FINAL

Flood DSS Memorandum

To: Carolyn Fritz, Ray Alvarado, Susan Lesovsky – CWCB
From: Steve Malers, Amnon Nevo, Ian Schneider – Riverside
Subject: Evaluation of Alternative Technologies for Flood DSS Web Mapping Application
Date: 9/14/2009

1. Introduction

The purpose of this memo is to summarize the evaluation of alternative technologies for the Flood DSS web mapping application, and recommend technologies for implementation.

This activity reflects the fact that although the Flood DSS Map Viewer prototype uses ESRI's ArcIMS software, this software is being phased out by ESRI (see http://www.esri.com/software/arcgis/arcims/index.html: "…Note: ESRI's development of ArcIMS is now limited. Our server GIS development efforts are devoted primarily to ArcGIS Server.…").

See the last section for a summary of recommendations.

2. Background

Background information is provided as a foundation for evaluation and recommendations.

2.1 Web Mapping Concepts

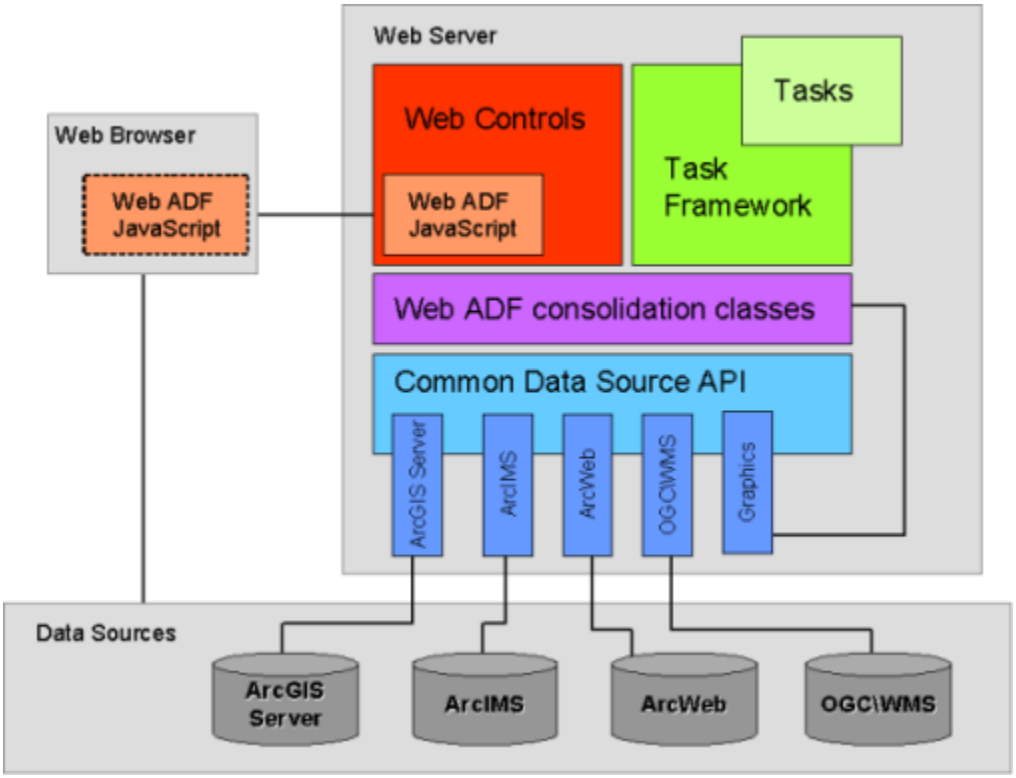
A web mapping application, such as the Flood DSS Map Viewer, is a software program that is accessible over a network (the Intranet or the Internet), and can provide dynamic, interactive spatial data, potentially from multiple sources, possibly with querying, analytical, and editing capabilities. The advantage of web applications compared to desktop applications is that software is maintained and updated on a central server and users only need a web browser to access the application. This also allows users to access the software from any computer with Internet access.

As shown in the figure below, technologies used to implement web mapping applications generally involve (from the right side): (i) spatial data storage and serving, (ii) server-side software (web server(s), map server(s), and geoprocessing), and (iii) client-side software (software with which users interact, such as a web browser).

Figure 1. Components of a Typical Web Mapping System

Commercial and free technologies can be used to implement each of the components of a web mapping system. Each implementation has specific requirements that can best be met by a specific combination of technologies. The selection of the right combination of technologies for the Flood DSS requires understanding the functionality, architecture, compatibility, and relative advantages of each technology in terms of issues such as performance and cost. For example, commercial tools have a high initial license cost but can be cost-effective in terms of development and implementation. In contrast, open source and other free solutions have no license cost but may require development and integration to achieve a solution.

All major web mapping technologies utilize server and client software to achieve a functional mapping solution. The communication between client and server components typically uses web services, where data is packaged in XML or some other format. Software toolkits are provided to request data, process responses, and visualize the data. For example, the ESRI Web Application Development Framework (Web ADF, see: http://edndoc.esri.com/arcobjects/9.2/NET_Server_Doc/developer/ADF/adf_overview.htm) utilizes client and server side software, in particular client-side JavaScript that communicates with the server and manipulates the map displays. The following figure illustrates this web mapping solution (taken from the above link):



2.2 Current and Future State of Colorado IT Infrastructure

Riverside conducted a survey of GIT infrastructure at DNR in February 2008. The map serving technologies used at that time were MapGuide, MapServer, ArcIMS, and ArcGIS Server. Of these, only ArcGIS Server was hosted at the DNR IT group, which was identified as the host of the Flood DSS. The other servers were hosted by individual divisions within DNR. In recent discussions with the State project team it was confirmed that ArcGIS Server is the preferred serving technology. This makes ArcSDE a preferred data storage technology because it is part of ArcGIS Server.

A summary of recent discussions related to IT ArcGIS Server support is as follows:

- DNR – in particular for DNRWeb, which is visible to the public
    - According to Tom Bergman (email from 6/29/09) the DNR currently has a production copy of ArcGIS Server Advanced Enterprise 9.2 installed. This product level is the most feature-rich option available (no limitations). IIS version 6 is running on a Windows 2003 server, with .NET 3.5.
    - Deb Bell (phone call, 7/22/09) indicated that that SDE is working with SQL 2000 (Tom mentioned SQL 2005). Image Server 9.3 is running stand-alone on a server other than the ArcGIS Server. The SQL 2000 database may need to be moved to the DNR cluster (SQL 2005) by the State.
    - Carolyn Fritz (email from 7/6/09) indicated that the State may have access to ArcGIS Server 9.3 software as part of a USGS grant. Bill Martin at the State Land Board is evaluating how the State can update to 9.3 within the constraints of the USGS grant. No estimate was given for when 9.3 would be installed. **However, the CWCB has indicated that the Flood DSS should be based on ArcGIS Server 9.3 (Carolyn Fritz email 8/25/09).**
    - Discussions with the State indicated that they have no GIS development server. Therefore, the FloodDSS implementation will need to be tested at Riverside and "dropped in" with as little disruption of the production server as possible.
- OIT – primary purpose is to support internal operations but is envisioning public access
    - hopes to have ArcGIS Server 9.3.1 (the latest version) by September.
    - **As indicated above, the CWCB has indicated that the Flood DSS should be based on ArcGIS Server 9.3 (Carolyn Fritz email 8/25/09), and be hosted on a DNR server.**

Network security and access to machines will require coordination with CWCB and DNR IT staff. Initial discussions indicate that it would be best to perform dynamic data processing (e.g., for real-time streamflow and SNODAS data) on a CWCB computer and then push the data to other servers. Access must be granted to LaserFiche and databases such as HydroBase. These specific issues will be resolved during development and deployment and should be evaluated by the State to determine if they impact hosting options.

2.3 Flood DSS Needs

Interviews with CWCB internal staff and potential external users, in addition to considering categories of data for the Flood DSS have identified the following themes and user groups.

System Theme	Anticipated Users
Floodplain boundaries	Floodplain managers CWCB (MapMod, etc) Public (limited access)
Historical flood information	Emergency managers Floodplain managers CWCB Public
Real-time data	Emergency managers CWCB (flood outlook) Public (limited access)
Community information	CWCB (community assistance) Communities
Restoration	CWCB (restoration)
Weather modification	CWCB (weather modification)

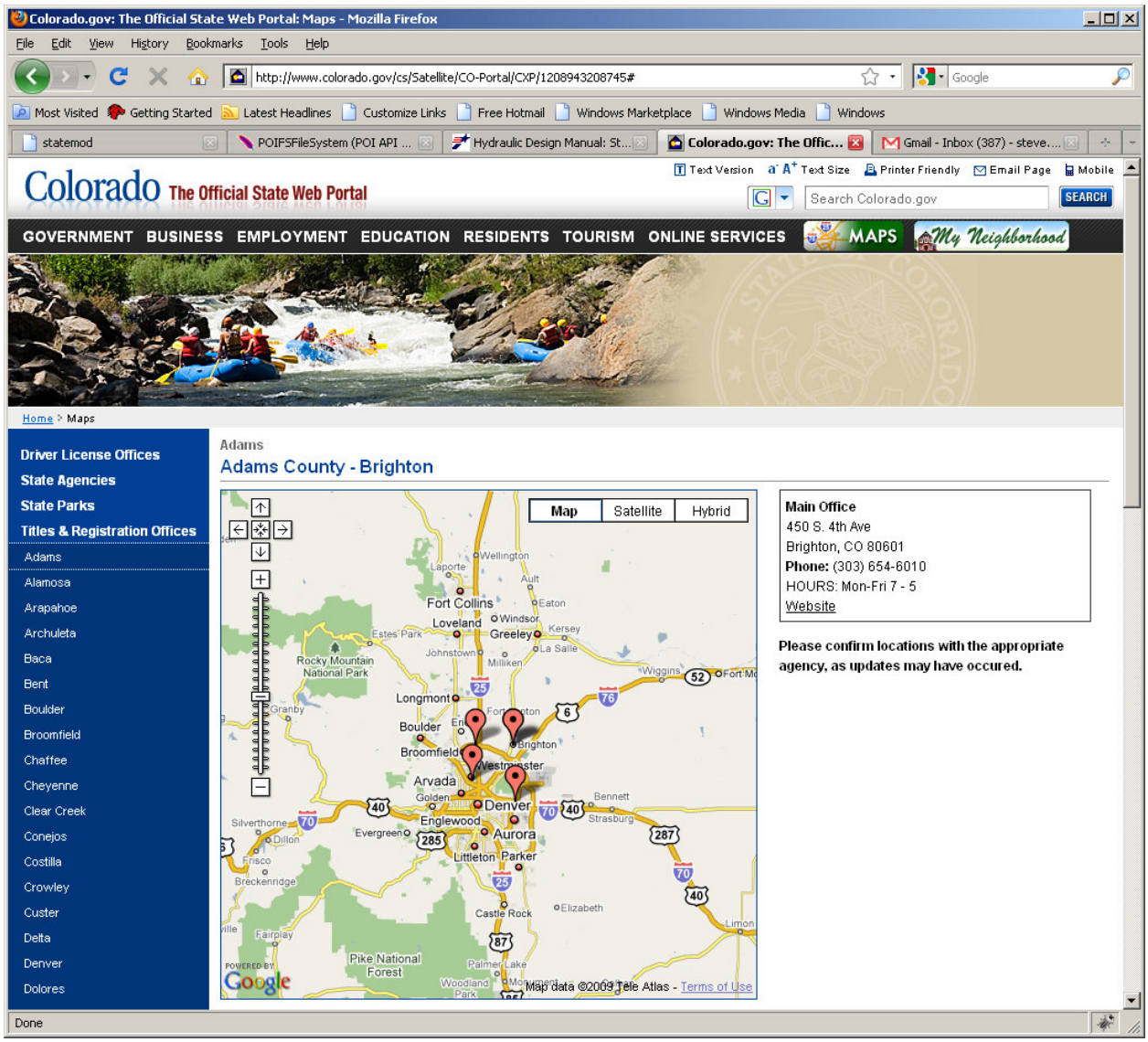
The colored text highlights how a user group may have an interest in multiple themes. However, it is not totally clear how data in multiple themes will be used together, although some interconnected use is likely (e.g., emergency managers may be interested in where historical floods have occurred and whether current real-time conditions might produce similar results).

The FEMA-related maps may require a separate site, due to the complexity of data layers, amount of data, impact of agency data providers, review cycle, etc. This decision should be further evaluated as data collection and user interviews are completed.

It is difficult to know the public's expectations and ability to use more complex tools; therefore a conservative assumption is that displays meant for the public should be as simple as possible and be consistent with other tools that they may have used.

Based on an evaluation from user interviews, personal experience, and a study of web design trends, it is unlikely that a single "power user" mapping web site will meet the expectations of all users. It may be able to meet their needs; however, the usability of such a site may not meet their expectations and therefore the end product may not be utilized.

Consequently, it is recommended that several simple web pages be implemented using tools such as Google Maps. A simple example is as shown in the following figure (see http://www.colorado.gov/ and click on MAPS graphic at the top – this site is hosted by OIT).



Example of Simple Web Mapping Application Using Google Maps

Simple maps can utilize map layers from Google, ArcGIS Server (such as the server that will be configured with Flood DSS data), and simple files such as KML.

Power user sites will need to utilize tools such as OpenLayers or ESRI APIs and are discussed in the technology recommendations sections below.

3. System Design Considerations

The selection of technologies must consider the design aspects of the system because the technologies need to be compatible with the design and meet technical requirements. Although details of the design for specific components have not been finalized, a few overarching design concepts have been determined:

- Utilize existing production tools as much as possible – this includes available commercial software from ESRI, CDSS software including TSTool (or components thereof), and open source software (if appropriate). This minimizes development costs.
- Minimize the dependence of components on specific features of other components – for example, exchange data using standard protocols and formats. This minimizes the impacts that replacing a component has on other components.
- Utilize a services based architecture, in particular using a Resources Oriented Architecture (ROA) and RESTful (Representational State Transfer) approaches to access resources in the system. This facilitates exchanging system resources between components using the web, and facilitates testing and troubleshooting.

Appendix A describes the ROA in more detail and will be utilized as the system is developed. An ROA approach is independent of specific technologies and builds upon the general internet and web resource concepts.

4. Evaluation of Technologies

Technologies that are suitable for the Flood DSS mapping website were evaluated with consideration of supported technologies, State of Colorado IT infrastructure and preferences, development and maintenance costs, and the needs identified for users.

4.1 Map Server Technology

Map server technologies that were considered include ESRI's ArcIMS, ESRI's ArcGIS Server, and open solutions such as MapServer. However, little effort was spent on the evaluation because ArcIMS is being phased out by ESRI and the State has committed to moving forward with ArcGIS Server. Based on State feedback, ArcGIS Server 9.3 will be used. The proposed technologies and ROA system design will allow upgrading to 9.3.1 if it becomes available at some point in the future. ArcGIS Server 9.3.1 provides additional tools for server-side configuration and includes some performance optimization.

ArcGIS Server provides the following capabilities:

- Map service (MS): Uses a map documents (mxd) to create map images. This service also has the capability of using a map document to create Keyhole Markup Language (KML) features.
- Geocode: returns a geographic location for a given address.
- Geodata: provides access to a geodatabase for data query, extraction, update, synchronization, and management.
- Geoprocessing: executes a geoprocessing model and returns the results.
- Image: Uses raster datasets to create map images.
- Web Mapping Service (WMS) – uses OGC Web Map Service standard to deliver maps as images.
- Web Feature Service (WFS) – uses OGC Web Feature Service standard for streaming features.
- Store spatial data in SDE. (At version 9.3 ArcGIS Server adds support for PostgreSQL and embedded SQL Server Express).

- Process and serve images using image server with the result made available as a standard image service (listed above).

ArcGIS Server can provide the same data via multiple services (e.g., as MS and WMS) to facilitate consumption by a variety of applications. ArcGIS Server 9.2 has limited APIs for client development, whereas ArcGIS Server 9.3 provides several options for developers, as discussed in the next section.

4.2 Client (Browser) Technologies

A number of browser solutions have been utilized by Riverside in production work and research projects. The trend in browser tools over the past several years has been towards rich client Web 2.0 technologies, using Ajax approaches or alternatives like Flash and Silverlight to minimize page refreshes and make applications behave more like desktop applications. All common web mapping technologies utilize Ajax to improve web application performance and usability.

Ajax is a combination of technologies that can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. Using Ajax, a web application can request only the content that needs to be updated, thus drastically reducing bandwidth usage and load time. The use of asynchronous requests allows the web application to be more interactive and to respond quickly to inputs, and sections of pages can also be reloaded individually. Users may perceive the application to be faster or more responsive, even if the application has not changed on the server side. Ajax is often implemented in web applications through the use of JavaScript toolkits such as Dojo (the ESRI preference, http://www.dojotoolkit.org/) and jQuery (widely used and now distributed with Microsoft tools, http://jquery.com/). These toolkits deal with differences in browser behavior so that developers can minimize effort spent on the differences. The ArcIMS HTML Viewer that was used for the Flood DSS prototype client doesn't support Ajax and at times is slow because the entire page is being updated.

The main plugin-based technologies are Flash and Silverlight. These technologies are based on viewers that require a one-time download and install on the client computer. These viewers are programmable and run in a browser to deliver rich interactive applications. Adobe Flash applications are developed using the Flex software development kit and ActionScript, which is similar to JavaScript and is the core language of Flash. Microsoft Silverlight applications are developed using the .NET framework. It is not recommended that Flash and Silverlight be used for the Flood DSS because they require special developer skills, which are currently not supported by the State. These technologies might be appropriate in the future.

Most web mapping applications consist of the following:

1. Map component that renders the map in the browser – supplied by the software vendor or organization
2. Programming to customize the map component – JavaScript modifications performed by the developer
3. Client/server data connection via services – requires that client can understand services that are provided by the server

The level of features and ability to customize applications is highly dependent on the intended purpose of the tools and the ability of developers to understand published APIs. For example, the Google Maps API is intended for simpler maps whereas OpenLayers and ESRI's APIs can be used by developers to build complex applications. Google Maps is the most-used web services API by far and consequently its

"look" is familiar to many users. Tools that have been available for the longest time have significant development communities that have provided extensive examples for more complex implementations.

ESRI's Web ADF utilizes server-side ASP.NET to form the content of web pages. This approach allows for significant out of the box functionality but has proven costly to customize and relies on proprietary ESRI tools. This approach also relies on JavaScript and Ajax on the client.

Riverside's review of client technologies compatible with ArcGIS Server 9.3 that are appropriate for a "power user" site focused on OpenLayers (open source) and ESRI APIs, of which there are several variations. All of the tools provide essentially the same capabilities, although some features are more challenging to implement than others (e.g., legend presentation – note that simple maps like Google Maps often have no legend). As one would expect, ESRI's client software provides additional capabilities when the approach is to rely on proprietary software that allows tighter coupling of the client and server software (e.g., Web ADF approach); however, the cost to develop custom tools in the ESRI environment is high. One approach is to wrap ESRI proprietary server features with an API (such as REST), allowing the result (resource) to be utilized in any number of client toolsets.

Section 4.1 indicated that ArcGIS Server can provide services in the following formats: ESRI Map, KML, WMS, and WFS. Most external map services, including some FEMA sites provide data in WMS and WFS using geographic coordinates. Therefore, to support initial and final versions of the system, it is critical that client software be able to consume services and formats that are expected to be used in the system. The following table summarizes this compatibility:

Web Mapping Tools and Ability to Consume Web Services

Client Mapping Component	ArcGIS Server Version Availability	Consumes ESRI Map Service	Consumes WMS	Consumes WFS	Consumes KML
Google Maps	Independent	Yes	Yes, with some effort	Yes, with some effort	Yes
OpenLayers	Independent	ArcGIS 9.3 REST, ArcIMS	Yes	Yes	Yes
ArcGIS Server Web ADF for .NET	9.2, 9.3+	Yes	Yes	No	No
ArcGIS JavaScript API	9.3+	Yes	Yes	No	No
ESRI Flex API	9.3+	Yes	Yes	No	No
ESRI Silverlight API	9.3+	Yes	Yes	No	Yes

'No' in the above table indicates that the API can't connect to the type of service directly. It is still possible to consume the service in the application by connecting to it from within another, supported service. This practice, called "cascading services", is not recommended (see **http://support.esri.com/index.cfm?fa=knowledgebase.techarticles.articleShow&d=35903**).

5. Recommendations

A summary of the recommendations is as follows:

- System: Utilize a Resource Oriented Architecture (ROA) using data services. This approach:
  - Focuses on the data that are of interest to users.
  - Provides unique addressing for all resources to facilitate access, bookmarking, searching, and testing.
  - Is independent of technologies and allows different technologies to work together in the same system.
  - Facilitates testing and troubleshooting.
- Web server:
  - Use Internet Information Server (IIS) 6.0 running on a DNR server with Windows Server 2003 and .NET 3.5.
- Map server: Use ArcGIS Server 9.3 for .NET as the map server.
  - Use version 9.3 for development and operational system.
  - Assume the final ArcGIS Server host will reside at DNR.
  - Configure ArcGIS Server to provide data using ESRI map services and open formats such as WMS, WFS, and KML to reduce reliance on ESRI and support use by multiple technologies (although Google Maps and OpenLayers do consume ESRI formats and can be further evaluated with Flood DSS data).
  - Minimize dependence on specific server features to minimize potential compatibility issues and minimize costs.
- Client (browser): Provide multiple web pages (sites), with similar look and feel. Data and features should be appropriate for different themes and user groups.
  - Provide multiple simple Google Maps, OpenLayers, and/or Web ADF out-of-the-box pages for specific user groups
    - Use WMS, WFS, and KML data from ArcGIS Server (and web server files).
    - Demonstrate access to and visualization of data before considering extensive software customization.
    - Google Maps are likely more familiar to users and will be utilized for some data, with an evaluation that includes user feedback.
    - An out-of-the-box ArcGIS JavaScript API approach will be evaluated for simple sites, with an evaluation that includes user feedback. The out-of-the-box Web ADF approach may also be utilized; however, due to high customization costs, the other solutions are preferred.
    - OpenLayers maps could also be used for simple maps and "power user" sites below, resulting in efficiency in implementation; this option may be evaluated if the ESRI API is not meeting requirements.
  - Provide one or more "power user" sites with more data layers and features, for more specific purposes
    - The out-of-the-box ArcGIS JavaScript API client will be used, focusing on data access and basic presentation. Customization of the client will occur with JavaScript development and use of the API.
    - OpenLayers will be considered if the ArcGIS JavaScript API solution does not meet requirements.
    - Even if using ESRI client software, avoid a reliance on proprietary features that reduce flexibility in upgrading or changing technologies.

- Deployment
  - Evaluate performance early in the project and address issues.
  - Ensure that all data connections and procedures are functioning early in the implementation.
  - Deploy at the State as early as possible to resolve deployment issues.
- Maintenance
  - ArcGIS Server 9.3 is recommended as the server solution. Maintenance of the software can occur as part of DNR support of the server machine. It is also recommended that the CWCB GIS contact (Carolyn Fritz or other) obtain ArcGIS Server 9.3 training in order to more directly provide support for the ArcGIS software, in conjunction with DNR IT staff.
  - Configuration of ArcGIS Server for the Flood DSS will be documented as part of the Flood DSS project, including procedures for maintaining data layers and services. This will facilitate future updates to data.
  - Configuration of the client application (e.g., the layers that are displayed and the services that provide the data) will be defined in a configuration file that can be updated by the State if layer names, locations, or attributes change. The configuration will be documented.
  - Client code maintenance will require knowledge of the web technologies being used, in particular JavaScript, and an understanding of the web mapping APIs that are utilized. If the secondary alternative, Web ADF, is used and requires custom development, then knowledge of .NET and the ESRI ADF will be required in addition to JavaScript.
  - The Java TSTool time series processing components can be maintained consistent with DSS efforts and requires Java expertise.
  - Other processing components (e.g., SNODAS, LaserFiche integration) utilize existing tools or APIs and will need to be maintained by someone with knowledge of those tools. For example, LaserFiche functionality is already maintained by CWCB staff with contracted LaserFiche support, as needed.

Impacts

1. The use of ArcGIS Server 9.3 provides several alternatives for implementing client-side solutions. To minimize overall risk, the mapping interface will initially have an out-of-the-box functionality rather than a custom interface. This will demonstrate overall system functionality and focus on data. The following features of the Flood DSS prototype will require custom development and will be evaluated as soon as possible in order to confirm the final client-side solution:

   - Map scale display (the denominator of the scale fraction)
   - Zoom to specified scale
   - Quick zoom to specified location (e.g., PLSS, district, coordinate)
   - Show LaserFiche documents for selected feature

   The use of OpenLayers for customized interfaces will be evaluated as an alternative if the ArcGIS JavaScript API implementation does not meet user requirements and requires significant development.
2. The impacts of using ArcGIS Server 9.3 compared to 9.3.1 are minimal and consist mainly of the time needed to upgrade ArcGIS software.
3. It is assumed that DNR will host the system, allowing for a number of data connections to be implemented and tested in an operational environment, including links to HydroBase, LaserFiche, and external data servers. We want to ensure that the operational system is robust and not prone to breakdowns with changes in State IT infrastructure. Therefore, appropriate State and project resources will need to be devoted to system configuration and testing. Issues will become more

evident as an initial version of the system is installed at the State. This should occur as soon as possible so that issues can be identified and resolved. Riverside will strive to implement a relatively complete preliminary system in the next 2-3 months that involves data processing, LaserFiche, HydroBase, and GIS components so that a State implementation can be tested in the target hosting configuration.

4. Feedback on the initial draft of this memo raised concerns about the use of Java, JavaScript, and Ajax. JavaScript and Ajax are utilized by all web mapping and many other web technologies – the mapping components will not function without the use of JavaScript and Ajax and therefore it would be difficult to implement the Flood DSS web site without using these technologies. Java is recommended only to utilize existing CDSS system components, such as time series processing using the TSTool Java software. Java use can be isolated and DNR staff (Tom Bergman) has indicated that it is acceptable to run on the web server. Moving away from Java for time series processing will result in expenditures that were not anticipated.

Appendix – Resource Oriented Architecture

This appendix describes concepts that will be utilized in the overall architecture of the Flood DSS, including automated data collection, processing and web mapping components.

A Resource Oriented Architecture (ROA) recognizes that every "thing" in a system can be accessed by a Universal Resource Identifier (URI), familiar to users as Universal Resource Locators (URLs) used in web browsers. The more general term URI will be used in the remainder of the discussion. Examples of resources in the Flood DSS include:

- A list of maps available to users
- A list of layers in a map
- Metadata for a map layer
- Attributes for a map layer (or single feature on a map)
- A list of real-time streamflow stations
- Real-time streamflow time series data
- A graph of time series

Whereas some approaches provide web services that hide the addresses to individual resources, a RESTful approach uses a unique address for every resource. Consequently, software that needs to access a resource can use the URI for the object. Because the approach is inspired by and often used with the Internet, a web browser can also be used to access the URIs.

References for ROA include:

- http://www.crummy.com/writing/RESTful-Web-Services/
- http://en.wikipedia.org/wiki/Resource_oriented_architecture
- http://resources.esri.com/help/9.3/ArcGISServer/apis/rest/index.html

The representation of resources depends on the use of the resources. For example, if a resource is hypermedia that will be viewed in a web browser, then the representation of the resources may be HTML (or XHTML). However, if the resource will typically be consumed by software, then the representation might be XML, JSON (JavaScript Object Notation – note that "JavaScript" has nothing to do with the Java programming language) or another suitable format. Resources can have multiple formats, and the format will be indicated in the URI (e.g., a resources with address ending in .html will return an HTML representation).

Representational state transfer (REST) APIs (Application Programming Interfaces) are published interfaces to the resources. Standard URIs for resources are made available and are supported by software on a web-enabled server. The meaning of addresses should not change once published and therefore addresses may contain a version (e.g., http://flooddss/v1/maplist). Development of an API requires that the developers determine which resources will be made available, the representations of those resources, and also deal with user authentication (security), error handling, etc. REST APIs are available for many systems, are implemented using a variety of technologies, and are becoming more and more popular. See http://www.programmableweb.com/ for a list of popular APIs that can be used to support development – ESRI, Microsoft, and others include some of these packages with their software. An alternative to REST APIs are the "heavy WS*" web services such as SOAP, which originated with Microsoft. A ROA can be implemented using a variety of technologies.

A ROA therefore consists of a system that allows resources to be accessed by different components as needed for full system implementation. Whereas other approaches might provide services that are processes/procedures/actions/verbs, an ROA focuses on things/nouns/resources. This may require a different approach to some work processes; however, for many implementations users are ultimately interested in resources and the execution of processes is up to the system implementers.

Implementing a ROA for the Flood DSS requires that server applications provide REST APIs (or other APIs that facilitate access to resources) and that client software (primarily browser JavaScript applications) can consume the resources returned from the APIs. Examples of how various resources can be handled are as follows:

- Content pages – normal HTML files are the resources (no API needed beyond normal URLs)
- HydroBase and Satellite Monitoring System data, in particular time series – time series are the resources and existing TSTool Java components will be used with open source RESTlet software (www.restlet.org) to publish a REST API. This approach has already been taken by Riverside on other projects. Note that the use of Java TSTool software is being advocated here because reimplementing TSTool in another language would be cost-prohibitive.
- LaserFiche resources may be accessed using an API or the web site may jump to a LaserFiche query page with pre-defined query parameters. The State is updating to LaserFiche 8 and the approach will be resolved soon (September). Ideally a programmable API will be available for LaserFiche to facilitate integration.
- Spatial data may be provided by a map server via web services or by placing KML or other files on a web server. Mapping clients can consume a variety of services as discussed in Section 4.1.
- Other data (reports, engineering data, and database query results) will be made available using approaches similar to the above.

Major considerations in implementing a ROA system include:

1. Understanding the resources ("raw" data and processed resources), organizing the resources, and providing unique published addresses to the resources.
2. Determining the representations for the resources that are suitable for consuming applications.
3. Implementing technologies that provide the final presentation of the resources – once the software knows how to consume the data, how is it presented? There may be multiple presentations of the same data (e.g., a simple site and a "power user" site).

An ROA approach will be used for various system components and is not restricted to just the web mapping application. In addition to facilitating data (resource) access, it facilitates automated testing and troubleshooting.