sharing and interoperation of geospatial data and model output. Technical problems with completing this GUI include (1) providing an extendable data structure and software framework to allow data of various formats to be integrative: (2) designing a flexible user interface that is compatibile with different internet browsers; (3) determining a middleware package that balances the development cost and supported functionalities to the users. In this report, we will focus on introducing the data organization, data to service conversion, and evaluating the GUI design tools and prototypes.

3. Sharing Geospatial Data for RELU-TRAN Simulation

As the LA-Plan project is based on the collaboration of a geographically distributed team, there is an urgent need for an efficient virtual platform with which to share the geospatial data. Figure 1 shows the required datasets for conducting the model simulation in RELU-TRAN. These datasets are inherently heterogeneous in data structure: some of the datasets were produced and maintained as ESRI shapefiles, some are in Microsoft Excel sheet format, and a few datasets are duplicated in both formats. The raw data is hosted on the UCSB web server but this server is difficult to manipulate by individuals not trained in GIS. In addition, the spatial analysis of any economic activity always requires the combination of several of different datasets. To facilitate the sharing of geospatial data and online visualization, we propose a Service-Oriented Architecture (SOA) and to utilize an advanced open source GIS development package with which to create an interactive online portal to improve the interoperability of spatial datasets and the real-time spatial analysis.

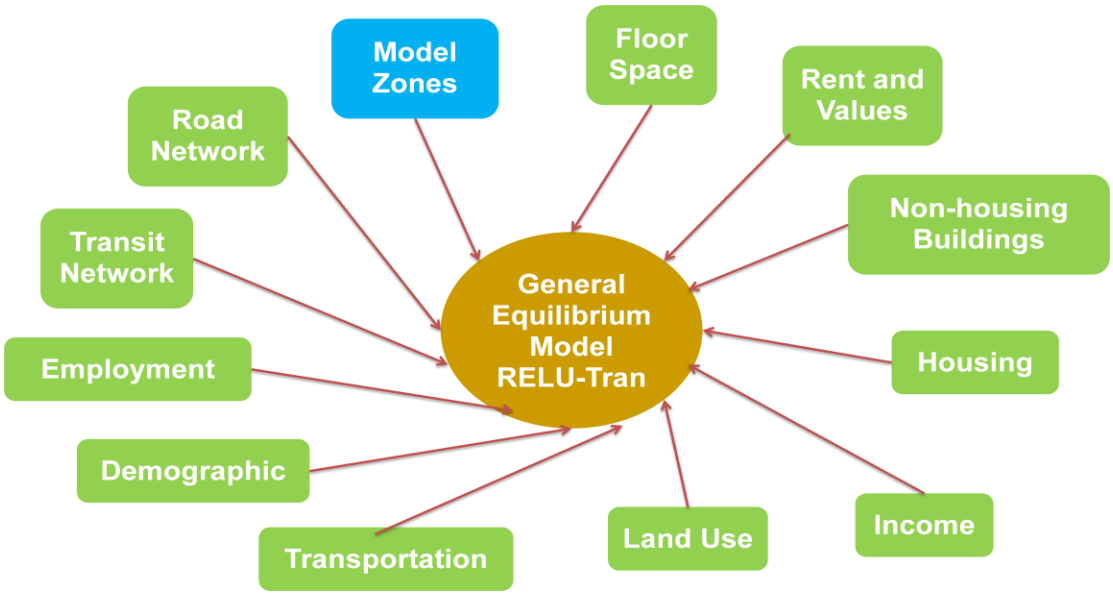

Figure 1. Required categorical dataset for model simulation. The sources of the datasets include SCAG (Housing, non-housing buildings, Transportation, Road Network, Land Use, Rent and Values), Census (Demographic, Income), CTPP (Transportation, Employment). The model zone dataset, which determines the geographical scale at which the urban economic activities are studied, is considered the most essential sets of data for the project. All other dataset are required to be aggregated to the scale of model zone for model simulation.

Figure 2 shows an object-oriented organization of datasets in our database. Each of the datasets can be abstracted to a class with attribute data. For example, a "GIS Dataset" class has attributes indicating the coverage of the data, the projection system of the data, the lineage of data, and other metadata describing this class. Each class can have multiple subclasses and one or more super-classes. This allows the data to be organized in a tree hierarchy. The subclasses will inherit all the attributes and properties of its parent class. For example, a GIS dataset may be represented by a polygon (such as county boundaries), a polyline (such as road networks) or a point (such as locations of retail stores). All three classes share the attribute information such as coverage, projection and source from their parent class, while maintaining their own distinctive attributes. Further, a polygon class has a function of calculating its area; a polyline class has a function of calculating the intersection between this polyline to another polyline; while a point class may have a

distance function to calculate the distance between one point to another. Once these classes are instantiated, they can be used to represent a real world visualization used in LA-Plan project. For example, a transportation dataset is a package of classes including traffic nodes, traffic links and local networks. The traffic node is a point class while traffic links and local road networks are polylines. To make them interoperable with each other, these datasets are encapsulated into web services, e.g. the "WMS (Web Map Service)" class in Figure 2. Web Map Service is a standardized web service proposed by Open Geospatial Consortium (OGC) for sharing heterogeneous datasets. This web service provides three standard APIs, GetCapabilities, GetMap and GetFeatureInfo through HyperText Transmission Protocol (HTTP) protocol. GetCapabilities interface allows requesting all metadata of a dataset, including data layers, projections, bounding box and map image format the service provides. GetMap is the major map request interface that transforms the backend dataset into a static image, so that the client only needs to overlay the geo-rectified images at the correct extent instead of conducting the data format transformation before the map composition. GetFeatureInfo supports the acquisition of attribute information of a specific feature to make the map more meaningful. In section 4, the deployment of web services will be introduced in details.

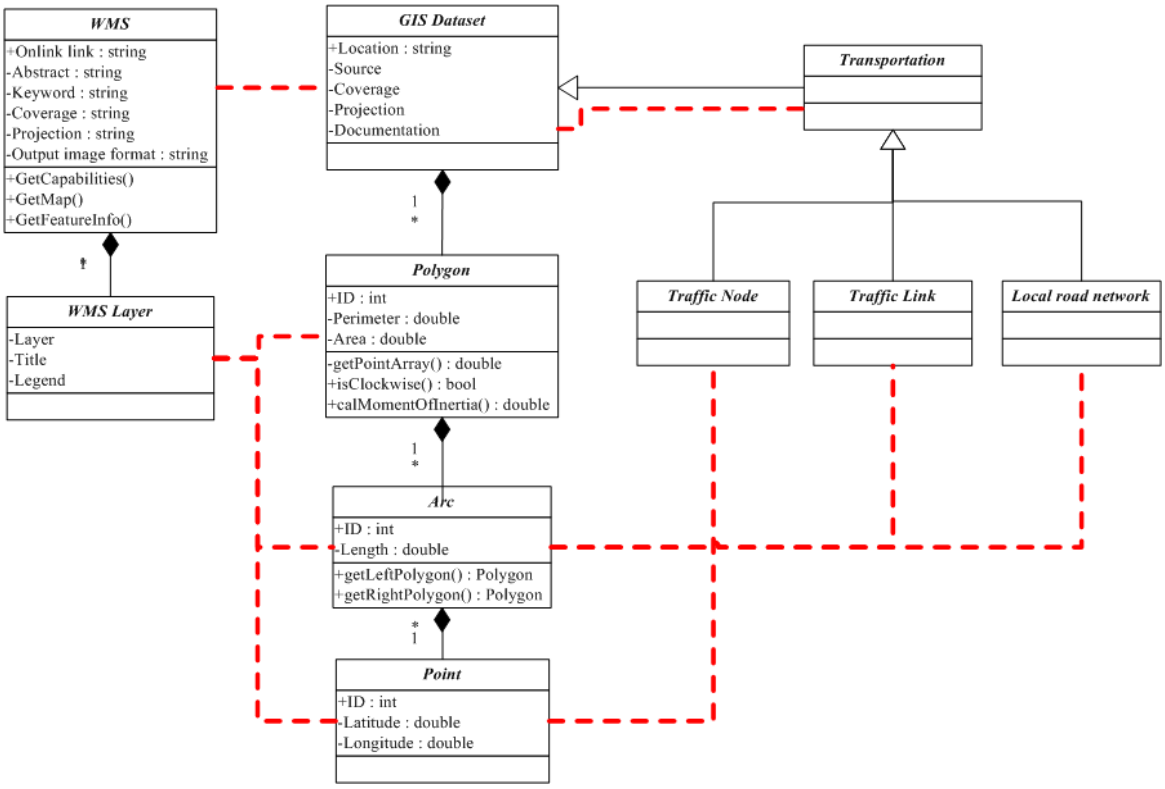

Figure 2. Object-oriented data organization

4. System Architecture Design

Figure 3 shows the overall architecture in designing the GUI. The left hand side is the database, in which all original geospatial data is stored. The datasets in the database are deployed through ESRI ArcGIS server into different Web Map Services (WMS). The GUI focuses on facilitating users in conducting searches, service combinations, and online visualization. An example workflow could be stated as: The service clearinghouse stores the descriptive information about WMSs, which are parsed and evaluated through a service parser. As the web services may go offline due to the problems with a network or server, it is necessary that the status of a web service is monitored such that the GUI can function as continuously as possible. Clients of the GUI can view available layers from a hierarchical tree and search additional datasets using the spatial-temporal search functions. Several communication protocols are introduced in developing the GUI: (1) JDBC: Java DataBase Connectivity is a standard interface-based Java mode abstraction of persistence. It is used to maintain the connectivity between Java programming language and a wide range of

databases, in the case of this project a MySQL database is used; (2)AJAX: **A**synchronous **Ja**vaScript and **X**ML is a grouping of interrelated web development methods used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to a server and retrieve data from a server without interfering with the display and behavior of the existing page. It's a mechanism of background multi-thread processing of concurrent requests. The advantage of AJAX is that the speed for remote data request could be improved due to introduced mechanism of parallel processing. Meanwhile, it resolves the problem of traditional GUI which tends to easily get stuck while the client and server are in communication.

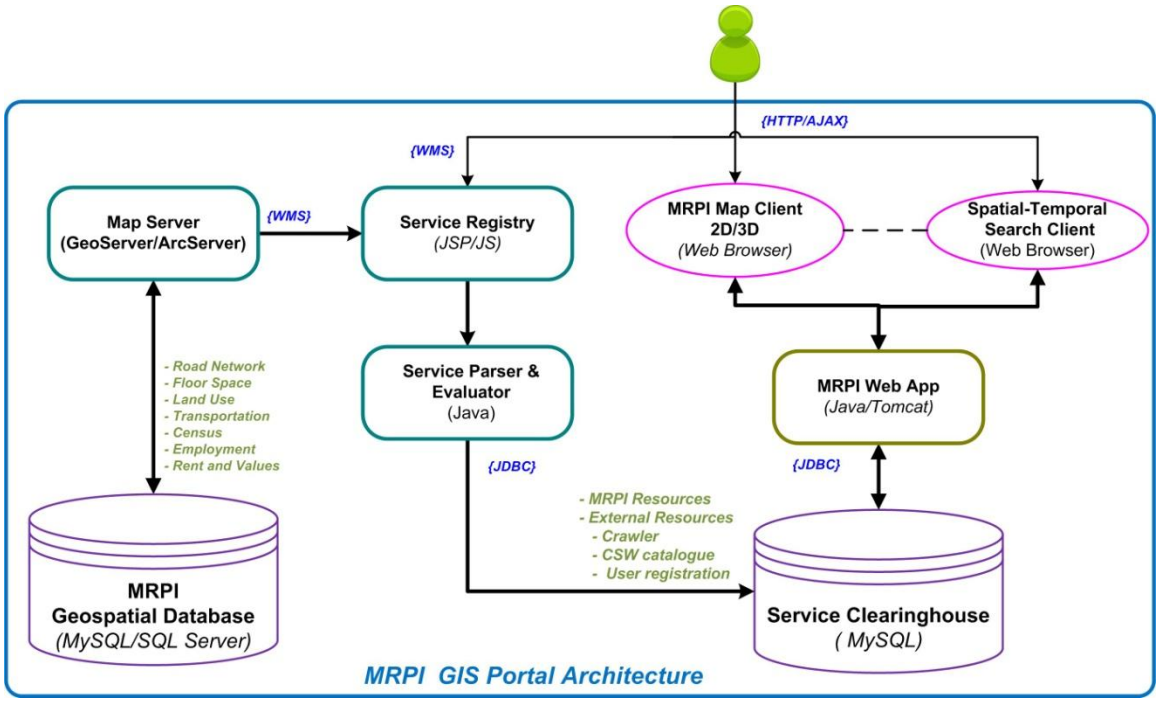

Figure 3. Architecture Design of LA-Plan Portal

5. Key Technologies

5.1 Creating Web Map Services Using ArcGIS Server

Data is organized into different groupings prior to creating WMSs for each. Currently, there are six groups:

a. Zoning datasets, including the intermediate result from zoning procedure

b. Base map data, including the layer file from Open Street Map, which is largely volunteer geographic information.
c. Transportation data, including the abstracted road network and road intersections, as well as freeways, interstate and local road network. These data are used for TRAN simulation
d. Floor space data, including the FAR statistics for residential area, commercial area, industrial area and all others computed for every model zone in our study area.
e. Other dataset, including elevation dataset, census tract, city layers, area and population centroid of a model zone, transit route and etc.

Each of the above data groups are being maintained in a map file (.mxd) file on the server and are deployed into web services by the following procedure

1. Start ArcGIS Server Manager hosted at mrpi.geog.ucsb.edu and log in.
2. Click the Services tab.
3. Click Publish a GIS Resource.
4. Use the Resource drop-down list to browse to the map document (.mxd) that you want to publish.
5. Type a name for the service in the Name text box. The name cannot be more than 120 characters long, and may contain only alphanumeric characters and underscores.
6. Accept the default folder level by clicking Next.
7. Click Next again to accept the default capability of Mapping.
8. Review the information about the service you are about to create, then click Finish to create the service.

Once the service is created, we need to ensure that it is working correctly. Below are the steps we used to check the status of a service:

1. Click the Services tab in Manager.
2. Make sure the "Services in" drop-down list contains the server name (the root level of the server) and not a folder name.

3. Click the plus (+) button to expand the information about the service you just published. If there is a preview image after a few seconds, we can tell that service is running correctly.

5.2 GeoExt for GUI design

GeoExt is a JavaScript library for creating rich web mapping applications. It combines the web mapping library OpenLayers with Ext JS, a cross-browser JavaScript library (for menus, trees, layout, window, trid, tab, panel etc) for building rich internet applications. Because GeoExt provides a number of useful widgets and a group of APIs to support data handling and it is OpenSource, it is chosen as the main technique for the GUI design. The current version in use is GeoExt 3.4.0 under Berkeley Software Distribution (BSD) license.

The following components are being used for creating the GUI:

a. Map Panel

Map panel is used to visualize and operate the maps provided by remote WMSs and is the most important component in the GUI. Each layer from remote web map server is declared as a Map type in the OpenLayer package: each is accessed through a LayerStore interface. Base map functions such as Zoom, Pan, Previous Scene, Next Scene, Zoomslider are provided by the "Control" components of OpenLayers. These functions are coded with Javascript.

b. WMS Browser

The "WMSBrowser" is a class that extends "Ext.Panel" and is responsible for generating all the panels and toolbars for previewing and selecting a data layer from remote server. Once loaded, it will at first initialize the "ServerComboBox", which provides the dropdown list of available WMS servers; and then create the "MapPanel", which is used to preview the content of the layer of interest. This "MapPanel" is different from the one introduced above as this MapPanel is part of the popup window for adding or removing a data layer. After the "MapPanel" is initiated, the status bar and the tree panel (discussed in section d) will be generated. All of these components are organized in a

layout by functions provided by “Ext”. The class “WMSBrowser” will initiate “WMSBrowserWMSCapabilitiesStore” (discussed in section c) too to make it ready to parse and display the available layers from a specific server.

c. WMS Capability Store and Parser

The “WMSBrowserWMSCapabilitiesStore” is an extension of the GeoExt class “GeoExt.data.WMSCapabilitiesStore”and is used to store the metadata information of the data layers of a WMS service. The following events (beforecapabilities, getcapabilitiessuccess, getcapabilitiesfail, genericerror, beforelayeradded and layeradded) are customized to handle the data fetching process. When a user tries to connect to a server, the GUI will check whether or not the URL (Universal Resource Locator) is valid before triggering a getCapabilities request. Once the capability file is loaded, a thread begins to loop through all the layer records, and check whether they support the map’s project and intersects with the current map’s extent. If the request returns records that means it is a valid server and the record will be added in the dropdown list in the loading panel. If not, the thread will send an event indicating that no layers will be returned. The available layers will then be loaded into the tree panel.

d. Tree Panel

The tree panel is used to display the data layers of interest by users. It is implemented by extending the Ext.tree.TreePanel and Ext.tree.AsyncTreeNode. For every WMS service, there will be one root node, which will be stored as an AsyncTreeNode. If the capability file of a WMS can be parsed, child nodes will be created to the root node and an event of “getcapabilitiessuccess” will trigger. The tree panel handles the “add layer” request. Once a user checks one or more layers to visualize by clicking the “add layer” button, a virtual new layer that combines the requested layers will be generated and will be loaded to the map panel.

The below diagram (Figure 4) shows the workflow and class organization for the GUI. Each box refers to a class in the program: the blue boxes are mainly part of the popup

window for previewing the layers of interest. They are controlled by the class "WMS Browser" as discussed above. The classes in the red boxes are components in the main GUI. These two components interact with class "WMS Browser" to change the list of available layers and the map content.

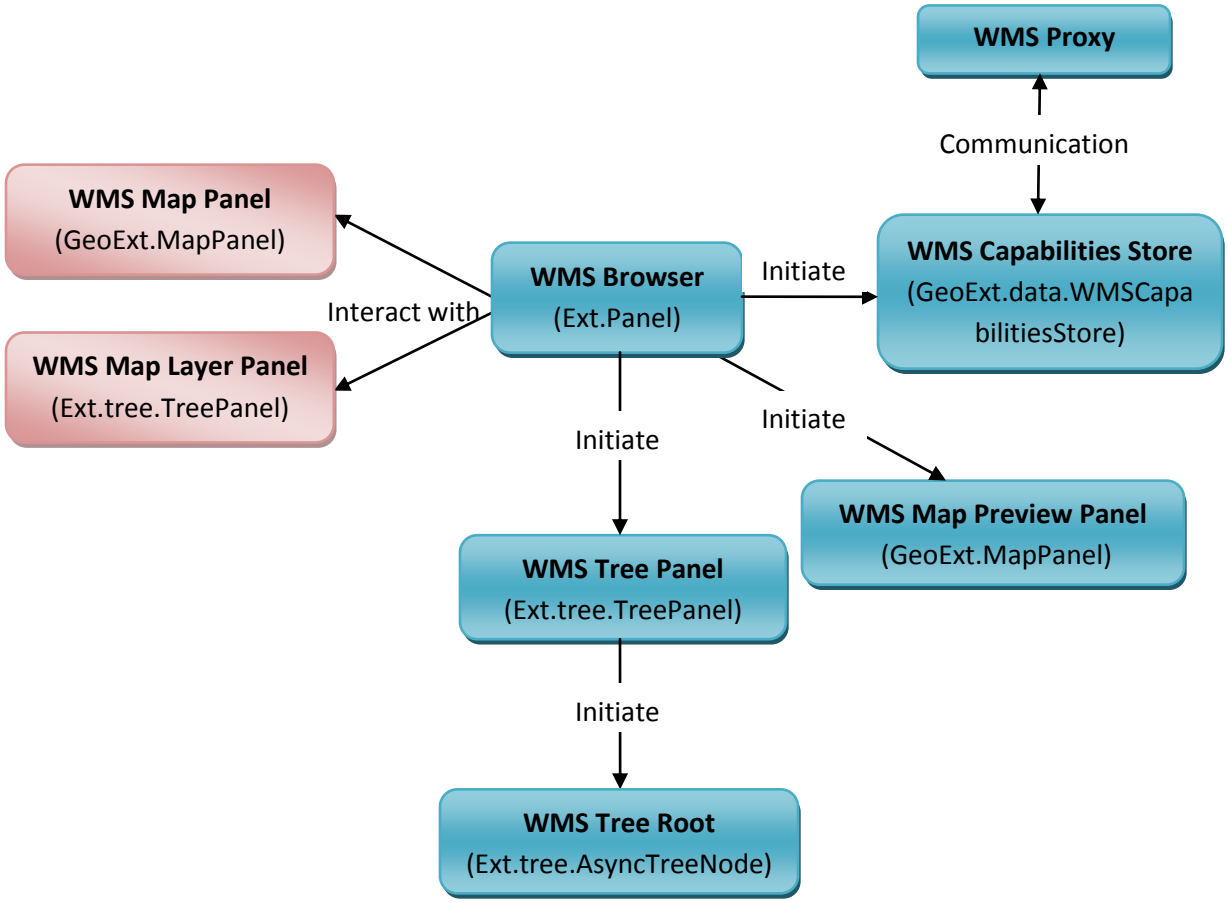

Figure 4. LA-Plan GUI workflow

5.3 Deploy and host web application using Apache Tomcat Server

This web application of the GUI is hosted by Apache Tomcat Server 6.0.28, which is an open source software implementation of Java Servlet and JavaServer Pages technologies. Apache Tomcat is very powerful and is a very stable platform for large-scale, mission-critical web applications. The following procedure is followed to deploy the application on the mrpi.geog.ucsb.edu server.

1. Installing Java Development Kit (JDK), under which the Apache tomcat will operate.
2. Installing Tomcat available at: http://jakarta.apache.org/downloads/binindex.html

3. The web application needs to be organized in the required format.

- ***.html, *.jsp, etc.** - The HTML (HyperText Markup Language) and JSP (JaveServer Pages) pages, along with other files that must be visible to the client browser (such as JavaScript, stylesheet files, and images).
- **/WEB-INF/web.xml** - The *Web Application Deployment Descriptor* for the web application. This is an XML file describing the servlets and other components that make up the application, along with any initialization parameters and container-managed security constraints.
- **/WEB-INF/classes/** - This directory contains any Java class files (and associated resources) required for LA-Plan, including both servlet and non-servlet classes, that are not combined into JAR (Java Archive) files.
- **/WEB-INF/lib/** - This directory contains JAR files that contain Java class files (and associated resources) required for LA-Plan, such as third party class libraries or JDBC drivers.

4. In order for the web application to be executed, it must be deployed on a servlet container. What is done here is to Copy unpacked directory hierarchy into a subdirectory in directory $CATALINA_HOME/webapps/. Tomcat will assign a context path to LA-Plan application based on the subdirectory name you choose.
5. Restart Tomcat after installing the application.

Currently, the Tomcat server runs on port 8080, to avoid the conflict with ArcGIS Server, which uses the default port 80.

6. Prototype

We have worked on developing the prototypes under three phases. The prototype I is shown in Figure 5, the prototype II is shown in Figure 6 and the prototype III is shown in Figure 7. The prototype I was customized by the Javascript and Flex support of ArcGIS server. It is a typical, uncomplicated GUI for data visualization. However, in part because of its simplicity, we have minimal control of the GUI and the response speed from ArcGIS server is usually very slow. The GUI prototype II is built upon the WMSUniportal technology – it is very flexible and we have complete control of the data processing

procedure since it merely contains third-party software packages. However, the GUI is not altogether user friendly (through subject interviews, discussed in the next section), therefore, we are working on the GUI prototype III to reconcile the advantages of the GUI prototype I and the GUI prototype II for a user friendly and multi-functional GUI. The specification and comparison of the three GUI is listed in Table 1.

6.1 Prototype I

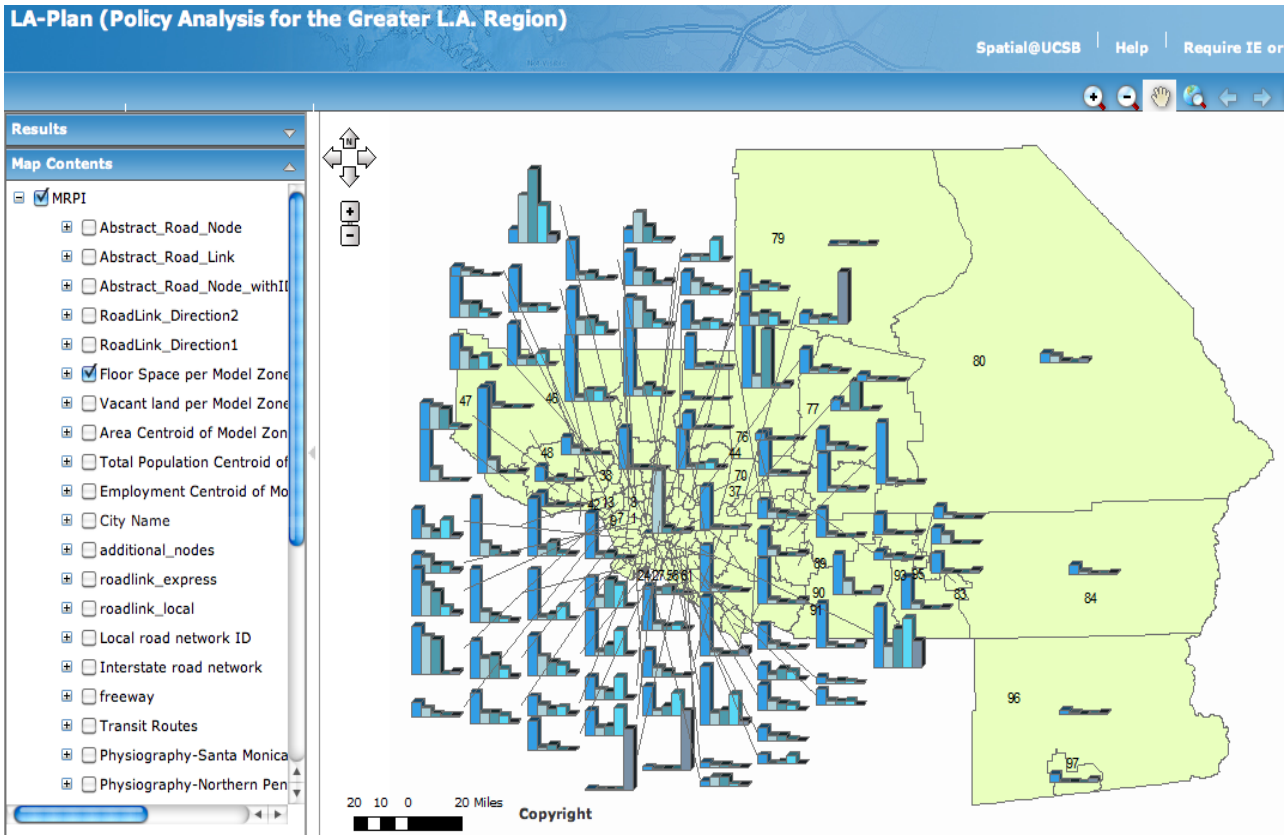

Figure 5. GUI Prototype I (http://mrpi.geog.ucsb.edu/mrpi)

6.2 Prototype II

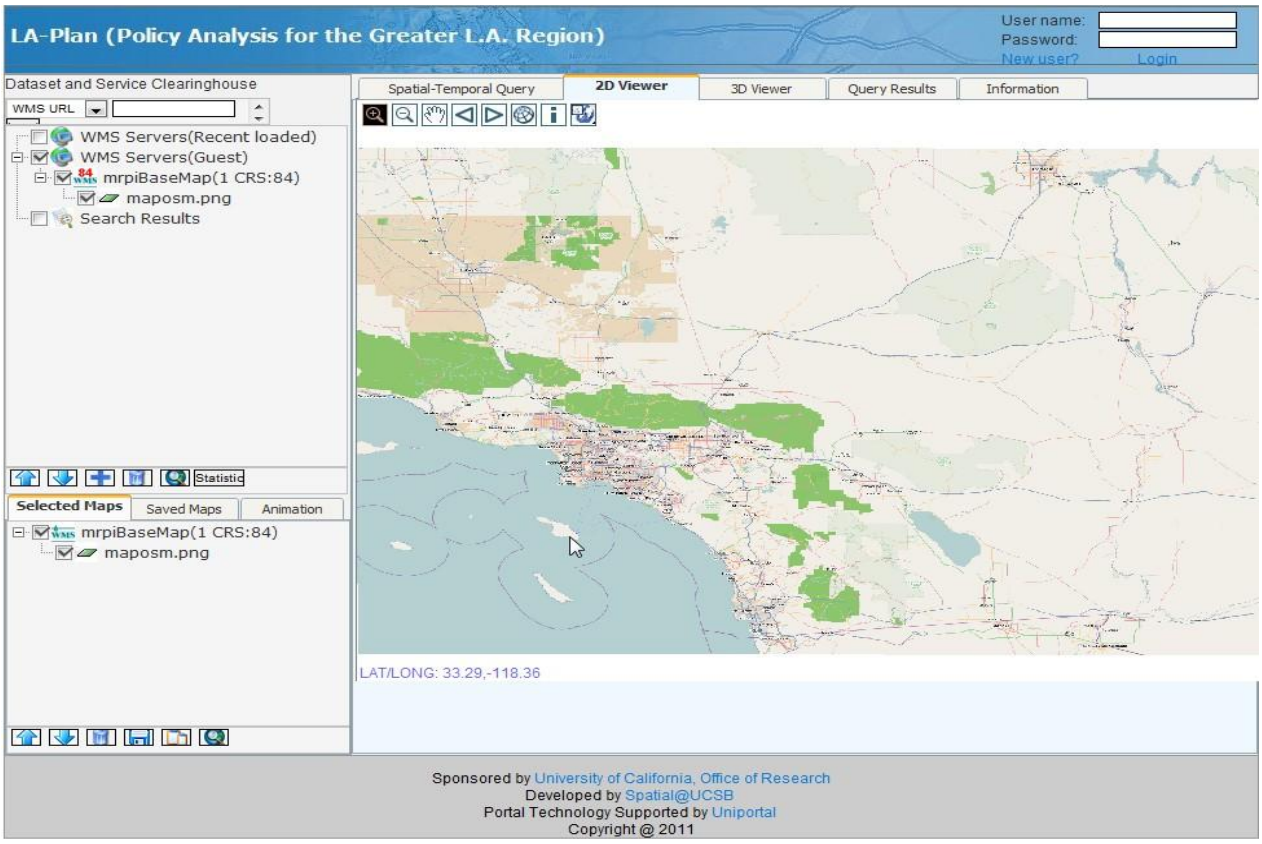

Figure 5. GUI Prototype II (http://mrpi.geog.ucsb.edu:8080/mrpi2)

6.3 Prototype III

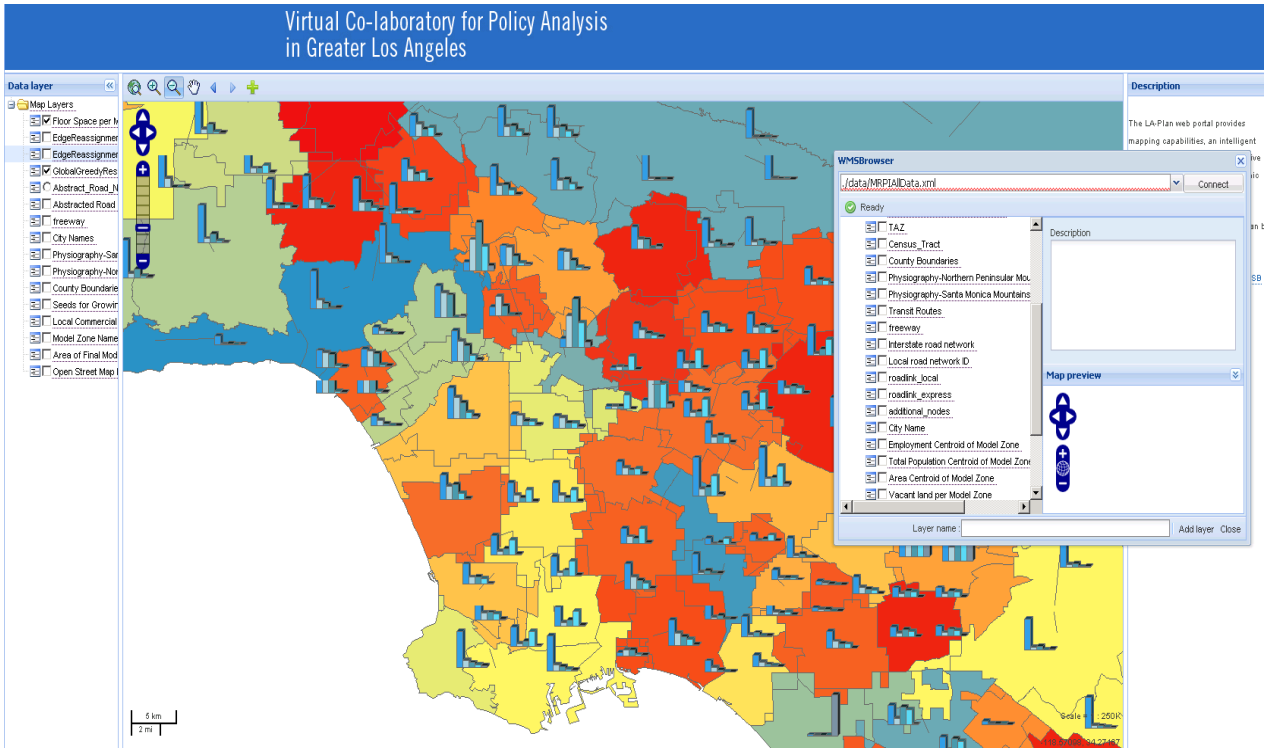

Figure 5. GUI Prototype III (http://mrpi.geog.ucsb.edu:8080/mrpi/GUI).

6.4 Comparison

	Phase I	Phase II	Phase III
GUI Flexibility	Good	Bad	Good
Open Source or Not	Commercial	Open Source	Open Source
API Encapsulation	Well	Not well encapsulated	Well
Control of GUI	Minimal	Maximal	In between
Development cost	Minimal	Maximal	Middle
Response Speed	Relative Slow	Fast	Fast

(client)			

Table 1. Comparison of pros and cons of the three GUIs

7. Test and evaluation

To evaluate the usability of the GUI prototypes, we conduct subject interviews and software test for feedbacks from general user and GIS experts. The LA-Plan user interface was tested during two separate trials. The first trial was completed in August 2011 and lasted two weeks. The second trial lasted four days in early October of 2011. Each trial was given to attendees of the University of California Santa Barbara who were affiliated with the UCSB geography program.

7.1 Trial One

Trial one was designed to test the LA-Plan user interface on undergraduate students enrolled in an introduction to Geographic information Systems course. Students were all over 20 years of age and either in their junior or senior year of their undergraduate program. Each student had taken an introduction to cartographer course or the equivalent as a prerequisite to attending the class. The trial was conducted in the latter half of the semester by which time the students had each had at least three weeks of exposure to ArcGIS products, in particular Arc Map 10.0. All students had previously used Google Earth.

The students were given no information regarding the purpose or usefulness of the LA-Plan user interface. None of the students had used an ARC Server user interface to their knowledge and were given no direction as to how to operate the interface. Students were asked to troubleshoot the user interface voluntarily. Those students who were able to detect the most problems or error with the system were given extra credit on their lab assignments for class. No instructions as to how to trouble shoot the product were given, only as to how to write up a report of problems found.

Three students completed a full report, identifying sixteen issues with the product which needed clarification or fixing to successfully operate.

7.2 Trial Two

Trial two was designed to test the LA-Plan user interface on graduate students enrolled in the UCSB geographer department. Students were all over 23 years of age and had completed several geography courses and were each familiar with Arc Server user interfaces and ESRI products. All students had previously used Google Earth.

The students were given no information regarding the purpose or usefulness of the LA-Plan user interface. Students were asked to troubleshoot the user interface voluntarily. Via an email request students were requested to try and break the user interface or cause it to crash without use of hacking. Students were asked to catalogue the errors or problems they had found with the user interface. The nature of the trial was also competitive, similar to trial one, encouraging the tester to find as many errors as possible. Fourteen respondents identified twenty-seven errors or problems with the user interface.

7.3 Catalogue of Tester Identified Problems

Combined both trials exposed seventeen separate issues with the user interface that needed to be address in order to improve the interface (prototype II). The below list of issues are organized by the frequency with which the issue was mentioned in the sum of reports.

1) Does not work with Safari web browser: a blank screen is all that is the user sees

2) Does not work with a Google Chrome browser: the interface's icons are scrambled and it is impossible to use

3) In the 3d viewer there is no map. The user of the interface has not downloaded Google Earth

4) In the 3d viewer there is no map. The Google maps API key was registered to a different map

5) The “I” info tab does not work, and the error message is rife with misspellings

6) Layers found or selected weren’t functioning: the problems of this sort were various, though a limited ability to zoom in or out to view the data were the most mentioned issues.

7) Saved maps can be deleted without the user logging into the system.

8) The site nearly freezes if one searches for nothing in the search box or use single characters to search.

9) Users can create usernames that are blank spaces (space bar hits) only.

10) One can freeze the site by asking it to search several times repeatedly.

11) When zooming excessively in the 2d viewer broken image links begin to appear as the data fails to load.

12) New user email addresses are not validated

13) Clicking repeatedly on the maps can cause the site to freeze

14) The LA-plan user interface has no full screen options. Non-standard size monitors either have a small portion of the screen filled by the UI (when using a larger monitor), or users are forced to scroll to see the entirety of the UI on small screens

15) The Animation Tab does not function: there is no way to input information into the To: or From:

16) In the 2d viewer instructional pop-ups did not appear when the icon was hovered over the buttons.

17) The initial view in both the 3d and 2d viewers is of the entire world. It should focus on LA.

7.4 Current Status and future test plan

Following testing the GUI was improved to address most of the problems identified. Of the cataloged issues there were two different varieties of problems. The first was with technical malfunctions of the GUI; buttons didn't work, data didn't load, etc. The second issue was with usability and user confusion. Technical issues that were presented have all been fixed with one exception: the sites inability to handle certain web browsers. Safari , in particular, does not appear to work with the GUI and evidence suggests that there are compatibility issues between the Apple browser and Arc Server. Smart phone browsers also tended to have either limited functionality or did not work at all. At this time the site is not designed for the usage of smart phones and questions at to the necessity in addressing this issue will be further discussed.

User confusion and usability issues detailed by the testers have been more problematic to address; much of the testers complaints stem from the underlying software used to create the GUI and server. Currently the GUI provides a window for the user into a wide variety of very large and a diverse collection of geospatial data through the use of software designed with strict limitations. The GUI is simply unable to handle very complex or data intensive searches, and is, at times, prone to crashing. Tester's frustration with waiting for the site to load information, or their inability to manipulate data in a way they desire is understandable, but there are few solutions to these problems within the limitations of the software being used.

8. Ongoing work and future plans

This report discussed the technical details of designing and implementing the graphic user interface for sharing geospatial data among the users of the LA-Plan GUI. We discussed the comparison of the three prototypes we have developed and have chosen the third prototype as the working GUI. Two rounds of tests were conducted and valuable feedback from the users was received. To address the existing problems and add new functions is the major focus in the few months remaining before the next project meeting. We aim to add the following functionalities to allow more interaction and data download from the GUI :

(1) Allow visual representation of data by model zone, which can be done with shading, bar graphs, a third dimension (a surface);

(2) Allow download of data at given extent in the format of spreadsheet. Let the model zones be the rows and the data fields corresponding to particular variables as the columns.

(3) Allow the coloring the model zones based on the values of attributes of a model zone.

Another important issue toward sharing the LA-Plan dataset are the security protocals, as some datasets such as Costar dataset are licensed and are only accessible by LA-Plan team. To deal with this issue, we will design hierarchical user management so that users with different permissions can access different sets of data. For service level access (rather than direct raw dataset access), any users who have access to the GUI will be able to operate the dataset through publicly available services however their data download permissions will be restricted.

We expect to make the GUI prototype up and running in early 2012 and we expect one more round of software tests to be needed. The new GUI will need to undergo a new round of testing and editing and software maintenance will likely be costly. This report will be continuously updated as new functions are added.