Proposed Architecture

Proposed Architecture and Implementation

To solve the limitations of the current architecture and implementation, we propose to move the geospatial rendering to the client-side with a more intuitive user interface that will be more responsive to user interaction and we will create an open and flexible server-side architecture that will allow for the addition of multiple data sources.

Architecture

In contrast to the current system, the service oriented architecture (SOA) allows for many services such as Google Maps, Web Feature Services (WFS), and ArcIMS to provide clients with data that is then rendered on the client-side using a Javascript-based tool called OpenLayers(see Figure 2).

OpenLayers is an open source Javascript library for delivering dynamic map content to any web page. It can display map tiles and markers loaded from any source and is supported by the open source community. By abstracting out the server side and setting up a common interface between the client and the server side, any map service or custom coded library can provided content and functionality to the client removing the limitation of waiting for the commercial vender to provide new features and bug fixes. User interaction will also be improved since the clients have more control over rendering and visualization; however, some of the advanced rendering capabilities provided by ArcIMS would have to be custom coded and there will be more computational load on the client. This increased computational load is partially mitigated by the increasing computational power available to common users.

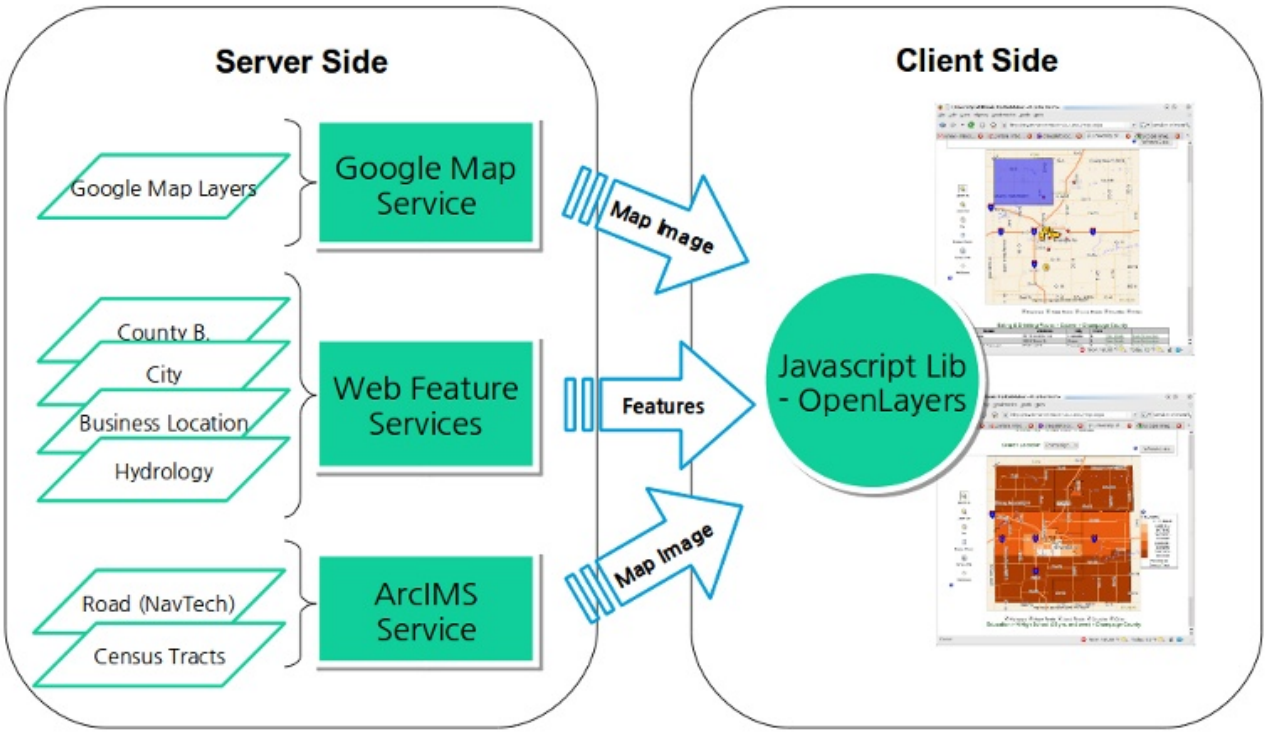

Figure 2. Proposed Architecture

Another advantage of this SOA is the ability to integrate current and future capabilities being developed by NCSA Cyberenvironments and Technologies (CET) division such as high performance computing access, scaling for large data, data analysis, provenance tracking, and metadata integration to deliver a richer user experience and to potentially collaborate on future projects to provide required functionality by the Market Maker Community.

Implementation and Technology

The development of the web mapping component will utilize an agile approach to software development and incorporate user feedback to deliver a high quality product. In the sections that follow, some high-level detail will be provided about the technologies that will be used and how they will be implemented in the component.

Server-Side

The initial server-side implementation of the component will consist of the OGC compliant Web Feature Server (WFS) and the Google Map API providing backend services to the client through an extensible service oriented architecture. These services will be made available to the client via a common API which will allow future technologies to be added as services and made available immediately to the client. The combination of WFS and Google Maps will illustrate the potential of this new architecture by seamlessly integrating data from the Web Feature Server, which will provide geospatial features like points, lines and polygons and services such as the Google Map API and ArcIMS Web Map Server which will provide map images.

The WFS will be implemented using Geoserver. The layers for the component, such as census tracts and business locations, will be defined as feature services under Geoserver.

Client-Side

The capability of handling both geospatial features and map images will be provided by OpenLayers, an open source Javascript library with the ability to combine multiple map services into one map rendering component. OpenLayers supports many popular services including Google Maps, MS Bing Maps, Yahoo Maps, OGC WFS, OGC Web Map Service (WMS), ArcIMS, and MapServer.

Spatial Database

The map component will use the PostgreSQL database with the PostGIS spatial database extension, which provides support for geographic objects, spatial indices, and many common spatial operations that will improve the performance and scalability of the map component and provide advanced query capabilities. PostGIS is free and open source and follows OGC-standard geometry types such as well-known-text (WKT) and well-known-binary (WKB) and OGC-standard spatial operations such as intersect, union, overlap, etc. This will make importing and manipulating data easier. In addition, the PostGIS spatial database is a widely used open source project that is compatible with numerous GIS applications and libraries (e.g. GeoTools, MapServer, Grass, etc).

Deliverables

By the end of the grant period the team will deliver a software package containing the above described functionality, including relevant source code. In addition to the package itself, help documentation and tutorials for executing the software will be delivered. For a time period of at least until the grant period ends, the team will provide additional online resources: a source code repository containing the latest source code, a publicly available bug tracking system, a daily build server containing the most recent builds, a web site containing up to date links to resources, and a publicly available data repository containing the appropriate sample datasets.

The team will also deliver a set of virtual machines configured as follows:

1. A virtual machine configured to run the PostGIS database as described above.
2. A virtual machine configured to run a web server with the web mapping component described above.

These virtual machines will be properly tested and configured to perform the tasks described in this document.

The component will consist of three major components:

1. Map
2. Query Builder
3. Search Results Table as shown in Figure 3.

Map

This component will be implemented by using OpenLayers toolkit and will display the selected map layers overlaid. For this project, we will support at least two layers extracted from the current Market Maker database, Illinois census tracts and Illinois business locations. In addition, background layers from Google Maps will also be supported. The map component will show detailed information for each geospatial entity by clicking on or hovering the mouse over the entity. A map legend and navigation bar will also be provided.

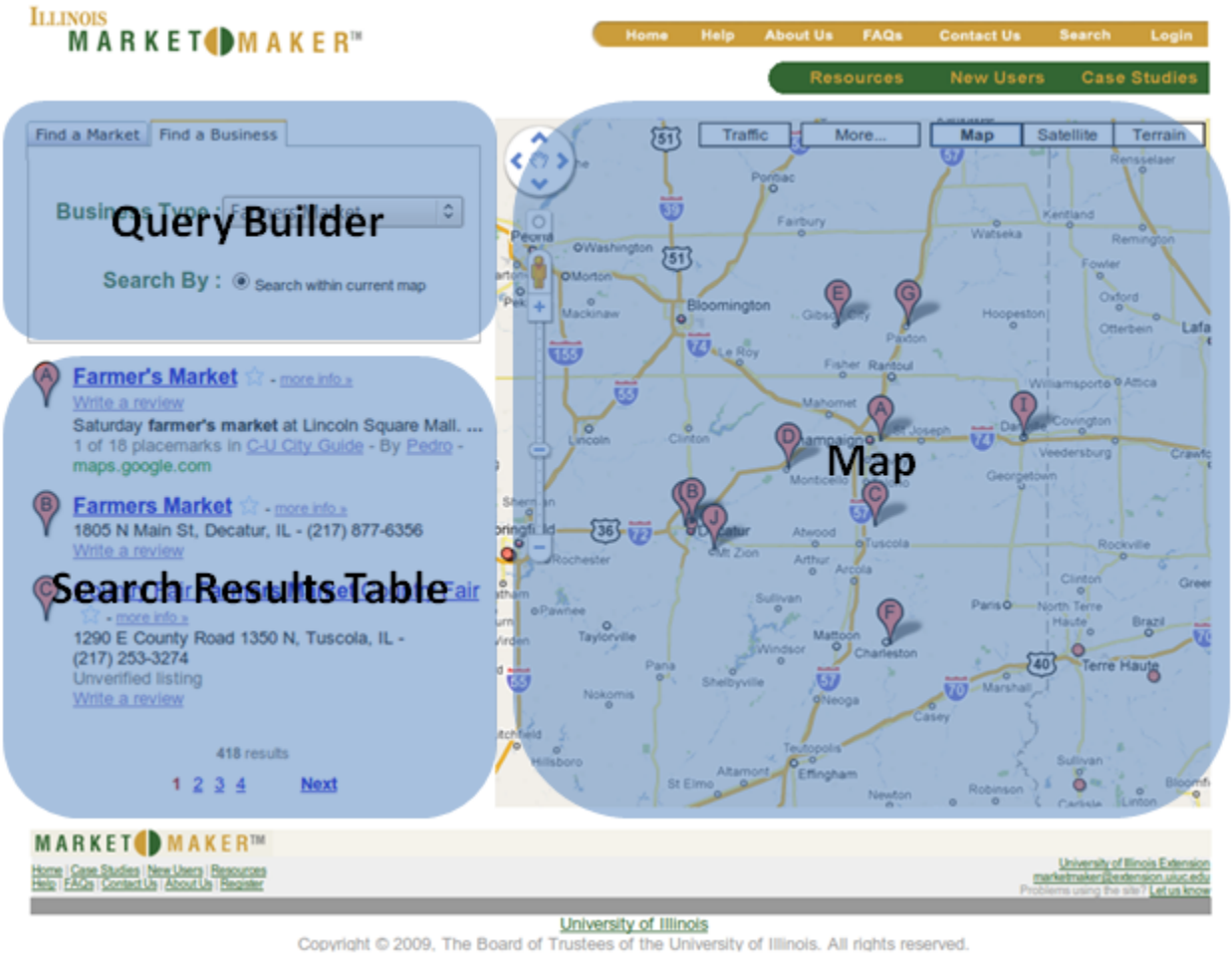

Figure 3. Mock-up user interface and major parts

Query Builder

Users will use the query builder to define and refine their criteria for the information they want to be displayed on the map and table (or list). As the user changes their criteria, the other components will be updated automatically.

Search Results Table

The results from the query builder will also be displayed in a search results table. This will provide a concise view of the result of the query.