"LOCATE, VIEW, MAP": DENVER'S MULTI-AGENCY INTEGRATED ARCIMS INTRANET SOLUTION

Allan Glen, Senior GIS Developer, City and County of Denver, Colorado USA

Abstract

Seamless access to the City and County of Denver's GIS resources by Denver personnel is essential to inter-agency coordination and decision-making. "Locate, View, Map" (LVM) is Denver's ASP/ArcIMS solution for delivering integrated GIS information to the desktops of all Denver staff using a standard web browser (no browser plug-ins or applets required). More than 400[1] GIS layers from more than 55 agencies are available. This city-wide GIS solution allows for centralized maintenance, support, and access to multi-agency GIS information. The extensible component-based architecture is critical to the success of this integrated solution. To support individual agency requirements, the application provides a rich plug-in API, allowing rapid development of custom tools, mapping capabilities, and user interfaces. Application integration interfaces are also provided, allowing interaction with other enterprise applications and allowing non-GIS applications to leverage existing GIS capabilities. The architecture and the lessons learned in creating this intranet GIS solution will be addressed.

1 Introduction

The City and County of Denver's Technology Services Geographic Information Systems group (DenverGIS) provides corporate GIS services to all city agencies and departments and to the public. Services provided include geographic data management, application development and data distribution. Providing city-wide access to integrated GIS information and applications is critical to support daily decision-making and to promote inter-agency communication and collaboration.

"Locate, View, Map" (LVM) is Denver's intranet web-based application that delivers integrated GIS information from the corporate GIS database to the desktops of all Denver employees (Figure 1). LVM uses ESRI ArcIMS and ArcSDE and is developed using Microsoft Active Server Pages (ASP). LVM provides a single interface to all data managed by DenverGIS and is the primary GIS application used by Denver personnel.

LVM is accessed using a standard web browser[2] and no additional browser plug-ins are required.

2 Goals

Denver's city and county are combined into a single jurisdiction and is comprised of a diverse group of offices, departments and agencies. For simplicity, these organizational groups will be referred to as 'agencies' throughout this paper.

[1] As of May 2005, LVM provides access to more than 450 individual ArcSDE data layers, an increase of 50 layers since the submittal of the paper abstract in February 2005.

[2] LVM supports W3C DOM Level 1 compliant web browsers. Tested browsers include Microsoft Internet Explorer 5+ and Mozilla Firefox.

Denver's agencies represent a diverse set of functional areas (1), many of which can benefit significantly from the application of GIS technology. Several of these agencies are producers of geographic data (Parcels, streets, zoning, etc.) and many more are consumers of the GIS resource.

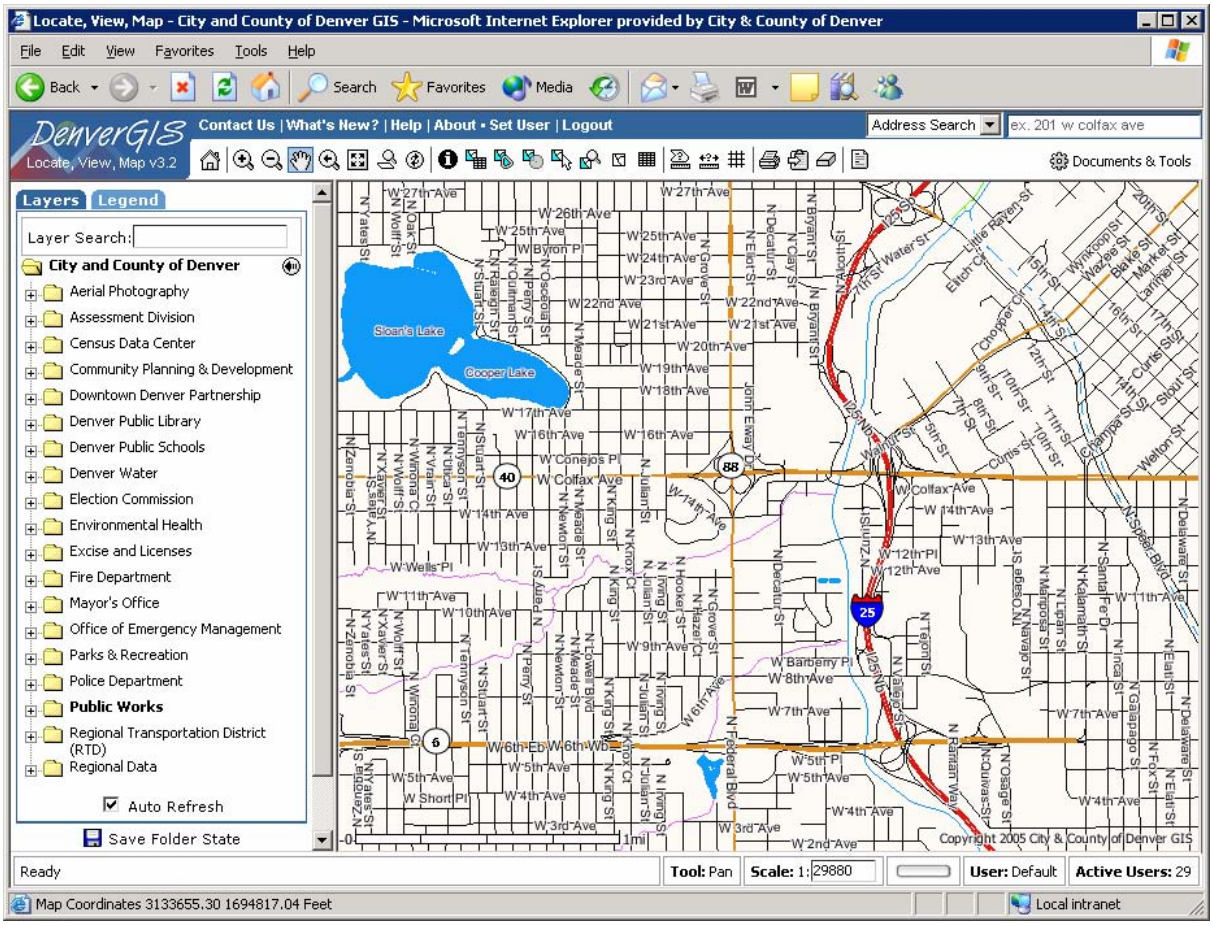


Figure 1: "Locate, View, Map" (LVM), Denver's intranet GIS application for providing GIS mapping, viewing and reporting capabilities to all City and County of Denver agencies and departments.

DenverGIS manages the corporate GIS and provides centralized data management and distribution of geographic data. A core goal of DenverGIS' mission is "to provide accurate, up-to-date, and readily available geographic information and databases to all City agencies" and to "[use] technology to build greater collaboration between agencies" (2).

Providing GIS services to a diverse set of agencies presents a two-fold challenge – it is valuable to be able to explore inter-agency relationships (ex. comparing property values from the Assessor's office with crime statistics) but it is also important to provide the specific GIS capabilities required to support the unique goals and workflows of each agency.

To meet this challenge, DenverGIS envisioned a web-based application that would allow data from all city agencies to be viewed concurrently and would also provide custom extension capabilities to support specific agency needs. All users would then use the same application, but would have access to specific components to support agency-specific functions. The application would also have the ability to be integrated with other enterprise applications to allow existing applications to leverage the mapping capabilities provided through LVM.

DenverGIS preferred this strategy over creating specific GIS applications for each agency for the following reasons:

- Centralized maintenance of application, all enhancements and fixes are available to all agencies, multiple applications do not need to be maintained
- Centralized hosting and management of application and supporting systems (consolidated servers, databases, storage and clustering)
- Single user-interface, single core training requirements, customized training for specific agency tools only
- All agency data is integrated in a single application avoiding additional data management or replication of data
- All data can be viewed concurrently with all other data promoting agency collaboration and communications
- Centralized marketing, single web address for users to remember
- Centralized authentication and access control to specific capabilities and data layers

3 Systems Architecture Overview

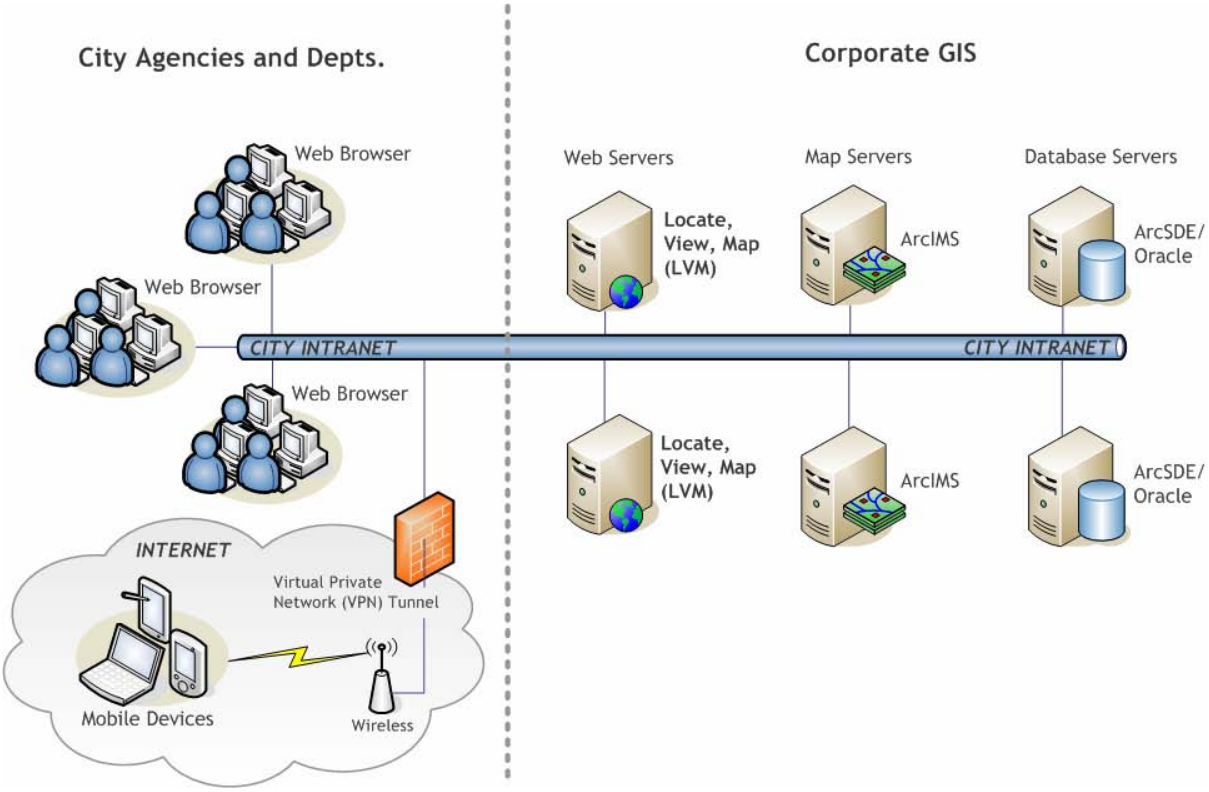


Figure 2. Overview of the DenverGIS data distribution environment showing the location of the LVM application and the ArcIMS and ArcSDE/Oracle servers. Users from all City agencies and departments access LVM and can view data from other city agencies. Data is published to the data distribution database servers from the data maintenance environment (not illustrated). The data maintenance environment is where all agency data updates are made before being published to the data distribution environment.

LVM is a web-based application developed using ASP, JavaScript, HTML, and XML. LVM is deployed on clustered IIS servers, providing redundancy and failover in the web server tier. Session state is maintained in the web browser, allowing for fully transparent session failover and load-balancing.

The LVM application communicates directly with ArcIMS on the map server tier. Two ArcIMS servers are utilized to provide redundancy and failover. DenverGIS designed a custom ArcIMS monitoring tool that is used by LVM to enable ArcIMS failover in the event of an ArcIMS failure on the primary ArcIMS server. Failover occurs transparently to the user and can occur in mid-session with no impact on the state of the current user request. Detailed ArcIMS monitoring and failover has been very important to ensure success of this city-wide solution and allows for high levels of uptime. This configuration also supports rolling updates to map service content without downtime.

The database tier utilizes ArcSDE on Oracle 9i. The database tier is read-only and also supports data distribution to desktop clients (ArcMap, ArcView etc.). The enterprise ArcSDE distribution database houses data layers from all agencies. The distribution database is replicated to support failover in the event of a failure on the primary database server and to support rolling updates to data layers with no application downtime.

4 Application Architecture Overview

LVM uses a component-based architecture where all tools, dialogs, and map operations (referred to as handlers) are implemented as components that tie into a core application framework (Figure 3).

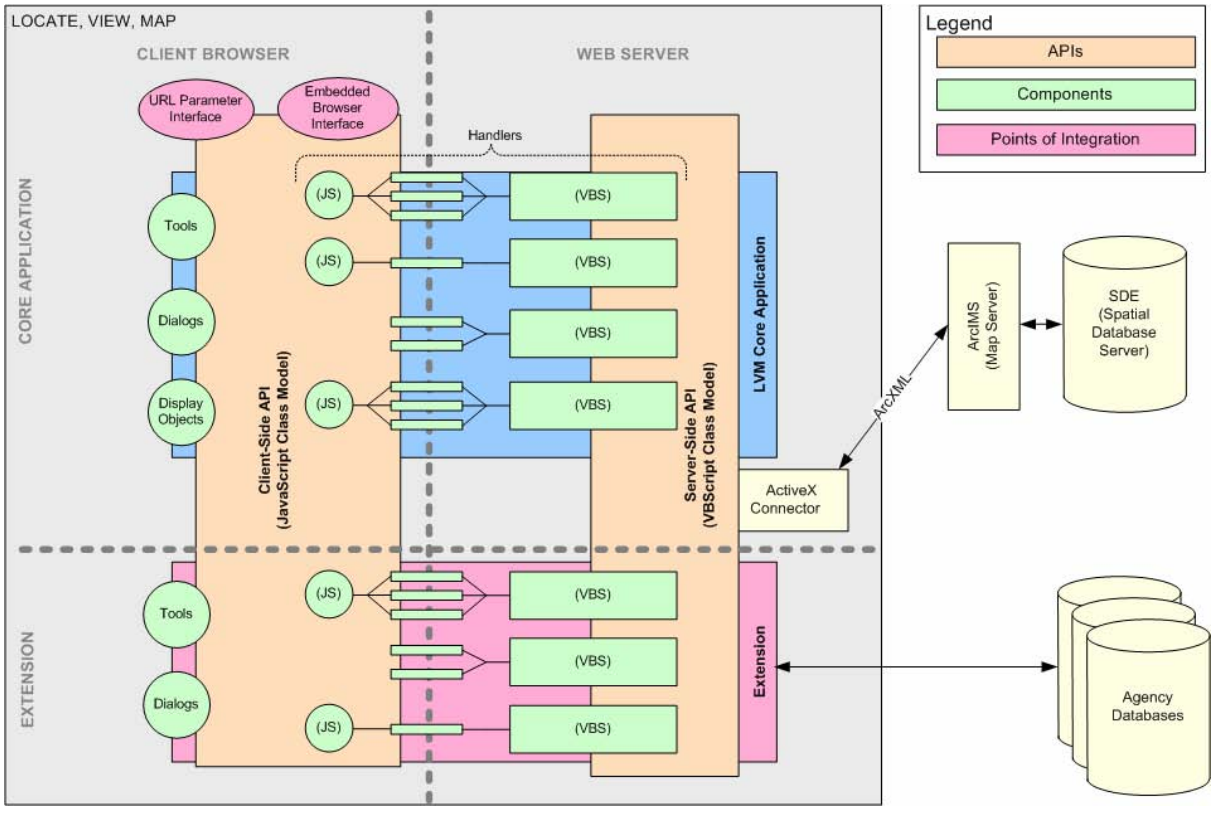


Figure 3: LVM architecture showing application components, application programming interfaces (APIs) and points of integration.

4.1 Core Framework

The core framework represents the core of the application, or the "glue" that holds the application components together. The framework represents the general application layout (browser frame layout, status bar, table of contents, etc.), manages all core application functions (startup tasks, session management, clustering, database access, etc.), and includes all core client and server side libraries.

4.2 Application Programming Interfaces

The client-side and server-side libraries are exposed to components via application programming interfaces (APIs). The client-side API is implemented using JavaScript classes and provides access to all client-side (web browser) functionality (window management, map control, tool handling, map events, etc.). The server-side API is implemented using ASP classes and provides access to all server-side functionality (ArcIMS requests, database access, map operations, etc.). The APIs are used by all application components and can be used to build custom extensions to support a wide variety of agency-specific functionality (Refer to section 4.4, Application Extensions).

4.3 Application Components

Application components include tools, dialogs, display objects, and handlers. The following is a summary of the component types and their general purpose.

Tools include both buttons and tools and are made available to the user in application button bars. A tool can either have interaction with the map or simply be a button that triggers other events (refresh map, open dialog, etc.). A tool component implements the defined tool interface to implement the tool functionality. All tools are implemented in this manner and each tool is a stand-alone component.

Dialogs are browser popup windows that contain specific functionality. Dialogs are controlled through the client-side API and are managed by the core framework. Dialogs are implemented via a dialog interface and can contain tools and other components as needed. Dialogs interact with the main application window via the client-side API and can also access server-side functionality during page requests.

Display objects are items that are used on the map area to provide display interactivity. This includes zoom boxes, interactive polygon drawing (to measure area, digitize polygons, etc.) and interactive line drawing.

Handlers are components that interface with the map and provide specific mapping functionality (zoom to previous, buffering, layer visibility, selection set rendering, etc.). Handlers process parameters submitted by a map request and collectively produce the output map. Handlers can be accessed by both the client-side and server-side APIs and are also used to exchange application state information between the client and the server. A handler component implements a defined handler interface.

The above component types are used together to provide all application functionality. For example, a buffering tool may require a dialog to collect buffering parameters, a tool to select the area to buffer, and a handler to process the buffer. The handler would select the features and render the selection on the map. Another dialog could be used to view the attributes of the selected features. The APIs allow the components to work together to complete such tasks.

4.4 Application Extensions

The component architecture described above also supports the ability to build custom application extensions.

A key goal of the application design is to allow for specific agency extensions to be created that extend the main application, thereby providing agencies with the tools that they require within a single, consistent application environment. The application extension interface enables extensions to be created that are architecturally similar to the main application but are implemented and managed as self-contained collections of components. Extensions use the client-side and server-side APIs and can implement all component types (excluding display objects which can only be implemented in the main application).

Application extensions can have user interfaces and can either be docked in the main application window (Figure 4, Figure 5) or appear in a popup window. Extensions can contain tool bars, maps, dialogs and other components that interact with the main application.

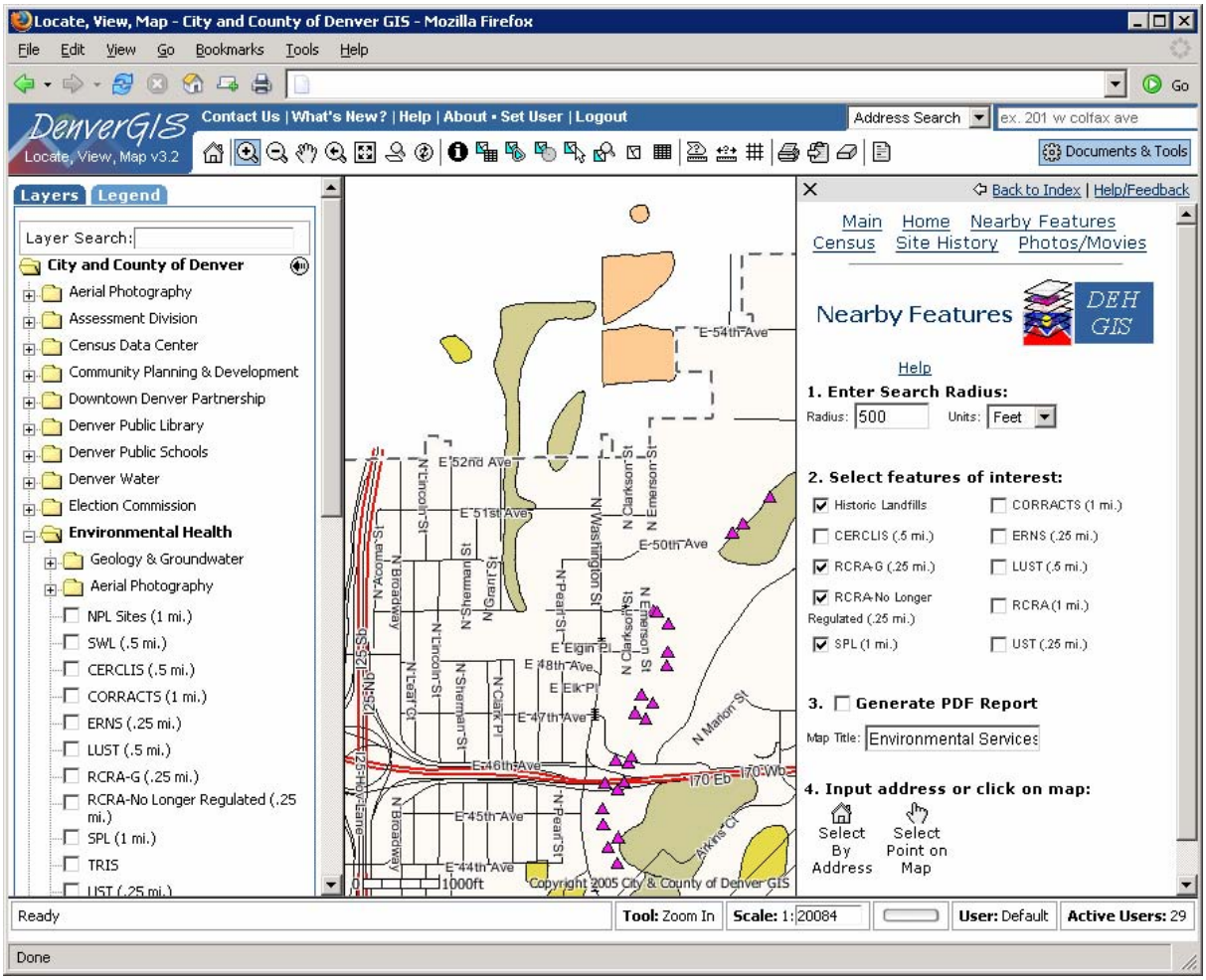


Figure 4: The LVM application showing a Department of Environmental Health extension docked to the right of the map area. This extension provides environmental site assessment tools including map-based reporting of environmental features, site history, census information, and access to site photos and media.

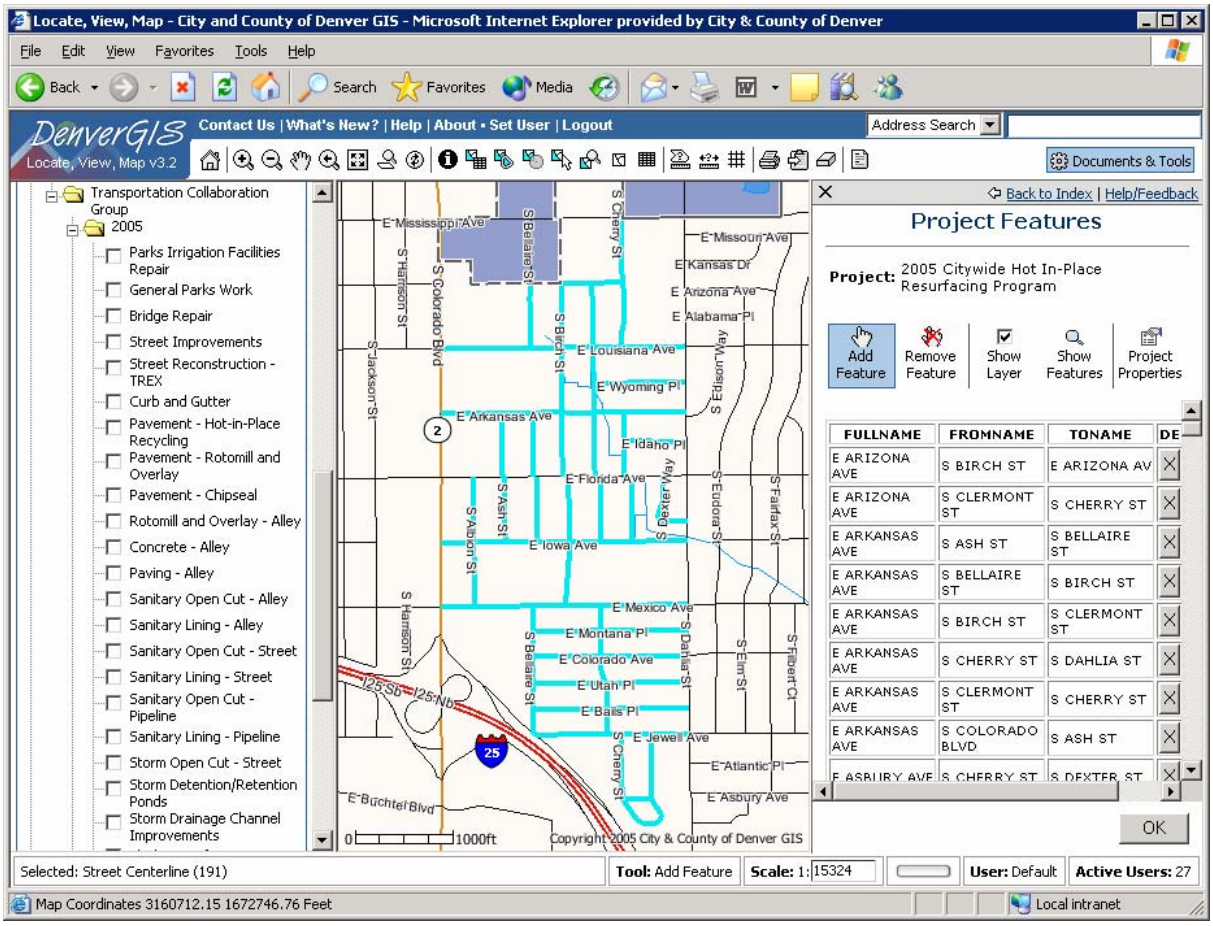


Figure 5: The LVM application showing a Transportation Collaboration Group extension docked to the right of the map area. This extension allows agencies to collaborate on transportation projects by selecting features that are to be included in projects. This example shows a set of streets that are planned to receive hot in-place resurfacing in 2005.

5 Design Challenges

5.1 Layer Organization

Delivering hundreds of layers of multi-agency GIS data in a single application presents several user interface challenges.

In early versions, LVM used a single-level folder structure to group layers by agency (Figure 6). This was found to be successful with less than a hundred layers but soon became cumbersome as the number of layers available increased.

To improve layer organization, LVM now includes an explorer style multi-level folder structure (Figure 7). This enables grouping of layers to improve organization within agency folders. This structure has been found to be very successful when users are familiar with the ownership of data within the city. It has been found that users that are unfamiliar with the ownership of GIS data tend to have difficulty finding particular data layers (For example, "Where are streams and lakes located?").

To help mitigate this, a quick search option was added that allows users to quickly find layers without having to navigate the folder structure. Additional methods for locating and managing

'favorite' data layers are being considered to further improve the ability to quickly locate information.

The ability to save to open folders is also provided so that the open folders remain open when the user returns in a future session. The ability to save the entire session (including layers that are visible) is planned for an upcoming version.

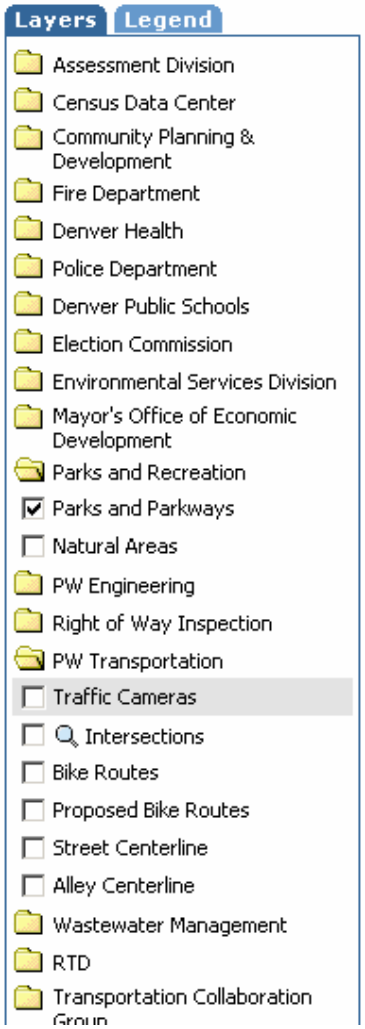


Figure 6: Locate, View, Map version 1.4 used a single-level folder structure. All available layers for an agency were listed in a single folder.

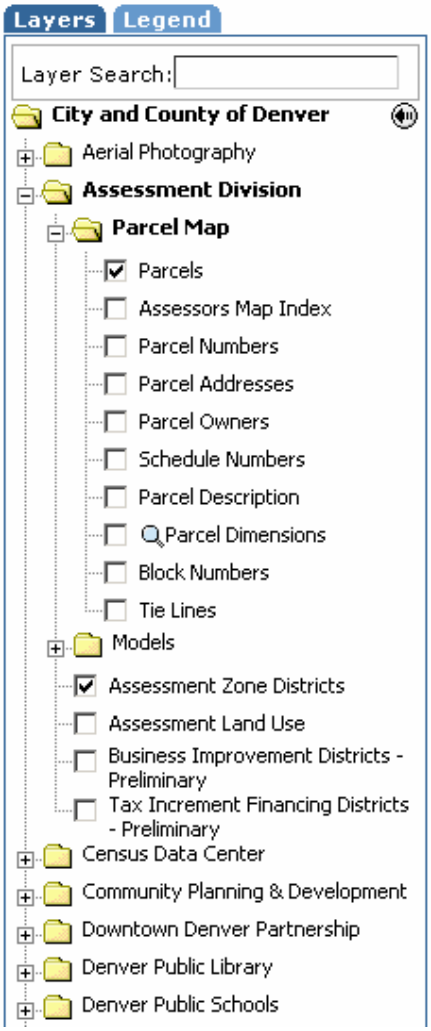


Figure 7: Locate, View, Map version 2.x added a multi-level folder structure. Layers are grouped within folders for improved navigation. The "Layer Search" function allows layers to be found quickly by searching the layer name and keywords. Bold folder names indicate that there are visible layers within the folder.

5.2 Access Control

When using a single application to publish data from multiple groups, access control is required to restrict access to information that is sensitive or is not ready to be published city-wide. Access control was implemented in an early version of LVM to restrict viewing of data layers to specific user groups and agencies. Restricted layers are not available in the layer list and are unable to be viewed or queried. Once a user logs in to the application, the layers that they can access are made available in the layer list.

At present, access control is only implemented for ArcIMS data layers. Extending access control to all component types would allow for the application to be further adapted to specific agency needs and is planned for a future version. For example, a particular user group would have access to specific tools, dialogs, handlers and extensions to support specific business needs. This would simplify the user interface for general users but provide the necessary components to support individual agency needs.

5.3 Performance

LVM uses a single ArcIMS map service to enable all data to be displayed in queried concurrently.

Early versions of LVM utilized the ArcIMS ActiveX Connector objects to manage all map requests. When the number of available layers in the map service increased into the hundreds, performance issues were identified. This was primarily found to be related to the initialization of the Map object (Map.InitMap) at the beginning of each map request. The Map.InitMap method of the ActiveX Connector was requiring approximately 3-4 seconds when using approximately 200 data layers.

To improve application performance, LVM was modified to use only the ArcIMSConnector object to send and receive ArcXML. This eliminated the additional 3-4 seconds of time required to initialize the Map object. This required some redesign of the application but the migration to using only ArcXML resulted in significant performance improvements.

A caching component was also designed to further increase performance. The caching mechanism stores map service information in memory on the web server to avoid making repeated requests to ArcIMS. Cache updating is automated, allowing changes to be made to the map service without any manual updating required.

6 Enterprise Application Integration

Many agencies in the City and County of Denver benefit from the integration of GIS capabilities into their business processes and workflows. Accessing LVM as a standalone application allows GIS to be used along-side other business applications but requires users to manage the state of both applications independently (ex. Copy the query from one application to another so that same street can be selected in both applications). To make this process more efficient, LVM provides a set of integration interfaces (Refer to Figure 3) that allow other applications to control and interact with the GIS application.

6.1.1 URL Parameter Interface

All map processing and interaction is implemented by handlers (Refer to Section 4.3, Application Components). Handlers receive parameters during a map request and are responsible for controlling the state of the map (extents, selections, buffers, queries, markup etc.). The core framework allows handler parameters to be passed via the application startup URL to perform actions on application startup. This enables external applications to pass parameters and control the state of the map (See example, Figure 8). This provides one-way integration into LVM from external applications and provides a simple technique for linking non-GIS applications to interactive GIS maps.

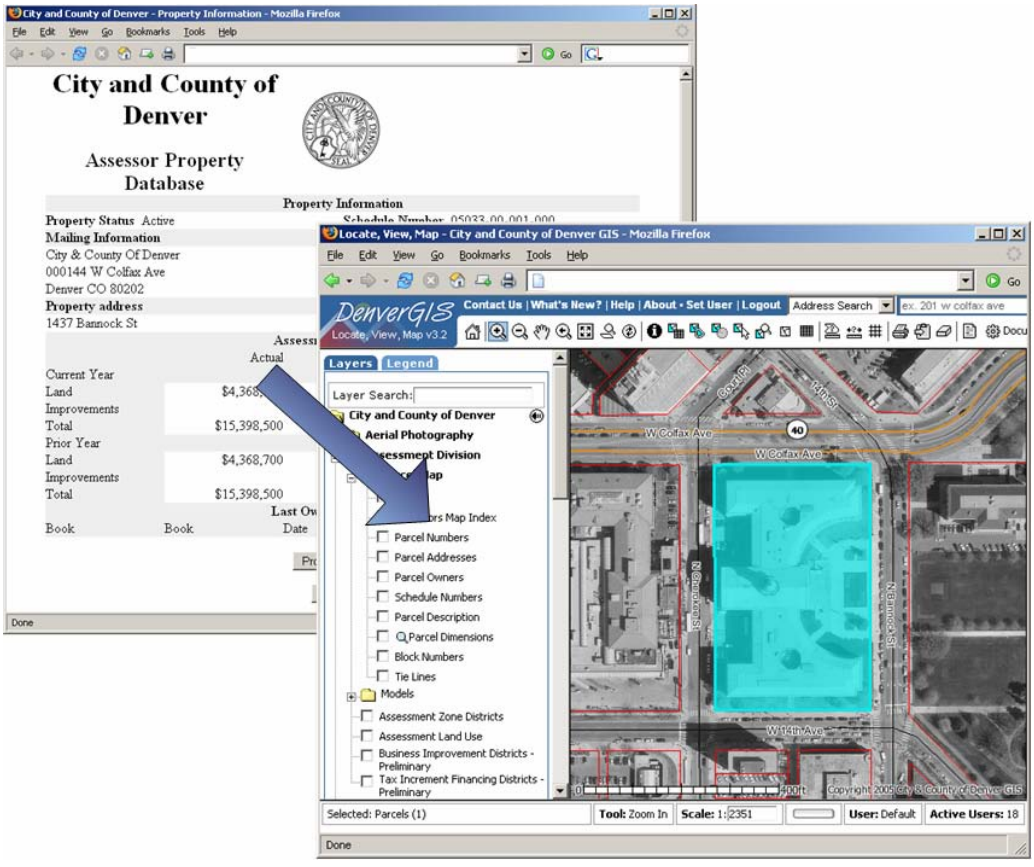
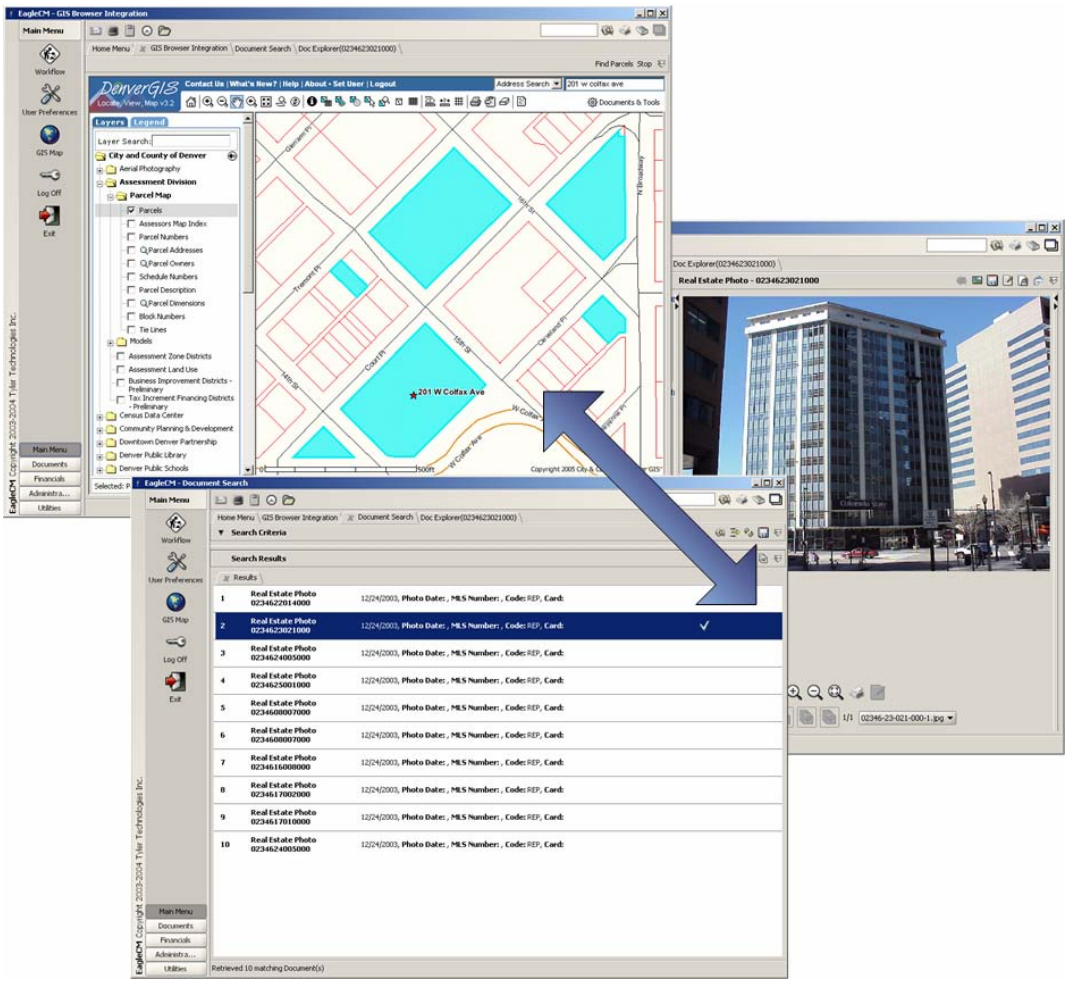


Figure 8: Example of URL Parameter Interface integration from the Assessor Property Database web application to LVM. The link from the Assessor web application selects the property, turns on parcels and aerial photos, and zooms to the selected property.

6.1.2 Embedded Browser Interface

LVM can be integrated into desktop applications using an embedded browser control (See example, Figure 9). The full client-side API is exposed to the embedded browser and allows LVM to be controlled programmatically from the host application (zoom, pan, show layers, query, open dialogs, print maps, etc.). This enables two-way communication and supports XML data exchange between applications. Browser integration allows applications to leverage LVM capabilities and take advantage of future updates without the costs associated with maintaining and supporting additional applications.



EagleCM Copyright 2003-2005 Tyler Technologies, Inc.

Figure 9: Example of Embedded Browser integration with the Assessor Document Management System (DMS). The integration supports two-way query of properties to support document management and workflow processing. Features can be queried either in LVM or in the DMS and selection information can be passed between the applications via XML messages. The LVM application is integrated into the DMS and appears on a tab within the DMS. In the screenshots above, a set of documents have been retrieved in the DMS and the user has requested to see the location of a document on the map. The map is opened on a tab within the DMS and shows the property that the document is associated with. The user can also select features on the map (using queries, buffers, interactive selections etc.) and then search the DMS for documents associated with the selected properties.

The Assessor DMS is a Java application (EagleCM) developed by Tyler Technologies, Inc.

6.1.3 Application Extension Interface

Application extensions (Refer to Section 4.4, Application Extensions) provide a means for closely integrating external systems with LVM. Application extensions are implemented like sub-applications to LVM and can use ASP to interact with other systems and data sources (Databases, application servers, etc.). By using the LVM APIs, extensions have the ability to integrate external data sources into the mapping environment.

For example, the Transportation Collaboration Group (TCG) extension (Refer to Figure 5) stores project information in an external database. This database includes references to all of the GIS features that are included in each project. The LVM APIs are used for data collection (to spatially identify the features that are to be included in the project) and for data display (to show project features on the map). By leveraging the extension interface, users are provided with an interactive means of identifying project locations. All capabilities of the core application are automatically available to TCG users (printing, querying, buffering, spatial overlay, etc.) and an additional stand-alone application was not required.

7 Conclusion

LVM is the primary application used by internal Denver staff for accessing GIS information. By providing citywide access to multi-agency data layers, LVM allows Denver's diverse agencies to explore inter-agency relationships and encourages increased communication and collaboration.

To leverage the value of the central GIS spatial database, layers from all City agencies are published in LVM. Providing access to a large number of layers in a single application presents several user interface challenges. Using a multi-level folder structure and search interface to the layer list improves users' ability to locate information. Users that are familiar with the functional responsibilities of Denver's agencies are most successful at quickly locating data layers and additional improvements can be made to improve efficiency for new users.

Using an extensible architecture, LVM also supports the development of custom agency extensions to meet specific business needs. The use of extensions has eliminated the development of duplicate functionality and has allowed applications to be developed at significant cost savings. Extensions gain immediate functionality from the application environment (printing, spatial operations, etc.) and also gain all core functionality added in application updates. Extensions allow for centralized support and maintenance, resulting in significant cost savings.

Application integration interfaces allow other enterprise applications to take advantage of the GIS capabilities provided in LVM. By providing these interfaces, the need for additional stand-alone GIS applications is diminished. Reducing the number of GIS applications that are required to be developed and supported results in cost savings to the City and allows development efforts to be focused on features that can immediately benefit all City personnel.

LVM is accessed by over 600 unique users per month and serves over 10,000 page requests per working day (Mon.-Fri.). Over 1GB of ArcIMS map images and printable maps (PDF format) are transferred daily.

8 Acknowledgements

LVM is developed and maintained by the City and County of Denver Technology Services Geographic Information Systems group (DenverGIS). DenverGIS is supervised by David Luhan.

Many of the features available in LVM are inspired by feedback from City and County of Denver staff.

For more information about "Locate, View, Map" please contact:

David Luhan
City and County of Denver
denvergis@ci.denver.co.us

9 References

(1) City and County of Denver Customer Information Services "City and County of Denver Government" *DenverGov.org* Dec. 2004. May 2005. <http://www.denvergov.org/images/orgchart/City_Org_Chart_2005.pdf>

(2) City and County of Denver Budget Management Office "2005 Budget – City and County of Denver" *DenverGov.org* Oct. 2004. May 2005. <http://www.denvergov.org/admin/template3/forms/budget_book_05.pdf> p. 250